

# TUGAS ALJABAR LINEAR | PENERAPAN SVD - IMAGE COMPRESSION

Disusun oleh : Zaidan Harith (23/512629/TK/56334)

---

## Landasan Teori

### A. Definisi Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) adalah metode aljabar linear yang memecah suatu matriks  $A$  yang berukuran  $m \times n$  menjadi tiga komponen :

- **Matriks  $U$**  : Matriks ortogonal yang berisi *eigenvector* kiri dari  $A^T A$  yang berukuran  $m \times m$ .
- **Matriks  $\Sigma$**  : Matriks diagonal berukuran  $m \times n$  yang berisi nilai singular dari  $A^T A$  diurutkan dari yang terbesar.
- **Matriks  $V$**  : Matriks ortogonal yang berisi *eigenvector* kanan dari  $A A^T$  yang berukuran  $n \times n$ .

SVD memiliki beberapa karakteristik yang membuatnya bermanfaat untuk melakukan kompresi pada gambar, seperti :

- **Dekomposisi kompak** : SVD dapat merepresentasikan matriks dengan jumlah kolom lebih sedikit dengan hanya menyimpan nilai singular yang signifikan.
- **Sifat ortogonal** : Matriks  $U$  dan  $V$  adalah ortogonal sehingga dapat memudahkan melakukan rekonstruksi pada gambar.
- **Ketahanan terhadap noise** : Nilai singular yang kecil biasanya mewakili *noise* dalam gambar dan dapat diabaikan untuk kompresi *lossy*.

### B. Langkah-langkah Kompresi Gambar Menggunakan SVD

1. **Membaca gambar** : Baca gambar digital dan representasikan sebagai matriks.
2. **Melakukan SVD** : Lakukan dekomposisi SVD pada matriks gambar.
3. **Memilih nilai singular** : Pilih nilai singular yang akan disimpan berdasarkan tingkat kompresi yang diinginkan. Nilai singular yang lebih kecil, yang mewakili informasi yang kurang penting, dapat diabaikan untuk mencapai kompresi lebih tinggi.

4. **Membangun matriks terkompresi** : Buat matriks baru yang terkompresi dengan menggunakan kolom yang terkait dengan nilai singular yang dipilih. Merekonstruksi gambar: Rekonstruksi gambar dari matriks terkompresi dengan mengalikan matriks  $U$ ,  $S$ , dan  $V^T$ .

## C. Kelebihan Menggunakan SVD untuk Kompresi Gambar

- **Efisiensi** : SVD dapat mencapai tingkat kompresi yang tinggi sambil mempertahankan kualitas gambar yang baik.
- **Ketahanan terhadap noise** : SVD dapat membantu menghilangkan *noise* dari gambar selama proses kompresi.
- **Kesederhanaan** : Algoritma SVD relatif mudah diimplementasikan.

## D. Kekurangan Menggunakan SVD untuk Kompresi Gambar

- **Kehilangan data** : Kompresi *lossy* dengan SVD akan menghilangkan sebagian informasi dari gambar asli.
- **Kompleksitas komputasi** : Perhitungan SVD dapat memakan waktu lama untuk gambar beresolusi tinggi.

## Source Code Program Kompresi Gambar Menggunakan Python

### 1. Memanggil *Library* yang dibutuhkan

```
In [41]: from PIL import Image  
import numpy as np  
import os
```

### 2. Masukkan gambar yang ingin dikompresi

Gambar yang diunggah akan ditampung ke dalam variabel yang bernama `gambarAsli`.

```
In [42]: gambarAsli = Image.open('gambar.jpg')  
gambarAsli
```

Out[42]:



### 3. Ubah gambar menjadi *grayscale* (Hitam Putih)

Gambar pada variabel `gambarAsli` diubah menjadi gambar *grayscale* dengan menggunakan *method* `.convert('L')` dan tampung hasil gambar yang baru ke dalam variabel `gambarGrayscale`. Gambar baru tersebut juga disimpan ke dalam file gambar baru bernama `gambar-grayscale.jpg`.

In [43]:

```
gambarGrayscale = gambarAsli.convert('L')
gambarGrayscale.save("gambar-grayscale.jpg")
```

### 4. Cetak ukuran gambar asli dan *grayscale*

Untuk mendapatkan ukuran gambar, dapat kita gunakan *method* dari *library* `os` yang bernama `os.path.getsize()`.

In [44]:

```
ukuranGambarAsli = os.path.getsize('gambar.jpg')
ukuranGambarGrayscale = os.path.getsize('gambar-grayscale.jpg')

print("Ukuran gambar asli : ", ukuranGambarAsli, " bytes")
print("Ukuran gambar grayscale : ", ukuranGambarGrayscale, " bytes")
```

Ukuran gambar asli : 4108897 bytes  
Ukuran gambar grayscale : 2302255 bytes

### 5. Konversi gambar *grayscale* menjadi sebuah matriks

Konversi gambar menjadi matriks dapat dilakukan dengan memanfaatkan *library* `numpy` yang telah didefinisikan sebagai `np`.

```
In [45]: matriksGambarGrayscale = np.array(gambarGrayscale)
matriksGambarGrayscale
```

```
Out[45]: array([[160, 163, 166, ..., 221, 221, 221],
   [164, 167, 172, ..., 221, 221, 221],
   [163, 168, 173, ..., 221, 221, 221],
   ...,
   [ 80,  85,  90, ...,  87,  90,  76],
   [ 90,  89,  86, ...,  70,  59,  74],
   [ 81,  89,  76, ...,  72,  64,  83]], dtype=uint8)
```

## 6. Memecah matriks gambar menjadi tiga komponen utama SVD, yaitu matriks $U$ , $\Sigma$ , dan $V^T$

```
In [46]: matriksU, matriksSigma, matriksVTranspose = np.linalg.svd(matriksGambarGrayscale, full_matrices=False)
print(matriksU.shape, matriksSigma.shape, matriksVTranspose.shape)

print("Matriks U = ", matriksU)
print("Matriks \Sigma = ", matriksSigma)
print("Matriks VTranspose = ", matriksVTranspose)
```

(3753, 3753) (3753,) (3753, 5630)

## 7. Lakukan kompresi pada gambar *grayscale* dengan cara memangkas matriks gambar dengan mengalikan dengan suatu nilai $k$

Gambar dapat dikompresi dengan cara mengalikannya dengan suatu nilai  $k$  yang merepresentasikan tingkat kompresi gambar. Semakin kecil nilai  $k$ , maka gambar akan semakin kecil ukurannya dan semakin buram. Nilai  $k$  sendiri berada di rentang  $0 \leq k \leq 255$ .

```
In [ ]: k = 50
matriksU_k = matriksU[:, :k]
matriksSigma_k = matriksSigma[:k]
matriksVTranspose_k = matriksVTranspose[:k, :]

print(matriksU_k.shape, matriksSigma_k.shape, matriksVTranspose_k.shape)
```

## 8. Bentuk matriks baru yang berupa matriks gambar hasil kompresi

Matriks gambar yang dibuat ditampung ke dalam variabel `matriksGambarTerkompresi`. Namun, sebelumnya, matriks  $\Sigma$  harus didiagonalisasi terlebih dahulu.

```
In [ ]: matriksSigma_k = np.diag(matriksSigma_k)

matriksGambarTerkompresi = np.dot(np.dot(matriksU_k, matriksSigma_k), matriksVTrans
matriksGambarTerkompresi
```

## 9. Konversi kembali matriks gambar yang baru menjadi sebuah gambar RGB

Untuk melakukan konversi matriks gambar menjadi gambar RGB, dapat kita gunakan *method* dari *library* `numpy` yang sudah didefinisikan sebagai `np` yang bernama `.clip()`, `.astype()`, dan `.fromarray()`.

```
In [ ]: matriksGambarTerkompresi = np.clip(matriksGambarTerkompresi, 0, 255)

#ubah menjadi ke type uint8
matriksGambarTerkompresi = matriksGambarTerkompresi.astype(np.uint8)

gambarTerkompresi = Image.fromarray(matriksGambarTerkompresi)
gambarTerkompresi
```

## 10. Simpan gambar RGB dan cetak ukuran gambar

Gambar yang sudah dikompresi dapat disimpan dengan menggunakan *method* `.save()`. Cetak ukuran file gambar asli, gambar *grayscale* dan gambar terkompresi sebagai perbandingan ukuran file sebelum dan sesudah dilakukan kompresi.

```
In [ ]: gambarTerkompresi.save('gambar-terkompresi.jpg')

print("Ukuran gambar asli :", os.path.getsize('gambar.jpg'), "bytes")
print('Ukuran gambar grayscale :', os.path.getsize('gambar-grayscale.jpg'), "bytes")
print('Ukuran gambar terkompresi :', os.path.getsize('gambar-terkompresi.jpg'), "by
```