Image compression

Image compression refer to reducing the dimensions, pixels, or color components of an image so as to reduce the cost of storing or performing operations on them. Some image compression techniques also identify the most significant components of an image and discard the rest, resulting in data compression as well.

Image compression can therefore be defined as a data compression approach that reduces the data bits needed to encode an image while preserving image details.

Listed below are some of the potential applications of image compression:

1. Data takes up lesser storage when compressed. As a result, this allows more data to be stored with less disk space. This is especially useful in healthcare, where medical images need to be archived, and dataset volume is massive.
2. Some image compression techniques involving extracting the most useful components of the image (PCA), which can be used for feature summarization or extraction and data analysis.
3. Federal government or security agencies need to maintain records of people in a pre-determined, standard, and uniform manners. Thus, images from all sources need to be compressed to be of one uniform shape, size, and resolution.

Lossy Compression

Lossy compression, as its name suggests, is a form of data compression that retains useful image details while discarding a few bits in order to reduce its size or to extract important components. Thus, in lossy compression, data is irreversibly lost and the original image cannot be completely re-created.

Lossless Compression

Lossless compression is a form of data or image compression under which any sort of data loss is avoided, and thus, compressed images are larger in size. However, the original image can be re-constructed using this kind of image compression.

Image Compression using principal component analysis

Principal component analysis(PCA) is a technique for *feature extraction* — so it combines our input variables in a specific way, at which point we can drop the least important variables while still retaining the most valuable parts of all of the variables. PCA results in the development of new features that are independent of one another.

Briefly listing the steps of PCA below:

1. Scale the data by subtracting the mean and dividing by std. deviation.
2. Compute the *covariance matrix*.
3. Compute *eigenvectors* and the corresponding *eigenvalues*.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues, with these becoming the principal components**.**
5. Derive the new axes by re-orientation of data points according to the principal components.

The principal components, when sorted in the order of their eigenvalues, preserve most of the information in the dataset within the first principal component, the rest in the second, and so on.

Thus, we can preserve most of the details in an image by applying PCA. We can detect the number of principal components required to preserve variance by a certain percentage—say, 95% or 98%—and then apply PCA to transform the data space.

All the steps mentioned above will be followed in the same manner, and in the end we can create the compressed image by using this transformed data space.

It's very space-efficient, since an image with n*m pixels or dimensions (say 28*28=784) can be preserved by a very small number of principal components (just around 20–30).

Image compression using k-means clustering

K-means clustering is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (k), which are represented by their centroids.

Briefly listing the steps of k-means clustering below:

1. We first choose k initial centroids, where k is a user-specified parameter; namely, the number of clusters desired.
2. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is called a cluster.
3. The centroid of each cluster is then updated based on the points assigned to the cluster.
4. We repeat the assignment and update steps until no point changes clusters, or similarly, until the centroids remain the same.

We pre-define the value of k as the number of color components that we want to preserve in the image. The rest of the k-means algorithm is performed according to the above-mentioned steps.

With an increase in the value of k, as the number of clusters increases, the image will get closer and closer to the original image, but at the cost of more disk space for storage and a higher computational cost. We can experiment with the values of k to get desirable results.

We can also calculate the within-cluster sum of squared error to gain insight on whether the clusters are well fitted and correctly assigned or not, since it provides us with the variance of the cluster centroids.