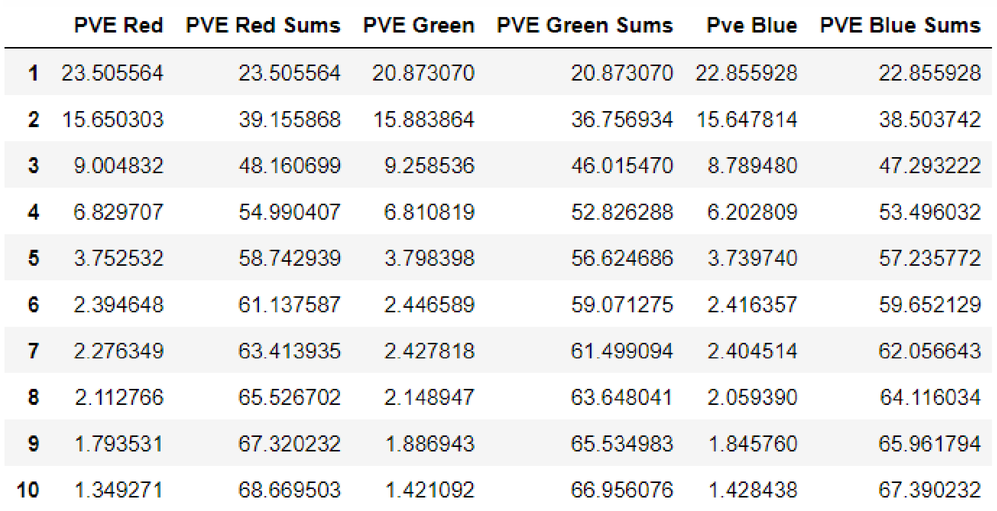
Homework 2

1 PCA & Cats

In this homework we are expected to analyze various cat images using Principal Component Analysis(PCA). In order to realize this procedure we have given a dataset which consists of 5653 different 512 $\times$ 512 sized cat images. Firstly, it is indicated that those images must be resized and datatype of image matrices must be changed. Therefore, I have resized the images to 64 $\times$ 64 and change the datatype to float. Then the data is split to 3 different sub data according to Red, Green and Blue. At the end I obtained 3 different 5653 $\times$ 4096 matrices.

1.1

After pre-processing I have applied pca to each color matrix. In this process firstly I have subtracted the mean from the data then find the eigenvalues and eigenvectors of the new matrix. After sorting eigenvalues, I have sorted eigenvectors with corresponds to proper eigenvalues. In this way principal components are found.
In the question we were asked to find the Proportion of Variance Explained (PVE) for each 10 principal components and their sum for each color. By taking the first 10 eigenvalue from each color matrix (X_i) and proportioning them to sum of color matrix PVEs are obtained easily. The PVE and their sums for first 10 principal components are given below.



	PVE Red	PVE Red Sums	PVE Green	PVE Green Sums	Pve Blue	PVE Blue Sums
1	23.505564	23.505564	20.873070	20.873070	22.855928	22.855928
2	15.650303	39.155868	15.883864	36.756934	15.647814	38.503742
3	9.004832	48.160699	9.258536	46.015470	8.789480	47.293222
4	6.829707	54.990407	6.810819	52.826288	6.202809	53.496032
5	3.752532	58.742939	3.798398	56.624686	3.739740	57.235772
6	2.394648	61.137587	2.446589	59.071275	2.416357	59.652129
7	2.276349	63.413935	2.427818	61.499094	2.404514	62.056643
8	2.112766	65.526702	2.148947	63.648041	2.059390	64.116034
9	1.793531	67.320232	1.886943	65.534983	1.845760	65.961794
10	1.349271	68.669503	1.421092	66.956076	1.428438	67.390232

Figure 1: PVEs for Each Colors and Their Sums

If the table is examined it can be seen that the PVE of preceding principle components are higher.

Namely, since eigenvectors were sorted from biggest to smallest, PVEs are also automatically sorted. In order to see this relation better I have also provided their plots, which can be seen below. However, we have lots of principal components so examining them together becomes hard. Therefore, I only plotted the first 50 principal components for each color. As seen from all the plots above, they all decreasing.

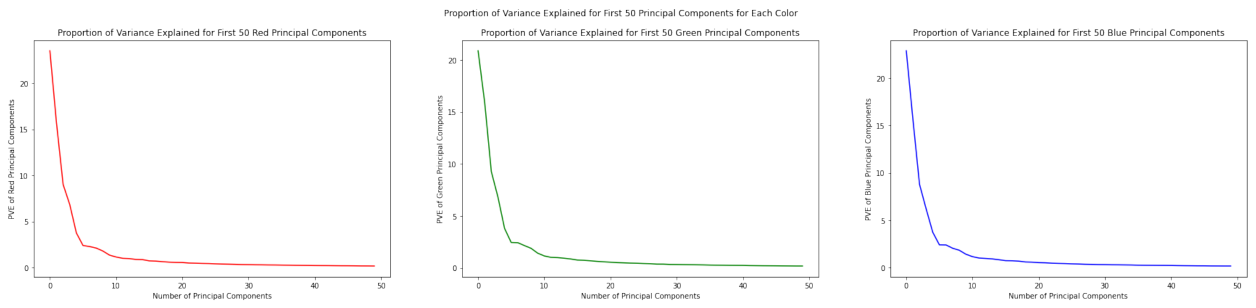


Figure 2: Plots of first 50 PVEs for Each Colors

On top of those we were also expected to find a threshold such that the minimum number of principal components that are required to obtain at least 70% PVE for all channels. After proper coding I fond that :
While the sum of first 11 principal component PVE is 69.815%, sum of first 12 principal component PVE is 70.815% for red color, so minimum number of principal components is 12.
While the sum of first 12 principal component PVE is 69.128%, sum of first 13 principal component PVE is 70.122% for green color,so minimum number of principal components is 13.
While the sum of first 12 principal component PVE is 69.584%, sum of first 13 principal component PVE is 70.565% for blue color,so minimum number of principal components is 13.

In order to see those thresholds better, the plots below can be examined.

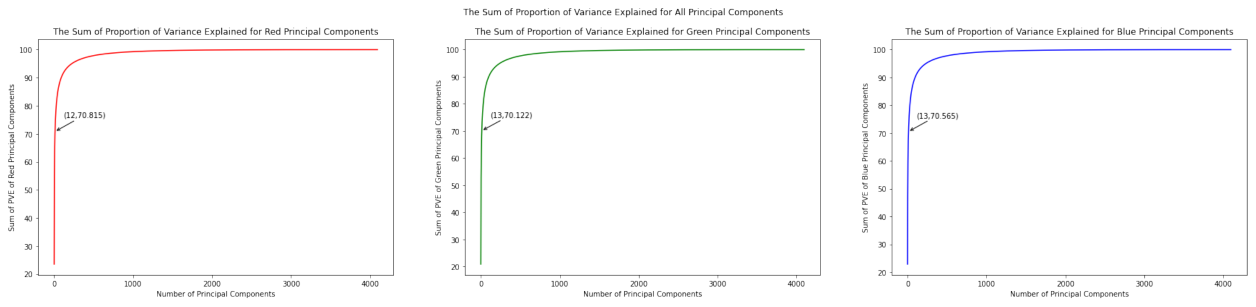


Figure 3: Sum of PVEs for Each Colors

1.2

In this section I have firstly reshaped the each principal component to a 64×64 matrix, then normalize those matrices using min(0) max(1) scaling method for each color. After those steps, in order to see the first ten eigenvectors by using all colors, matrices are stacked together, (according to corresponding RGB channels). In this way I have obtained a $10 \times 64 \times 64 \times 3$ data. Then I have displayed the those 10 eigenvectors, which are $64 \times 64 \times 3$ now. Which can be seen below.

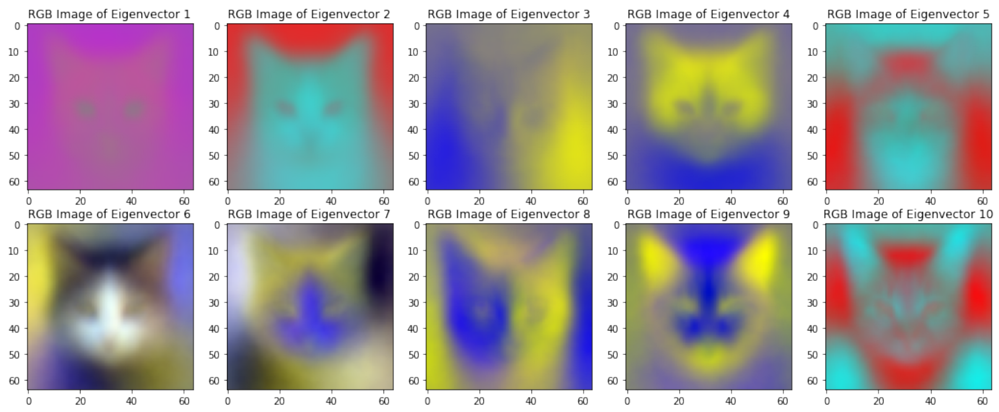


Figure 4: RGB Images of First 10 Eigenvectors

At the above you can see the $64 \times 64 \times 3$ RGB visualization of first 10.At first glance they may seem like randomly printed cat images. However, if you look from first to last, you can see that at the beginning an overall cat structure is seen. Then at every step you can see different highligthed details. That is to say those eigenvectors are the most closer components to overall cat dataset, and so the dataset can be reconstructed using those matrices.

1.3

In this section we are expected to reconstruct an image using the first k principal components such that $k \in [1, 50, 250, 500, 1000, 4096]$. To realize this let's call original image one color matrix as X and first k principal components of one color as Y. Firstly we need to find XY and then we need to project XY to Y which corresponds to XYY^T. This process is repeating for each color channel and then all channels are stacked. After stacking the created matrix should be reshaped to $64 \times 64 \times 3$. At the end of this process we got the reconstruction of the original image. The reconstructions for each k value is shown below.

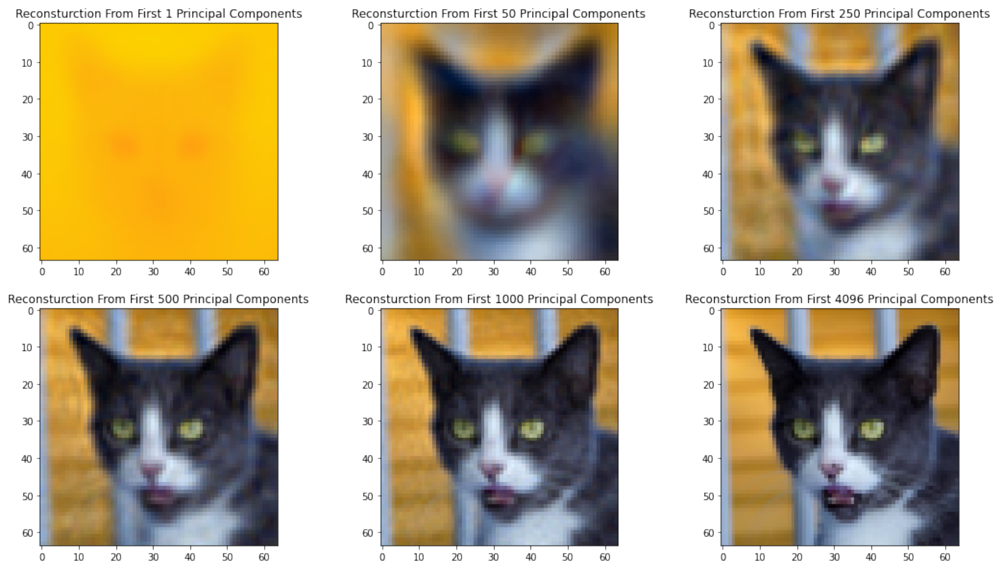


Figure 5: Reconstruction for Different First Principal Components Sizes from K values

As you can see above in the first construction (for only first principal component)we can almost only see the first principal component. However, with more principal components, the image becomes better. At the end (with all principal components) the image become flawless. In order to see the difference between reconstructions and original image numerically you can check the plot below. As metrix I have used Mean squared error.

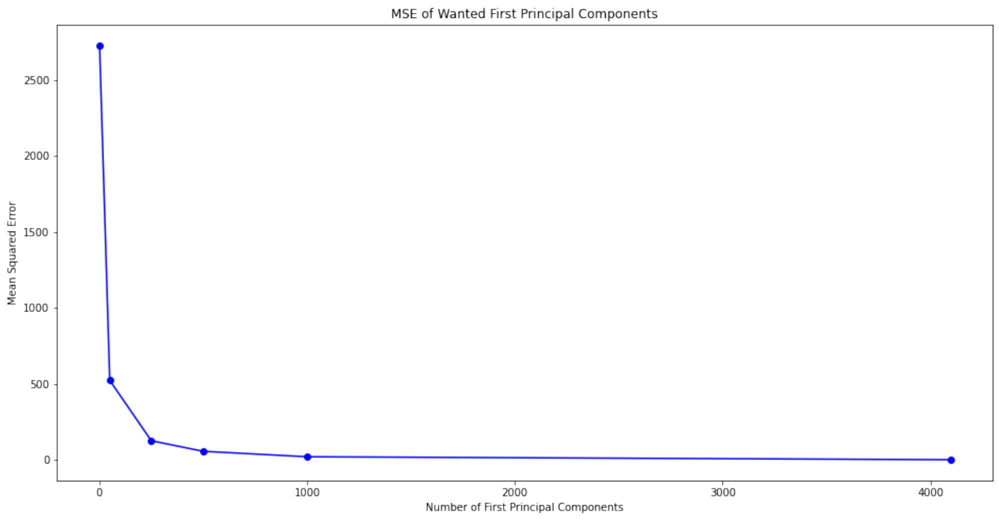


Figure 6: MSE of Wanted First Principal Components

As you can see from the plot above the MSE decreases dramatically for more first principal components. For 4096 (all) principal components we get 4.01e-18 MSE which is almost zero and most probably this error come from basic computations. In order to compare the best reconstruction with the original image you can check figure 7 below.

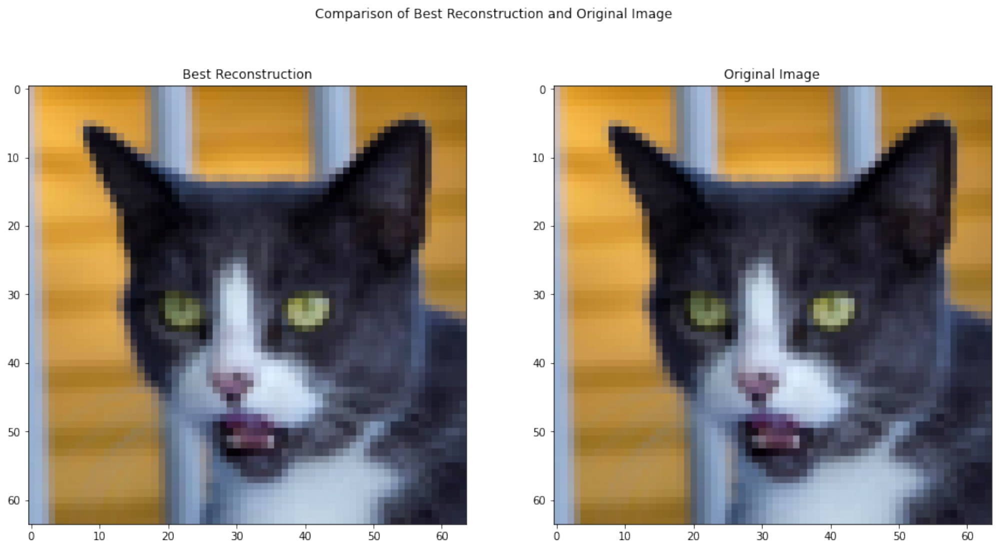


Figure 7: Comparison of Best Reconstruction and Original Image

2 Appendix

```
import os
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt

####Change direction####
dir_name = r"C:\Users\Oguz\Desktop\Bilkent\3.2\CS464\hw2\afhq_cat"

print("Please wait, it can take few minutes to run completely")
pics=[]
for i in os.listdir(dir_name):
    pics.append(Image.open(os.path.join(dir_name,i)).resize((64,64), Image.BILINEAR))

imgs=[]
for j in pics:
    imgs.append(np.array(j,dtype=float).reshape(-1,3))

X=np.array(imgs)

X_0=X[:,:,0]
X_1=X[:,:,1]
X_2=X[:,:,2]

# # Question 1.1

def pca(matrix):
    matrix_mean=matrix-np.mean(matrix,axis=0)
    cov=np.cov(matrix_mean, rowvar = False)
    value, vector = np.linalg.eig(cov)
    sort_index = np.argsort(value)[::-1]
    sort_eigenval = value[sort_index]
    sort_eigenvec = vector[:,sort_index]
    return sort_eigenval, sort_eigenvec

val_0,vec_0=pca(X_0)
val_1,vec_1=pca(X_1)
val_2,vec_2=pca(X_2)

def pve(sort_val):
    n=len(sort_val)
    pve = sort_val*100/np.sum(sort_val)
    pve_sum=pve.dot(np.triu(np.ones((n,n))))
    return pve, pve_sum

pve_R,pve_sum_R=pve(val_0)
pve_G,pve_sum_G=pve(val_1)
pve_B,pve_sum_B=pve(val_2)

pve_table=pd.DataFrame(np.array([pve_R,pve_sum_R,pve_G,pve_sum_G,pve_B,pve_sum_R]).T \
,index=np.arange(1,len(pve_G)+1),columns= \
["PVE Red","PVE Red Sums","PVE Green","PVE Green Sums","Pve Blue","PVE Blue Sums"])
#pve_table[:10]
print(pve_table[:10])

fig,ax=plt.subplots(1,3,figsize=(30,6))

ax[0].plot(pve_R[:50],"r")
ax[0].set_ylabel("PVE of Red Principal Components")
ax[0].set_xlabel("Number of Principal Components")
ax[0].set_title("Proportion of Variance Explained for \
First 50 Red Principal Components")

ax[1].plot(pve_G[:50],"g")
ax[1].set_ylabel("PVE of Green Principal Components")
ax[1].set_xlabel("Number of Principal Components")
ax[1].set_title("Proportion of Variance Explained for \
First 50 Green Principal Components")

ax[2].plot(pve_B[:50],"b")
ax[2].set_ylabel("PVE of Blue Principal Components")
ax[2].set_xlabel("Number of Principal Components")
ax[2].set_title("Proportion of Variance Explained for \
First 50 Blue Principal Components")

fig.suptitle("Proportion of Variance Explained for \
 First 50 Principal Components for Each Color")
fig.show()

def threshold(th,pve):
    th_mat=np.argwhere(pve>th)
    return (th_mat[0][0]+1, pve[th_mat[0][0]])
```

```
th_R=threshold(70,pve_sum_R)
th_G=threshold(70,pve_sum_G)
th_B=threshold(70,pve_sum_B)

print("While the sum of first {} principal component PVE is\
 {:.3f}%, sum of first {} principal component PVE is {:.3f}% \
for red color.".format(th_R[0]-1,pve_sum_R[th_R[0]-2],th_R[0],th_R[1]))

print("While the sum of first {} principal component PVE is \
{:.3f}%, sum of first {} principal component PVE is {:.3f}% \
for green color. ".format(th_G[0]-1,pve_sum_G[th_G[0]-2],th_G[0],th_G[1]))

print("While the sum of first {} principal component PVE is \
{:.3f}%, sum of first {} principal component PVE is {:.3f}% \
for blue color.".format(th_B[0]-1,pve_sum_B[th_B[0]-2],th_B[0],th_B[1]))

fig,ax=plt.subplots(1,3,figsize=(30,6))

ax[0].plot(pve_sum_R,"r")
ax[0].annotate("({},{:.3f})".format(th_R[0],th_R[1]),th_R,\
 xytext=(th_R[0]+100,th_R[1]+5),arrowprops=dict(arrowstyle="->") )
ax[0].set_ylabel("Sum of PVE of Red Principal Components")
ax[0].set_xlabel("Number of Principal Components")
ax[0].set_title("The Sum of Proportion of Variance \
Explained for Red Principal Components")

ax[1].plot(pve_sum_G,"g")
ax[1].annotate("({},{:.3f})".format(th_G[0],th_G[1]),th_G,\
 xytext=(th_G[0]+100,th_G[1]+5),arrowprops=dict(arrowstyle="->") )
ax[1].set_ylabel("Sum of PVE of Green Principal Components")
ax[1].set_xlabel("Number of Principal Components")
ax[1].set_title("The Sum of Proportion of Variance \
Explained for Green Principal Components")

ax[2].plot(pve_sum_B,"b")
ax[2].annotate("({},{:.3f})".format(th_B[0],th_B[1]),th_B,\
 xytext=(th_B[0]+100,th_B[1]+5),arrowprops=dict(arrowstyle="->") )
ax[2].set_ylabel("Sum of PVE of Blue Principal Components")
ax[2].set_xlabel("Number of Principal Components")
ax[2].set_title("The Sum of Proportion of Variance \
Explained for Blue Principal Components")

fig.suptitle("The Sum of Proportion of Variance \
Explained for All Principal Components")
fig.show()

## Question 1.2

normalization = lambda x:(x-x.min()) / (x.max()-x.min())

def first_10(mat):
    _10_pca=mat[:,:10]
    reshaped_10_pca=np.transpose(_10_pca, axes= (1,0)).reshape(10,64, 64)
    return reshaped_10_pca

reshaped_R=normalization(first_10(vec_0))
reshaped_G=normalization(first_10(vec_1))
reshaped_B=normalization(first_10(vec_2))

ten_img=np.array([reshaped_R,reshaped_G,reshaped_B])
ten_img=np.transpose(ten_img,axes=(1,2,3,0))

row = 2
col = 5
plt.figure(figsize=(18,7))
for q in range(10):
    plt.subplot(row,col,q+1)
    plt.imshow(ten_img[q])
    plt.title("RGB Image of Eigenvector {}".format(q+1))
plt.suptitle("RGB Images of First 10 Eigenvectors")

## Question 1.3

def reconstruct(data, eigvec):
    mean=np.mean(data,axis=0)
    data=data-mean
    project = np.matmul(data,eigvec.dot(eigvec.T)) + mean
    return project

k_list= [1, 50, 250, 500, 1000, 4096]

sec_img=Image.open(os.path.join(dir_name,"flickr_cat_000003.jpg"))\
.resize((64,64), Image.BILINEAR)
sec_img=np.array(sec_img,dtype=np.float32).reshape(-1,3)

R_list=[]
```

```
G_list=[]
B_list=[]
for k in k_list:
    R_list.append(reconstruct(sec_img[:,0], vec_0[:,:k]))
    G_list.append(reconstruct(sec_img[:,1], vec_1[:,:k]))
    B_list.append(reconstruct(sec_img[:,2], vec_2[:,:k]))

rec=np.array([R_list,G_list,B_list])
rec=np.transpose(rec,axes=(1,2,0)).reshape(6,64,64,3)

plt.figure(figsize=(18,10))
for l in range(6):
    plt.subplot(2,3,l+1)
    plt.imshow(normalization(rec[l]).reshape(64,64,3))
    plt.title("Reconsturction From First {} \
Principal Components".format(k_list[l]))
plt.suptitle("Reconstruction for Different \
First Principal Components Sizes")

errors=[]
for i in range(len(k_list)):
    errors.append(np.square(sec_img.reshape(64,64,3)-rec[i]).sum()/(64*64*3))

plt.figure(figsize=(16,8))
plt.plot(k_list,errors,"bo-")
plt.xlabel("Number of First Principal Components")
plt.ylabel("Mean Squared Error")
plt.title("MSE of Wanted First Principal Components")

plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plt.imshow(normalization(rec[5]).reshape(64,64,3))
plt.title("Best Reconstruction")
plt.subplot(1,2,2)
plt.imshow((normalization(sec_img).reshape(64,64,3)))
plt.title("Original Image")
plt.suptitle("Comparison of Best Reconstruction and Original Image")
plt.show()
```