Colour Quantization with K-Means Clustering

Tan Yu Hoe
DAAA, School of Computing
Singapore Polytechnic

This technical paper aims to conduct Colour-Based Segmentation on an Image based on the chromaticity plane leveraging over an unsupervised learning technique, K-Means Clustering, to reduce the number of distinct colours. The implementation is done using Python programming and sci-kit learn machine learning library.

Keywords—image, compression, KMeans, Mini-Batch KMeans, machine learning, clustering, unsupervised, python

I. Introduction

Data in the present day comes in many forms, such examples are text, video, images, and sound. These brought innovation to the Data Science community, such as natural language processing and biomedical research. A recent innovation by GovTech Singapore, the Data Science and Artificial Intelligence Division developed a video analytics project, VigilantGantry, on a fully automated, contactless gantry system for temperature screening, improve the rate of contactless scanning. [1] In this technical paper, I will demonstrate an unsupervised learning approach on a type of Unstructured data, Colour Quantization. Colour Quantization is a process of assigning a label or category to every pixel in an image such that clusters of pixels would share common properties, reducing the number of distinct colours. The objective is to reduce the number of distinct colours based on similar RGB colour characteristics using K-Means Clustering on the chromaticity plane without losing too much of its visual properties.

II. Related works

In the period when computers were just needed invented, there was a need for multimedia compression due to the limited memory in their RAM (Random Access Memory) and the limited support for video inputs. As such compression techniques such as Colour Quantization came about.

In earlier days, using of K-Means for colour quantization is deem unsuitable due to its high computational requirements to reach proper convergence. In year 2011, E. Emre Celebi revisited K-Means effectiveness as a algorithm to perform colour quantizer. [2] He showed that a well-implemented K-Means approach, including initialisation method, outperforms a variety of colour quantization methods.

III. Image processing

The image used here is a MIME (Multipurpose Internet Mail Extensions) image/jpg media type, a picture of myself taken in Singapore Polytechnic's Library. After loading the image as a into a Jupyter Notebook, the NumPy array indicates that it is 6 MB, and (3363, 6000, 3) in shape. Within the shape, 3 represents each individual colour channel in the RGB colour model. From a simple search on google, this image is considerably large in terms of memory size relative to the average size 11 kB of image/jpg.

Fig. 1. Image used for Demostration

The input image is read as binaries and returns as a ndarray, where each element is pixel-based on the RGB colour scale. A rough introduction on the RGB colour model, each colour is formed using three light channels (one red, one green, and one blue), where the channel is added together to form a colour, thus being additive. Each component in the RGB model contains an unsigned integer number in the range 0 to 255, the range that a single 8-bit byte can offer. The value in each light component represents light intensity in the colour channels before being added up. Therefore, in theory, there are 16,777,216 possible colours within an image.

The first approach is to perform normalization on the image as clustering algorithms such as K-Means uses Euclidean Distance to form any conclusions. Typically for normalization, I would use Min-Max normalization such that all three light intensities will transform into the range [0,1], meaning that the minimum and maximum value of RGB will be 0 and 1, respectively.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Since the RGB light intensity scale is always between 0 to 255 inclusively, we can solely divide each pixel by 255 to normalize the image as the minimum is a constant 0.

$$Pixel\ Intensity_{norm} = \frac{Pixel\ Intensity}{255}$$

The next step is to reshape the image data from a three-dimensional array to a two-dimensional array, such that each column vector represents an individual light component of Red, Green and Blue.

```
# reading the image as an ndarray
image = imread('raw.jpg')

# normalization and reshaping the data
image_data = (image / 255).reshape(-1, 3)
```

Fig. 2. Code implementation for Image Preprocessing

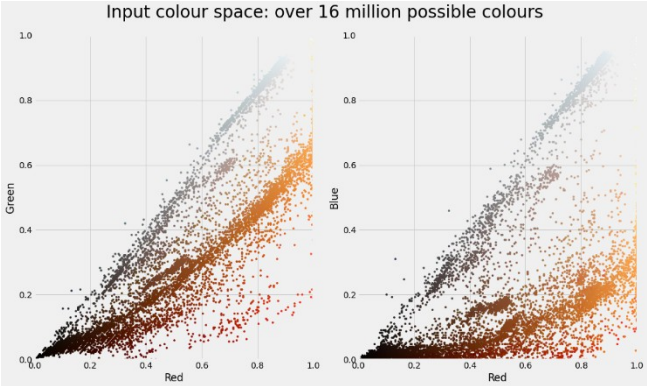


Fig. 3. Two Dimensional Color Space Visualisation

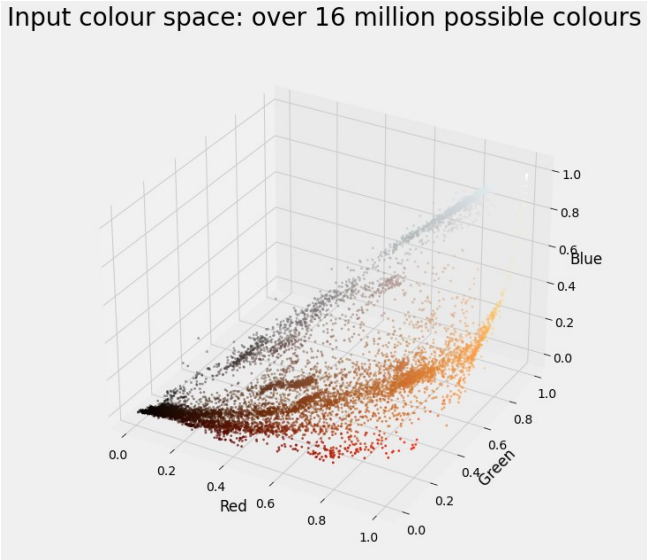


Fig. 4. Three Dimensional Color Space Visualisation

In the colour space, it can be observed that the colours are quite spread apart respective to their unique colours. It is as if there is a colour gradient of diversified colours. For Colour Quantization, we will attempt to make full use of K-Means Clustering to perform segmentation on the colours on the colour space, to find clusters of colours with similar RGB attributes. By doing this way, we will keep the visual properties of the image while reducing the number of distinct colours by replacing the pixels with the centroid pixel.

IV. K-Means Clustering

K-Means Clustering is an iterative clustering algorithm that attempts to find the centroids based on the input number of clusters (denoted as k) given. The procedure for K-Means is as follows:

1. Initialization: initiate random centroids based on k
2. Assignment: assign to the nearest centroid
3. Update: update the centroid with a new cluster
4. Repeat steps 2 and 3 until convergence

The idea is to locate the (r_n, g_n, b_n) at the centroids which is done using the Euclidean Distance or "straight-line distance".
By using K-Means, the algorithm could effectively locate and replace the cluster region with the centroid RGB values.

$$Distance = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

For this objective, I will use the Mini Batch K-Means algorithm instead of vanilla K-Means as the data (image) is very large in size, and it requires much computational time if vanilla K-Means is used. Mini Batch K-Means converges faster than K-Means by using the random sampling method, however, the quality of the results is reduced though the difference is not that far apart from results using K-Means. [4]

```
from sklearn.cluster import MiniBatchKMeans

# initiate class and fit image data
model = MiniBatchKMeans(n_clusters=4, batch_size=3072)
model.fit(image_data)

# update every pixel with the assigned colour/centroid
k_colours = model.cluster_centers_[model.labels_]
```

Fig. 5. Code Implementation of Mini Batch K-Means, k = 4

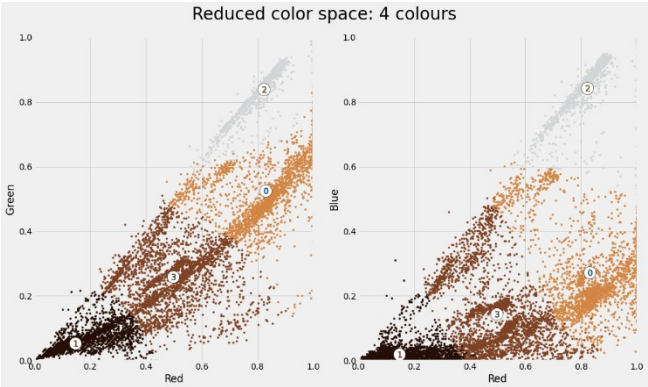


Fig. 6. Two Dimensional Color Space after 4 centroids segmentation

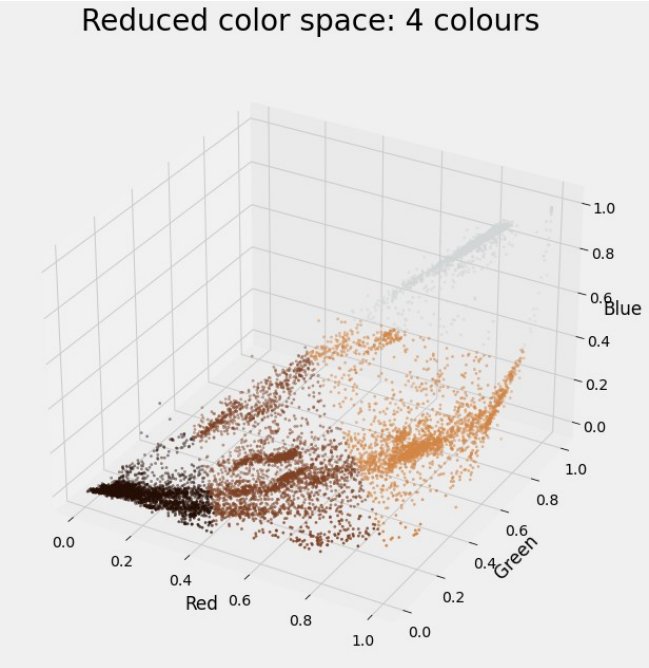


Fig. 7. Three Dimensional Color Space after 4 centroids segmentation

After implementing K Means Clustering with k = 4, it could be seen in the colour space that there are 4 different distinct colours and clusters. When compared to the original colour space, it can be seen that the reduced colour space has less diversified colours.

Fig. 8. Original Image and Compressed Image, $k = 4$

In figure 8, there is a clear effect of K-Means clustering, an overall decrease in quality but still retains its visual properties. K-Means clustering here takes the centroid colour clusters and reduce the number of colours to represent the image. This is also called lossy compression, as the image quality is reduced to decrease the file size. In a sense, we can create a compressed version of our image to a predefined extent by choosing the number of distinct colours we would want to include, at a trade-off of losing some visual properties.

V. RESULTS AND DISCUSSION

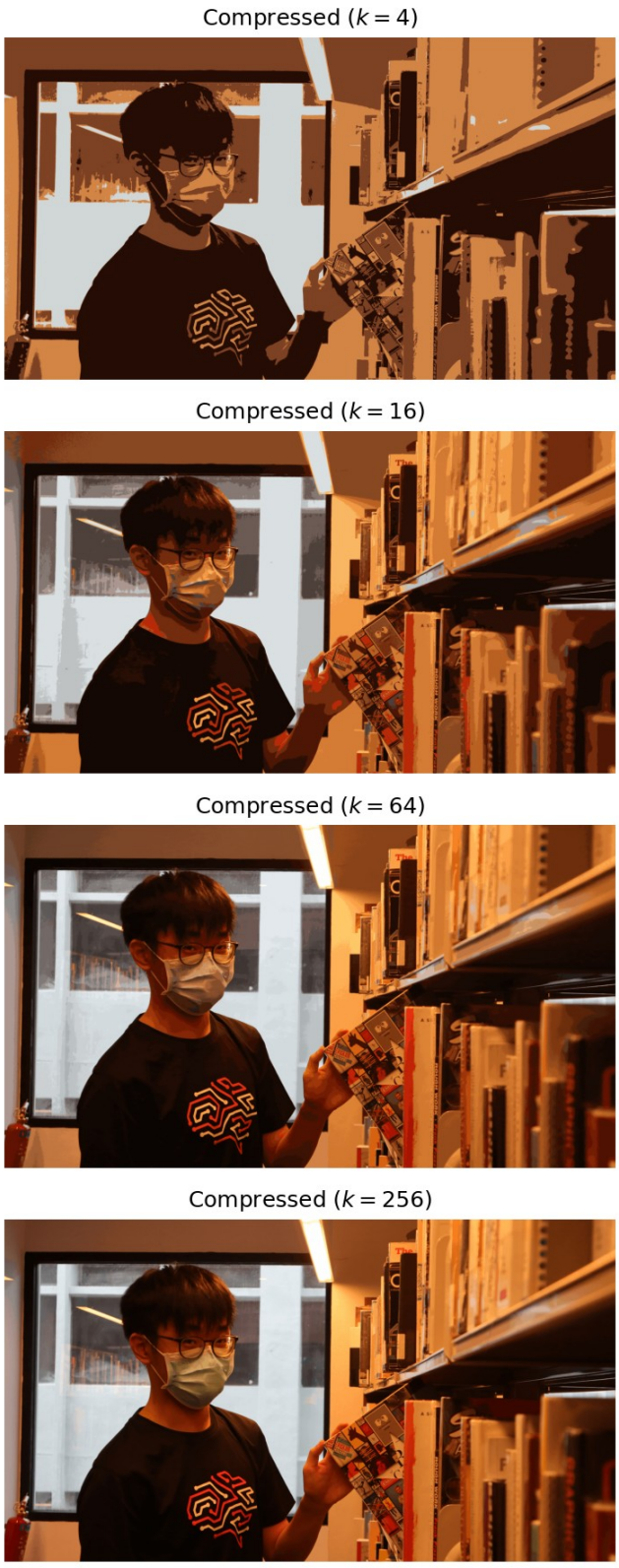


Fig. 9. Compressed Images of different number of k

In figure 9, it can be observed that as k increases, their image becomes clearer. In this case, k = 256 is the clearest when it comes to replicate the original image. It can be noticed in k = 4, the image quality is significantly diminished due to the lack of variation of colours.

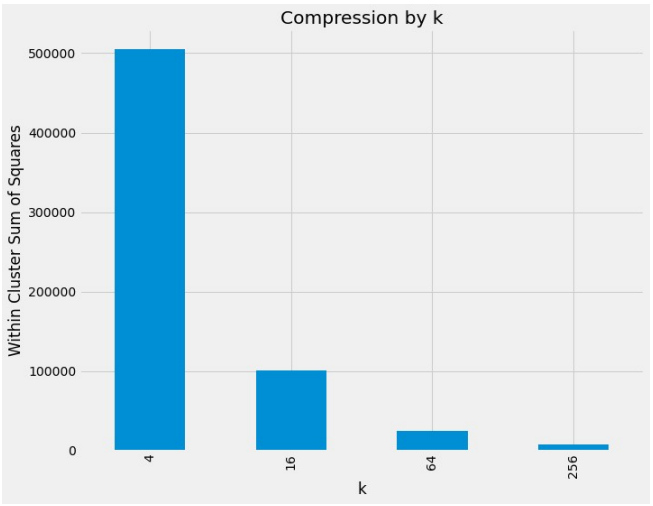
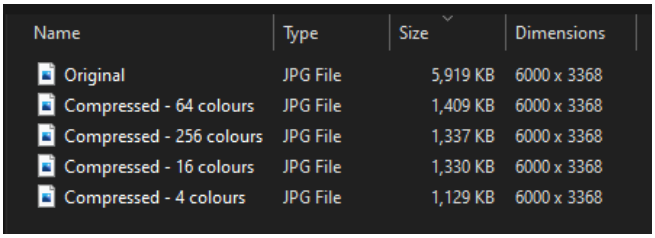


Fig. 10. Intertia Scores for each k

WCSS (Within Cluster Sum of Squares) or also known as inertia score is basically the mean euclidean distance to the cluster centroid. The inertia scores looks pretty standard to any K-Means clustering task, larger k results to smaller inertia score, the same goes for vice versa. Based on the "elbow" point, it is likely to go for k = 16 for the best clusters, but choosing a specified number of colours is not required for this colour quantization. Usually for image compression using colour quantization, it is better to check for hardware specification on how many colours does the video graphic support at one time.



Name	Type	Size	Dimensions
Original	JPG File	5,919 KB	6000 x 3368
Compressed - 64 colours	JPG File	1,409 KB	6000 x 3368
Compressed - 256 colours	JPG File	1,337 KB	6000 x 3368
Compressed - 16 colours	JPG File	1,330 KB	6000 x 3368
Compressed - 4 colours	JPG File	1,129 KB	6000 x 3368

Fig. 11. File Directory containing all of the images

By using K-Means clustering, it can be seen that the file size of the compressed image by an average of 20% of the original. It can also be seen that the file size increases as the number of distinct colours in an image increases, except for an anomaly in the compressed image with 64 colours. Possible reasoning is due to the compressing algorithm for MIME image/jpg files.

VI. CONCLUSION

In this technical paper, we have explored colour quantization using K-Means Clustering. While colour quantization does not perfectly replicate the original image, it illustrates the same image while reducing the number of visual properties in an image, which is also called lossy compression. Applications of colour quantization can be applied in places where computational power is limited and there is a need for large multimedia transmission. While the image used here is being compressed using K-Means Clustering an average of 20% of the original memory size, there is a trade-off with image quality where image quality will also decrease along with file size.

ACKNOWLEDGMENT

I thank Dr Wilson and Dr Peter, who implemented this technical paper section into the assignment. I would also like to thank Tingxiao for taking the photos for me.

REFERENCES

1. VigilantGantry - access control with AI and video analytics. Singapore Government Developer Portal. (2021, August 6). https://www.developer.tech.gov.sg/technologies/digital-solutions-to-address-covid-19/vigilantgantry.

2. Celebi, M. E. (2011). Improving the performance of k-means for color quantization. Image and Vision Computing, 29(4), 260–271. https://doi.org/10.1016/j.imavis.2010.10.002

3. Clustering¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans.