Özyeğin University

CS552 – Data Science with Python

Assignment-2

Image Compression by K-Means Clustering

Furkan Cantürk

13.12.2020

1. INTRODUCTION

To represent a high-volume data with less number elements, similarity of data can be exploited so that we obtain a representative less-volume data for the raw data. With this basic motivation clustering algorithms can be even used to compress images. 5 images are subject to compress in this report, which are represented in Figure 1. As human eye cannot distinguish all color levels, images are compressed by reducing the number of colors to an acceptable quality level for some storage requirement or network performance.

K-Means is one of commonly used clustering algorithm. It splits a dataset into a given number clusters, k. Each cluster is represented by a centroid, which is the mean of clustered data points. The objective of this algorithm is to classify data points into clusters so that variance within the same cluster is low while it is high between clusters as much as possible. Working principle of k-Means is easy to implement. k different points are chosen randomly or based on a policy from the dataset as initial centroids. Then each data point is assigned to its nearby centroid based on a distance measure. The new centroids are computed by taking the mean of all points assigned to the same cluster. Then, each point is reassigned to the new centroids. These epochs continue until centroid values are stabilized or a given maximum number of iterations is reached.

Figure 1. baboon, landscape, lenna, peppers, and umbrella

2. PREPROCESSING AND COMPRESSION OF IMAGES

All presented images have the size of 512×512. Size of each image in terms of kilobytes and their number of unique colors are presented in Table 1. Firstly, using Pillow library, a given image is resized to 256×256. Then, opacity value (if any) level is dropped so that the image data has a 256×256×3 matrix shape. Then, this downscaled image data is converted to a 2-D matrix (at shape 256*256×3) in which each row represents a 3-dimensional pixel value.

After preprocessing image data, kmeans++ method is used to initialize k centroids. In this report, each image is compressed with 8 different number of colors (k), where $k \in \{2, 4, 8, 16, 32, 64, 128, 256\}$.

Table 1. Size and Number of Unique Colors of the Images

Image	Size (KB)	#colors
baboon	617.9	230,427
peppers	504.8	183,525
lenna	468.5	148,279
umbrella	467.8	135,560
landscape	315.9	58,694

k-means++ steps are as follows [1].

Step 1. Choose one center uniformly at random among the data points.

Step 2. For each data point x not chosen yet, compute D(x), the distance between x and the nearest center that has already been chosen.

Step 3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.

Step 4. Repeat Steps 2 and 3 until k centers have been chosen.

After initialization of centroids, k-Means clustering is applied for given k value, and centroid colors are obtained. Since each centroid represents the mean value of a cluster, it can be a real number RGB code. Regarding this issue, centroid values are rounded so that we have integer intensity values. Finally, each pixel is replaced by its nearby centroid pixel based on Euclidian distance, so, we have a 2-D matrix, which is the compressed image represented by k unique colors. Lasty, this 2-D matrix is converted to a $256 \times 256 \times 3$ matrix to use the right form of compressed image data.

3. EVALUATION

To measure performance of k-Means models, several measures are interested such as how much variance of data is explained, how much similar points in the same cluster are, and how much different than others each cluster is. Such each measure is computed based on a distance function, which is Euclidian in this report. The related basic measures are Within Cluster Sum of Squares (WCSS) and Between Clusters Sum of Squares (BCSS). The less WCSS (or the higher BCSS) is, the better clustering is. These measures are calculated as follows.

$$WCSS = \sum_j^k \sum_{i \in S_j} (x_i - \mu_j)^2 \quad (1)$$

$$TSS = \sum_i (x_i - \mu)^2 \quad (2)$$

$$BCSS = TSS - WCSS \quad (3)$$

μ_j is the mean value (centroid) of data points i belong to cluster j (S_j) and μ is the mean value of all data points. BCSS is obtained by (3) according to the law of total variance [1] where total sum of squares (TSS) is calculated as (2). The other performance score is Explained Variance (EV), which is the proportion of BCSS to TSS.

$$EV = BCSS/TSS \quad (4)$$

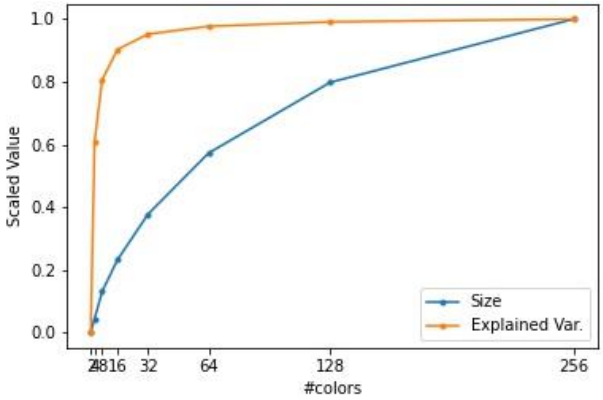

Figure 2. (Scaled) Explained Variance and (Scaled) Size of Each k-Color "baboon" Images

Table 2. Clustering Performance Measures for Each k value for Each Image

Image	#colors	Size (Kb)	WCSS	BCSS	Explained Var.
baboon	2	9.3	3.02E+08	2.23E+08	0.425
	4	14.5	1.22E+08	4.04E+08	0.769
	8	25.1	6.24E+07	4.63E+08	0.881
	16	37.3	3.35E+07	4.92E+08	0.936
	32	54.9	1.89E+07	5.06E+08	0.964
	64	78.7	1.14E+07	5.14E+08	0.978
	128	106.0	7.17E+06	5.18E+08	0.986
	256	130.5	4.56E+06	5.21E+08	0.991
Image	**#colors**	**Size (Kb)**	**WCSS**	**BCSS**	**Explained Var.**
landscape	2	2.5	2.02E+08	5.28E+08	0.723
	4	5.6	5.98E+07	6.70E+08	0.918
	8	13.1	3.10E+07	6.99E+08	0.958
	16	18.0	1.45E+07	7.16E+08	0.980
	32	27.1	7.30E+06	7.23E+08	0.990
	64	38.0	3.94E+06	7.26E+08	0.995
	128	51.5	2.25E+06	7.28E+08	0.997
	256	67.0	1.36E+06	7.29E+08	0.998
Image	**#colors**	**Size (Kb)**	**WCSS**	**BCSS**	**Explained Var.**
lenna	2	5.8	1.47E+08	2.61E+08	0.639
	4	11.3	5.17E+07	3.56E+08	0.873
	8	18.3	2.46E+07	3.83E+08	0.940
	16	26.8	1.23E+07	3.96E+08	0.970
	32	40.7	6.56E+06	4.01E+08	0.984
	64	58.8	3.83E+06	4.04E+08	0.991
	128	78.9	2.38E+06	4.05E+08	0.994
	256	98.7	1.52E+06	4.06E+08	0.996
Image	**#colors**	**Size (Kb)**	**WCSS**	**BCSS**	**Explained Var.**
peppers	2	4.9	2.38E+08	3.78E+08	0.614
	4	9.3	1.05E+08	5.11E+08	0.829
	8	15.3	4.44E+07	5.72E+08	0.928
	16	24.2	2.37E+07	5.92E+08	0.961
	32	35.3	1.37E+07	6.02E+08	0.978
	64	48.9	7.69E+06	6.08E+08	0.988
	128	65.4	4.55E+06	6.11E+08	0.993
	256	82.8	2.79E+06	6.13E+08	0.995
Image	**#colors**	**Size (Kb)**	**WCSS**	**BCSS**	**Explained Var.**
umbrella	2	5.0	6.13E+08	5.88E+08	0.490
	4	10.4	3.14E+08	8.86E+08	0.739
	8	13.9	1.52E+08	1.05E+09	0.873
	16	21.9	8.98E+07	1.11E+09	0.925
	32	30.9	6.43E+07	1.14E+09	0.946
	64	43.3	5.18E+07	1.15E+09	0.957
	128	58.0	4.53E+07	1.15E+09	0.962
	256	74.3	4.22E+07	1.16E+09	0.965

Explained variance represents the quality of compressed image. As the number of colors increases, better images are obtained, but at the same time the image size increases. So, this tradeoff requires to find the optimal number of colors to compress given image, and can be quantized with the following function where Size is in terms of kilobytes.

$$P_1 = EV/Size \qquad (5)$$

When we only consider the maximum value of P_1, which is depicted for baboon image in Figure 3, 4-color compressed baboon image should be chosen as the best. However, visual comparison of 4-color baboon with its original image (See Figure 6. at Appendix) shows that this much high compression becomes too poor in terms of image quality. Besides, 4-means clustering of baboon image data has a high WCSS value, which is observed in Figure 3. For a better selection of compression level, effect of WCSS should be more enforced. Therefore, the second tradeoff function is proposed to find the optimal number of colors for the compression, which combines the quality-size tradeoff with the Elbow method.

$$P_2 = EV/Size/WCSS \qquad (6)$$

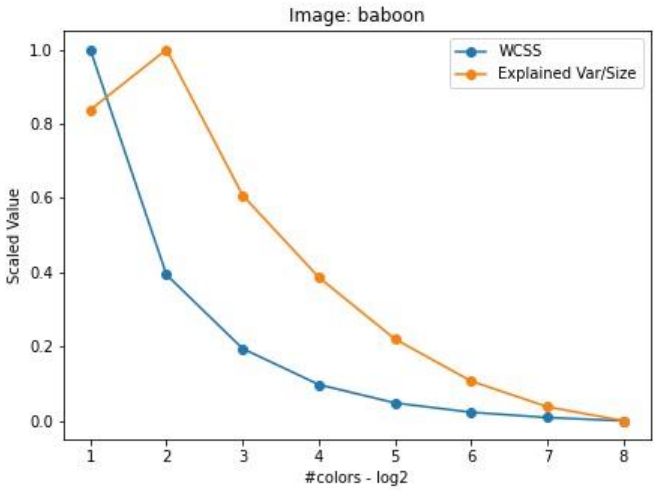

Figure 3. (Scaled) Explained Variance / Size and (Scaled) WCSS of Each k-Color "baboon" Image

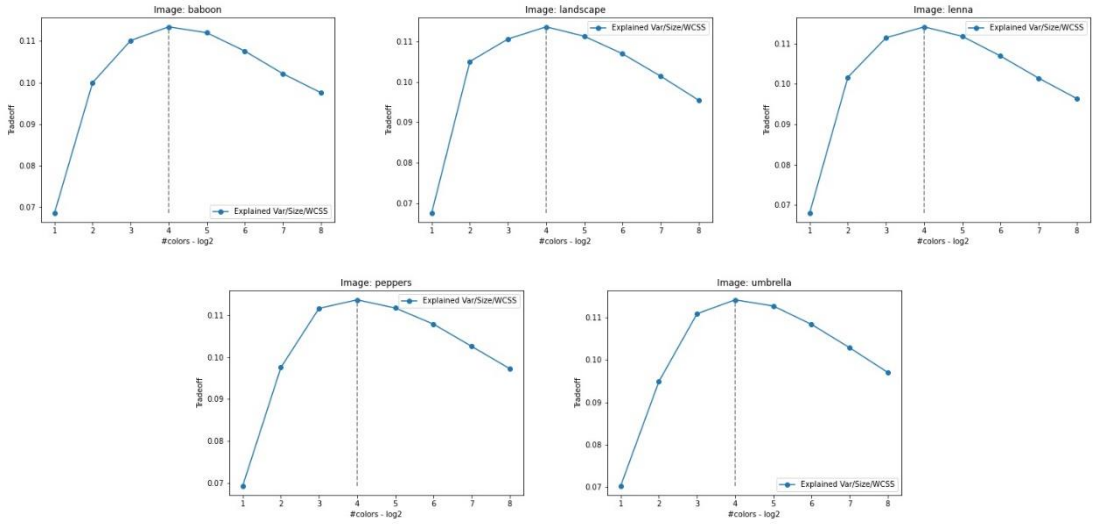

Figure 4. Tradeoff Function in Terms of k for Each Image

Since the used measures for P_2 have different scales, firstly each measure is standardized by removing its mean and scaled to unit variance, in other words, each measure is standard-scaled. Then, this scaled measures are shifted up by some value which is greater than its maximum value so that all measures are strictly positive (i.e. negativity is eliminated). By using these scaled-shifted measure values, P_2 values are calculated, which are represented in Figure 4 for each image.

Based on tradeoff values by P_2, 16-color compression is the best in terms of the tradeoff for each image. In addition, it is observed that P_2 values are very close for 8, 16, and 32 colors. Thus, based on different use cases of compressed images, 8-color images can be chosen in favor of storage or 32-color compressed images can ben chosen in sake of better quality as well.

4. CONCLUSION

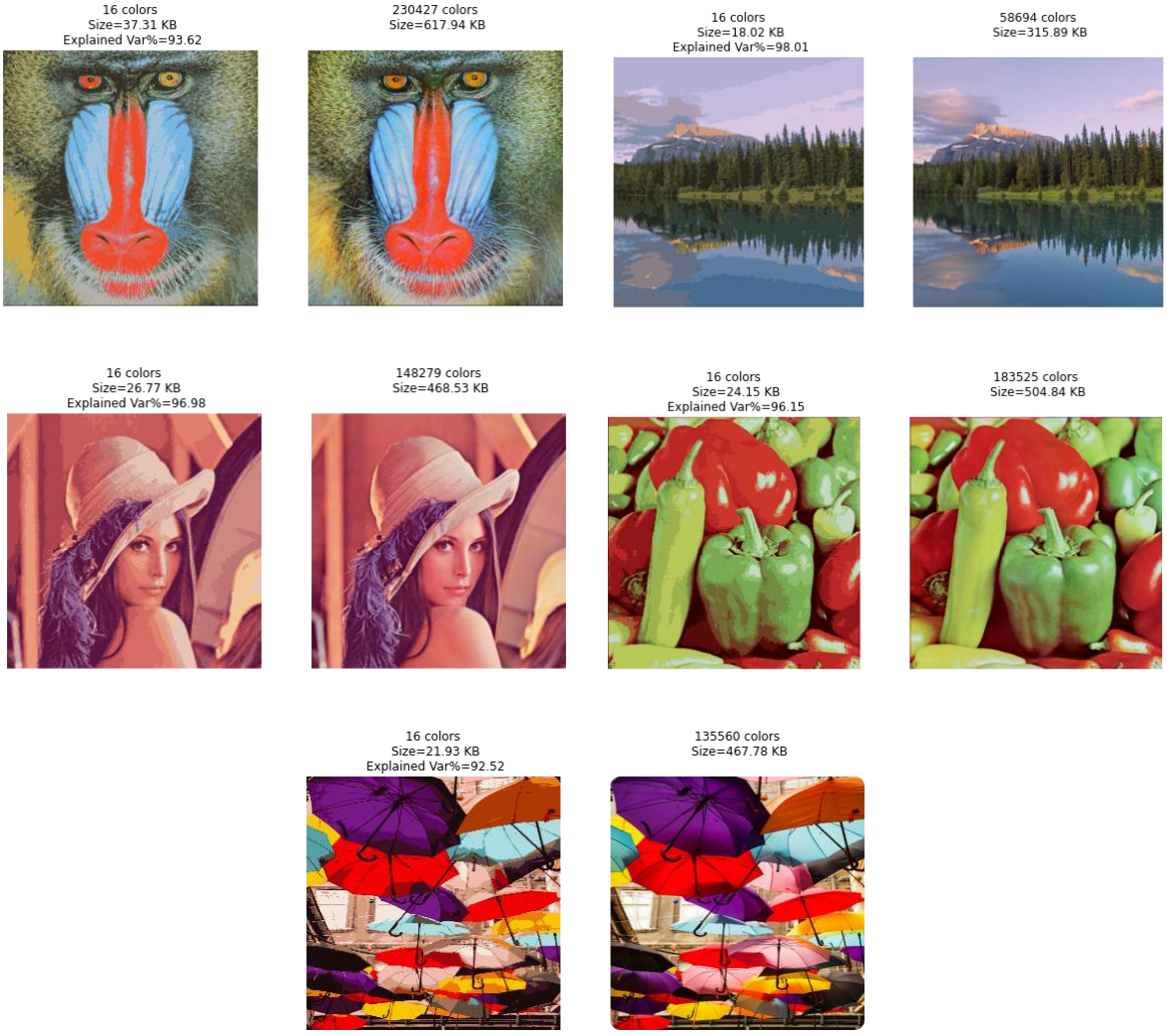

Figure 5. The Chosen Compressed and Original Images

Using k-Means clustering, the downscaled forms of 5 images (to 256x256) are compressed by 8 different k values, which are powers of 2 up to 256. The proposed tradeoff function considering explained variance, image size, and WCSS is utilized to choose the best compression for each image, which is 16-color for all images. The presented 16-color images in Figure 5 provide 95% less kilobytes at average and 95% quality at average compared to their original forms.

REFERENCES

[1] Law of total variance

[2] k-means++

APPENDIX

The implementation code is attached as a Jupyter Notebook.

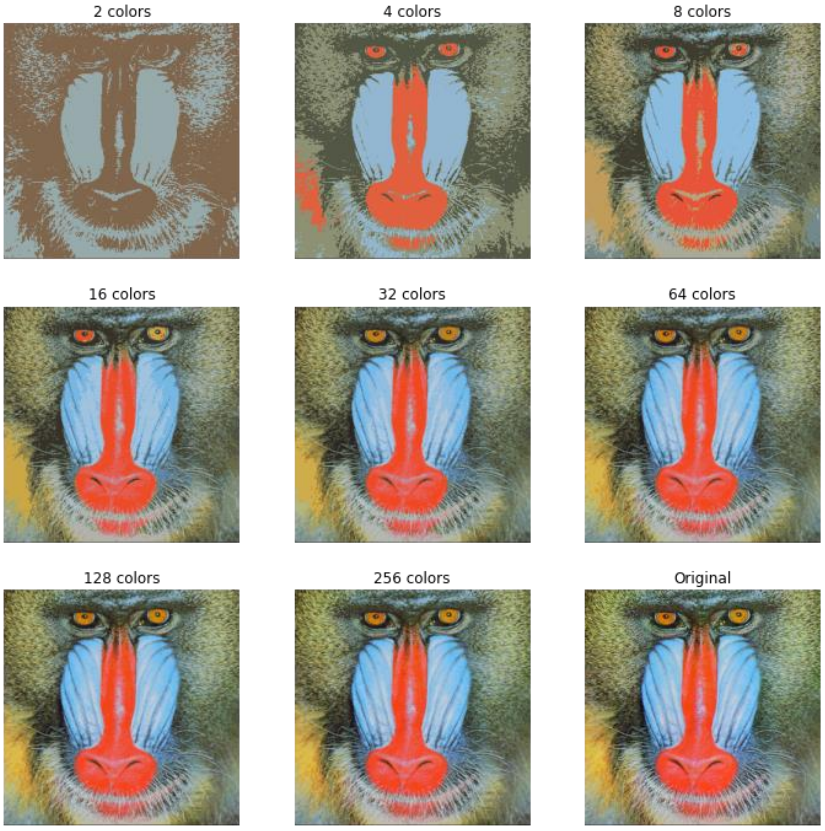

Figure 6. Compressed and Original baboon Images

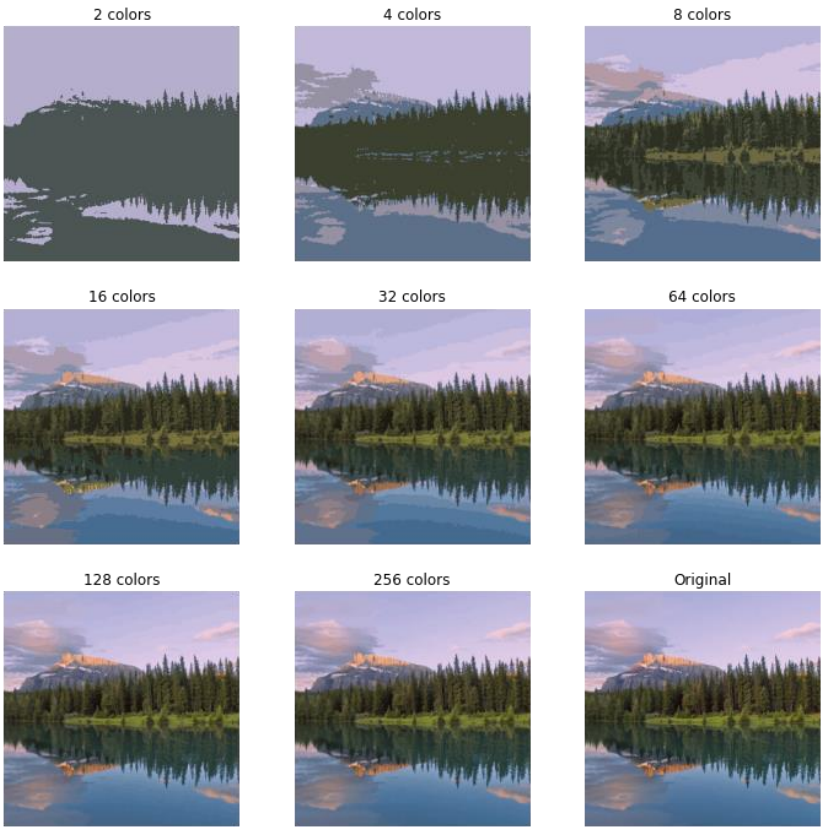

Figure 7. Compressed and Original "landscape" Images

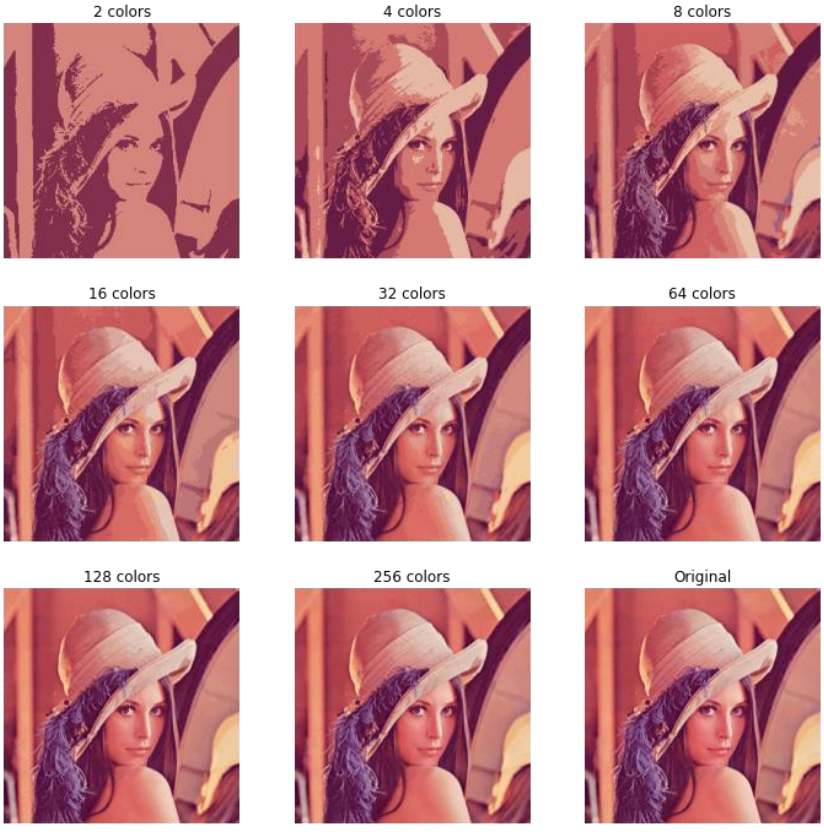

Figure 8. Compressed and Original "lenna" Images

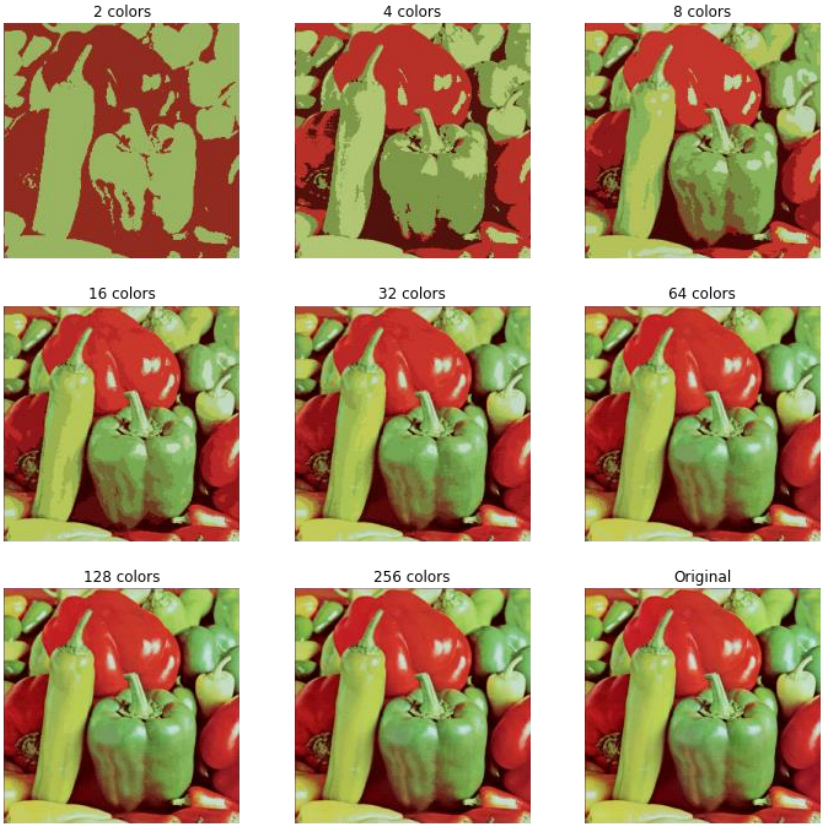

Figure 9. Compressed and Original “peppers” Images

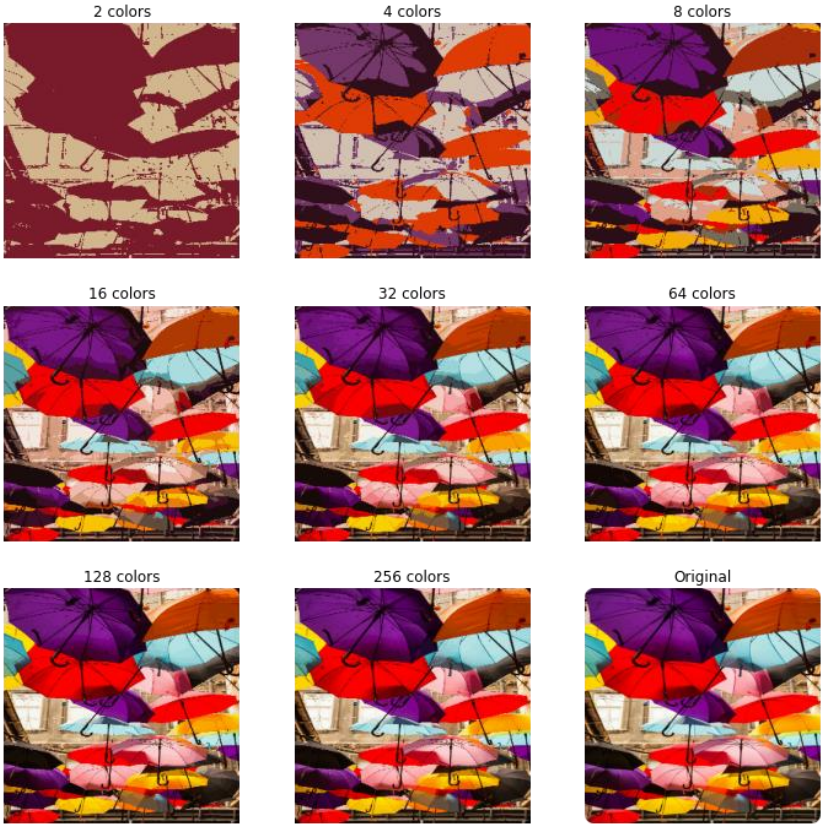

Figure 10. Compressed and Original “umbrella” Images