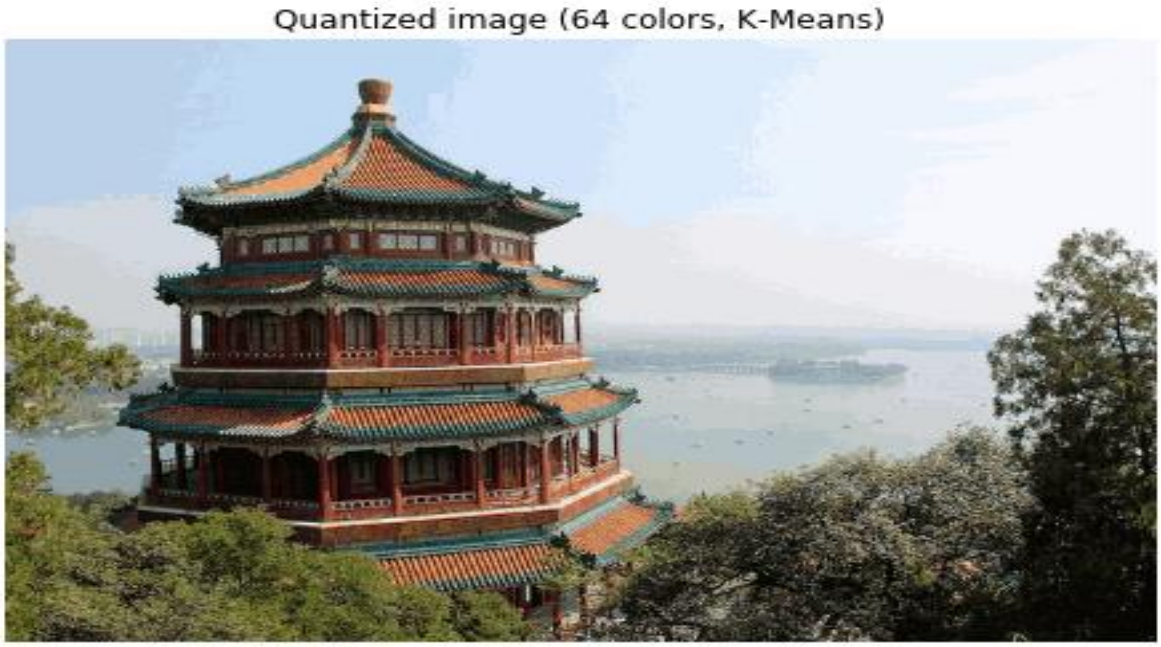
Sahil Paneri sp4313
Jayakrishna Narra jn1711

Homework 4

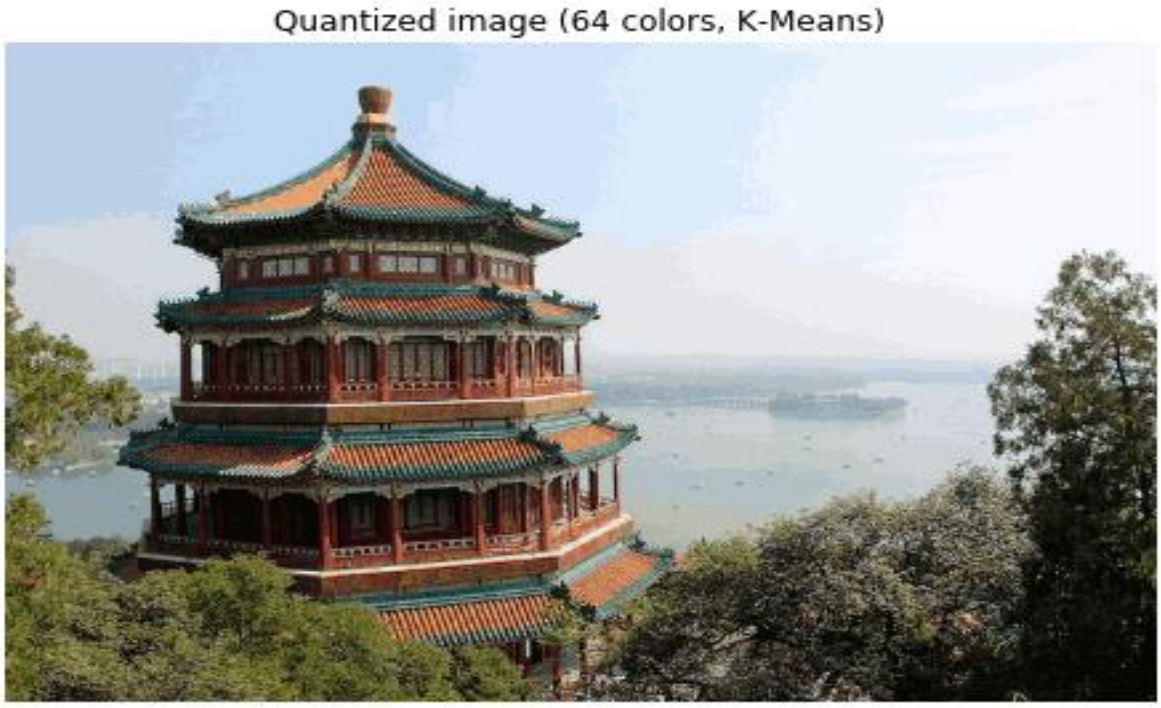
1.a.
Representing an image in number of classes which are lower than the original number of classes results in a segmentation of image with the quality of the output image lower than the input image. This is very much evident in the duration of this homework. We can clearly see in this problem that when the input image is represented in k=64 and k=256, degradation happens.

Original Image:

The images obtained after k-means algorithm and the benchmarking algorithm are as follows:
K=64, k-means algorithm



K=64, benchmarking algorithm, k-means++



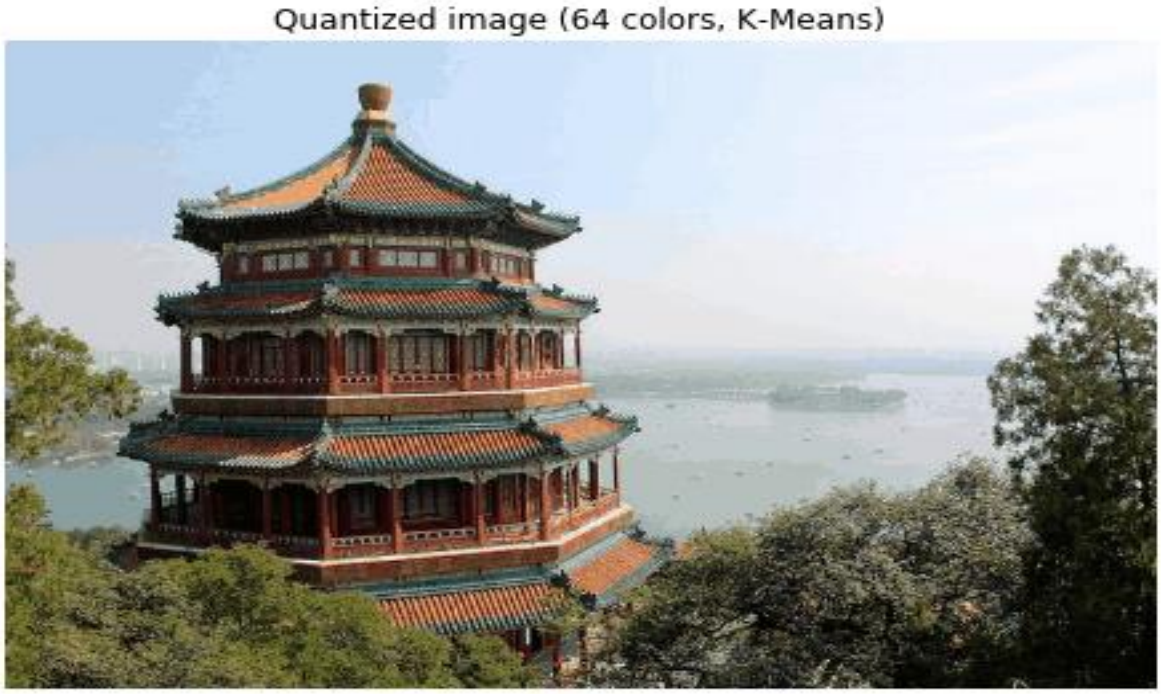
K=256, k-means algorithm, k-means++

K=256, benchmarking algorithm, k-means++

We can clearly see the degradation effect. Also degradation is much more in case of K=64 than K=256 for a particular algorithm which can be thought of as intuitively also. Moreover, when K is same, degradation is more in case of benchmarking algorithm than k-means algorithm.

1.b.
We now change the default parameters and set them to “random” and to “1” trial than “k-means++” and “10”. The output images are as follows: (next page)

K=64, k-means algorithm, random initialisation, 1 trial



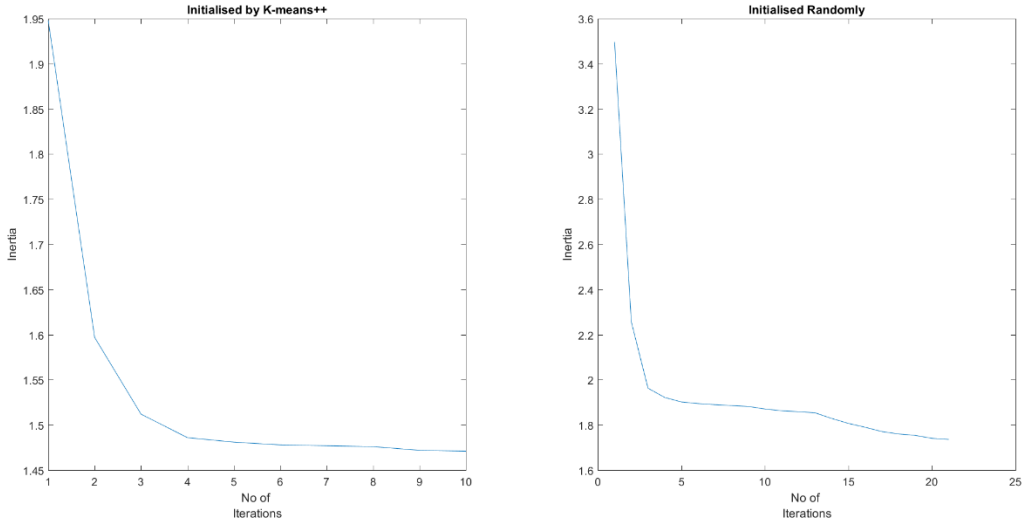
K=64, benchmarking algorithm, random initialisation, 1 trial

K=256, k-means algorithm, random initialisation, 1 trial

K=256, benchmarking algorithm, random initialisation, 1 trial

We observe now that when we do random initialisation and just 1 trial, segmentation is worse than what we had for k-means++ initialisation and 10 trial. And this is true for both K=64 and K=256.

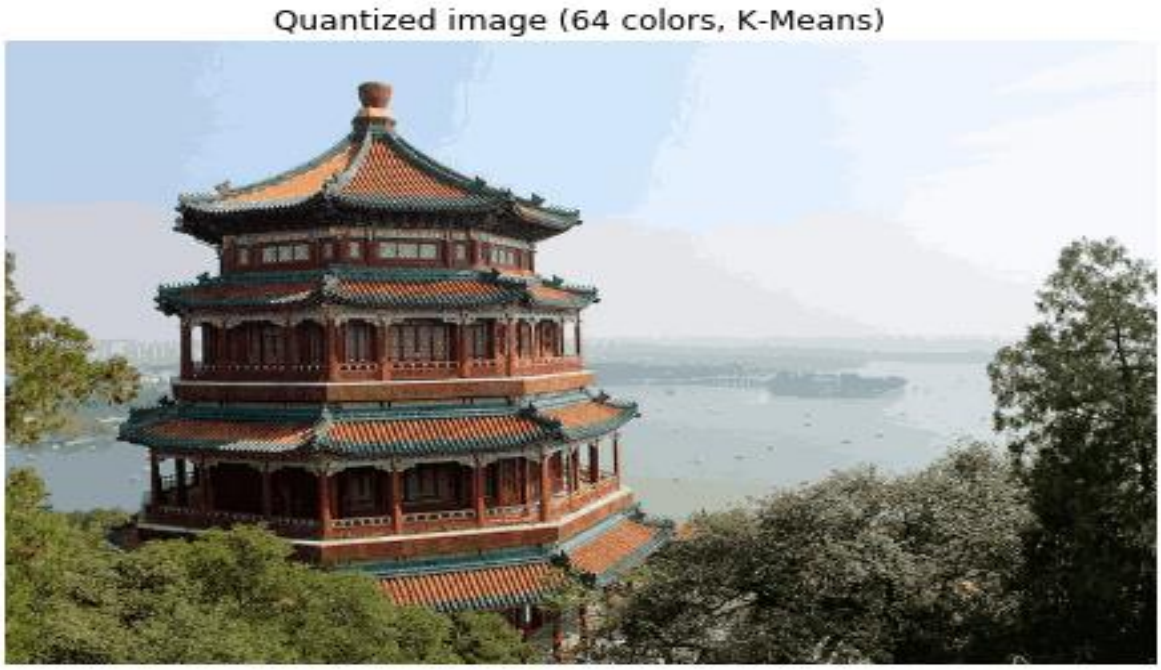
The error plots for "k-means++" and "random" initialisation with 1 trial are as follows:



We clearly see that the k-means algorithm with "k-means++" initialisation converges much faster than with "random" initialisation. This is expected as the seeds are placed in a much better manner in case of "k-means++" than "random" initialisation.

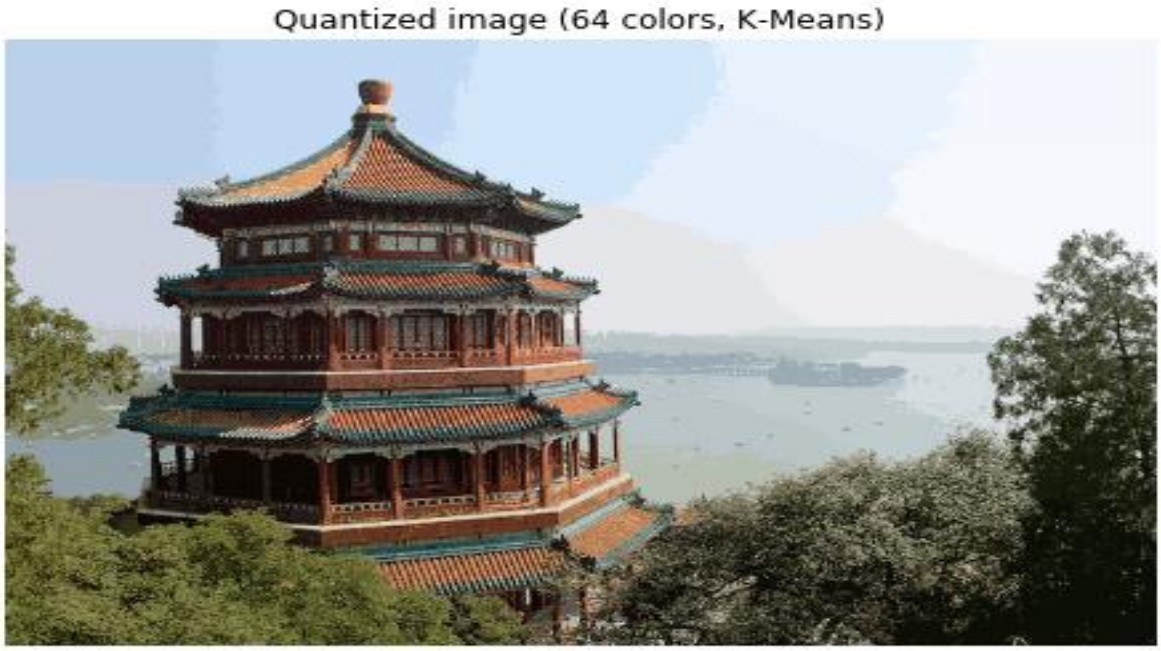
2.
We now also consider the pixel-coordinates along with pixel intensities to find out better labels. The results are as follows:

Weight = 0.25, k-means algorithm



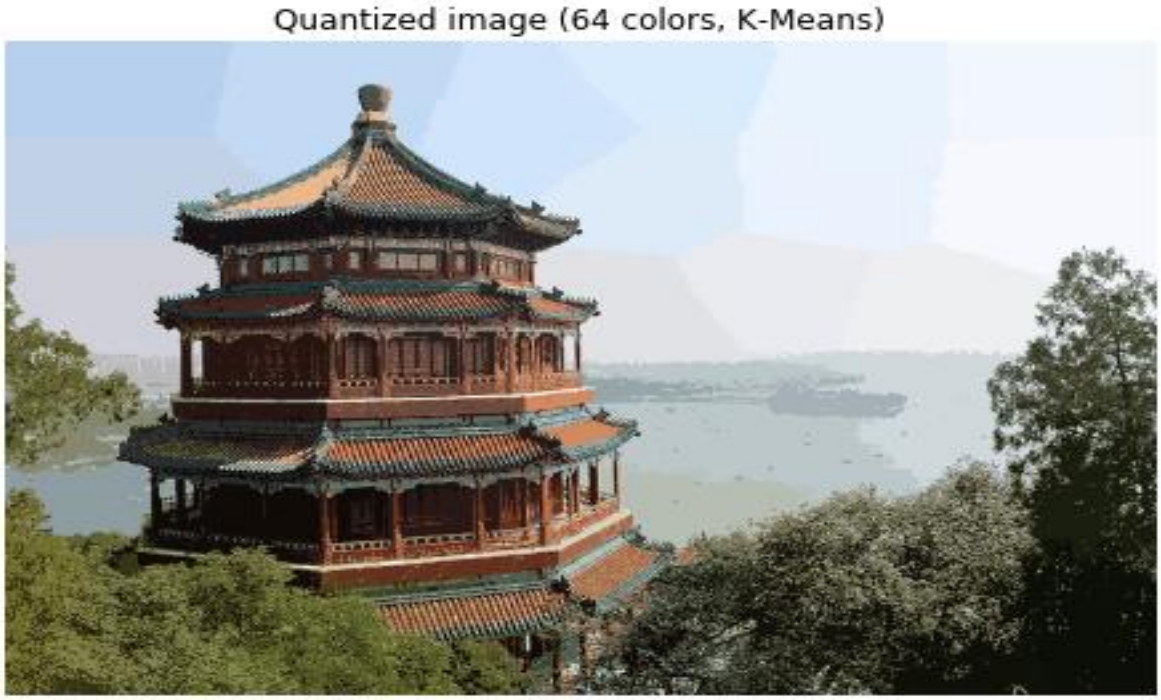
Weight = 0.25, benchmarking algorithm

Weight = 0.5, k-means algorithm

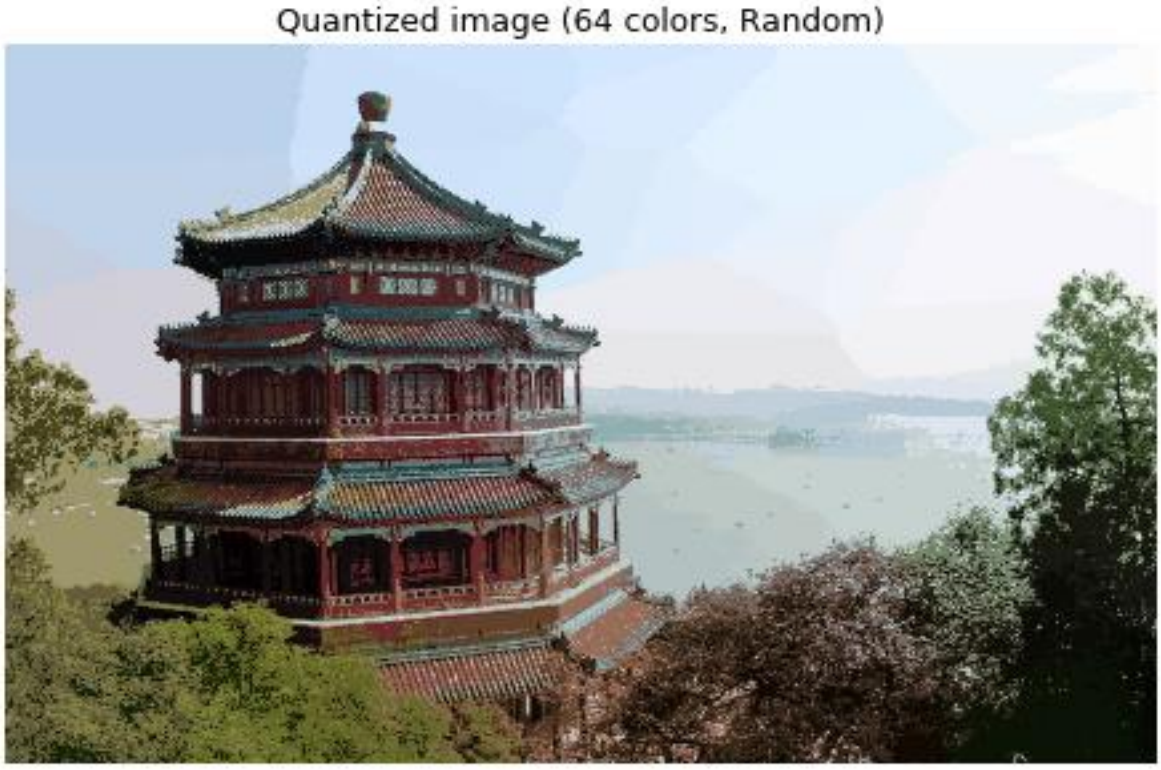


Weight = 0.5, benchmarking algorithm

Weight = 1, k-means algorithm

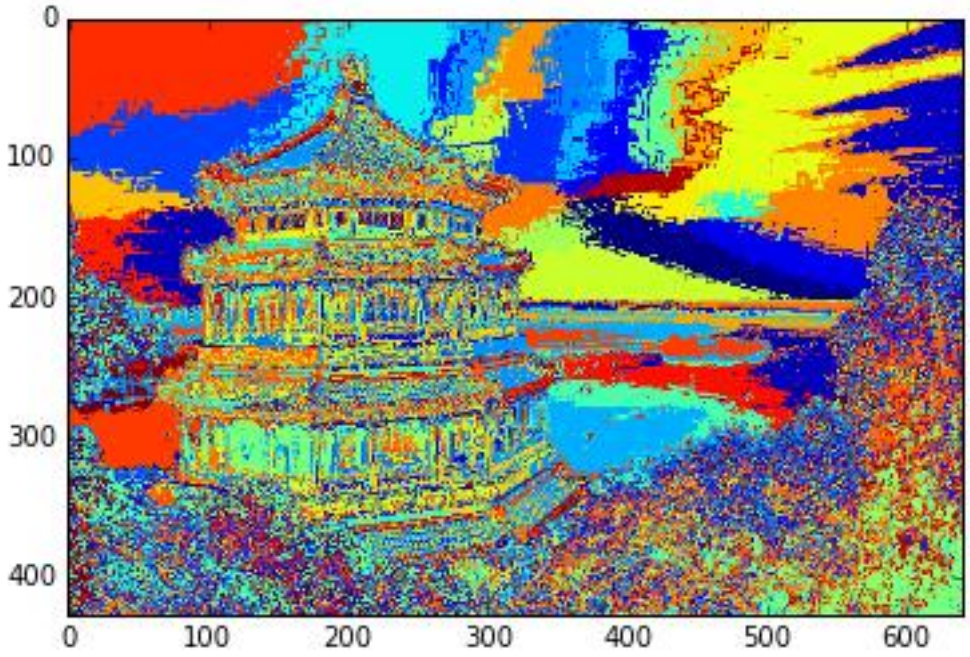


Weight = 1, benchmarking algorithm

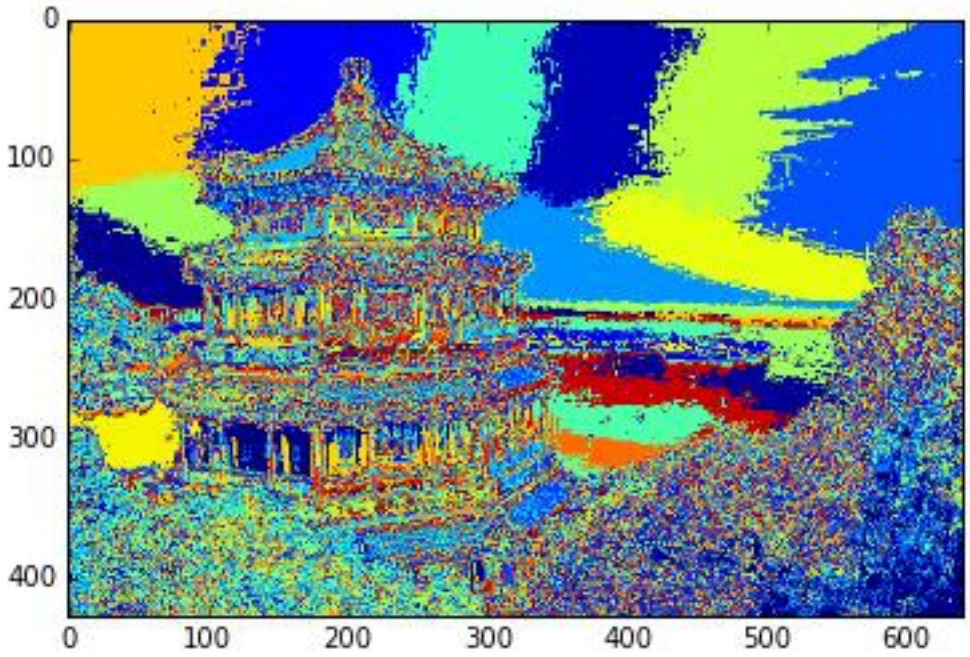


Thus, we see that the results are much better when we use another feature for calculating the labels which is visible specifically in the sky and lake areas. But we also notice that when we increase the weight to 1, some of the nuances of the original image are lost.

Codebook k-means:



Codebook benchmark



: