CS 452 Assignment 2

Image Compression by K-Means Clustering

FALL 2020

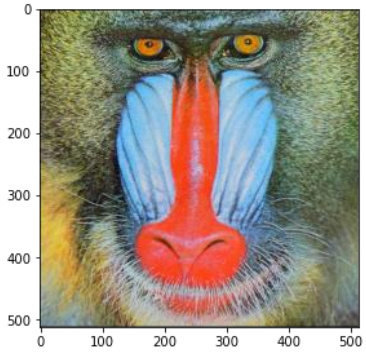
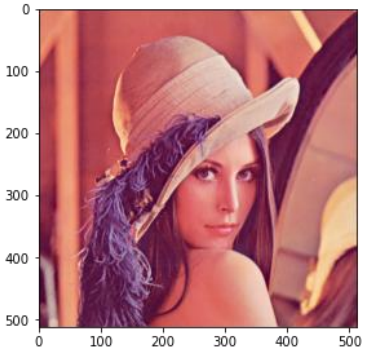
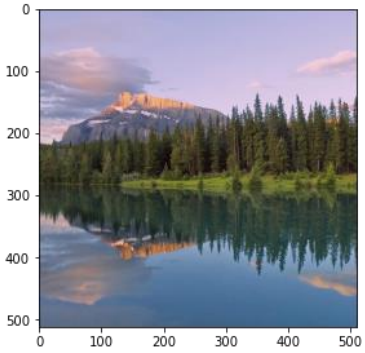
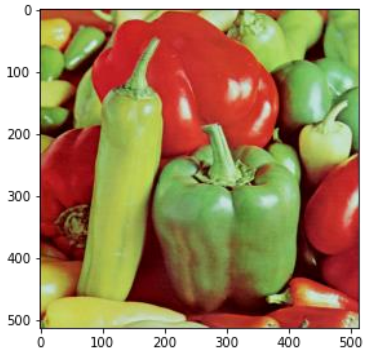
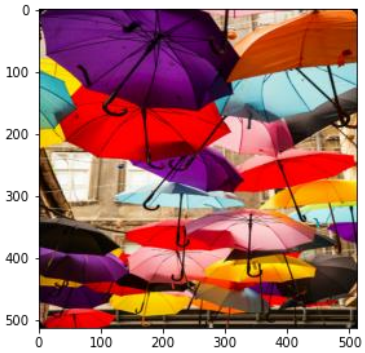
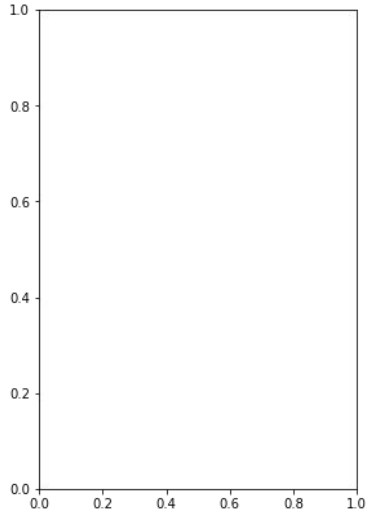
Report

S015674

Mustafa Alper Sayan

1. Introduction

In this assignment we are excepted to compress images using k-means algorithm. We are given 5 input images as shown below

We are excepted to do preprocessing on the images meaning reshaping, flattening images. After this point we are going to use k-means algorithm to compress the images to 2^k where k= {2, 4, 8, 16, 32, 64, 128, 256} cluster sizes. These cluster points are going to represent the unique colors ex. For 2 cluster centers we are going have 2 unique colors and assign the rest of the colors to the closest center. Then we will plot each compressed image, give information about their sizes in bytes, calculate the WCSS, BCSS, TSS, explained variance of each object from scratch. Then we are going to write our own elbow calculation algorithm from scratch. And visualize all the relevant information we gathered and put the relevant information on a Pandas data-frame object and show it in the report.

2. Methodology

K-Means Clustering

K-means clustering is a method of vector quantization. It aims to partition n samples into k clusters in which each sample belongs to the nearest mean.

Given a set of samples from $\langle X_1, X_2, \dots, X_n \rangle$ where each sample has d dimensions, partition n samples into k sets $S = \{S_1, S_2, \dots, S_k\}$ so we can minimize within-cluster sum of squares (WCSS). The equation is as follows:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} \quad \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Where μ_i is the mean point in S_i. This is same as minimizing the pairwise squared derivations of the points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

The algorithm computation is done in two steps assignment step and update step. Given an initial set of k number of means $m_1^1, m_2^1, \dots, m_k^1$

Assignment step:

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\}$$

Where each x_p is assigned to exactly one S^t

Update step: Recalculate means for observations assigned to each cluster.

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

The algorithm converged when the steps no longer change. The algorithm can be stuck in a local minimum.

Naïve Initialization Method: Consists of a single step

1. Choose k initial points randomly from domain

K-means++ Initialization Method:

1. Choose one center at random among the sample points
2. For each sample point not chosen yet, compute Distance(x), the distance between x and the nearest center chosen
3. Choose a new random center point, using a weighted probability distribution
4. Repeat 2, 3 until k centers has been chosen
5. Proceed with the naïve k-means steps.

K-means Evaluation Metrics:

WCSS (Within Cluster sum of errors): This method is used for understanding when to stop the k-means algorithm meaning we keep running the algorithm until no significant improvement can be made on WCSS. WCSS is defined formally as:

$$WCSS = \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2.$$

Where S is the set of clusters, with centers μ being each center associated with the relevant cluster. The operation inside sums is defined as Euclidian distance.

TSS (Total Sum of Squares): If there are m points numbered x_1, x_2, x_i then

$$M = \frac{1}{n} \sum_{i=1}^{k} x_i$$

Where M is the mean of each dimension. Then TSS is defined as:

$$TSS = \sum_{i=1}^{n} ||x - M||^2$$

Explained Variance (Silhouette Coefficients): This metric is used to evaluate the performance of the model. In normal circumstances the higher score should be better, but when considering the size as evaluation metric there is a tradeoff in the following sections I will explain this in more detail. Explained variance is found as follows:

$$Explained\ Variance = \frac{BCSS}{TSS}$$

Elbow Point

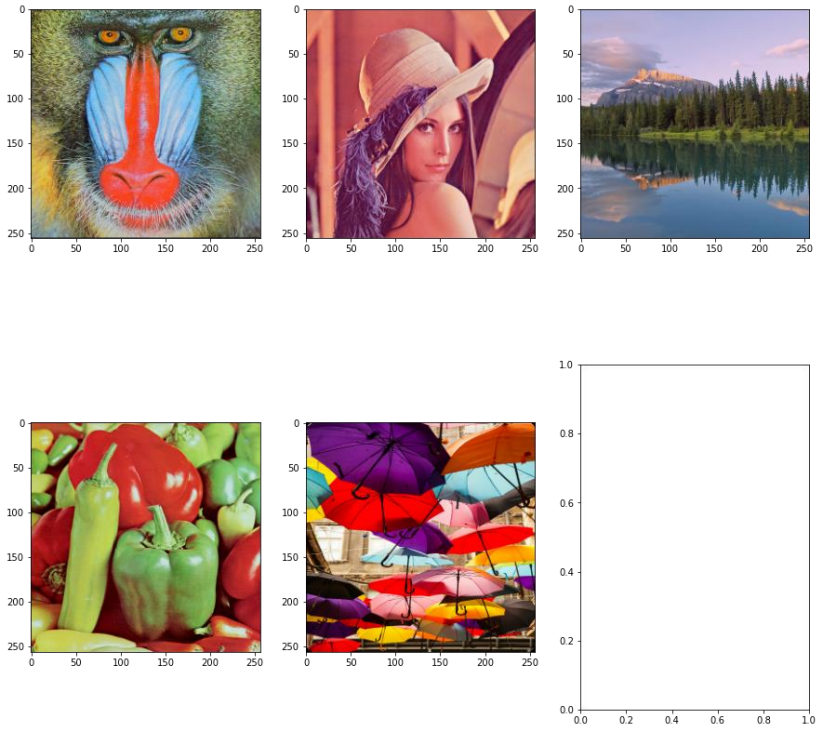
In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a dataset. In our assignment which is image quantization this can be interpreted as trade-off between explained variance and size of the outputted image. So we need a function to determine this tradeoff with parameters being explained variance and size of the compressed image. To achieve this goal, we can use explained variance to measure the distance of each explained variance value on a 2d plane. Assume we have n number of cluster points ranging from $k = \{k_1, \dots, k_n\}$. Assume there are n number of explained variances from $y = \{y_1, y_n\}$. The steps are as follows:

1. Calculate point initial as $init = (k_1, y_1)$
2. Calculate point ending point as $end = (k_n, y_n)$
3. Assume there is a line l between init and end point
4. For each (k_i, y_i) $i \in n$ calculate the Euclidian distance between line l
5. The maximum distance is the elbow point
6. Return the elbow point

3. Implementation Details

Preprocessing Data

1. Reshaping the input images to shape (256, 256, 3) from (512, 512, 3). Outputted images can be seen below:



2. Flattening input images to 2d * D (RGB) arrays from (256, 256, 3) meaning obtaining (65536, 3)
3. Normally at this step for different scaled features we should use the standard scaler, but since every pixel value is between (0, 255) we do not need to scaled our data.

Image Quantization

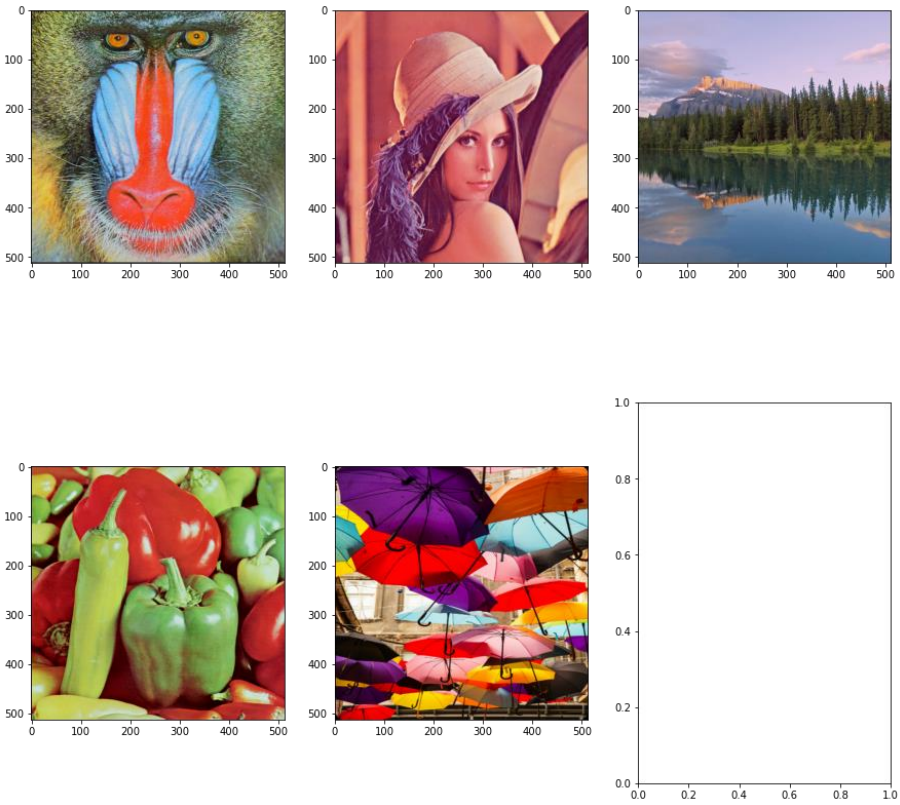
1. Initializing the k-means algorithm for each category and assigned number of clusters. Algorithm parameters are n_clusters, which determines the number of clusters. Init is which kmeans initialization method should be used, in my case I used k-means++ which I explained in the previous sections. Random_state, is the seed used for the reproducibility of the code.
2. Fit each category to their relevant categories and then append them to a list for future calculations
3. Assign each label of the k-means object to the relevant cluster center
4. Reshape pixels from (65536,3) to (256, 256, 3)

Metric Calculations

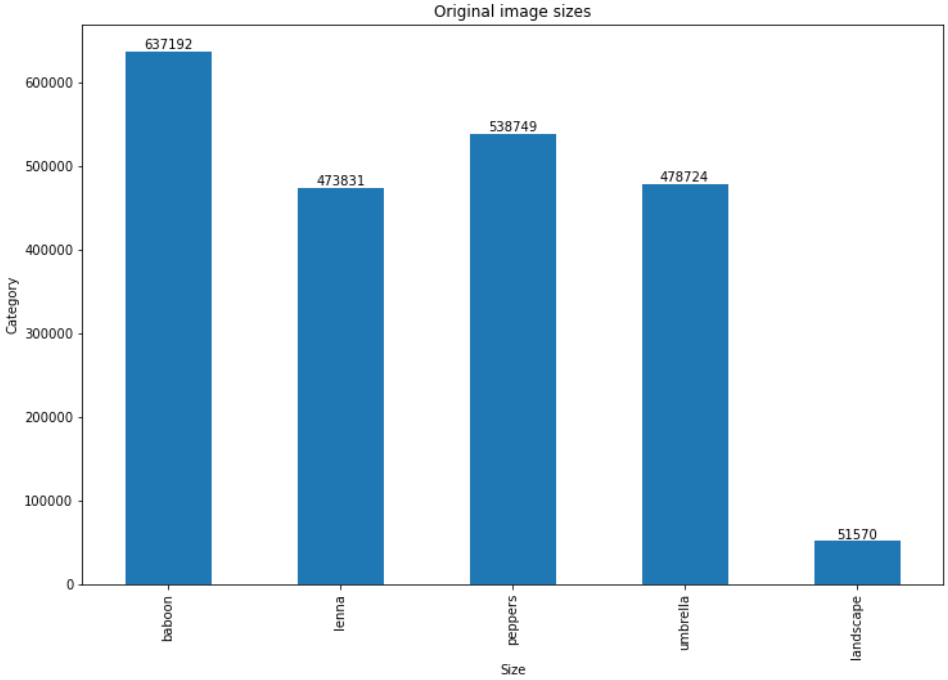
1. Calculate WCSS
    a. For every (R, G, B) value in the original array calculate the Euclidian distance between all the cluster centers
    b. Get the minimum distance which the input value is equal to. This is the error
    c. Sum all the errors and return the error
2. Calculate TSS
    a. Calculate the mean of the each (R, G, B) channel
    b. Find the Euclidian distance between mean of each channel with respect to original image R, G, B channels
    c. Sum the differences
    d. Return differences, this is TSS
3. Calculate BCSS
    a. Return the difference between WCSS and TSS
4. Calculate Explained variance
    a. Return BCSS divided by TSS

4. Results

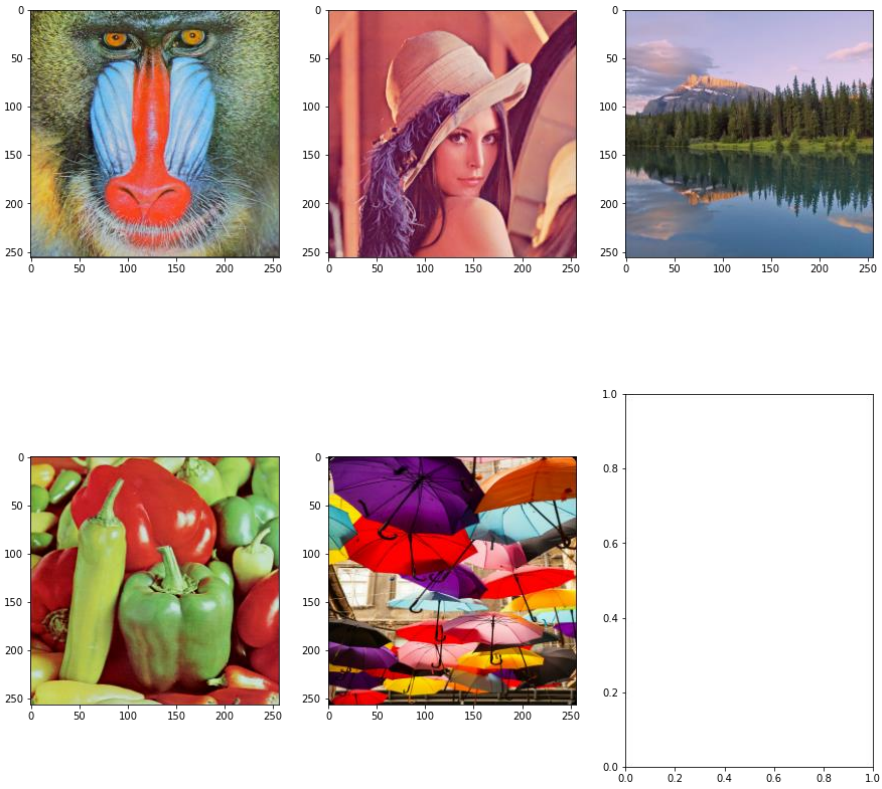
i. Input images



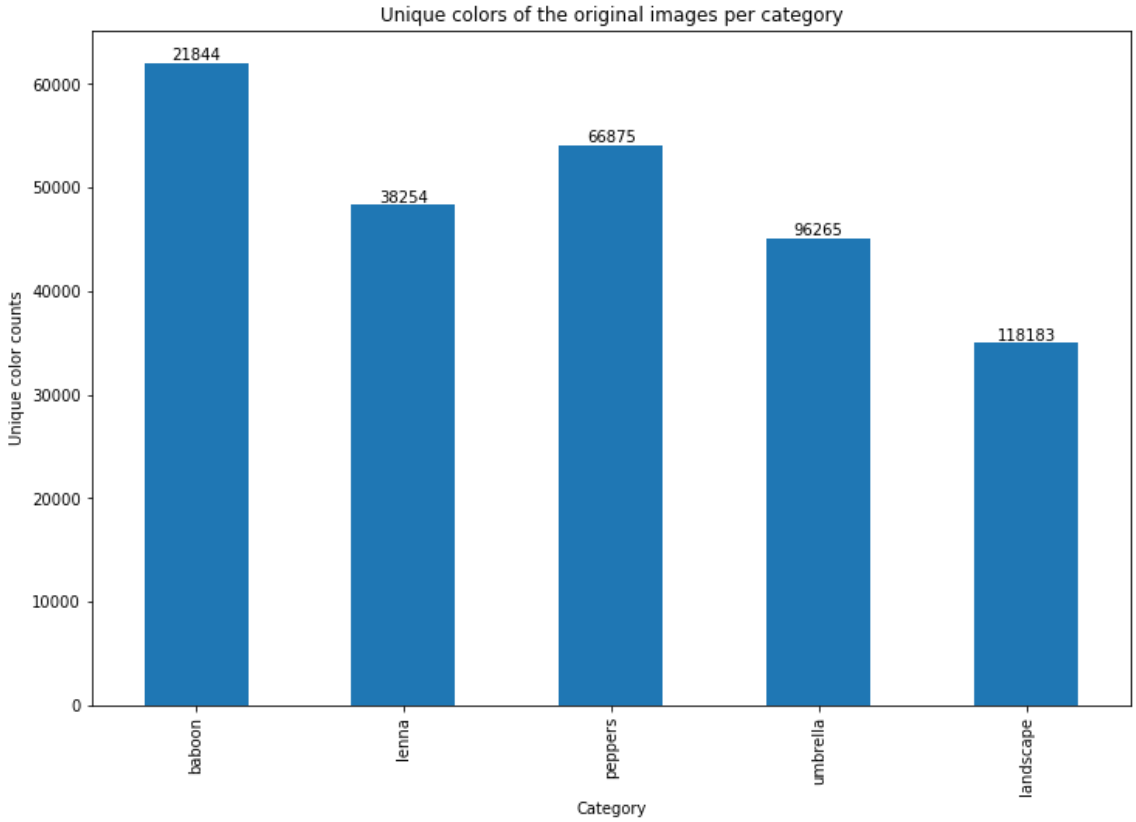
ii. Input image sizes:



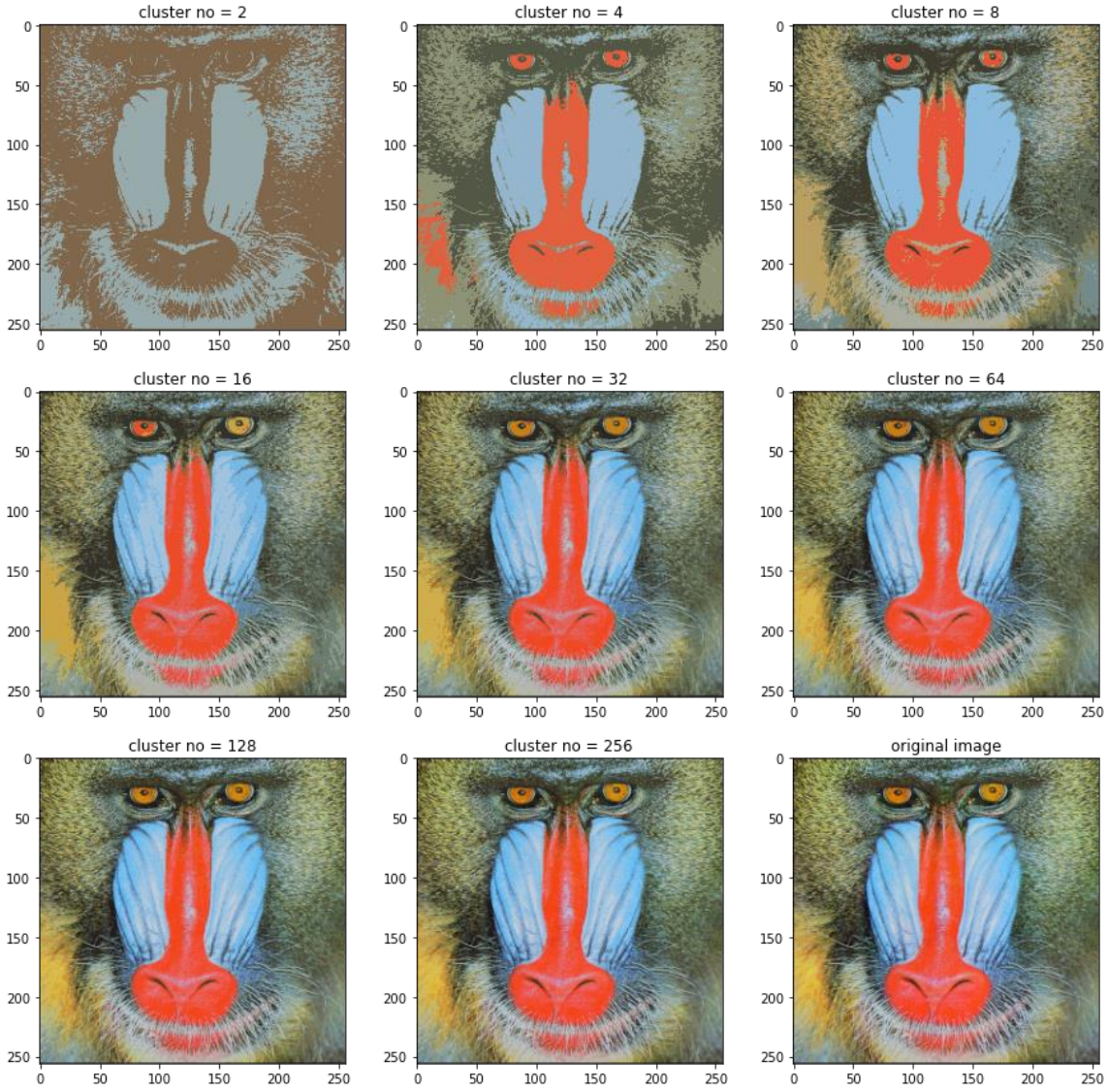
iii. Resized images:



iv. Unique colors original images:

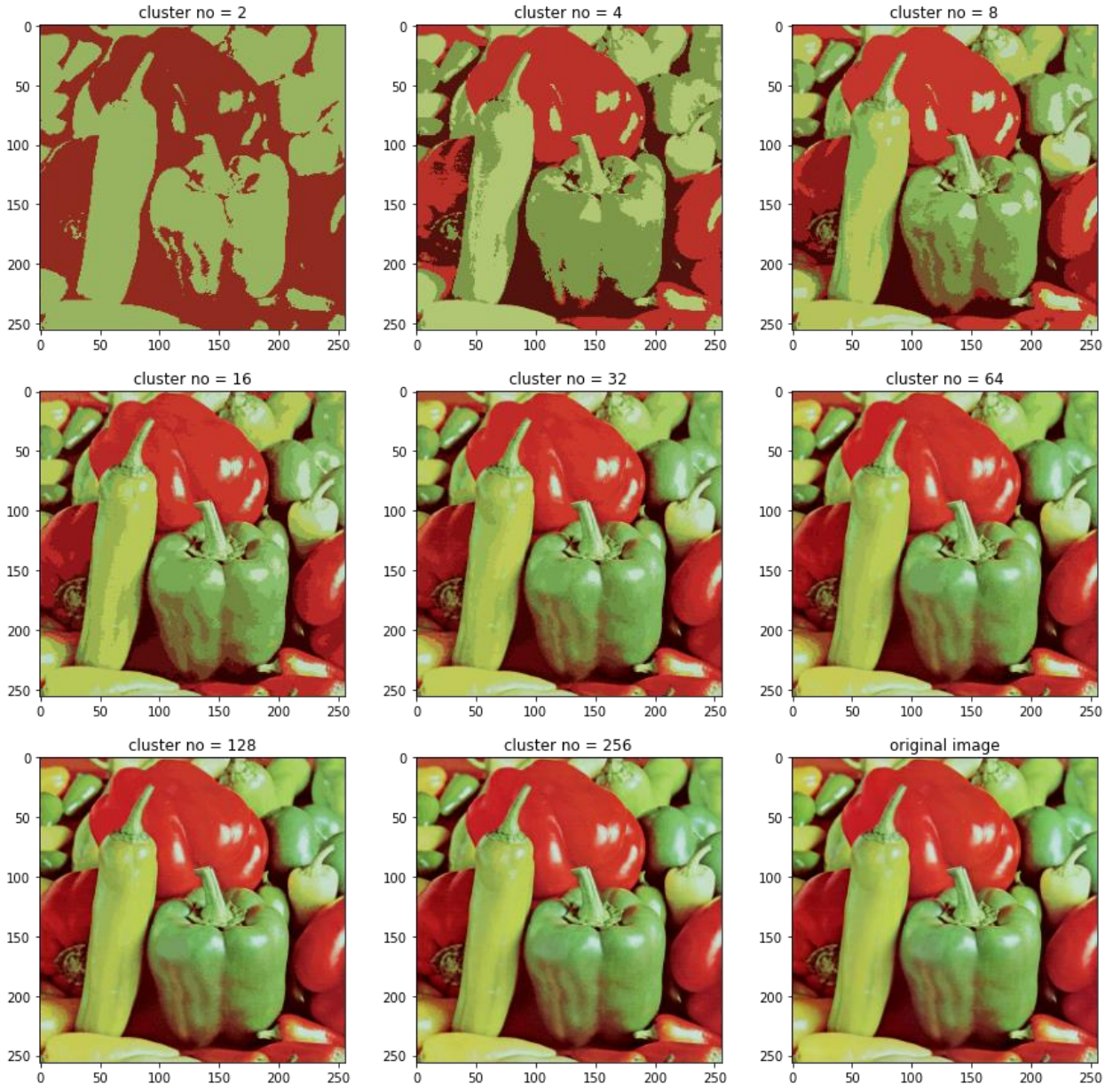


v. Compressed images of the baboon, last image is the original image. After cluster number 8, image does not gain much detail

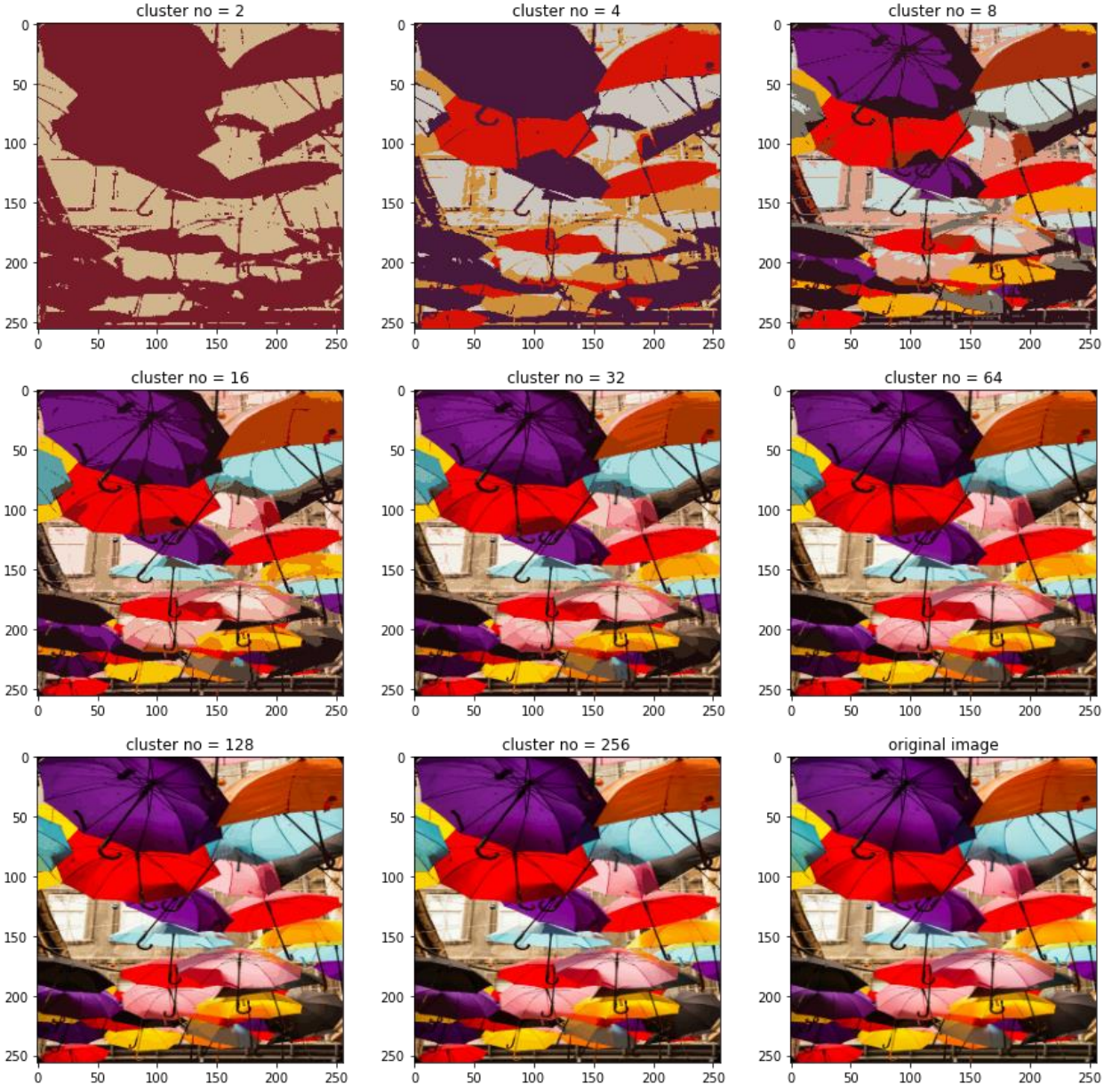


vi. Compressed images of the lenna, last image is the original image. After cluster number 8, image does not gain much detail

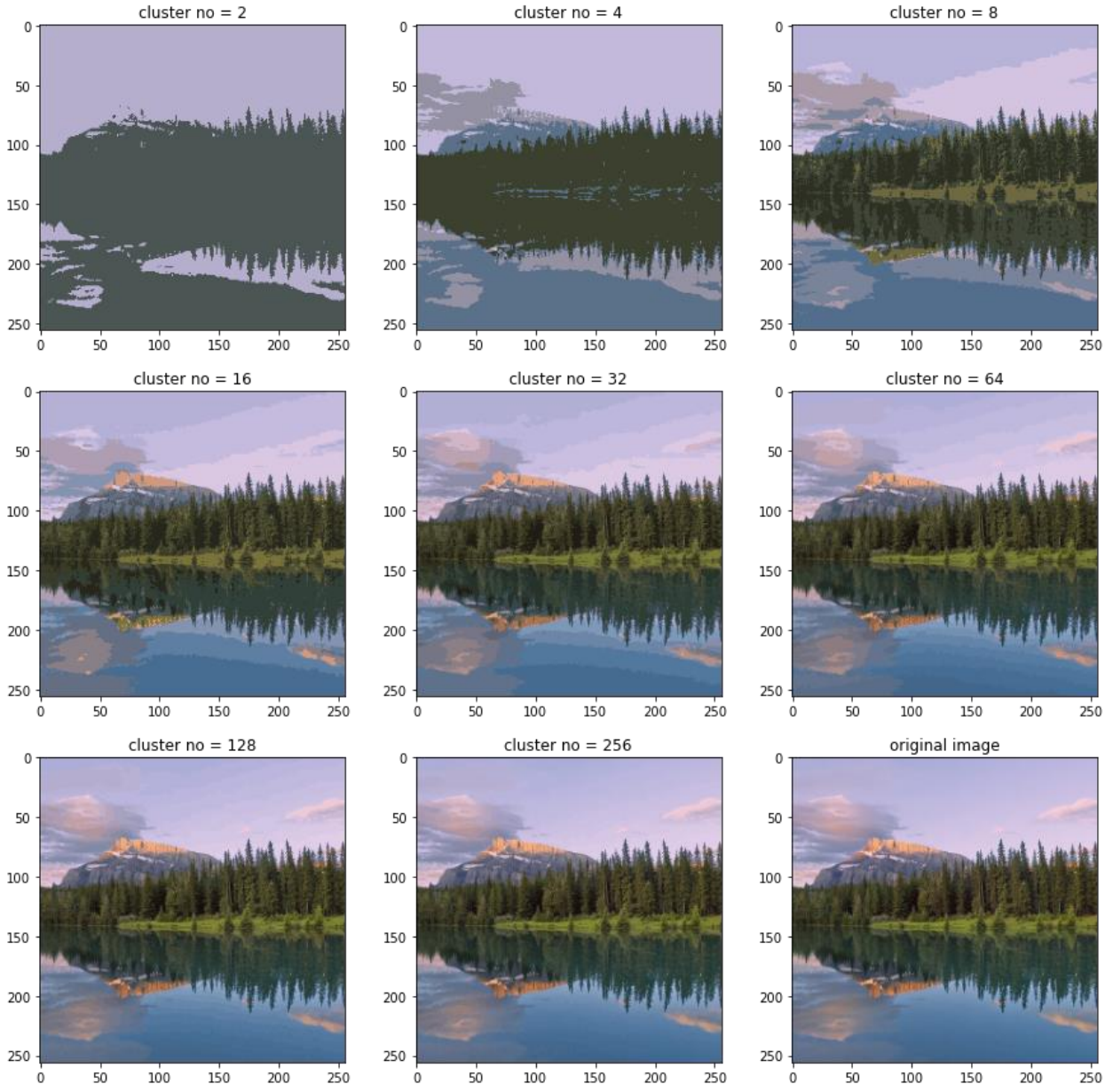
vii. Compressed images of the peppers, last images are the original image. After cluster number 8, image does not gain much detail



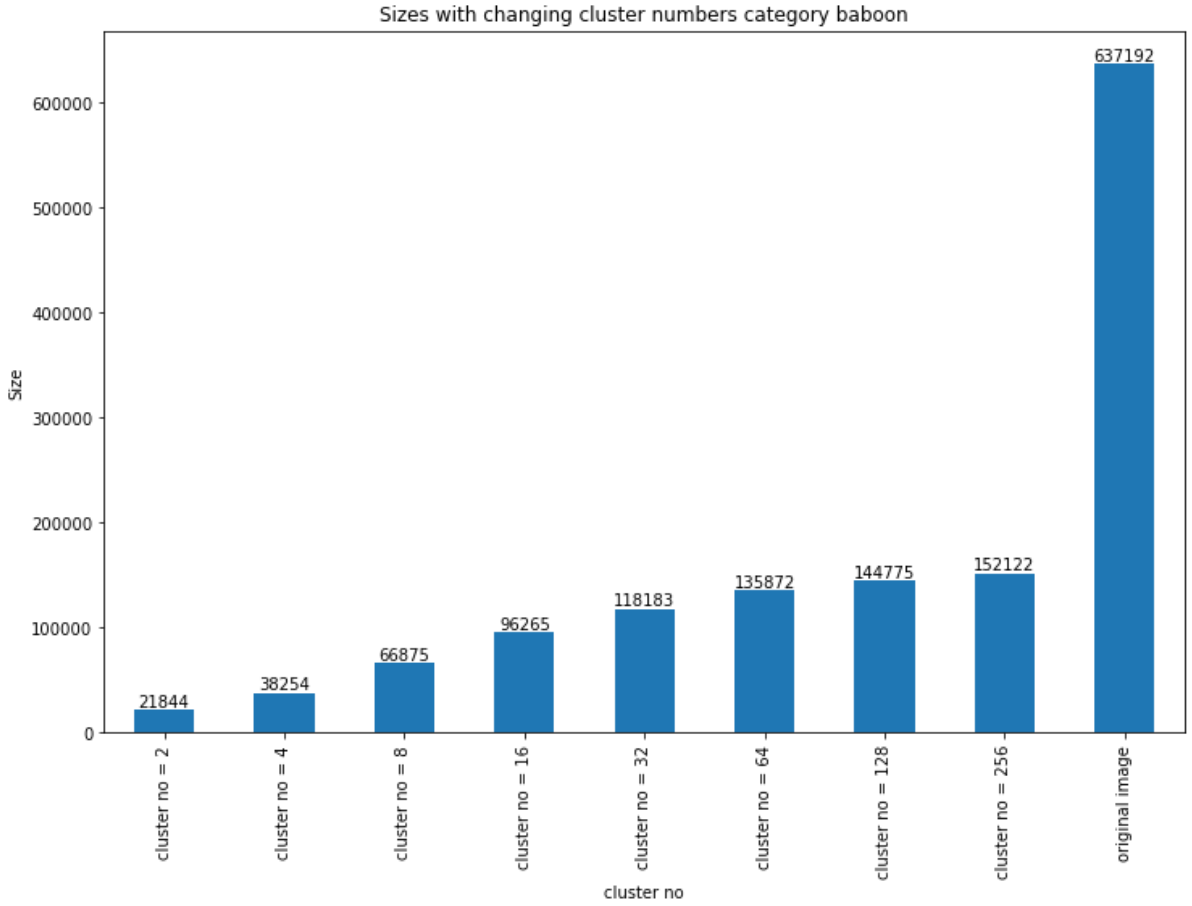
viii. Compressed images of the umbrellas, last image is the original image. After cluster number 8, image does not gain much detail



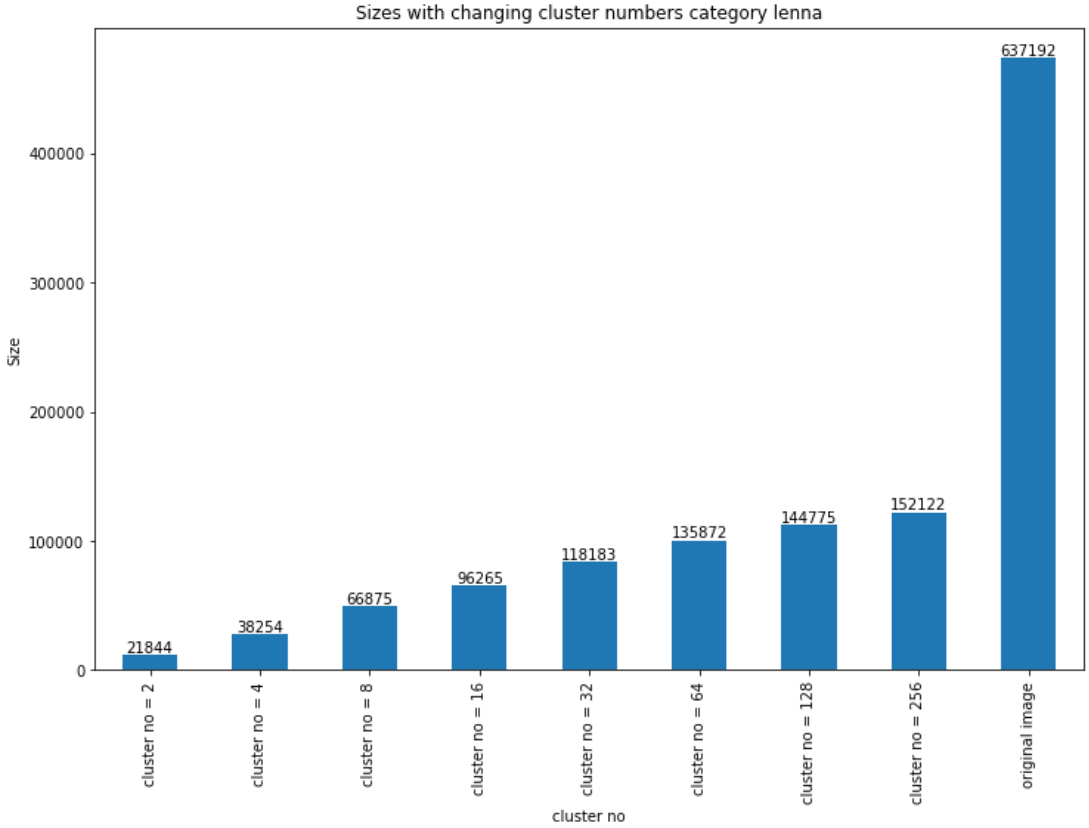
ix. Compressed images of the landscape, last image is the original image. After cluster number 8, image does not gain much detail



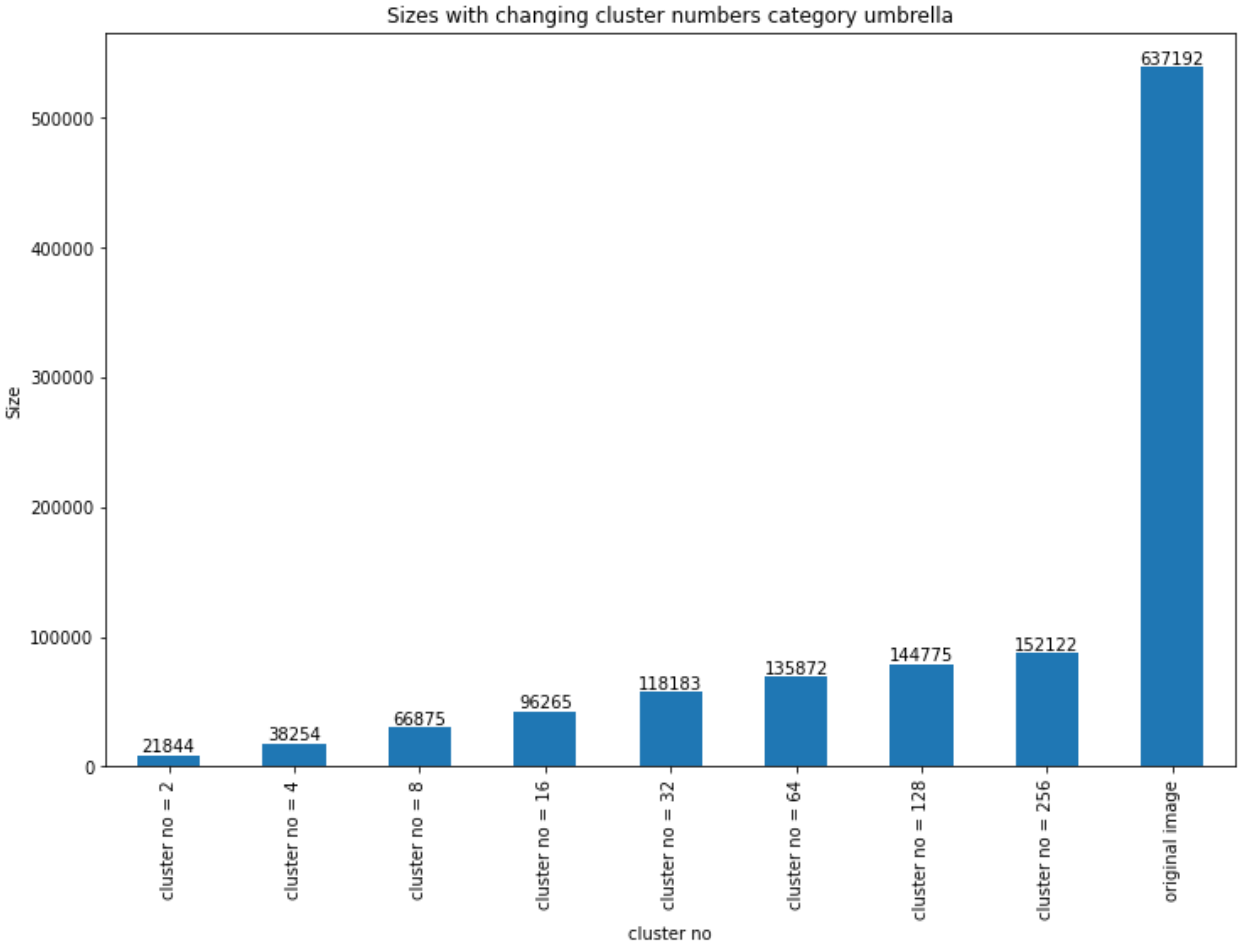
x. Sizes with changing cluster numbers category baboon. By compressing the image, even with 256 clusters we gained much in size



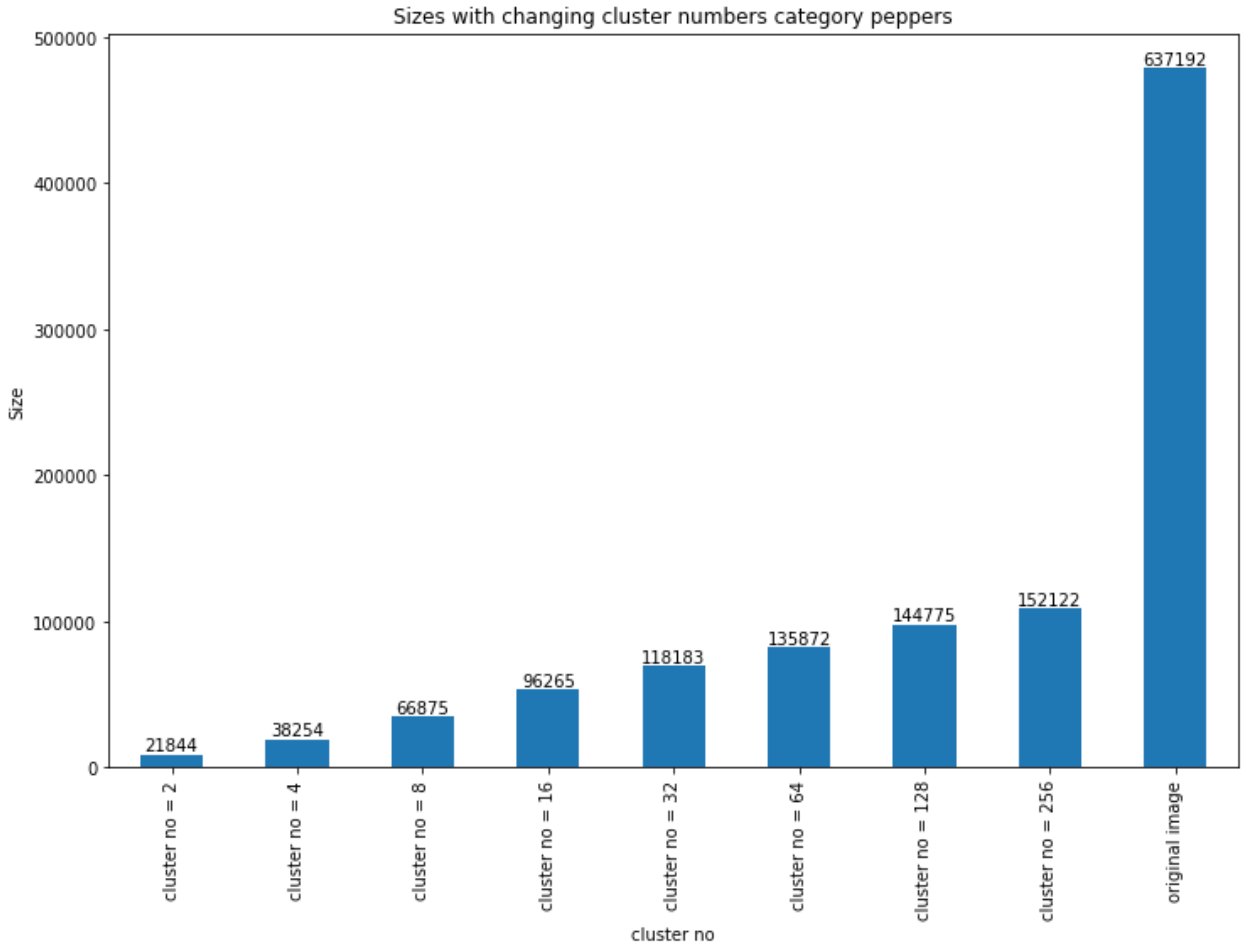
xi. Sizes with changing cluster numbers category lenna. By compressing the image, even with 256 clusters we gained much in size



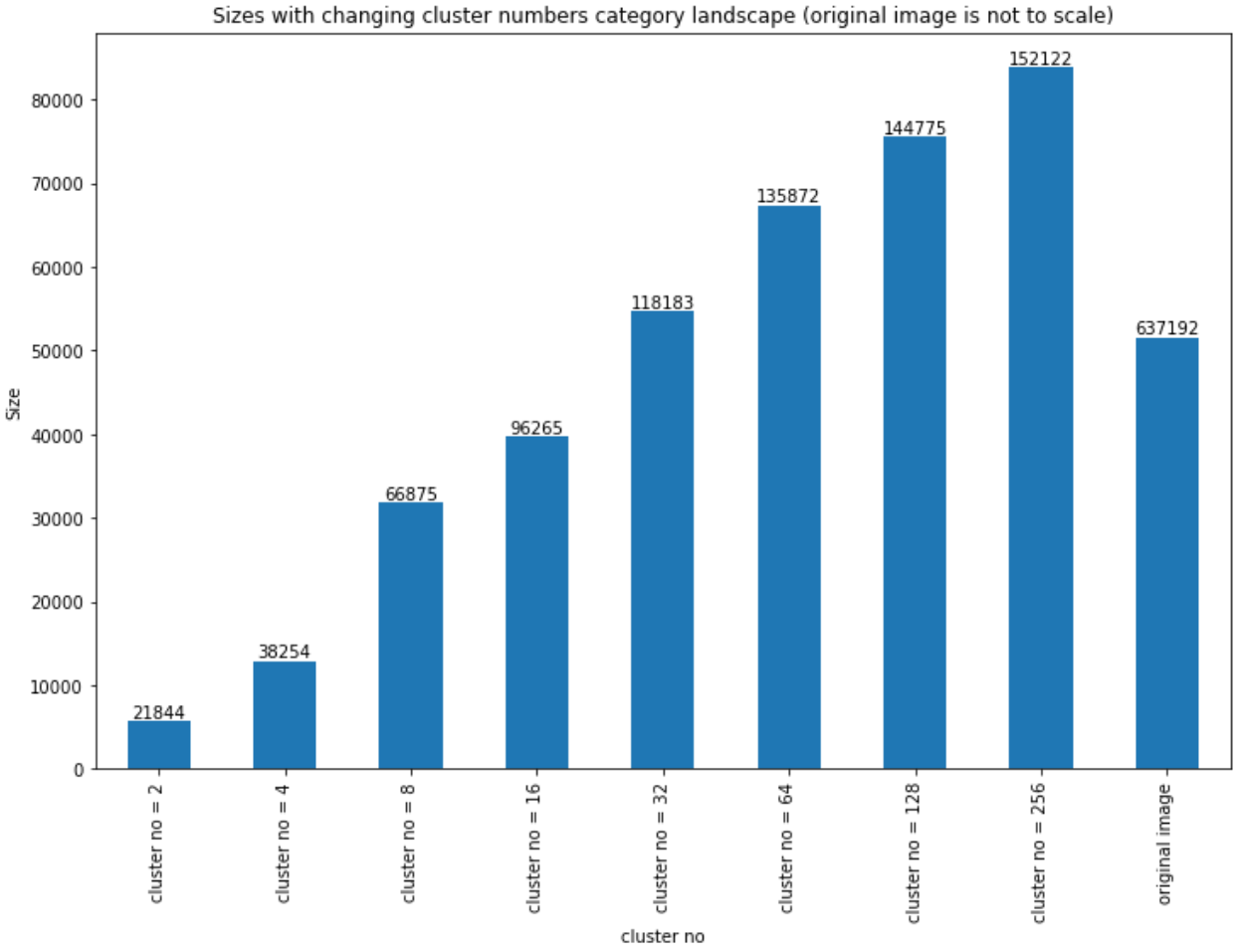
xii. Sizes with changing cluster numbers category umbrella. By compressing the image, even with 256 clusters we gained much in size



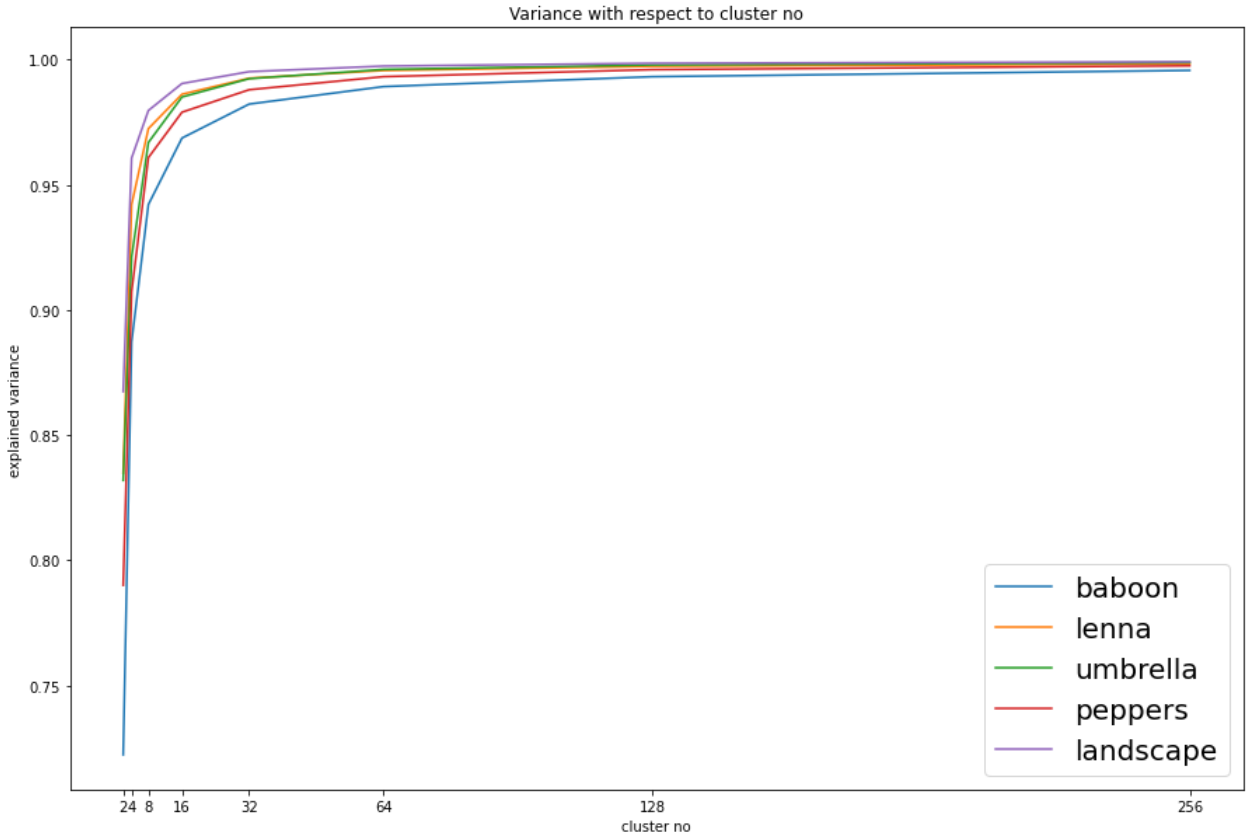
xiii. Sizes with changing cluster numbers category peppers. By compressing the image, even with 256 clusters we gained much in size



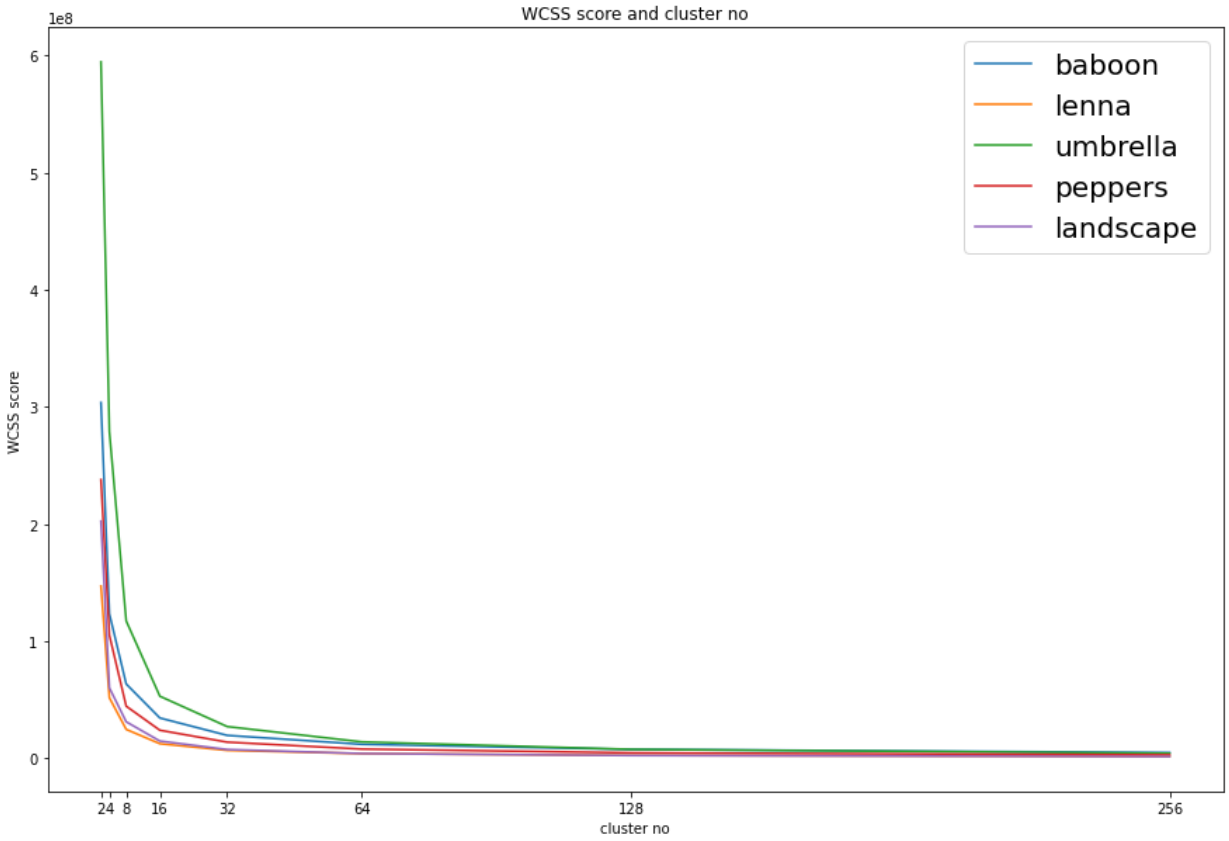
xiv. Sizes with changing cluster numbers category landscape. By compressing the image, even with 256 clusters we gained much in size. Because the scale is of some much the plot is not working as intended



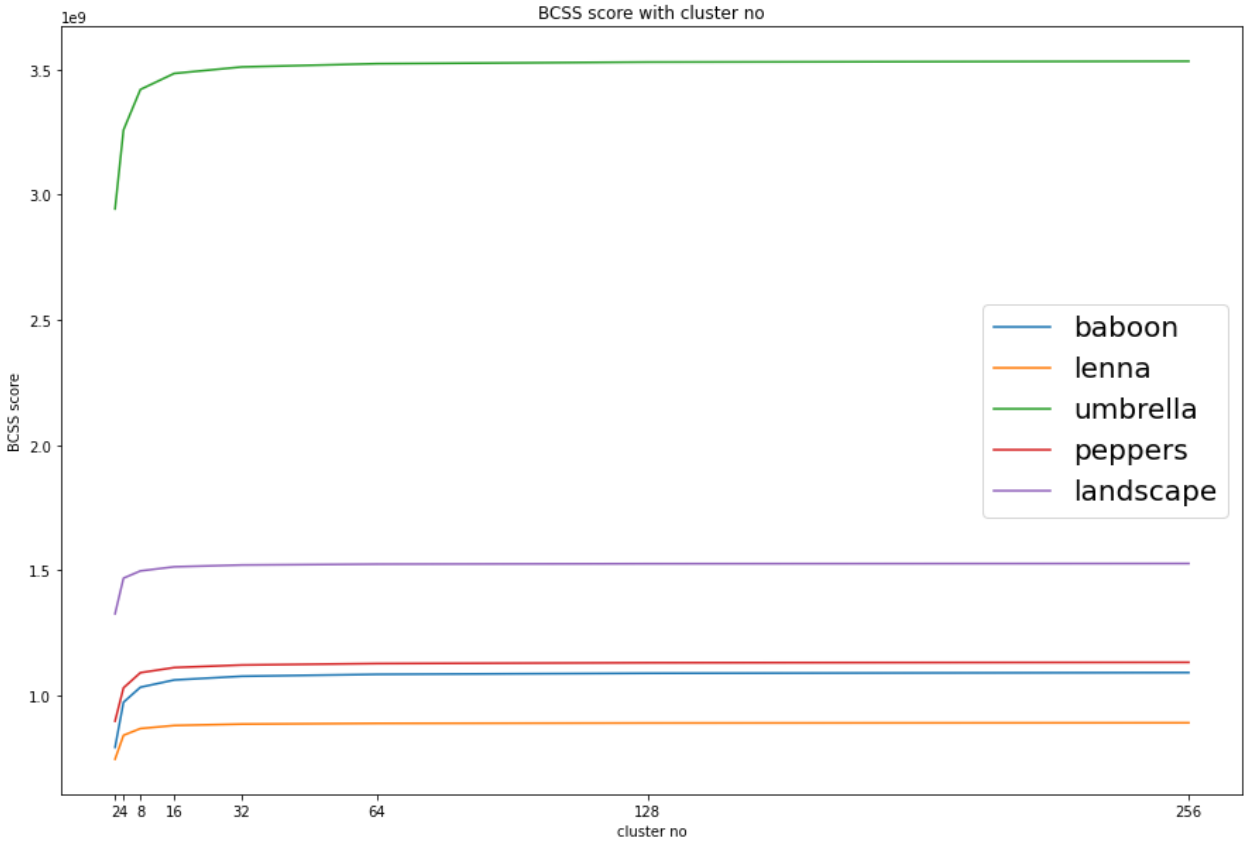
xv. Variance with respect to cluster no. It seems by looking at this plot the optimal cluster size i.e. elbow point is 8 for all the categories



xvi. WCSS with respect to cluster no. Elbow point is again 8 clusters



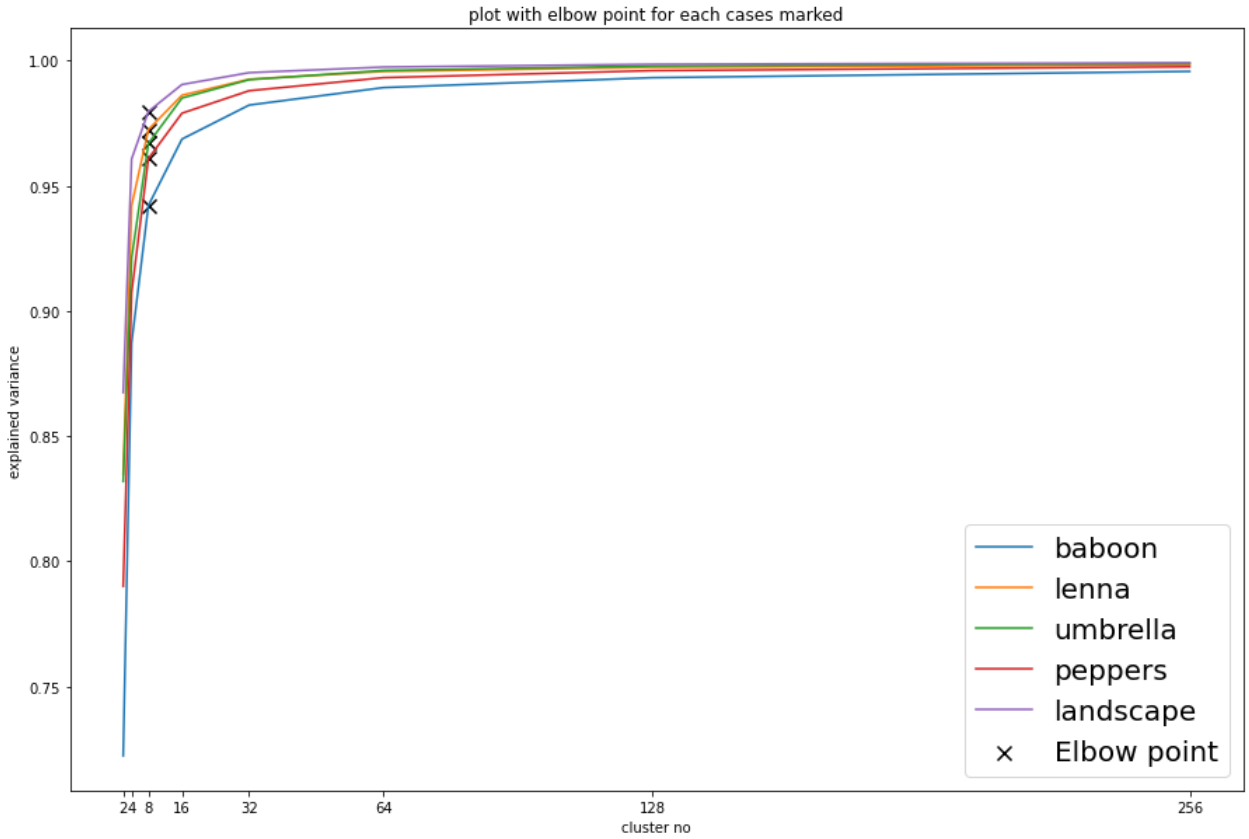
xvii. BCSS score and cluster number



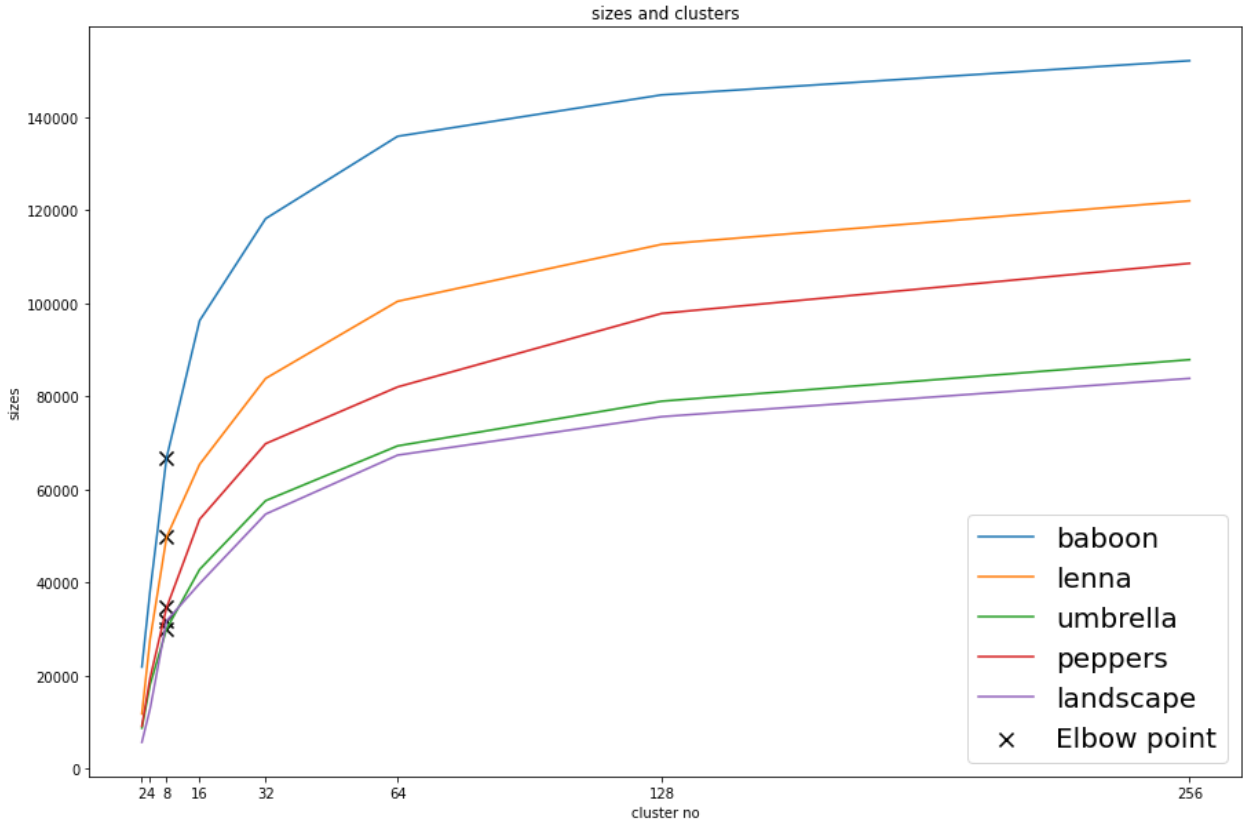
xviii. The table which holds, image names, cluster numbers, size of the images, WCSS, BCSS, explained variance. I dropped the column unique color names since it was impossible to show 256 unique color names in a valid way. In the notebook all unique colors for each image can be reached

	image names	cluster number	size	wcss	bcss	explained variance
0	baboon_compressed_2_no_of_clusters.png	2	21844	303688890	790166560	0.722368
1	baboon_compressed_4_no_of_clusters.png	4	38254	123334263	970521187	0.887248
2	baboon_compressed_8_no_of_clusters.png	8	66875	63302088	1030553362	0.942129
3	baboon_compressed_16_no_of_clusters.png	16	96265	34275882	1059579568	0.968665
4	baboon_compressed_32_no_of_clusters.png	32	118183	19420770	1074434680	0.982246
5	baboon_compressed_64_no_of_clusters.png	64	135872	11784074	1082071376	0.989227
6	baboon_compressed_128_no_of_clusters.png	128	144775	7456593	1086398857	0.993183
7	baboon_compressed_256_no_of_clusters.png	256	152122	4710313	1089145137	0.995694
8	lenna_compressed_2_no_of_clusters.png	2	11768	146950560	742853978	0.834851
9	lenna_compressed_4_no_of_clusters.png	4	27872	51534103	838270434	0.942084
10	lenna_compressed_8_no_of_clusters.png	8	49808	24526723	865277814	0.972436
11	lenna_compressed_16_no_of_clusters.png	16	65397	12299646	877504891	0.986177
12	lenna_compressed_32_no_of_clusters.png	32	83843	6608711	883195827	0.992573
13	lenna_compressed_64_no_of_clusters.png	64	100418	3857263	885947274	0.995665
14	lenna_compressed_128_no_of_clusters.png	128	112663	2379998	887424539	0.997325
15	lenna_compressed_256_no_of_clusters.png	256	122011	1498338	888306200	0.998316
16	peppers_compressed_2_no_of_clusters.png	2	9138	237842987	895179256	0.790081
17	peppers_compressed_4_no_of_clusters.png	4	19505	104995779	1028026464	0.907331
18	peppers_compressed_8_no_of_clusters.png	8	34628	44341121	1088681121	0.960865
19	peppers_compressed_16_no_of_clusters.png	16	53585	23803292	1109218950	0.978991
20	peppers_compressed_32_no_of_clusters.png	32	69835	13630257	1119391985	0.987970
21	peppers_compressed_64_no_of_clusters.png	64	82019	7714156	1125308086	0.993192
22	peppers_compressed_128_no_of_clusters.png	128	97806	4530067	1128492175	0.996002
23	peppers_compressed_256_no_of_clusters.png	256	108578	2786352	1130235890	0.997541
24	umbrella_compressed_2_no_of_clusters.png	2	8714	594605710	2943756821	0.831955
25	umbrella_compressed_4_no_of_clusters.png	4	18092	278983663	3259378868	0.921155
26	umbrella_compressed_8_no_of_clusters.png	8	30058	117131548	3421230983	0.966897
27	umbrella_compressed_16_no_of_clusters.png	16	42804	52858623	3485503908	0.985061
28	umbrella_compressed_32_no_of_clusters.png	32	57574	26935446	3511427085	0.992388
29	umbrella_compressed_64_no_of_clusters.png	64	69353	13867700	3524494831	0.996081
30	umbrella_compressed_128_no_of_clusters.png	128	78956	7468801	3530893730	0.997889
31	umbrella_compressed_256_no_of_clusters.png	256	87880	4131280	3534231251	0.998832
32	landscape_compressed_2_no_of_clusters.png	2	5678	202407416	1324356631	0.867427
33	landscape_compressed_4_no_of_clusters.png	4	12895	60014297	1466749750	0.960692
34	landscape_compressed_8_no_of_clusters.png	8	31788	31027098	1495736949	0.979678
35	landscape_compressed_16_no_of_clusters.png	16	39765	14576483	1512187564	0.990453
36	landscape_compressed_32_no_of_clusters.png	32	54699	7339081	1519424966	0.995193
37	landscape_compressed_64_no_of_clusters.png	64	67367	3918104	1522845943	0.997434
38	landscape_compressed_128_no_of_clusters.png	128	75630	2227366	1524536681	0.998541
39	landscape_compressed_256_no_of_clusters.png	256	83855	1316438	1525447609	0.999138

xix. Calculated elbow point marked on the plot with respect to cluster numbers. For all the images calculated elbow point is 8



xx. Sizes and cluster numbers, with optimal elbow point marked. It seems our algorithm does a good job taking size into consideration.



5. Conclusion

In this assignment we were expected to compress images using k means algorithm. The algorithm performs this operation by quantizing the images by minimizing the WCSS score such that it can represent the whole image colors by just using 2, 4, …, 256 unique colors. It is an interesting concept that we can explain so much of the images just by using 8 different colors and compress the image more than half of its original size. Although WCSS scores keeps decreasing as we increase the cluster numbers I cannot differentiate between 8 unique colors and 256 unique colors with ease. With the elbow point calculator, we have written, we have managed to capture this aspect as well. Although I did not include image sizes as a function parameter, the elbow method managed to capture this parameter, as can be seen by the figure above. In conclusion we have successfully managed the compress the image to its half size by using image quantization method with the minimum loss on explained variance.