

- a- Ödevde kullanmak için aşağıdaki fotoğrafta belirtilen kütüphaneleri import ettim. “yuzler” adında oluşturduğum klasöre .pgm uzantılı olan dosyaları topladım. Boyut yüksek olmasından dolayı işlem süresi uzun olacağından boyutları düşürdüm. yuz\_matrisi ismi adlı matrisle değerleri tutarak array haline getirdim. Yukarıdaki açıklamaya ilişkin kod aşağıda belirtilmiştir.

```
import numpy as np
from skimage.io import imread
from skimage.measure import block_reduce
import glob
from sklearn.decomposition import PCA
from PIL import Image
from sklearn.decomposition import PCA

foto_yolu = r"C:\Users\yigit\Desktop\odev2\yuzler\*.pgm"
foto_yolu = glob.glob(foto_yolu)

yuz_matrisi = []

for face in foto_yolu:
    foto = imread(face)
    foto = block_reduce(foto, block_size=(4,4), func=np.mean)
    foto = np.array(foto.flatten())
    yuz_matrisi.append(foto)

yuz_matrisi = np.array(yuz_matrisi)
```

10\*2016 boyutundaki öz nitelik matrisinin kovaryansını almamız gerekiyor. Aşağıda yazdığım kodda bu işlemi yapabiliriz. Ardından bu kodunun çıktısında ise 2016\*2016 boyutunda kovaryans matrisi çıktısı alıyoruz. Aşağıda ilgili kodlar ve çıktılar belirtilmiştir.

```
yuz_matrisi = np.array(yuz_matrisi)
pca = PCA()
pca.fit(yuz_matrisi)

kovaryans = pca.get_covariance()
```

	0	1	2	3	4	5	6
0	77.4375	87	98.9375	103.625	106.625	117.812	122.375
1	58.5	63.125	68	75.1875	88.3125	86.25	88.4375
2	28.875	36.3125	49	56.4375	63.5625	77.5	87.375
3	31.3125	36.5	51.125	55.625	62	71.375	87
4	48.9375	50.9375	67.0625	84.125	100.625	118.625	130.875
5	3.6875	4.0625	4	9.5	14.25	24.1875	51.0625
6	38	41.375	45.5625	50.1875	56.3125	61	71
7	62.5625	76	78	81	95.375	100.625	110.438
8	69.875	75.625	83.3125	92.875	97.625	101.438	104.375
9	67.125	74.5625	80.3125	82	83.875	93.9375	101.312

10\*2016 öz nitelik matrisi

	0	1	2	3	4	5	6
0	532.288	582.877	592.17	579.037	564.828	533.959	389.024
1	582.877	647.258	663.205	651.68	647.473	620.404	463.938
2	592.17	663.205	710.024	711.139	712.844	702.174	538.061
3	579.037	651.68	711.139	731.088	744.296	743.858	578.815
4	564.828	647.473	712.844	744.296	779.916	786.491	626.363
5	533.959	620.404	702.174	743.858	786.491	813.152	662.337
6	389.024	463.938	538.061	578.815	626.363	662.337	563.596
7	434.537	521.336	598.546	630.48	685.591	712.839	593.189
8	312.92	394.655	456.997	487.753	552.499	587.818	511.743
9	210.17	279.552	336.854	365.43	426.584	464.219	423.584
10	327.951	413.549	485.247	503.992	559.696	592.064	512.929

2016\*2016 kovaryans matrisi

- b- A şıkında çıktı olarak aldığımız kovaryans matrisinin özvektörlerini ve özdeğerlerini aşağıda kodda belirtildiği gibi elde ediyorum. (PCA kütüphanesi.)

```
vektor = pca.components_
value = pca.explained_variance_
```

value - NumPy object

	0
0	604547
1	263250
2	221252
3	197951
4	148602
5	129402
6	93748.5
7	77558.7
8	51470.5
9	7.06528e-26

Özdeğer matrisi

vektor - NumPy object array

	0	1	2	3	4	5	6
0	-0.0146191	-0.0184074	-0.0207721	-0.0215319	-0.0241067	-0.0252245	-0.0234363
1	0.0317136	0.0335855	0.034609	0.033177	0.0303339	0.0301719	0.0211334
2	0.0212575	0.0202149	0.0134653	0.0113718	0.00779474	0.000649829	-0.0048458
3	-0.00994696	-0.0124325	-0.0194987	-0.0240575	-0.0263728	-0.0284298	-0.0201087
4	0.00949727	0.0108955	0.00897406	0.00432148	0.00440762	-0.000327315	-0.00615677
5	0.000612021	0.00483263	-0.00212215	-0.00414184	0.00187359	0.000555871	0.0014176
6	-0.00400663	-0.00345941	-0.00446669	0.0038916	0.0121821	0.0142972	0.0148697
7	-0.00697739	-0.00587558	-0.00705888	-0.0120898	-0.0151657	-0.0106247	-0.00521473
8	-0.000682764	-0.000221628	0.00194014	-0.00148575	0.00170574	-0.00389983	4.19845e-05
9	0.273323	0.125216	0.054229	-0.110598	0.0375402	-0.221672	0.170164

Özvektör matrisi

Kullandığım PCA kütüphanesiyle birlikte çıktı olarak seçilmiş özvektörler verildi. Fotoğrafların siyah beyazlığını daha görünür kılmak adına aşağıda görüldüğü üzere 2048 ile çarptım. İlgili kod ve koda ait çıktılar belirtilmiştir.



```
for i in range(10):
    foto_cikti = Image.fromarray(vektor[i].reshape(48,42)*2048)
    foto_cikti.show()
```

- c- Bu aşamadan sonra ise elimizdeki projeksiyon hesaplama formülünü kullanarak seçilmiş özvektörler ve kovaryans matrisiyle çarpıp fotoğrafların izdüşümünü buldum. Bu hesaplamayı yapmak için numpy ve PIL kütüphanesinden yararlandım.

```
cikti = np.matmul(vektor,kovaryans)

cikti = cikti.T
ters = np.linalg.pinv(np.matmul(cikti.T,cikti))
project = np.matmul(cikti, ters)
project = np.matmul(project, cikti.T)
cikti_yeni = np.matmul(yuz_matrisi, project)

for i in range(10):
    foto_cikti = Image.fromarray(cikti_yeni[i].reshape(48,42))
    foto_cikti.show()
```



- d- Her bir izdüşürülmüş fotoğrafın oklid uzaklığını hesaplamak için aşağıdaki kodu kullandım. Kod sonucu 10 fotoğrafın da öklid uzaklığı eşit çıktı.

```
oklid = []
for i in range(10):
    temp = np.linalg.norm(yuz_matrisi[i]- cikti_yeni[i])
    oklid.append(temp)
for i in range(10):
    print("",i+1,". ", "",oklid[i])
```

```
1 . 1898.272458501607
2 . 1898.272458501607
3 . 1898.272458501607
4 . 1898.272458501607
5 . 1898.272458501607
6 . 1898.272458501607
7 . 1898.272458501607
8 . 1898.272458501607
9 . 1898.272458501607
10 . 1898.272458501607
```

- e- Orijinal fotoğraflar ile izdüřürölen göröüntöler arasında anlaşılabilirliklerin yüksek olduğunu ve benzerliklerinin yüksek olduğunu düşünüyorum bu yüzden 10 yüz göröüntösünün yeterli olduğunu düşünüyorum. Daha fazla fotoğrafın modellenmesi benzerlik açısından pozitif etki gösterecektir. Daha fazla model kullanması durumunda baz vektörlerleri daha fazla yer kaplayacaktır. Bu durumda ise negatif etki göstererek hem bellek açısından hem işlem hızı açısından negatif etki gösterecektir.

## KODUN TAMAMI

```
1 import numpy as np
2 from skimage.io import imread
3 from skimage.measure import block_reduce
4 import glob
5 from sklearn.decomposition import PCA
6 from PIL import Image
7 from sklearn.decomposition import PCA
8
9 foto_yolu = r"C:\Users\yigit\Desktop\odev2\yuzler\*.pgm"
10
11 foto_yolu = glob.glob(foto_yolu)
12
13 yuz_matrisi = []
14
15 for face in foto_yolu:
16     foto = imread(face)
17     foto = block_reduce(foto, block_size=(4,4), func=np.mean)
18     foto = np.array(foto.flatten())
19     yuz_matrisi.append(foto)
20
21 yuz_matrisi = np.array(yuz_matrisi)
22 pca = PCA()
23 pca.fit(yuz_matrisi)
24
25 kovaryans = pca.get_covariance()
26 vektor = pca.components_
27 value = pca.explained_variance_
28
29 """
30 for i in range(10):
31     foto_cikti = Image.fromarray(vektor[i].reshape(48,42)*2048)
32     foto_cikti.show()
33 """
34
35 cikti = np.matmul(vektor, kovaryans)
36
37 cikti = cikti.T
38 ters = np.linalg.inv(np.matmul(cikti.T, cikti))
39 project = np.matmul(cikti, ters)
40 project = np.matmul(project, cikti.T)
41 cikti_yeni = np.matmul(yuz_matrisi, project)
42 """
43 for i in range(10):
44     foto_cikti = Image.fromarray(cikti_yeni[i].reshape(48,42))
45     foto_cikti.show()
46 """
47
48 oklid = []
49 for i in range(10):
50     temp = np.linalg.norm(yuz_matrisi[i] - cikti_yeni[i])
```