Using Singular Value Decomposition for Image Compression

Jay Menon[1]
[1]School of Manufacturing Systems and Networks, Arizona State University, Arizona, USA

Article Info

Keywords:

Singular Value Decomposition
Image Processing

ABSTRACT

This project investigates the use of an important concept in Linear Algebra: Singular Value Decomposition (SVD) in Python image processing. The goal is to find out how to effectively analyze and reconstruct images using SVD which also helps in memory management for edge computing. SVD is performed, image data is loaded and preprocessed, and the image is then rebuilt using a subset of singular values. The effect of changing the quantity of singular values that are kept is assessed on the quality of the image. The results offers a quick insight into the possible uses of SVD and advance our understanding of its usefulness in image analysis.

1. OBJECTIVE

The project's goal is to investigate and illustrate the use of Singular Value Decomposition (SVD) in image processing on a bird image using Python. Our goal is to reduce the dimensionality of the image data by applying SVD. Using a subset of singular values, we are also reconstructing images.

2. METHOD

Let A be the original matrix (image), and U, S, V^T be the matrices obtained from the Singular Value Decomposition.

$$\mathbf{A = U\,S\,V^T}$$

U is an m × m orthogonal matrix,
Σ is an m × n diagonal matrix with singular values on the diagonal,
V^T is an n × n orthogonal matrix

After importing necessary libraries, check if the loaded image is in color (3D array), and if so, convert it to grayscale by taking the mean along the last axis. Apply the svd function to the image matrix, obtaining three matrices: U, S, V^T. 's_v' variable holds the number of singular values that must be retained. Finally reconstruct the image.

3. CODE IMPLEMENTATION

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import svd

image = plt.imread("C:/Users/menon/OneDrive - Arizona State
University/OpenCV/final_project/bird.jpeg")

# Convert the image to grayscale if it's a color image
if len(image.shape) == 3:
    image = np.mean(image, axis=-1)
```

```
# Perform Singular Value Decomposition
U, S, Vt = svd(image, full_matrices=False)

# Reconstruct the image
s_v = 10  # No of singular values to retain
reconstructed_image = U[:, :s_v] @ np.diag(S[:s_v]) @ Vt[:s_v, :]

# Compare
plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title(f'After applying SVD where (s_v = {s_v})')
plt.imshow(reconstructed_image, cmap='gray')
plt.axis('off')

plt.show()
```

4. MEDIA

5. CONCLUSION

The experiment demonstrated the effectiveness of SVD as a linear algebraic tool for breaking down matrices into understandable parts. The image matrix's factorization into U, S, and V^T showed how linear algebra can be used to extract crucial information from intricate data structures. Using SVD to reduce dimensionality and then reconstruct images demonstrated how useful linear algebra is for managing high-dimensional datasets.

REFERENCES

[1] Salih Marangoz blog, https://salihmarangoz.github.io/blog/Image-Compression-With-SVD/