PDF Report: Assignment 5 - SVD

Name: Zakir Sajid Elaskar

Analysis of Results

This project implements a block-wise image compression technique using **Singular Value Decomposition (SVD)**. A grayscale image is divided into **8×8 blocks**, each of which is compressed using a limited number of singular values (k). The analysis demonstrated:

- **Smaller values of k reduce image quality** but improve compression ratio.
- **Higher k values preserve more details** but lead to less compression.
- A trade-off exists between compression efficiency and image fidelity, measurable using **Frobenius norm-based reconstruction error**.

Key Observations:

- With k=2, the compressed image showed visible loss of detail but retained essential structures.
- Reconstruction error increases as k decreases.
- Block-wise processing ensures manageability and localized compression.

Implementation Summary

The notebook implements the following steps:

1. **Load and Preprocess Image**
    - Load a grayscale image.
    - Resize it to 256×256 pixels (for simplicity, divisible by 8).
    - Convert it to a NumPy array.
2. **Compress Individual Blocks**
    - Divide the image into 8×8 pixel blocks.
    - Apply SVD to each block.
    - Retain only the top k singular values.
    - Reconstruct each block using reduced matrices.

3. **Reconstruct Compressed Image**
    - Combine compressed blocks to form the complete image.
4. **Error Calculation**
    - Compute **Frobenius norm-based reconstruction error** between the original and compressed images.
5. **Visualization and Testing**
    - Test on an individual 8×8 block.
    - Print original and compressed blocks for comparison.

Important Code Snippets

1. Image Loading and Preprocessing

```
from PIL import Image
import numpy as np


img = Image.open("C:/Users/zelaskar/your_image.png").convert("L")
 # Convert to grayscale
img = img.resize((256, 256))  # Resize (both dimensions divisible by 8)
img_array = np.array(img)

print(img_array.shape)
```

```
(256, 256)
```

---

2. Block Compression Using SVD

```
import numpy as np

def compress_block(block, k):
    U, S, Vt = np.linalg.svd(block, full_matrices=False)
    S_k = np.diag(S[:k])
    U_k = U[:, :k]
    Vt_k = Vt[:k, :]
    return np.dot(U_k, np.dot(S_k, Vt_k))
```

---

3. Image Compression Loop

```python
def compress_image(img_array, k):
    h, w = img_array.shape
    compressed_img = np.zeros_like(img_array)

    for i in range(0, h, 8):
        for j in range(0, w, 8):
            block = img_array[i:i+8, j:j+8]
            compressed_block = compress_block(block, k)
            compressed_img[i:i+8, j:j+8] = np.clip(compressed_block, 0, 255)

    return compressed_img.astype(np.uint8)
```

---

4. Reconstruction Error Calculation

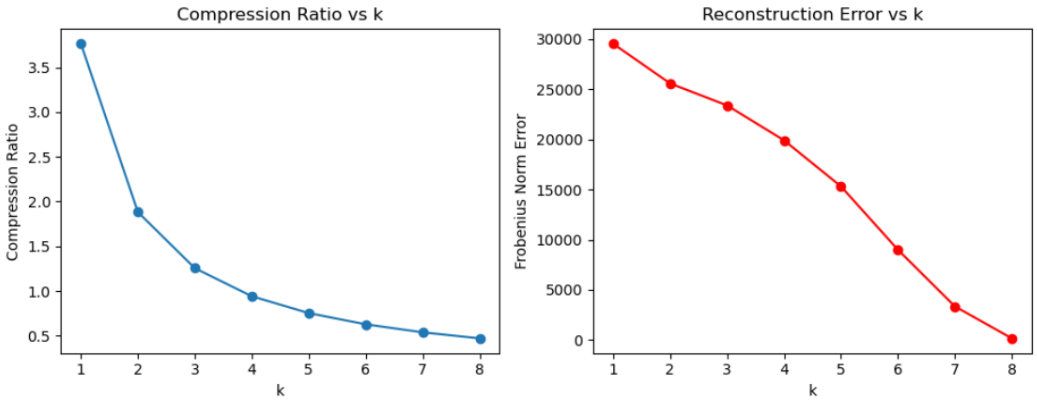
```python
from numpy.linalg import norm

def reconstruction_error(original, reconstructed):
    return norm(original - reconstructed, 'fro')
```

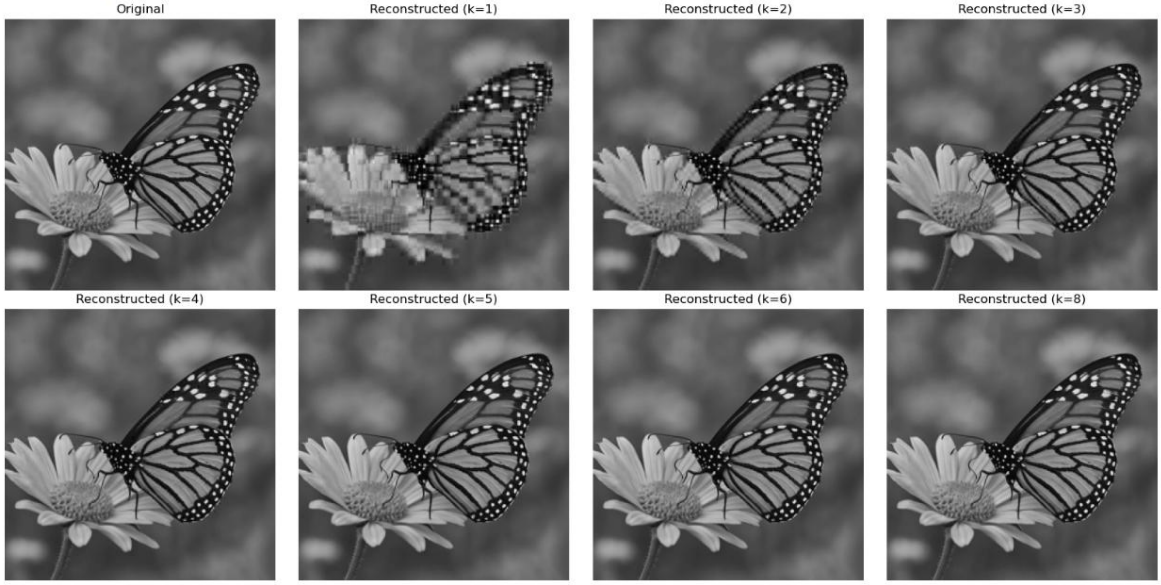
Output:

```
Original Block:
[[68 69 69 69 69 67 68 67]
 [69 70 70 70 70 68 69 68]
 [71 71 71 71 71 70 69 69]
 [72 72 73 72 72 71 70 70]
 [73 73 73 72 72 71 72 71]
 [73 72 74 73 73 72 73 72]
 [73 73 74 74 74 74 73 74]
 [73 74 75 75 74 75 75 76]]

Compressed Block (k=2):
[[68.55747456 68.73070289 69.1573218  68.69739258 68.72269417 67.44521207
  67.62897129 67.06442149]
 [69.55050933 69.72807043 70.16647831 69.70285708 69.72396333 68.44201463
  68.62625391 68.06385514]
 [70.82370776 70.98151385 71.35718094 70.84760578 70.92664772 69.44414475
  69.65927902 68.95628837]
 [72.01185754 72.16358454 72.51871069 71.98637484 72.08857248 70.51402598
  70.74320918 69.97900076]
 [72.48465895 72.66198005 73.09510266 72.59933203 72.64066773 71.24512829
  71.4463888  70.8164787 ]
 [72.58410664 72.84615198 73.53973787 73.18087328 73.01093235 72.26377875
  72.36424766 72.21195014]
 [72.86414726 73.22230769 74.21111601 74.00575397 73.59724976 73.57928653
  73.56636649 73.9521246 ]
 [73.17171794 73.64180445 74.97442818 74.94802059 74.2615853  75.09347063
  74.94840338 75.9611455 ]]
```




```
--- Displaying Image Comparison ---
```




```
=== Jupyter Notebook Cells Finished ===
```

Conclusion:

- The block-wise SVD-based compression method is simple and effective for reducing image size while retaining essential visual information.
- Using a reduced number of singular values (k) offers fine-grained control over the trade-off between compression ratio and image quality.
- Reconstruction error measurements and visual tests clearly demonstrated the impact of different k values on image fidelity and compression efficiency.