K-Means Image Compressor

Brandon Salgado
Submitted: 4 December 2022
CSCI 183: Data Science

Introduction

Image compression is something that has revolutionized the digital world. It's importance can be seen in every corner of the online space from storing thousands of photos on a device to sending internet videos to a friend, image compression is effectively unavoidable. Image compression algorithms have existed since the early 1950s and there is a large variety of algorithms which are useful for different scenarios. Within information theory, data compression is the process of encoding information using fewer bits of information than the original representation. Image compression is a type of data compression that is applied to digital images to reduce the cost of storage or transmission. There are two types of image compression, these include "lossy" and "lossless". Lossy image compression removes data from the file permananely, and irriversibly. Lossless image compression removes data however, the original representation can be reconstructed after compressing the image.

Unsupervised Learning

During the class CSCI 183: Data Science, the topic of unsupervised learning was touched upon. Unsupervised learning sits within the realm of data science and machine learning, and is a method of learning patterns in data. These patterns specifically are learned from "untagged data" which is data given without a specific dependent variable or expected output. This differs from supervised learning where along with the "features" or independent variables, comes with dependent variables or target variables that the algorithm can use to create associations between the target variables and the features. Without the target variables, the machine is forced to create associations within the features and then build a representation through its "imaginative" algortithm based solely on the features.

One subset of unsupervised learning is "clustering", where the features are grouped together based on similarity. These algorithms are built upon two properties. These are (1) all the data within a cluster should be as similar to eachother as possible and (2) the data from different clusters should be as different as possible.

K-Means Algorithm

Within clustering, there are a variety of algorithms to choose from. The algorithm used in this project is the K-Means algorithm. This algorithm involves grouping data points into a number (K) of mutually exclusive clusters.

Method of Compression

Regarless of the size of an image, every image is encoded using RGB values. These values are coded for 0 to 255 for each color. Meaning using combinatorics, for every pixel, the image must contain 256*256*256, equal to 16,777,216 bytes of data. However, the human eye cannot distinguish between 16 million different colors, therefore taking advantage of this, it is possible to eliminate some of the colors within the image. Compressing the data using K-Means can be implemented and conceptualized by reducing the range of colors needing to be stored. This is done by grouping together regions of similar color and representing them as the mean of their color values.

Algorithm

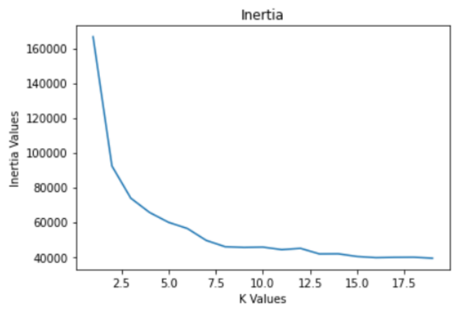
The algorithm for implementing K-Means as an image compressor is as follows. After importing the image, it needs to be preprocessed to be easer to work with. The current size of the image is (rows, columns, 3(RGB)) to flatten the image from three dimensions to two dimensions, multiply the rows and columns to get (rows*cols, 3). After this, the image is ready for the K-Means clustering where it will find K clusters (initialized randomly) that will represent the surrounding colors. Once the centroid(cluster centers) are found, the RGB value all of the pixels within each cluster will be replaced by the average of all of the pixels within the cluster. Finally, the image is re-shaped into three dimensions (rows cols, 3) and displayed.

Error Analysis

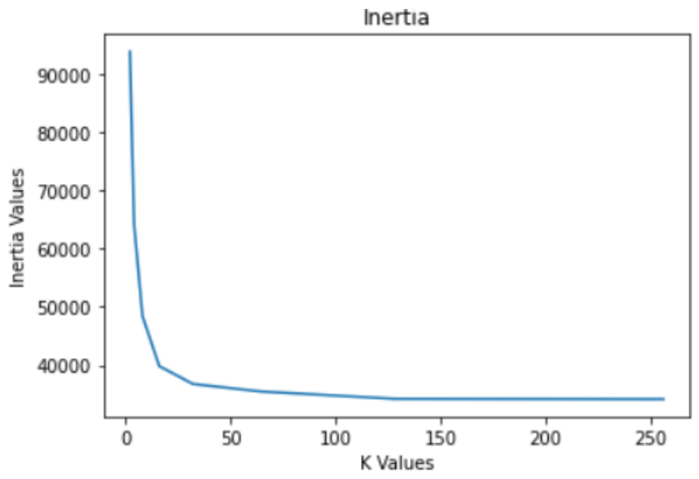
To find error for K-Means, the Inertia is used which is the sum of all the distances from all data points to their respective clusters. As the K values increased, the inertia should decrease. When using K values from 1-19 the inertia graph looked like this:

```
In [97]: kValues = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

         plt.plot(kValues, inertiaValues)
         plt.title('Inertia')
         plt.xlabel('K Values')
         plt.ylabel('Inertia Values')
         plt.show()
```



And For Larger K Values:

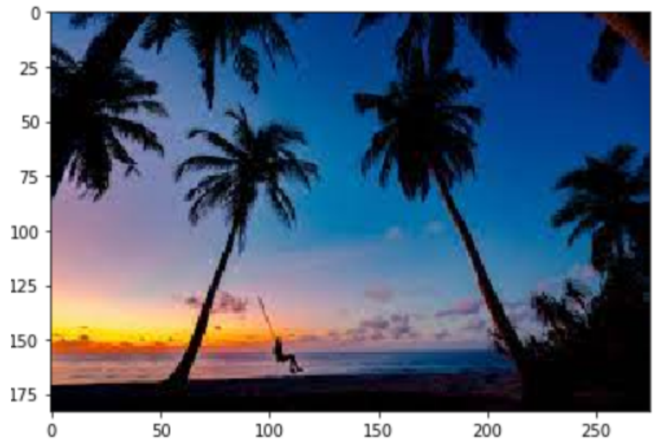


Here it is clear to see that there is a leveling off of the inertia as the K values increase. Usually the best results are found at the “elbow” of the curve however for image compression, the best results are subjective to how small or good looking the image should be. However, here the elbow of the curve is a fantastic ratio of size to image quality. This is the point where the quality of the image is “good enough” to justify how small the file is. Overall, I do believe that the model learned and performed well at the given K values, and produced a respectable image from them.

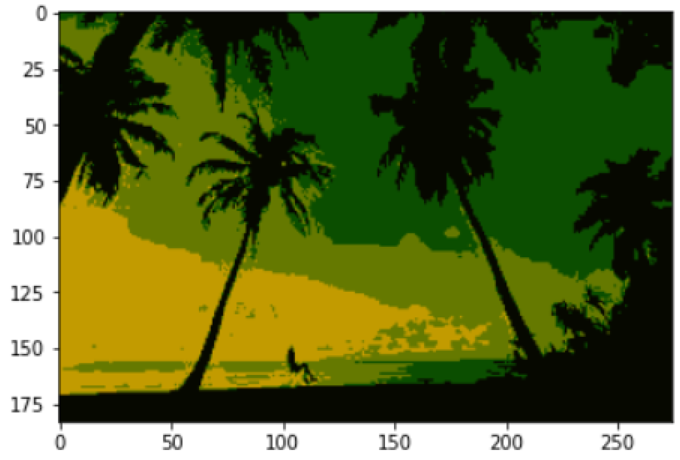
Final Image Results:

Below are the results of the image compression which do look far from the original image however they are a small portion of the data of the original. The original image was 5832 x 3888 pixels with the RGB values that makes 544 MB. With a K value of 16 that makes the resulting image of size 90 MB which is 16 percent of the original size. This makes a huge difference in image compression and storage required.

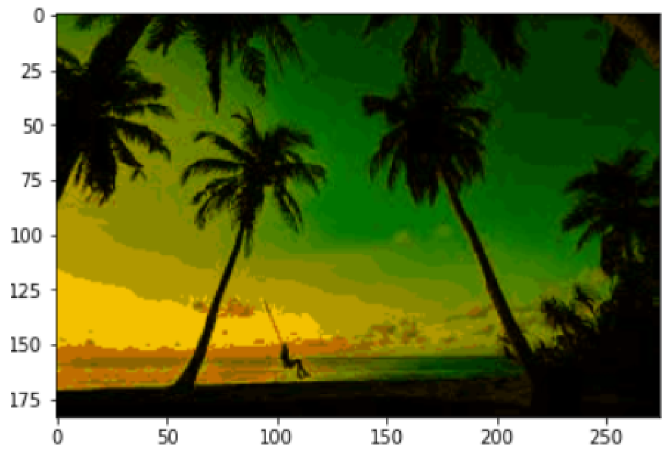
Original Image:



New Image (K value 4 (elbow of inertia graph)):



New Image (K value 16):



Here is a graph representation of the size of the file after compression:

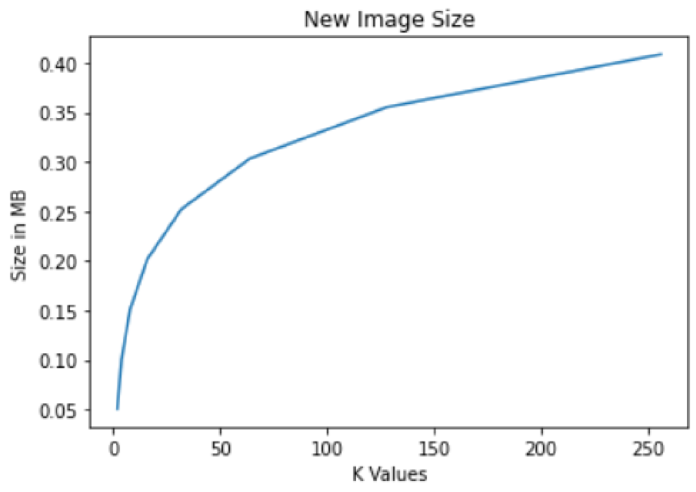


Image Analysis

As the values of K or the number of clusters increases, it provides a more visually accurate representation of the image. Through the image-size graph, it is easy to see this trend within the size of the image. Low values of K providing an exceptionally low amount of space. Since the K-Means algorithm doesn't produce the same output every time, after running this algorithm multiple times, it seems to hold up well especially with images that have less colors. This is due to the lack of clusters needed to represent the image.

Conclusion

K-Means is a very interesting and effective way to compress an image that should not be overlooked. Yes, the quality takes a hit with lower values of K, however a majority of the data is also discarded for what makes up a pretty good visual representation of the original image. No, it isn't typically the best method for image compression especially because it is of "Lossy" format but it's definitely one of the more interesting applications of K-Means especially from a visual point of view. In the future I would love to continue to try this program with different images and larger K values to see more relationships and how they compare to each other.