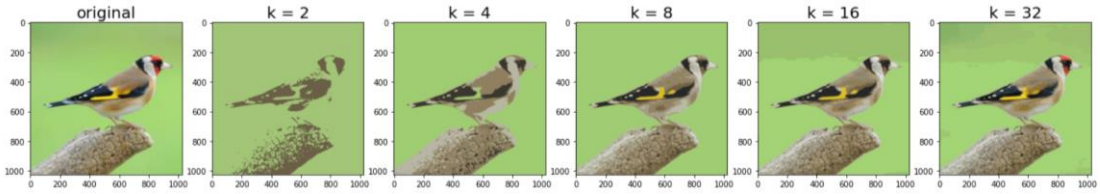
DLCV HW1 Report

學號：B06901045 系級：電機四 姓名：曹林熹

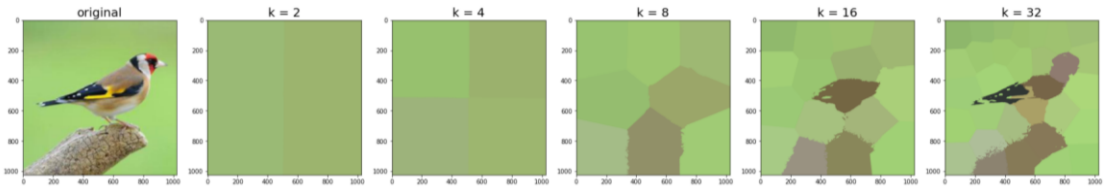
1. Problem 1: K-Means Clustering (24%)

1. (10%) “no collaborators”



由上圖可以看到當 k = 2 時，已經可以將背景圖與鳥分成兩個部分，當 k 愈來愈大也可以將細節劃分更清楚。

2. (6%) “no collaborators”

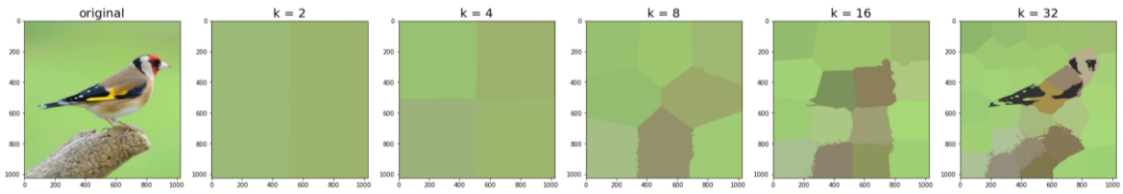


納入 x, y 維度之後，k = 2 直接將圖單純分為左右，到 k = 32 才隱約看出有鳥的圖案。

3. (8%) “no collaborators”

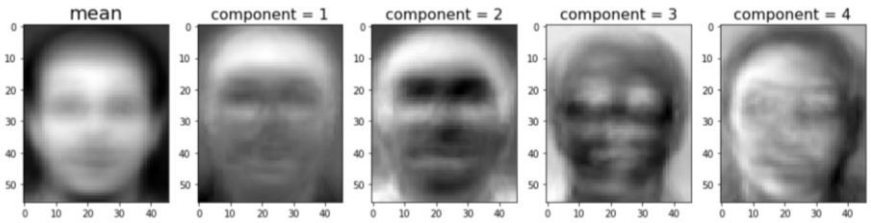
結果顯示，1. 的結果會比 2. 還要好，或許是多了 x, y 這兩個維度，使得 K-means 有過多不必要的資訊（ex.方向看起來不是那麼重要，會造成圖片在 k = 2 直接分成左右兩部分）。

我針對 2. 有稍微做了一點改進，此時我將 K-means 的最大 iteration 從 10 變為 20，而初始的點從採用參數 cv.KMEANS_RANDOM_CENTERS（隨機生成）變成 cv2.KMEANS_PP_CENTERS（先搜尋最可能是 center 的 pixel，再開始初始 K 個點進行分群）。可以看到在 k = 32 時，我們可以更近一步地將翅膀的黑毛還有臉部的黑毛區分出來，不過使用更好的區分方式的代價，就是演算法的所花時間。



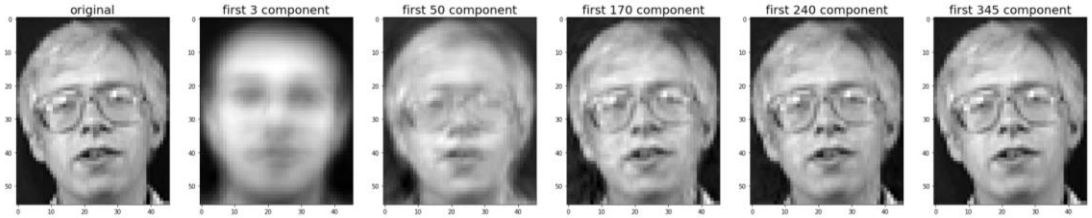
2. Problem 2: Principal Component Analysis (60%)

1. (15%) “no collaborators”



可以看到 mean 隱約是一張人臉，而其他 component 是人的輪廓，但細節較不清楚。

2. (12%) “no collaborators”



只用三個 component 時與 mean 差不多相同，而很顯然地，component 愈多，效果愈好。

3. (6%) “no collaborators”

```
for n = 3 , MSE = 746.7993860092394
for n = 50 , MSE = 236.36913976155375
for n = 170 , MSE = 43.763457491785694
for n = 240 , MSE = 13.406921805396351
for n = 345 , MSE = 0.21543397076006437
```

我認為當 n = 170 時基本上已經還原 80% 以上的圖片，可以清楚認出此人的樣子，然而仍與 n= 345 有相當大的 MSE 差距。

4. (15%) “no collaborators”

在這邊我將 training data 先打亂後分成三組 A、B、C，各 120 筆資料，此分資料的方法採用 sklearn 的既有函式如下：

```
sklearn.model_selection.train_test_split(*arrays, **options)
```

有趣的是，裡面有個 random_state 參數會影響 val 的結果。當 random_state = 0 時，(k=1, n=170) 出來的預測結果有最高的分數。

```
for n = 3 , k = 1 , score = 0.6888888888888888
for n = 3 , k = 3 , score = 0.6638888888888889
for n = 3 , k = 5 , score = 0.6055555555555556
for n = 50 , k = 1 , score = 0.9444444444444443
for n = 50 , k = 3 , score = 0.9166666666666666
for n = 50 , k = 5 , score = 0.9027777777777778
for n = 170 , k = 1 , score = 0.9472222222222223
for n = 170 , k = 3 , score = 0.9111111111111111
for n = 170 , k = 5 , score = 0.8972222222222223
```

然而選取不同的 random_state 時，我發現（k=1, n=50）的預測結果並不會輸給（k=1, n=170）的狀況，或許當 n 大到一個值之後，k 值的選取重要性比較大。

```
for n = 3 , k = 1 , score = 0.6916666666666668
for n = 3 , k = 3 , score = 0.6472222222222223
for n = 3 , k = 5 , score = 0.6027777777777779
for n = 50 , k = 1 , score = 0.9388888888888888
for n = 50 , k = 3 , score = 0.9138888888888889
for n = 50 , k = 5 , score = 0.8777777777777778
for n = 170 , k = 1 , score = 0.9388888888888888
for n = 170 , k = 3 , score = 0.8888888888888888
for n = 170 , k = 5 , score = 0.8333333333333334
```

（a）當 random_state = 1，(k=1, n=50)、(k=1, n=170）分數相同

```
for n = 3 , k = 1 , score = 0.6916666666666668
for n = 3 , k = 3 , score = 0.6611111111111111
for n = 3 , k = 5 , score = 0.6555555555555556
for n = 50 , k = 1 , score = 0.9583333333333334
for n = 50 , k = 3 , score = 0.9444444444444443
for n = 50 , k = 5 , score = 0.9111111111111111
for n = 170 , k = 1 , score = 0.9555555555555556
for n = 170 , k = 3 , score = 0.9305555555555557
for n = 170 , k = 5 , score = 0.8777777777777778
```

（b）當 random_state = 3，(k=1, n=50）分數 > (k=1, n=170）分數

關於最後的 k、n 取值，我選擇 k=1 與 n=170 作為我最後預測的模型超參數，不過關於（k=1, n=50）我也有在第五小題做出了預測，揭發我心中的好奇。

5.（12%）“no collaborators”

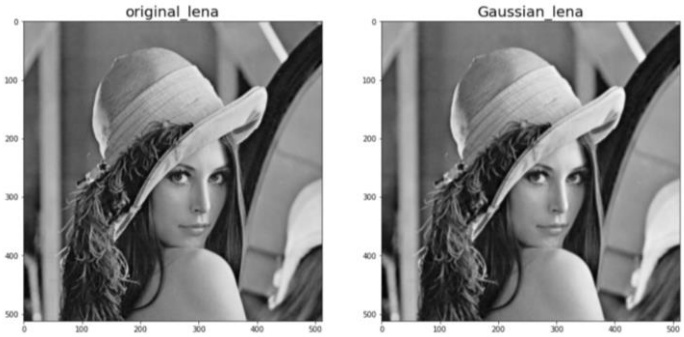
```
predicted score = 0.95
```

（a）(k=1, n=170）分數

我們最後的預測結果有 95% 的正確率，算是高的準確度，而我也順便預測了（k=1, n=50）的分數，也有 92.5% 的正確率。雖然在 val 時，感覺 n=50 已經相當足夠預測不錯的成績，然而在最後的 test 是將更多的資料拿去 train，再去預測（因為把 val data 也放進去 train)，因此不意外地當 n 愈大能夠有更多向量使機器去判別圖形，也有較高準確率。

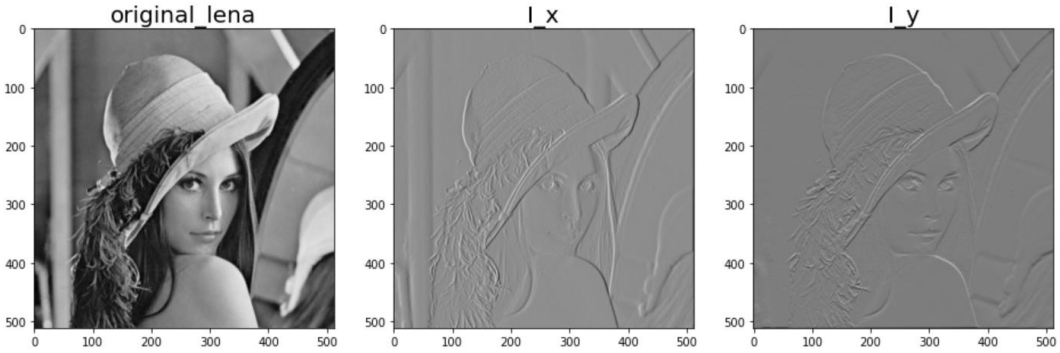
3. Problem 3: Image Filtering (36%)

1. (12%) “no collaborators”



我們可以稍微看出使用 Gaussian filter 後，圖片變得比較模糊，有 smoothing 的效果，在有光影的部分可以看到較明顯的差別。

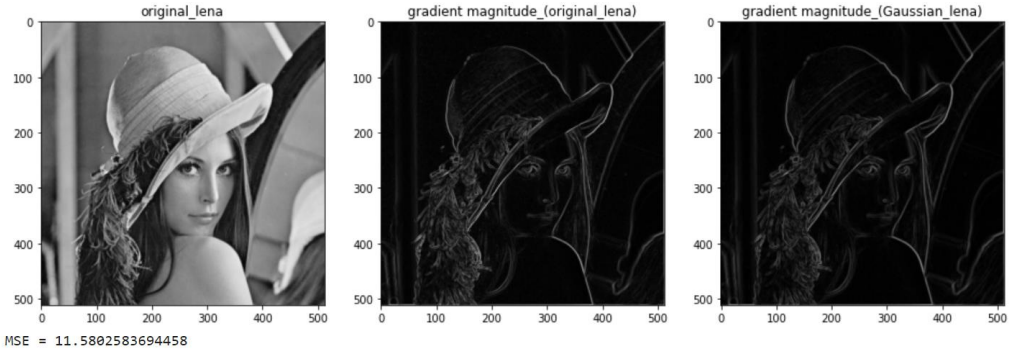
2. (16%) “no collaborators”



$$k_x = \frac{1}{2} \cdot [-1, 0, 1] \qquad k_y = \frac{1}{2} \cdot [-1, 0, 1]^T$$

我的觀察發現 Ix 產生的 lena，鼻樑部分比較凸，然而 Iy 產生的 lena，鼻樑較凹，且在邊緣也有黑白的差別。

3. (8%) “no collaborators”



我覺得兩張結果蠻像的，主要輪廓都有畫出來，然而在細節上比如帽子、頭髮等，拿原圖的邊緣檢測能夠給出更多的線條，而兩張的 MSE 大約是 11.58，也不算太大。