**PRACTICAL ACTIVITY: IMAGE COMPRESSION USING PCA**

1. **Step-by-Step Solution: Loading the Image Dataset**

First, you need a simple image dataset. For this activity, we will assume we are using a **grayscale image** stored in a .csv file, where each pixel's intensity (0-255) is represented as a feature in the matrix.

**Example Dataset**: Imagine a 32x32 pixel grayscale image. The .csv file might look like this: Download the image datasets here: <https://github.com/ayots/COM618/blob/main/image_data.csv>

1. **Reading and Visualizing the Image**

First, we need to read the dataset and visualize the original image.

**Python Code to Load and Visualize the Image:**

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# Load the image dataset from CSV file (assuming the dataset is in image\_data.csv)  
data = pd.read\_csv('image\_data.csv', header=None)  
  
# Select the first image (row 0) and reshape it to 32x32 pixels  
image\_data = data.iloc[0].values.reshape(32, 32)  
  
# Plot the original image  
plt.figure(figsize=(4, 4))  
plt.imshow(image\_data, cmap='gray')  
plt.title('Original Image')  
plt.axis('off')  
plt.show()

1. **Applying PCA for Dimensionality Reduction**

Now, let's apply PCA to reduce the dimensionality of the image and compress it.

**Python Code to Apply PCA:**

from sklearn.decomposition import PCA  
  
# Flatten the image into a 1D array of 1024 pixels (for PCA input)  
flat\_image = image\_data.flatten()  
  
# Apply PCA, specifying how many principal components to keep  
pca = PCA(n\_components=100) # Keep only the top 100 components  
image\_pca = pca.fit\_transform(flat\_image.reshape(1, -1))  
  
# Inverse the PCA to reconstruct the image  
reconstructed\_image = pca.inverse\_transform(image\_pca)  
  
# Reshape the reconstructed image back to 32x32 for visualization  
reconstructed\_image = reconstructed\_image.reshape(32, 32)  
  
# Plot the compressed/reconstructed image  
plt.figure(figsize=(4, 4))  
plt.imshow(reconstructed\_image, cmap='gray')  
plt.title('Compressed Image with PCA (100 components)')  
plt.axis('off')  
plt.show()

**What you have done - Explanation:**

* We flatten the 32x32 image into a 1D array of 1024 pixels because PCA operates on 1D data.
* PCA is applied, and we keep only 100 components (meaning we're reducing the dimensionality of the image from 1024 to 100).
* After applying PCA, we reconstruct the image by performing the inverse PCA transformation.
* The reconstructed image is reshaped back to 32x32, and we plot it.

1. **Comparing the Original and Compressed Images**

Now, let's compare the original image and the compressed image side by side to see the difference. TASKS TO BE DONE.

**4A. SOLUTION**

# Plot side by side

fig, ax = plt.subplots(1, 2, figsize=(8, 4))

# Original image

ax[0].imshow(image\_data, cmap='gray')

ax[0].set\_title('Original Image')

ax[0].axis('off')

# Compressed image

ax[1].imshow(reconstructed\_image, cmap='gray')

ax[1].set\_title('Compressed Image (100 components)')

ax[1].axis('off')

plt.show()

**What the code does:**

* We plot the original and compressed images side by side to visually inspect the quality of the compression.
* As the number of principal components decreases, more details from the original image are lost, but the overall structure is preserved.

**Finally:**

1. **Load and visualize** the original image.
2. **Apply PCA** to reduce the dimensionality of the image (compress it).
3. **Reconstruct the image** from the reduced data.
4. **Compare** the original and compressed images.