Analyzing a Python Program for Image Color Compression using K-Means

Phan Lam Anh - 21127580

July 19, 2023

1 Introduction

This report presents an analysis of a Python program designed for image processing. The program uses the K-Means clustering algorithm to perform color quantization on an image. It first imports libraries, including Pillow (PIL) used for image processing, NumPy for handling numerical operations, and matplotlib for creating visualizations.

This report will cover how these libraries are used in this program, what the K-Means clustering is, and how it's used for image quantization. Along with the program's explanation, this report will provide a detailed discussion on the handling of potential 'divide by zero' or 'invalid value' warnings during the centroid computation process.

2 Background

2.1 Pillow (PIL)

Pillow is the friendly fork of PIL (Python Imaging Library). It is used for opening, manipulating, and saving many different image file formats. In this program, it's used to load an image which is to be quantized.

2.2 NumPy

Short for 'Numerical Python', NumPy is one of the most important foundational packages for numerical computing in Python. In this program, it's used for its powerful n-dimensional array object and its function to randomize and calculate means, among other operations.

2.3 Matplotlib

Matplotlib is a visualization library in Python used for 2D and 3D plots of arrays. In this program, it's used to display the original and quantized images.

2.4 K-Means clustering

K-Means is a type of unsupervised machine learning algorithm used to classify items into k groups of equality variance. In the context of image quantization, K-Means clustering groups similar color values together, reducing the overall number of unique colors in an image.

2.5 Image Quantization

Image quantization is a process that reduces the number of distinct colors used in an image, important for displaying images on devices that support fewer colors than the image contains. It's also useful for compressing image data without losing too much quality.

3 Program Analysis

The program can be broken down into the following sections:

3.1 Importing Libraries

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

3.2 K-Means Function

The ‘kmeans‘ function receives an image represented as a 1D array, the number of clusters, maximum iterations, and a method to initialize the clusters (”random” or ”in_pixels”).

Within the ‘kmeans‘ function, another function ‘initialize_centroids‘ is defined. It takes the same parameters to decide how to initialize centroids (either starting with random color values or by using pixel values taken directly from the image).

The ‘kmeans‘ function then enters a loop in which it assigns each pixel to the nearest centroid and recalculates the centroids based on these assignments.

In this piece of code:

```
for _ in range(max_iter):
    distances = np.sqrt(((img_1d - centroids[:, np.newaxis]) ** 2).sum(axis=2))
    labels = np.argmin(distances, axis=0)
    centroids = np.array([img_1d[labels == k].mean(axis=0) for k in range(k_clusters)])
```

3.3 Running the K-Means Algorithm

After loading and reshaping the image, the K-Means function is called. The labels (cluster assignments) are reshaped to match the shapes of the images and finally, a new image is created where each pixel's color is replaced by the close matching or the centroid color of its nearest cluster, resulting in a quantized version of the original image.

3.4 Visualize the Result & Comment

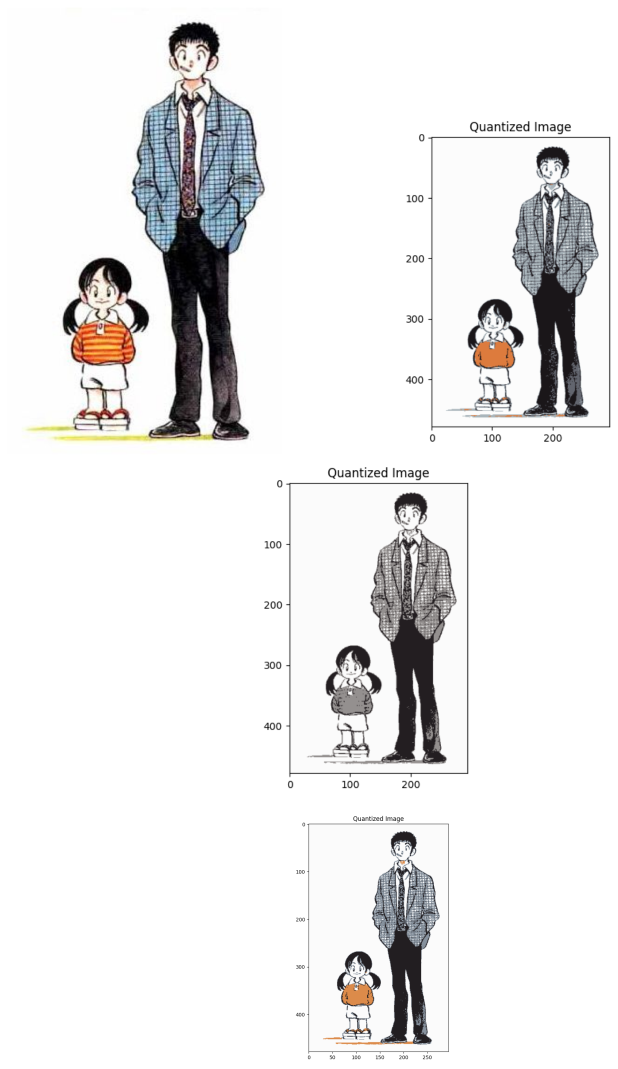

Figure 1: Image after the compression

The color intensity of the image, after the color reduction process, may not be as striking as before, but it continues to remain quite pleasing.

Even with fewer colors, the images have managed to retain their original forms and distinct characteristics, making them easily recognizable to the viewers.

All in all, the end results are satisfactory and there is scope for improvements in the future.

4 Error Handling

The division by zero and invalid value warnings are handled well by the NumPy seterr function, ensuring a smooth execution of the program. It uses the 'ignore' value to prevent the program from throwing a warning in the event that there's a division by zero or if an invalid value is encountered during the mean calculation.

5 Conclusion

The report provides an in-depth analysis of the Python program and explains the underlying logic of K-Means based Color Quantization. The program error handling safely prevents potential 'divide by zero' or 'invalid value' warnings, demonstrating excellent error management practices.

References

[1] SciPy community. *scipy.cluster.vq.kmeans*. SciPy Reference Document. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/cluster.vq.html

[2] Amit Banerjee. *Image Color Quantization using K-means Clustering python implementation*. GitHub, 2021. [Online]. Available: https://github.com/AmitBaanerjee/Image-Color-Quantization-using-K-means-Clustering-python-implementation

[3] Chingisoo Infinity. *K-means-Image-Compression*. GitHub, 2021. [Online]. Available: https://github.com/chingisooinar/K-means-Image-Compression.