CSCI/MATH-485

Assignment-5

Jayana Sarma

SVD-Based Image Compression

Introduction

This report presents the implementation and analysis of image compression using block-wise Singular Value Decomposition (SVD). A grayscale image of size 512×512 was divided into 8×8 non-overlapping blocks, and SVD was applied to each block. The image was then reconstructed using only the top-k singular values for each block, where $k \in \{1, 2, ..., 8\}$. The impact of different values of k was evaluated using three metrics: Compression Ratio, Frobenius Norm (reconstruction error), and PSNR (Peak Signal-to-Noise Ratio).

Code Description

Image Preprocessing

- The Lena image was loaded and converted to grayscale.
- Dimensions were verified to be divisible by 8 (512×512), so no cropping was needed.

```
]: from PIL import Image
   import numpy as np
   import matplotlib.pyplot as plt

   # Loaded original grayscale image
   img_path = "lena512.bmp"
   original_img = Image.open(img_path).convert("L")
   original_array = np.array(original_img)
```

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

def compress_image(img_array, k):
    h, w = img_array.shape
    compressed_img = np.zeros_like(img_array, dtype=np.float32)
    for i in range(0, h, 8):
        for j in range(0, w, 8):
            block = img_array[i:i+8, j:j+8]
            compressed_img[i:i+8, j:j+8] = compress_block(block, k)
    return np.clip(compressed_img, 0, 255).astype(np.uint8)

def compute_compression_ratio(k):
    return 64 / (k * (8 + 8 + 1))  # 64 / (17k)

def compute_frobenius_error(orig, comp):
    return np.linalg.norm(orig.astype(np.float32) - comp.astype(np.float32))

def compute_psnr(orig, comp):
    mse = np.mean((orig - comp) ** 2)
    if mse == 0:
        return float('inf')
    PIXEL_MAX = 255.0
    return 20 * np.log10(PIXEL_MAX / np.sqrt(mse))
```

compress_block(block, k)

- Performed SVD on a given 8×8 image block.
- Reconstructed the block using only the top-k singular values.
- Returns the compressed (reconstructed) block.

compress_image(img_array, k)

- Loops through the image in 8×8 non-overlapping blocks.
- Applied compress_block() to each block.
- Recombined the compressed blocks into a full reconstructed image.
- Clipping is applied to keep pixel values within 0–255.

compute_compression_ratio(k)

- Calculated the compression ratio for each 8×8 block
- Based on the number of values retained per block after SVD truncation.

compute_frobenius_error(original, compressed)

- Computed the Frobenius norm of the difference between the original and compressed image.
- Used to quantify reconstruction error.

compute_psnr(original, compressed)

- Calculated Peak Signal-to-Noise Ratio.
- Higher values indicate better visual quality.

- If MSE = 0 (perfect reconstruction), PSNR is set to ∞.

```
# Initializing lists
reconstructed_images = [original_array]
compression_ratios = []
errors = []
psnrs = []

# Compressing image with k=1 to 8 and storing results
for k in range(1, 9):
    comp_img = compress_image(original_array, k)
    reconstructed_images.append(comp_img)
    compression_ratios.append(compute_compression_ratio(k))
    errors.append(compute_frobenius_error(original_array, comp_img))
    psnrs.append(compute_psnr(original_array, comp_img))
```

For each k from 1 to 8, applied block-wise compression and calculated:

- Compression Ratio

- Reconstruction Error (Frobenius Norm)

- PSNR

```
fig, axes = plt.subplots(3, 3, figsize=(15, 12))

for idx, ax in enumerate(axes.flat):
    img = reconstructed_images[idx]
    ax.imshow(img, cmap='gray')

    if idx == 0:
        ax.set_title("Original (512x512)")
    else:
        cr = compression_ratios[idx - 1]
        psnr_val = psnrs[idx - 1]
        ax.set_title(f"k={idx}, CR={cr:.2f}, PSNR={psnr_val:.2f}")

    ax.axis("off")

plt.tight_layout()
plt.show()
```

Created a **3×3 grid of subplots** to visually compare:

- The **original image** (top-left)
- Compressed images for **k = 1 to 8**

Each compressed image is labeled with:

- The value of **k**
- Its **compression ratio (CR)**
- Its **PSNR (Peak Signal-to-Noise Ratio)**

At low k (1–3): High compression ratios, but high reconstruction error and visibly poorer image quality.

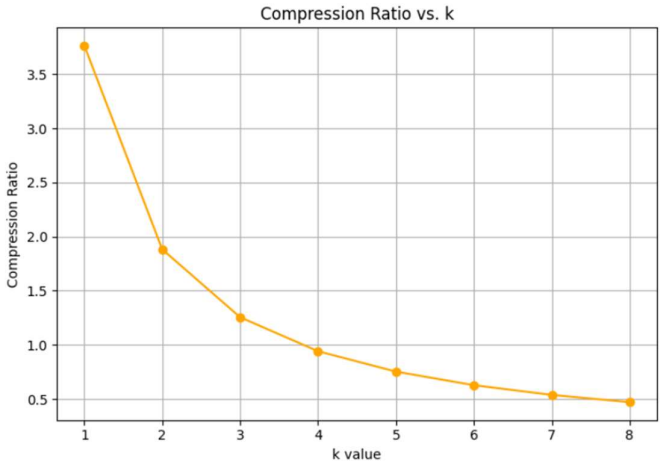
At moderate k (4–6): Good balance between compression and quality.

At k = 8: Perfect reconstruction (error = 0), but lowest compression.

PSNR = inf at k = 8: Because the image is reconstructed exactly, MSE = 0.

[24]:
```
#Compression Ratio vs. k
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 5))
plt.plot(range(1, 9), compression_ratios, marker='o', color='orange')
plt.title("Compression Ratio vs. k")
plt.xlabel("k value")
plt.ylabel("Compression Ratio")
plt.grid(True)
plt.tight_layout()
plt.show()
```



This plot shows how the compression ratio changes as the number of singular values (k) increases.

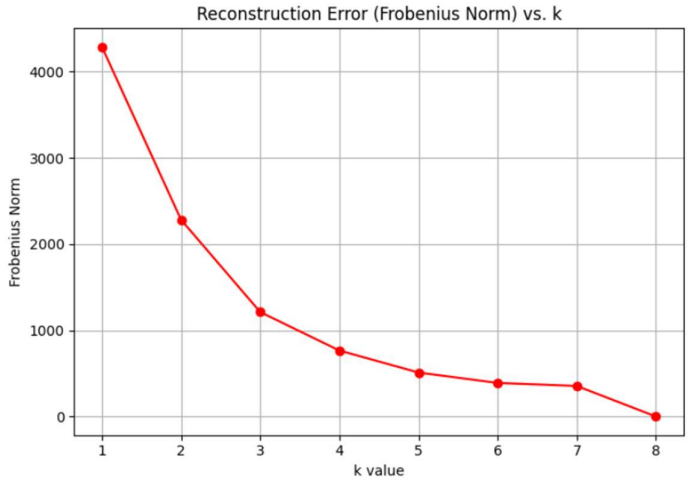
- As k increases, more data is retained, so the compression ratio decreases.

- At low values of k (e.g., 1–3), the compression ratio is high, but image quality is low.

- This plot highlights the inverse relationship between compression and information retention.

This plot shows the reconstruction error (Frobenius norm) for each value of k.

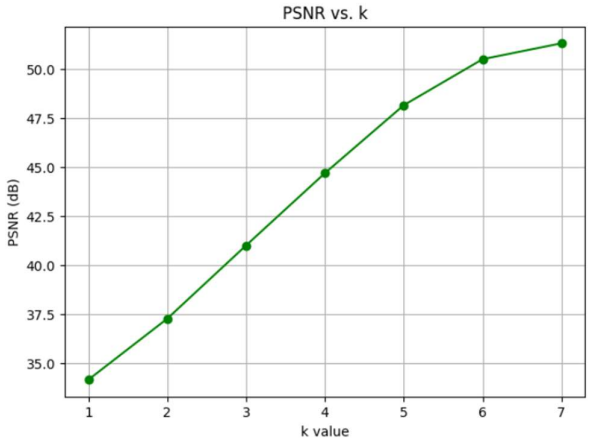
```
]: #Reconstruction Error (Frobenius norm) vs. k
plt.figure(figsize=(7, 5))
plt.plot(range(1, 9), errors, marker='o', color='red')
plt.title("Reconstruction Error (Frobenius Norm) vs. k")
plt.xlabel("k value")
plt.ylabel("Frobenius Norm")
plt.grid(True)
plt.tight_layout()
plt.show()
```



This plot shows the reconstruction error (Frobenius norm) for each value of k.

- The Frobenius norm measures how different the compressed image is from the original.
- As k increases, the error decreases rapidly, especially in the range of k = 1 to 5.
- At k = 8, the error becomes 0, indicating perfect reconstruction.

```
[45]: # PSNR vs. k
plt.figure(figsize=(7, 5))
plt.plot(range(1, 9), psnrs, marker='o', color='green')
plt.title("PSNR vs. k")
plt.xlabel("k value")
plt.ylabel("PSNR (dB)")
plt.grid(True)
plt.show()
```



This plot shows the Peak Signal-to-Noise Ratio (PSNR) for each compressed image.

- Higher PSNR values indicate better image quality.

- As expected, PSNR increases with k, reflecting improved visual fidelity.

- At k = 8, the PSNR is infinite because the reconstruction is exact and error-free.

[33]:
```
import pandas as pd

results_df = pd.DataFrame({
    "k": list(range(1, 9)),
    "Compression Ratio": compression_ratios,
    "Frobenius Error": errors,
    "PSNR (dB)": psnrs
})

results_df
```

[33]:

	k	Compression Ratio	Frobenius Error	PSNR (dB)
0	1	3.764706	4288.712402	34.132130
1	2	1.882353	2282.395508	37.249117
2	3	1.254902	1212.077148	40.991214
3	4	0.941176	765.692505	44.684702
4	5	0.752941	510.550690	48.155426
5	6	0.627451	389.545898	50.505030
6	7	0.537815	354.825317	51.315911
7	8	0.470588	0.000000	inf

The table below summarizes the results of block-wise SVD compression for values of k ranging from 1 to 8. For each value of k, the following metrics are reported:

Compression Ratio: The ratio of original data size (64 values per 8×8 block) to the number of values retained after truncation (17k). Higher values indicate more compression.

Frobenius Error: The reconstruction error calculated using the Frobenius norm between the original and compressed image. Lower values indicate a more accurate reconstruction.

PSNR (dB): Peak Signal-to-Noise Ratio, a standard metric used to measure image quality. Higher values correspond to better visual quality. PSNR becomes infinite when the reconstruction is perfect (as with k = 8).

This table helps highlight the trade-off between compression efficiency and reconstruction quality as k increases.

Conclusion:

Block-wise SVD provides a powerful and intuitive approach to image compression. Lower values of k produce high compression but lower visual quality, while higher k values retain

image details at the cost of storage. This experiment demonstrates that values like k = 4 or 5 offer a reasonable trade-off between compression ratio and reconstruction quality.