#Problem 2

The Problem is to reconstruct face images using principal component analysis (PCA). PCA is a dimensionality reduction technique that identifies the most important patterns or components in the data, allowing us to represent the data in a lower-dimensional space while preserving as much of the original information as possible.

Methodology

Data Preprocessing: First computed the mean image across all the images in the dataset and subtracted it from each image. This step ensured that the data was centered around zero, which is a common requirement for PCA.

Computing the Covariance Matrix: The covariance matrix R was computed as R = (X^T * X) / N, where N is the number of images. This matrix captures the covariance between all pairs of pixels in the images.

Performing PCA: Performed PCA on the covariance matrix R by computing its eigenvalues and eigenvectors. The eigenvectors (U) represent the principal components, and the corresponding eigenvalues (Σ) represent the amount of variance explained by each principal component.

Selecting Principal Components: Analyzed the eigenvalues to determine how many principal components to retain for reconstruction. This was done by plotting the cumulative explained variance ratio and setting a threshold (e.g., 95%) to select the number of components that capture a certain percentage of the total variance.

Image Reconstruction: Reconstructed the images by projecting the original data onto the selected principal components and then adding back the mean image. This process resulted in approximations of the original images using a reduced number of dimensions (principal components).

Results

Reconstruction Image: Visualized a few of the reconstructed images alongside the original images.

Reconstruction Error: Computed the reconstruction error as the mean of the L2 norms between the original and reconstructed images. The error decreased as more principal components were included in the reconstruction, indicating a better approximation of the original data.

Eigenvalue Analysis: By adjusting the threshold for selecting the number of principal components, we observed the trade-off between reconstruction quality and dimensionality

reduction. Higher thresholds (retaining more components) resulted in lower reconstruction errors but less dimensionality reduction.

Clustering: Clustered the data using the PCA coefficients. While the clustering did not perfectly separate all images of the same individual, we observed some degree of clustering, suggesting that the PCA coefficients captured relevant information for distinguishing between different individuals.

Conclusion

Principal component analysis proved to be an effective technique for reconstructing face images while reducing the dimensionality of the data. By selecting an appropriate number of principal components, we could balance the trade-off between reconstruction quality and dimensionality reduction. The clustering results indicated that the PCA coefficients captured relevant features for distinguishing between different individuals.

#Problem 3

Implemented EM algorithm for a mixture of Gaussian models based on color features to segment the peppers_color.tif image. The goal is to segment the image into different regions or clusters based on the pixel colors.

Implementation:

The implementation begins by loading the peppers_color.tif image using the tifffile library and reshaping it into a feature matrix X, where each row represents a pixel, and each column represents a color channel (red, green, and blue). Set the number of Gaussian components (K) to 3 and initialize the means, covariances, and weights using K-means clustering.

The EM algorithm is then implemented through an iterative process consisting of two main steps: the Expectation (E) step and the Maximization (M) step. In the E-step, calculate the responsibilities (or posterior probabilities) of each data point (pixel) belonging to each Gaussian component. This is done by evaluating the Gaussian probability density function (PDF) for each component and normalizing the results.

In the M-step, update the parameters of the Gaussian mixture model based on the responsibilities calculated in the E-step. Specifically, we update the weights (mixing coefficients) as the sum of responsibilities for each component divided by the total number of data points. The means are updated as the weighted sum of the data points, using the responsibilities as weights. The covariances are updated as the weighted sum of the outer products of the deviations from the means, again using the responsibilities as weights.

The E-step and M-step are repeated iteratively until convergence, which is determined by monitoring the log-likelihood of the data under the current model parameters. The algorithm stops when the change in log-likelihood between successive iterations falls below a predefined threshold.

After convergence, the segmentation is performed by assigning each pixel to the Gaussian component with the highest responsibility (posterior probability). The resulting segmentation is visualized using matplotlib, with different colors representing different segments or clusters.

Results:

The implementation of the EM algorithm for the mixture of Gaussian models based on color features successfully segments the peppers_color.tif image. The resulting segmentation image clearly distinguishes different regions or clusters based on the pixel colors.

Conclusion:

In summary, the implementation of the EM algorithm for a mixture of Gaussian models based on color features successfully segments the peppers_color.tif image into different regions or clusters.

#Problem 4

This problem explored two distinct approaches to image segmentation: DeepLabv3, a state-of-the-art deep learning model for semantic segmentation, and the Expectation-Maximization (EM) algorithm combined with k-means clustering, a classic unsupervised learning technique. The objective was to apply these methods to the iconic "Lena" color image and compare their segmentation outputs.

Methodology:

DeepLabv3:

- DeepLabv3 is a powerful convolutional neural network architecture designed for semantic segmentation tasks.
- Utilized a pre-trained version of the DeepLabv3 model from the PyTorch library, which had been trained on a large dataset of labeled images.
- The Lena image was preprocessed and fed into the DeepLabv3 model, which produced a segmentation map as output.

EM Algorithm:

- First employed k-means clustering to partition the Lena image into a specified number of clusters (in this case, 5 clusters).
- These initial clusters were then used to initialize the EM algorithm, an iterative technique for estimating the underlying probability distributions that best explain the image data.

- The EM algorithm iteratively refined the cluster assignments and estimated the parameters of the probability distributions until convergence.
- The final segmentation map was obtained by assigning each pixel to the cluster with the highest responsibility.

Results:

DeepLabv3 Segmentation: DeepLabv3 accurately identified and segmented the central figure from the background.

EM Algorithm Segmentation: The EM algorithm segmented the entire image into multiple regions, each with distinct color and texture characteristics.

Conclusion:

This problem is a comparison of the contrasting strengths of the two segmentation approaches. DeepLabv3, as a deep learning model trained on labeled data, excelled at identifying and isolating semantically meaningful objects, such as the central figure in the Lena image. On the other hand, the EM algorithm, being an unsupervised technique, provided a more comprehensive, data-driven segmentation of the entire image based on low-level image statistics.

#Problem 5

The Fourier Transform has long been a fundamental concept in the field of image processing, enabling the decomposition of images into their constituent frequency components. The Fourier Style Transfer, allows for the stylistic elements of one image (the target image) to be transferred onto another image (the source image), resulting in a new image that combines the content of the source with the visual style of the target.

Methodology:

The core principle behind Fourier Style Transfer is rooted in the ability of the Fourier Transform to separate an image into its low-frequency components, representing the overall structure and content, and high-frequency components, representing the fine details and textures. By manipulating these frequency components, it becomes possible to blend the low-frequency components of the source image with the high-frequency components of the target image.

The provided code implements this technique through the following key steps:

- Convert the source and target images to grayscale.
- Compute the 2D Fourier Transform of the grayscale images.
- Separate the Fourier Transforms into amplitude and phase components.
- Create a mask to isolate the low-frequency components of the source image's amplitude spectrum.

- Resize the target image's amplitude spectrum to match the source image's dimensions.
- Swap the low-frequency components of the source image's amplitude spectrum with the corresponding components from the target image's amplitude spectrum, using the mask.
- Reconstruct the image by combining the swapped amplitude spectrum with the original phase spectrum of the source image.
- Compute the inverse Fourier Transform to obtain the final styled image.

Results:

The provided code takes a source image and a target image as input, and outputs a new image that combines the content of the source image with the visual style of the target image. This is achieved by swapping the low-frequency components of the source image's amplitude spectrum with those of the target image's amplitude spectrum, while preserving the high-frequency components of the source image.

Conclusion:

The Fourier Style Transfer technique provides a novel approach to combining the content of one image with the visual style of another, leveraging the power of the Fourier Transform and frequency domain manipulations. By selectively swapping the low-frequency components of the amplitude spectra, this method allows for the transfer of stylistic elements while preserving the overall content and structure of the source image.