CS452 – Data Science with Python Assignment-3
Image Compression by K-Means Clustering

Created by Aylin Aydın S018183

1-Introduction

Color images are stored in digital media via pixels. Pixels are the very small intensity values (dots) that comprise an image. Each pixel in a colorful image contains color information. Each pixel in a colorful image is composed of three bytes containing RGB (Red-Blue-Green) data, with red, blue, and green intensity values for each pixel. A colored digital image, as you may have noticed, can also be very large due to the fact that each pixel requires 3 bytes or (3X8) bits of intensity data. As a result, these images are frequently stored as compressed images with fewer bits for intensity values and, thus, less storage memory. Image compression is a data compression technique used to lower the cost of storing or transmitting digital photos.

In this paper, the K-means algorithm is used to compress the colored image based on its pixel values. The optimization approach K-means clustering is used to locate the 'k' clusters or groups in a set of data points. The data points are grouped together based on a degree of resemblance. It begins by randomly initializing the 'k' clusters, and then seeks to minimize the distance between every data point and the cluster center in each cluster based on some. There are mainly two iterative steps in the algorithm. First one is the assignment step, each data point is assigned to a cluster whose center is nearest to it. Second one is the update step, new cluster centers (centroids) are calculated from the data points assigned to the new cluster by choosing the average value of these data points. These recurrent procedures are repeated until the centroids stop moving away from their clusters. **[1]**

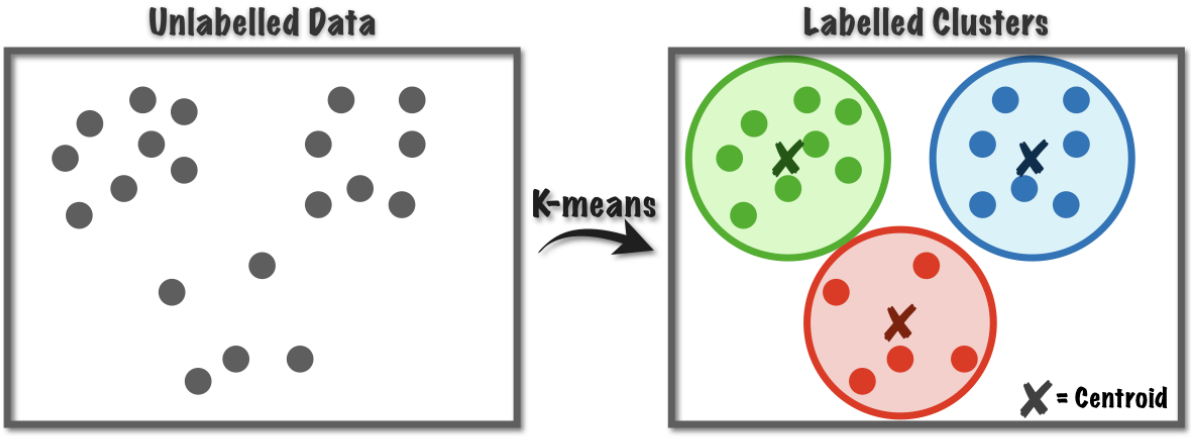


Figure 1. K-Means Cluster Algorithm Visualization **[2]**

2-Methodology

This study was prepared using numpy, pandas, matplotlib, seaborn and skimage from python's existing libraries by using five different images in python software language and performing operations via Jupyter Notebook.

In this study, we looked at 5 distinct images and discussed them one by one. To compare the data gathered before, all of these images were converted to .png format. After that, each image was read one by one with the skimage library and visualized on the notebook. For each image, the number of unique colors it contains and the number of bytes expressing its size were calculated. Then, for each image [k=2, 4, 8, 16, 32, 64, 128, 256] eight different types of k-means clustering were applied. All of them were evaluated on the basis of three different metrics. The metrics used during these evaluations were calculated using methods written in pure python without using any additional libraries. Information about the methods will be provided in the implementation details section. After all the kmeans clustering algorithms, the closest color names that the clustroids have on the webcolors were determined. After all of these steps were completed for each picture, the best k value for compressing the image was determined based on the compressed image's size and the derived metrics with the help of elbow method.

3-Implementation Details

The necessary libraries for image processing have been imported.

```
from skimage import io
from sklearn.cluster import KMeans
import numpy as np
import pandas as pd
from skimage.transform import rescale, resize, downscale_local_mean

import matplotlib.pyplot as plt
import matplotlib.image as img

from PIL import Image
import cv2
import os
import webcolors
```

Skimage library were used to read the images. After that, the pictures were resized to ensure ease of operation. In order to perform the correct operations with the numpy library in the future, the received image was converted from array type uint to int. After reading the Flowers image, when its shape was taken, there were 4 channel - most likely caused by the frame around it -, there should have been 3 expressing r,g, b in the color channel. The frame around it was removed and made 3-channel in order to avoid confusion. After these operations, the number of bytes of the original image and the number of unique colors were calculated for each image. The k-means cluster algorithm was executed and fit [k=2,4,8,16,32,64,128,256].

```
#Implement k-means clustering to form k clusters
kmeans_baboon_2= KMeans(n_clusters=2)
kmeans_baboon_2.fit(baboon_image)

#Replace each pixel value with its nearby centroid
compressed_image_baboon_2 = kmeans_baboon_2.cluster_centers_[kmeans_baboon_2.labels_]
compressed_image_baboon_2  = np.clip(compressed_image_baboon_2.astype('uint8'), 0, 255)

#Reshape the image to original dimension
compressed_image_baboon_2  = compressed_image_baboon_2.reshape(rows_baboon,cols_baboon,3)
#Save and display output image
io.imsave("compressed_image_baboon_2.png", compressed_image_baboon_2 )
io.imshow(compressed_image_baboon_2 )
io.show()
```

The images generated as a result of clustering algorithms were recorded in the locale. Byte numbers were calculated by taking these images from the local repository. The parameters of the model within the cluster sum of squares (WCSS), between the cluster sum of squares (BCSS), and explained variance (Silhouette Coefficients) were calculated. The metrics were written without library functions. The squared average distance between all locations within a cluster and the cluster centroid is measured by the within cluster sum of squares (WCSS).

$$\mathrm{WSS} = \sum_{i=1}^{N_C} \sum_{x \in C_i} d(\mathbf{x}, \bar{\mathbf{x}}_{\mathbf{C_i}})^2$$ **[4]**

```
def wcss(image,center):
    dist=np.sum((np.subtract(image,center))**2)
    return dist/len(image)
```

The squared average distance between all centroids is measured by the between clusters sum of squares (BCSS).

$$\mathrm{BSS} = \sum_{i=1}^{N_C} |C_i| \cdot d(\bar{\mathbf{x}}_{\mathbf{C_i}}, \bar{\mathbf{x}})^2$$ **[4]**

```
def bcss(image,center):
    mean_=np.mean(image,axis=0)
    dist=np.sum(np.subtract(mean_,center)**2)
    return dist*len(image)
```

The silhouette coefficient is a statistic for determining how well a clustering process is. Its value is between -1 and 1. The Silhouette Coefficient is defined for each sample and is composed of two scores:

a: The mean distance between a sample and all other points in the same class.

b: The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample is then given as: **[5]**

$$s = \frac{b - a}{max(a, b)}$$

As a result of these definitions, the silhouette coefficient was calculated using the following method using only the numpy library.

```
def silhouette(image,kmeans):
    a=[]
    b=[]
    df=pd.DataFrame(image)
    df['cluster'] = kmeans.labels_
    for i in range(len(image)):

        filtered=df[df.cluster==df.iloc[i].cluster]
        filtered=np.array(filtered.drop("cluster",axis=1))
        dist=np.sum(np.sqrt(np.sum((np.subtract(filtered,image[i])**2), axis=1)))
        a.append(dist)

        #find distance with cluster centers
        x = np.sqrt(np.sum(np.subtract(kmeans.cluster_centers_,image[i])**2, axis=1))
        #find second nearest cluster center
        near = np.where(x==np.amin(np.array(x)[x != np.amin(x)]))[0][0]
        #choose nearest cluster points
        filt=df[df.cluster==near]
        filt=np.array(filt.drop("cluster",axis=1))
        #take distance betweeen nearest cluster points and our data points
        dist=np.sum(np.sqrt(np.sum((np.subtract(filt,image[i])**2), axis=1)))
        b.append(dist)

    a=np.sum(a)
    b=np.sum(b)

    return abs(b-a)/max(a,b)
```

At the end of all these operations, the approximate values of the colors in each compressed image that correspond to the colors in the webcolors library were calculated using the following method. **[6]**

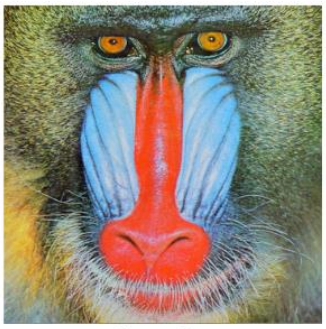
```
from scipy.spatial import KDTree
from webcolors import (
    CSS3_HEX_TO_NAMES,
    hex_to_rgb,
)
def convert_rgb_to_names(rgb_tuple):

    # a dictionary of all the hex and their respective names in css3
    css3_db = CSS3_HEX_TO_NAMES
    names = []
    rgb_values = []
    for color_hex, color_name in css3_db.items():
        names.append(color_name)
        rgb_values.append(hex_to_rgb(color_hex))

    kdt_db = KDTree(rgb_values)
    distance, index = kdt_db.query(rgb_tuple)
    return names[index]

def names_of_colors(kmeans):
    clustroids=kmeans.tolist()
    names=[]
    for i in range(len(clustroids)):
        name=convert_rgb_to_names(clustroids[i])
        print("Closest Match for the name of ", i,"th centroid color : ", name)
        names.append(name)
    return names
```

The sequence came to the evaluation stage after all the clustering algorithms of each image were finished. In the outcome section, all of the gathered metrics were displayed, and the optimum cluster numbers were stated using the elbow method approach.

4-Results and Conclusion

The original photos of each image, as well as the compressed images that are the outputs of the algorithm developed with each cluster, are shown below in order. We watch how the image approaches the original as the number of distinct colors in the cluster of compressed photos rises after the original image. In the following stages, how the difference between the clusters gradually decays is among what we observe in each picture. The values of the metrics belonging to each of the images that are the product of clusters and are also described in the implementation details section, the approximate web color names of the colors in the compressed image, and the byte numbers of all images, including the original image, have been visualized by summing them in a single table. According to this tables, as the number of clusters used to compress images increases, the image sizes increase. It was observed that as the number of clusters increased by two times during this growth process, the size of the images also almost doubled. It was also observed from the table that the within cluster sum of squares decreased greatly with the increase in the number of clusters. Because as the number of clusters increases, the distance between the cluster's decays. In contrast to wcss, we observe that between clusters sum of squares increases as the number of clusters increases. Because as the number of clusters increases again, a large number of cluster distances are added up with mean, which increases the number. It is also observed that they show increasing and decreasing properties at the same rate when they are visualized graphically. Below, all the names of the colors do not appear because they do not fit in the table, but they can be accessed from the inside of the notebook file.

Baboon Image



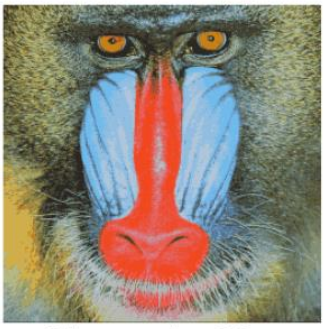
Original Baboon.png

Cluster k=2

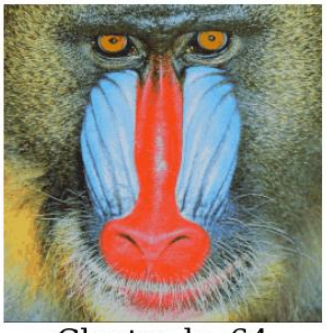
Cluster k=4

Cluster k=8

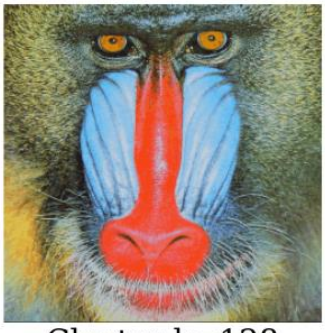
Cluster k=16



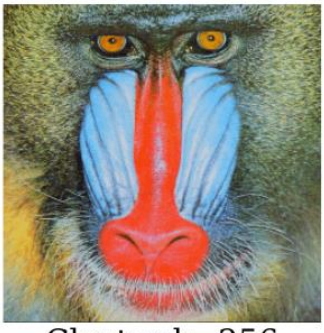
Cluster k=32



Cluster k=64

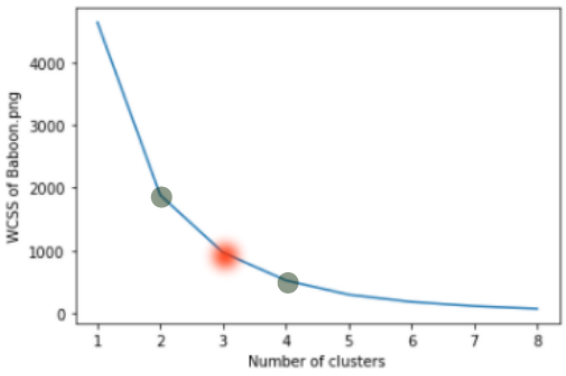


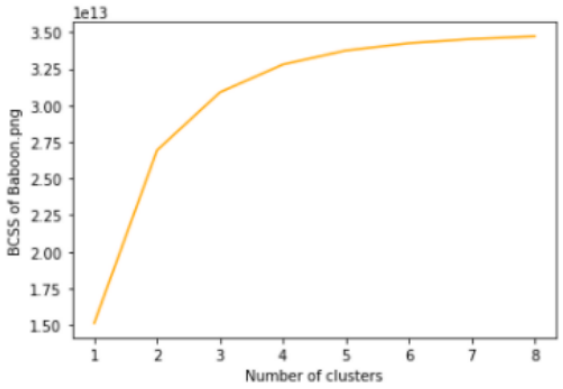
Cluster k=128

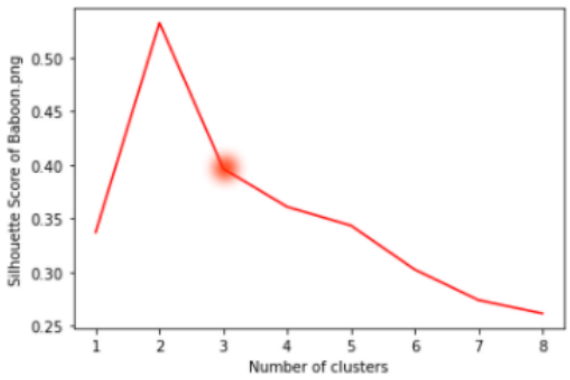


Cluster k=256

	WCSS	BCSS	Silhouette Score	Color names	Size of image
Original	0.000000	0.000000e+00	0.000000	62070	651142
k=2	4633.809636	1.512701e+13	0.337057	[darkgray, dimgray]	9413
k=4	1881.609050	2.693610e+13	0.532157	[darkolivegreen, gray, lightsteelblue, tomato]	15379
k=8	965.779438	3.089075e+13	0.396309	[dimgray, darkgray, tomato, darkslategray, dim...	27866
k=16	522.812888	3.278673e+13	0.360698	[peru, dimgray, skyblue, darkslategray, chocol...	43845
k=32	295.832900	3.374094e+13	0.343288	[dimgray, silver, darkolivegreen, tomato, ligh...	67448
k=64	179.859758	3.423861e+13	0.302227	[dimgray, darkgray, orangered, darkolivegreen,...	96489
k=128	113.446833	3.452672e+13	0.273722	[gray, tomato, lightsteelblue, darkolivegreen,...	122244
k=256	71.928915	3.470520e+13	0.261324	[lightsteelblue, darkolivegreen, indianred, li...	139286







	Size of image
Original	651142
k=2	9413
k=4	15379
k=8	27866
k=16	43845
k=32	67448
k=64	96489
k=128	122244
k=256	139286

In the light of the above data for the Baboon image, it was decided that the optimum number of clusters for compression is in accordance with the definition of the elbow method at the elbow point where the difference begins to decrease according to the wcss graph. Although it is unstable due to the fact that there are many elbow points on the graph, it was considered that k=8 is more suitable when looking at the number of bytes. In addition, the silhouette score was also higher, which made it easier to make this decision. But if the choice is made by eye, it dec actually observed that there are obvious differences between the original image and the compressed 8-cluster image. At this point, perhaps the 32-cluster compressed image, which is the point where the difference decreases completely, should be selected because of its similarity to the original.

Flowers Image

Original Flowers.png

Cluster k=2

Cluster k=4

Cluster k=8

Cluster k=16

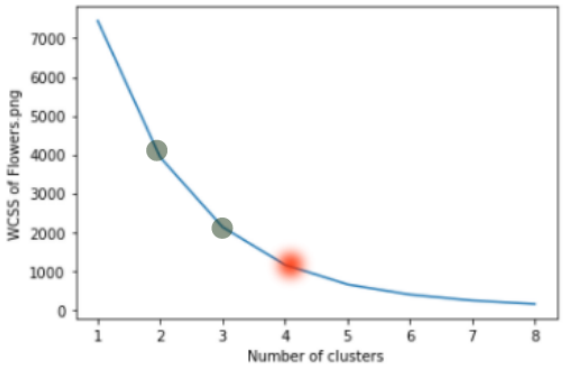
Cluster k=32

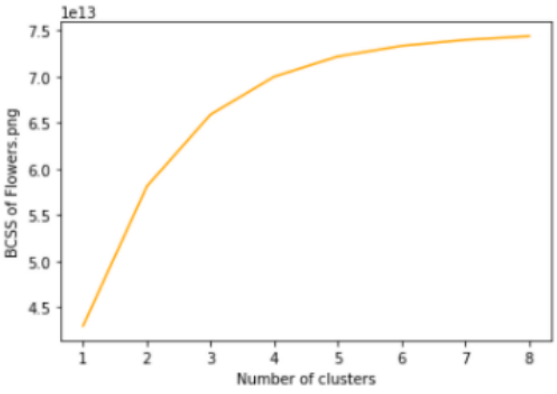
Cluster k=64

Cluster k=128

Cluster k=256

	WCSS	BCSS	Silhouette Score	Colors name	Image size
Original	0.000000	0.000000e+00	0.000000	65314	614122
k=2	7451.635013	4.302087e+13	0.503836	[darkgray, darkslategray]	7549
k=4	3930.231952	5.812872e+13	0.304975	[darkslategray, plum, sienna, slateblue]	14313
k=8	2128.080503	6.591269e+13	0.394997	[lightpink, black, crimson, darkslateblue, med...	26176
k=16	1154.458108	7.001941e+13	0.408492	[gray, saddlebrown, mediumpurple, bisque, blac...	43282
k=32	648.299440	7.221300e+13	0.375591	[plum, darkslategray, sienna, darkslateblue, o...	65488
k=64	385.628111	7.335310e+13	0.332471	[lightcoral, indigo, darkslategray, mediumslat...	87716
k=128	233.058008	7.400101e+13	0.328971	[mediumpurple, black, crimson, darkslategray, ...	107674
k=256	142.265633	7.439735e+13	0.312652	[darkolivegreen, plum, black, mediumslateblue,...	123240





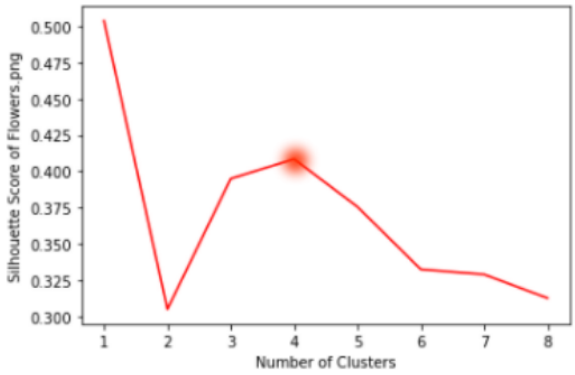


	Image size
Original	614122
k=2	7549
k=4	14313
k=8	26176
k=16	43282
k=32	65488
k=64	87716
k=128	107674
k=256	123240

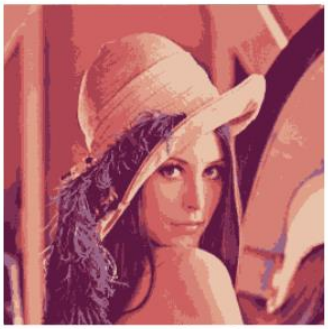
The elbow method was applied as described above. Again, because there are multiple breakpoints, it was decided that the silhouette score was ideal for optimal compression by looking at the 16 cluster, both because it is one of the 14 original sizes, and because it is the best candidate for the silhouette and elbow method.

Lena Image

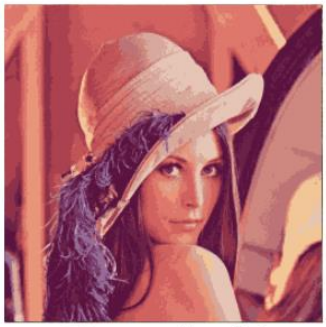
Original Lena.png

Cluster k=2

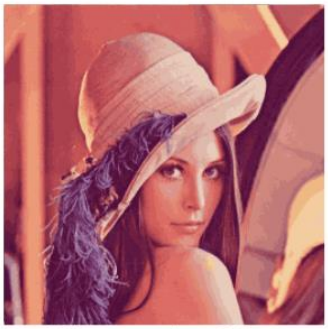
Cluster k=4



Cluster k=8

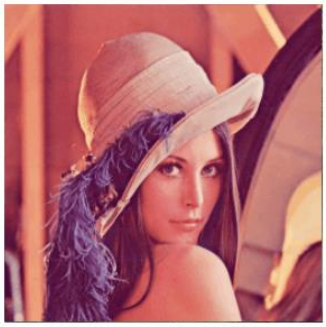


Cluster k=16

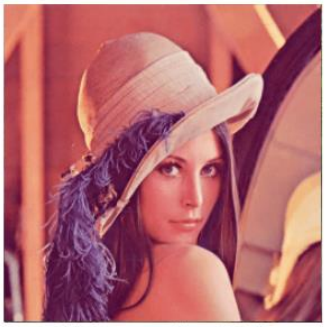


Cluster k=32

Cluster k=64

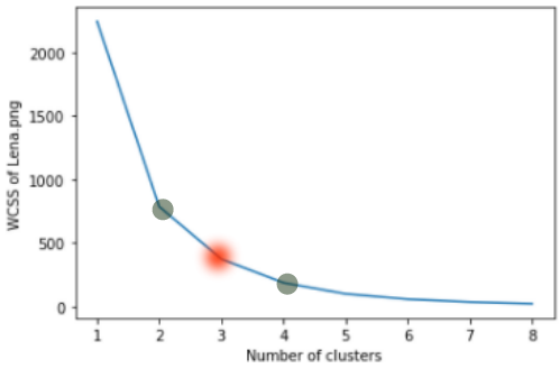


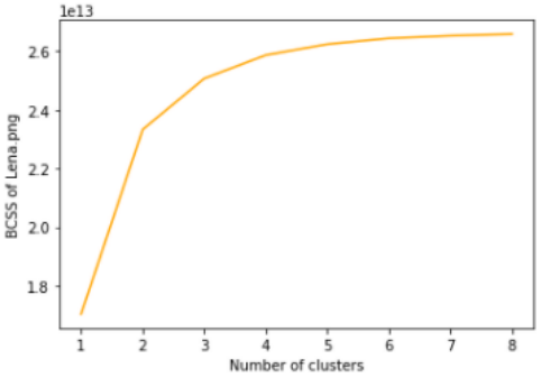
Cluster k=128



Cluster k=256

	WCSS	BCSS	Silhouette Score	Color names	Image sizes
Original	0.000000	0.000000e+00	0.000000	48331	473831
k=2	2242.253322	1.704287e+13	0.532248	[brown, darksalmon]	4693
k=4	786.299819	2.332813e+13	0.525161	[lightcoral, brown, indianred, burlywood]	10562
k=8	374.243682	2.506611e+13	0.417960	[gray, indigo, darksalmon, sienna, lightcoral,...	18827
k=16	187.623744	2.584178e+13	0.389946	[indianred, darksalmon, indigo, rosybrown, tan...	28304
k=32	100.395810	2.623607e+13	0.337459	[indianred, brown, tan, sienna, darksalmon, di...	44152
k=64	58.554860	2.643721e+13	0.306820	[rosybrown, brown, indianred, tan, indigo, sie...	62884
k=128	36.351518	2.652691e+13	0.285268	[indianred, brown, silver, lightcoral, sienna,...	82721
k=256	22.843678	2.658062e+13	0.270678	[lightcoral, purple, sienna, silver, dimgray, ...	97763





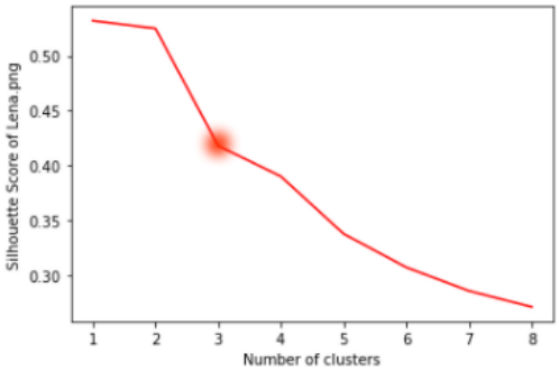


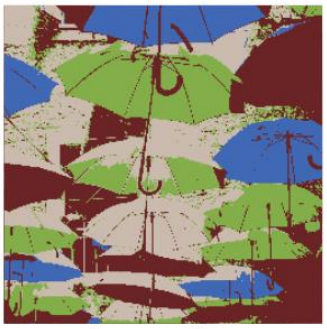
	Image sizes
Original	473831
k=2	4693
k=4	10562
k=8	18827
k=16	28304
k=32	44152
k=64	62884
k=128	82721
k=256	97763

The process of choosing the optimal number of compression clusters for other images proceeded in the same way within the Lena image. Again, as in other pictures, the silhouette score chart was used due to the fact that there are many elbow points. Although it was theoretically decided that 8 clusters were optimal, it was again observed that the 32 clusters with the most recent breakpoint were much more similar when viewed with the eye.

Umbrella Image

Original Lena.png

Cluster k=2



Cluster k=4

Cluster k=8

Cluster k=16

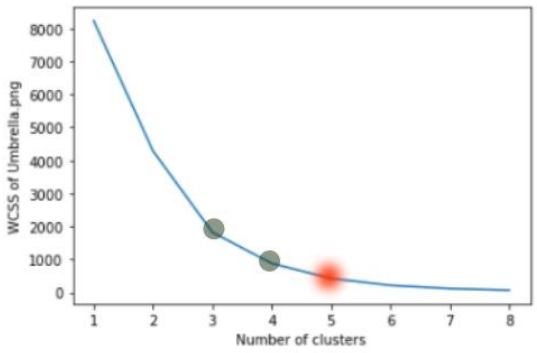
Cluster k=32

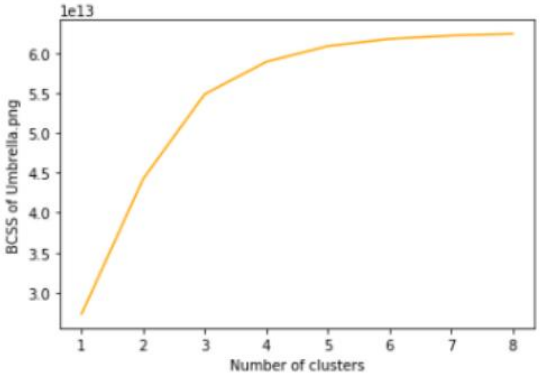
Cluster k=64

Cluster k=128

Cluster k=256

	WCSS	BCSS	Silhouette Score	Color Names of Cluster Centroid	Image Size
Original	0.000000	0.000000e+00	0.000000	49461	1658471
k=2	8230.981471	2.732877e+13	0.402206	[darkseagreen, darkolivegreen]	6513
k=4	4284.673016	4.425305e+13	0.422617	[saddlebrown, silver, steelblue, yellowgreen]	11024
k=8	1822.983673	5.485500e+13	0.394316	[silver, olivedrab, rosybrown, firebrick, stee...	17295
k=16	879.789051	5.890227e+13	0.404556	[olivedrab, darkseagreen, darkslateblue, darkg...	25101
k=32	423.899925	6.087038e+13	0.434897	[crimson, darkgray, darkslategray, steelblue, ...	35917
k=64	214.863384	6.175744e+13	0.426969	[darkseagreen, brown, steelblue, yellowgreen, ...	52677
k=128	118.246444	6.218094e+13	0.384310	[lightgray, gray, darkslategray, slateblue, re...	67392
k=256	68.702807	6.239488e+13	0.351627	[yellowgreen, darkslateblue, darkslategray, ro...	83572





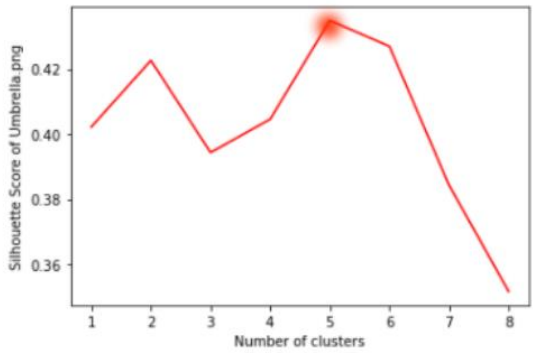


	Image Size
Original	1658471
k=2	6513
k=4	11024
k=8	17295
k=16	25101
k=32	35917
k=64	52677
k=128	67392
k=256	83572

When the same procedure was applied for the Umbrella image, the output of the algorithm with a much smaller number of clusters and a little more number of clusters was selected instead of selecting small-sized images in other images when the same procedure was applied. Because although there are still too many elbows in the elbow method, the silhouette score was so high that it affected this situation. In addition, when you look at the picture with your eyes, it's not so easy to decipher the difference between the original picture and them.

Grafitti Image

Original Grafitti.png

Cluster k=2

Cluster k=4

Cluster k=8

Cluster k=16

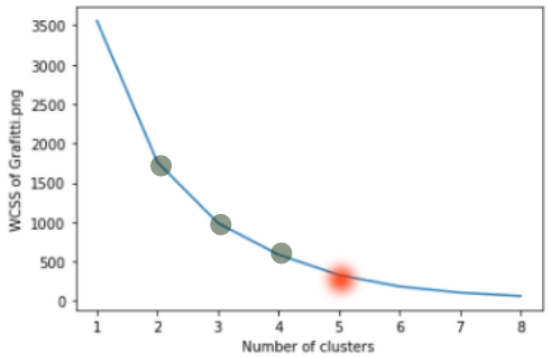
Cluster k=32

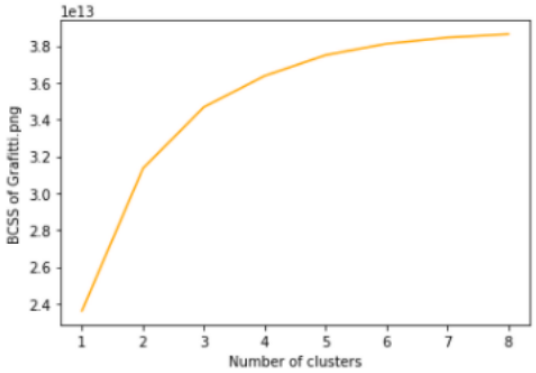
Cluster k=64

Cluster k=128

Cluster k=256

	WCSS	BCSS	Silhouette Score	Color Names of Cluster Centroid	Image Size
Original	0.000000	0.000000e+00	0.000000	47091	6567011
k=2	3557.239505	2.361834e+13	0.531942	[darkslategray, rosybrown]	7913
k=4	1761.895149	3.134328e+13	0.472208	[darkslategray, indianred, sandybrown, silver]	14892
k=8	987.448386	3.467311e+13	0.381012	[gray, indianred, darkslategray, silver, sienn...	28054
k=16	586.268168	3.637239e+13	0.341989	[dimgray, chocolate, darkgray, darkslategray, ...	43417
k=32	326.730695	3.749515e+13	0.370343	[indianred, darkslategray, silver, dimgray, ch...	60542
k=64	181.470316	3.811322e+13	0.376263	[indianred, darkslategray, sandybrown, silver,...	78625
k=128	102.804607	3.845101e+13	0.360762	[indianred, darkslategray, powderblue, sandybr...	96530
k=256	60.635510	3.863392e+13	0.331646	[darkslategray, darksalmon, dimgray, coral, in...	114976





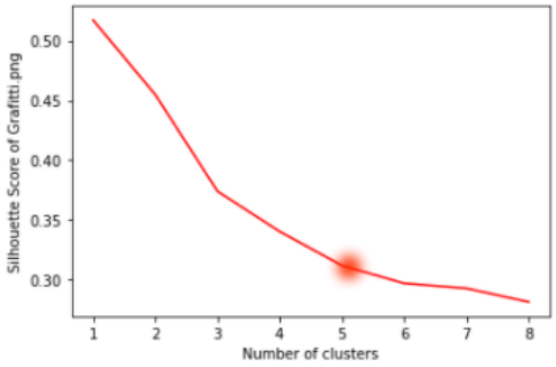


	Image Size
Original	6567011
k=2	7913
k=4	14892
k=8	28054
k=16	43417
k=32	60542
k=64	78625
k=128	96530
k=256	114976

The Graffitti image was one of the images converted to png. For this reason, it actually has a fairly high resolution and size. There are many breaking points in the wcss chart. At this point, when we add the size factor into the work, the 32 cluster also provides both the best quality and the least memory consumption. In addition, the size of the selected image is also 10 times smaller than the size of the actual image.

In our research, we converted 5 different images to png format and transferred them to our notebook. Then, with reshape and resize methods, we brought our images to workable and appropriate sizes. In the process of this process, we used the k-means algorithm. As a result, we chose the best quality, least space-consuming compressed image using the elbow and silhouette method, as well as taking into account the size of the images. While performing these operations, I realized how important image compression is and that it can make a big difference. It was concluded that specifying the colors of the images with their names could be very useful, especially in industrial conditions.

As a result, image compression is important and necessary to reduce the irrelevance and redundancy of image data, to minimize the number of bits required to represent an image, in order to be able to store or transmit data efficiently. For this process, it is an efficient method to use the k-means algorithm.

References

[1] https://medium.com/@agarwalvibhor84/image-compression-using-k-means-clustering-8c0ec055103f

[2] https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

[3] https://odsc.medium.com/unsupervised-learning-evaluating-clusters-bd47eed175ce

[4] https://www.kdnuggets.com/2019/05/golden-goose-cohort-analysis.html/2

[5] https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient

[6] https://medium.com/codex/rgb-to-color-names-in-python-the-robust-way-ec4a9d97a01f