6CS012 - Artificial Intelligence and Machine Learning. Image Compression and Decompression using PCA.

Prepared By: Siman Giri {Module Leader - 6CS012}

Feb 23, 2025

——————— Worksheet - 1. ———————-

1 Instructions

This worksheet contains programming exercises based on the material discussed from the slides.This is a graded exercise and are to be completed on your own and is compulsory to submit.
Please answer the questions below using python in the Jupyter Notebook and follow the guidelines below:

- This worksheet must be completed individually.
- All the solutions must be written in Jupyter Notebook.

2 Getting Started with Image Processing with Python.

Introduction to Python Imaging Library(PIL)

2.1 Exercise - 1:

Complete all the Task.

1. Read and display the image.
   - Read the image using the Pillow library and display it.
   - You can also use matplotlib to display the image.
2. Display only the top left corner of 100x100 pixels.
   - Extract the top-left corner of the image (100x100 pixels) and display it using **NumPy and Array Indexing.**
3. Show the three color channels (R, G, B).
   - Separate the image into its three color channels (Red, Green, and Blue) and display them individually, labeling each channel as R, G, and B.{Using NumPy.}
4. Modify the top 100×100 pixels to a value of 210 and display the resulting image:
   - Modify the pixel values of the top-left 100×100 region to have a value of 210 (which is a light gray color), and then display the modified image.

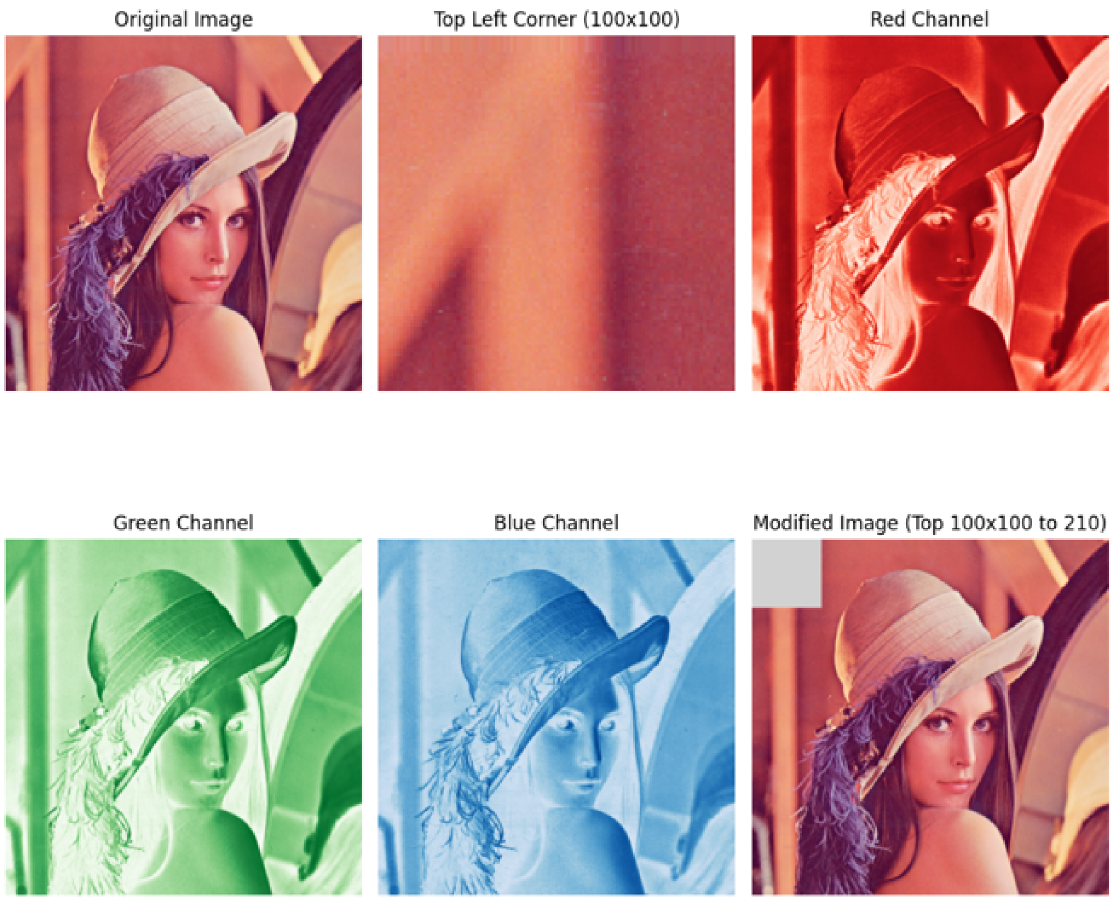
Original Image
Top Left Corner (100x100)
Red Channel
Green Channel
Blue Channel
Modified Image (Top 100x100 to 210)

Figure 1: Sample Output for Exercise - 1.

Cautions - Explanation:

The visible **red**, **green**, and **blue** backgrounds in the sample output are due to the use of the `cmap` argument in `matplotlib`. When displaying the individual color channels (Red, Green, Blue) using `imshow()` with specific colormaps such as `Reds`, `Greens`, and `Blues`, `matplotlib` visualizes each channel with those distinct color backgrounds to highlight them.

If you were to display the same channels using **Pillow (PIL)** directly, the output would be in **grayscale**, as `PIL` doesn't apply the color maps automatically. This is because the channels contain the intensity values for each color and don't have the full color representation unless we specifically map them to RGB or use a colormap as `matplotlib` does.

Summary:

- **Matplotlib with cmap:** Displays each channel with its respective color.
- **Pillow (PIL):** Displays the channels in grayscale, reflecting the intensity of each channel.

2.2 Exercise - 2:

Complete all the Task.

1. Load and display a grayscale image.
   - Load a grayscale image using the `Pillow` library.
   - Display the grayscale image using `matplotlib`.
2. Extract and display the middle section of the image (150 pixels).
   - Extract a 150 pixel section from the center of the image using **NumPy array slicing**.
   - Display this cropped image using `matplotlib`.
3. Apply a simple threshold to the image (e.g., set all pixel values below 100 to 0).
   - Apply a threshold to the grayscale image: set all pixel values below 100 to 0, and all values above 100 to 255 (creating a binary image).
   - Display the resulting binary image.
4. Rotate the image 90 degrees clockwise and display the result.
   - Rotate the image by 90 degrees clockwise using the `Pillow` `rotate` method or by manipulating the image array.
   - Display the rotated image using `matplotlib`.
5. Convert the grayscale image to an RGB image.
   - Convert the grayscale image into an RGB image where the grayscale values are replicated across all three channels (R, G, and B).
   - Display the converted RGB image using `matplotlib`.

Sample Output

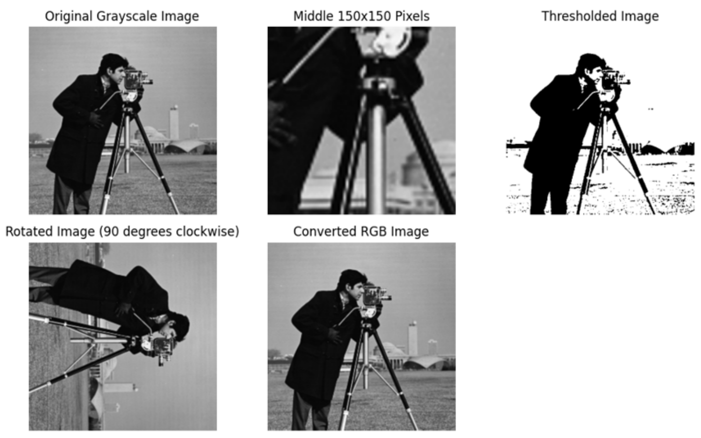

Figure 2: Sample Output Exercise - 2

3 Image Compression and Decompression using PCA.

In this exercise, build a PCA from scratch using explained variance method for image compression task. You are expected to compute the necessary matrices from the scratch. Dataset: Use image of your choice.

1. Load and Prepare Data:
   - Fetch an image of you choice.{If colour convert to grayscale}
   - Center the dataset - Standaridze the Data.
   - Calculate the covaraince matrix of the Standaridze data.
2. Eigen Decomposition and Identifying Principal Components:
   - Compute Eigen Values and Eigen Vectors.
   - Sort the eigenvalues in descending order and choose the top k eigenvectors corresponding to the highest eigenvalues.
   - Identify the Principal Components with the help of cumulative Sum plot.
3. Reconstruction and Experiment:
   - **Reconstruction:** Transform the original data by multiplying it with the selected eigenvectors(PCs) to obtain a lower-dimensional representation.
   - **Experiments:** Pick Four different combination of principal components with various explained variance value and compare the result.
   - Display the Results and Evaluate.

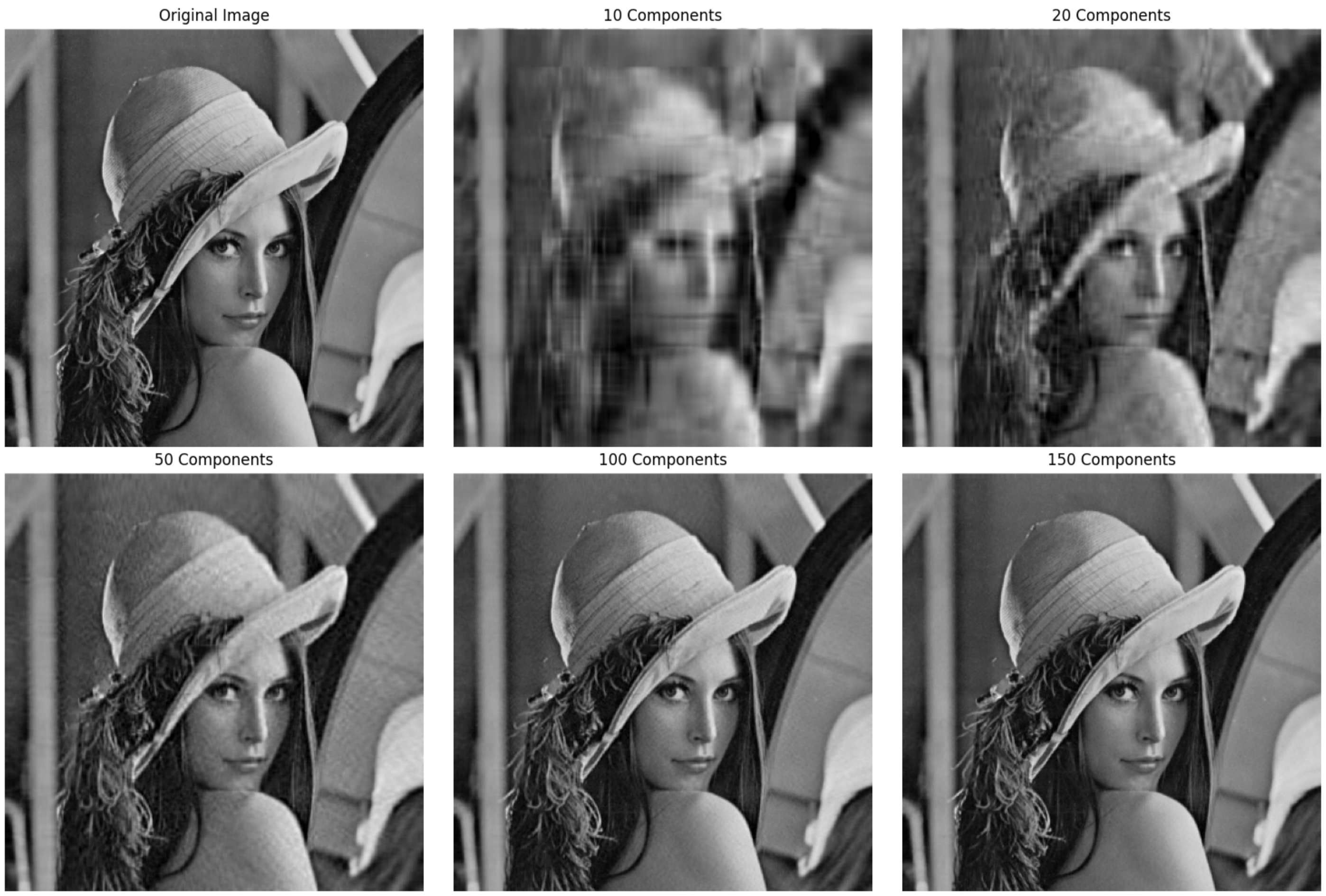

Figure 3: Sample Output for various principal Components.

——————— Fasten Your Seat Belt and Enjoy the Ride. ———————-