

LAPORAN TUGAS BESAR
Aplikasi Nilai Eigen dan Vektor Eigen
dalam Kompresi Gambar

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2123 Aljabar Linier dan Geometri pada Semester I Tahun Akademik 2021/2022

Disusun oleh:

| | |
|------------------------------------|-----------------|
| Saul Sayers (K2) | 13520094 |
| Patrick Amadeus Irawan (K2) | 13520109 |
| Rania Dwi Fadhilah (K3) | 13520142 |



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

DAFTAR ISI

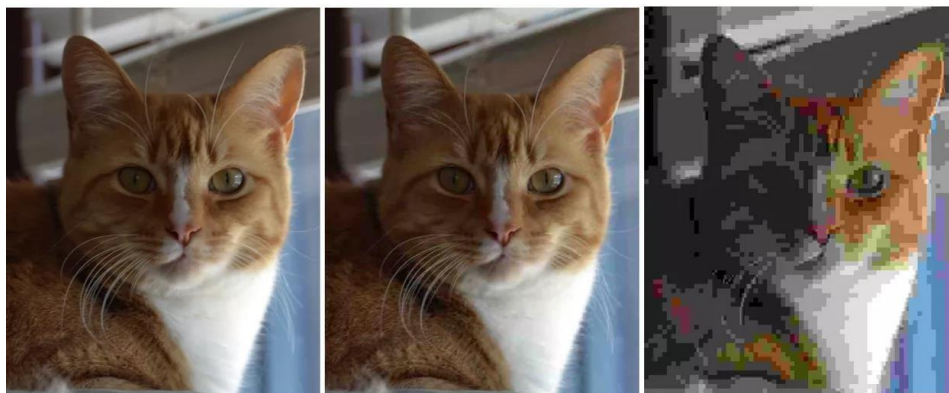
| | |
|---|-----------|
| DAFTAR ISI | i |
| BAB I DESKRIPSI MASALAH..... | 1 |
| BAB II TEORI SINGKAT | 5 |
| BAB III IMPLEMENTASI PROGRAM | 9 |
| BAB IV EKSPERIMEN | 15 |
| BAB V | 27 |
| I. Kesimpulan | 28 |
| II. Saran..... | 28 |
| III. Refleksi | 29 |
| REFERENSI | ii |

BAB I

DESKRIPSI MASALAH

Gambar adalah suatu hal yang sangat dibutuhkan pada dunia modern ini. Kita seringkali berinteraksi dengan gambar baik untuk mendapatkan informasi maupun sebagai hiburan. Gambar digital banyak sekali dipertukarkan di dunia digital melalui file-file yang mengandung gambar tersebut. Seringkali dalam transmisi dan penyimpanan gambar ditemukan masalah karena ukuran file gambar digital yang cenderung besar.

Kompresi gambar merupakan suatu tipe kompresi data yang dilakukan pada gambar digital. Dengan kompresi gambar, suatu file gambar digital dapat dikurangi ukuran filenya dengan baik tanpa mempengaruhi kualitas gambar secara signifikan. Terdapat berbagai metode dan algoritma yang digunakan untuk kompresi gambar pada zaman modern ini.



Three levels of JPG compression. The left-most image is the original. The middle image offers a medium compression, which may not be immediately obvious to the naked eye without closer inspection. The right-most image is maximally compressed.

Gambar 1. Contoh kompresi gambar dengan berbagai tingkatan

Sumber : [Understanding Compression in Digital Photography](#)
([lifewire.com](#))

Salah satu algoritma yang dapat digunakan untuk kompresi gambar adalah algoritma SVD (Singular Value Decomposition). Algoritma SVD didasarkan pada teorema dalam aljabar linier yang menyatakan bahwa sebuah matriks dua dimensi dapat dipecah menjadi hasil perkalian dari 3 sub-matriks yaitu matriks ortogonal U, matriks diagonal S, dan transpose dari matriks ortogonal V. Dekomposisi matriks ini dapat dinyatakan sesuai persamaan berikut.

$$A_{m \times n} = U_{m \times m} S_{m \times n} V_{n \times n}^T$$

Gambar 1. Algoritma SVD

Matriks U adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks AA^T . Matriks ini menyimpan informasi yang penting terkait baris-baris matriks awal, dengan informasi terpenting disimpan di dalam kolom pertama. Matriks S adalah matriks diagonal yang berisi akar dari nilai eigen matriks U atau V yang terurut menurun. Matriks V adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks A^TA . Matriks ini menyimpan informasi yang penting terkait kolom-kolom matriks awal, dengan informasi terpenting disimpan dalam baris pertama.



Gambar 2. Ilustrasi Algoritma SVD dengan rank k

Dapat dilihat di gambar di atas bahwa dapat direkonstruksi gambar dengan banyak *singular values* k dengan mengambil kolom dan baris sebanyak k dari U dan V serta *singular value* sebanyak k dari S atau Σ terurut dari yang terbesar. Kita dapat mengaproksimasi suatu gambar yang mirip dengan gambar aslinya dengan mengambil k yang jauh lebih kecil dari jumlah total *singular value* karena kebanyakan informasi disimpan di *singular values* awal karena *singular values* terurut mengecil. Nilai k juga berkaitan dengan rank matriks karena banyaknya *singular value* yang diambil dalam matriks S adalah *rank* dari matriks hasil, jadi dalam kata lain k juga merupakan rank dari matriks hasil. Maka itu matriks hasil rekonstruksidari SVD akan berupa informasi dari gambar yang terkompresi dengan ukuran yang lebih kecil dibanding gambar awal.

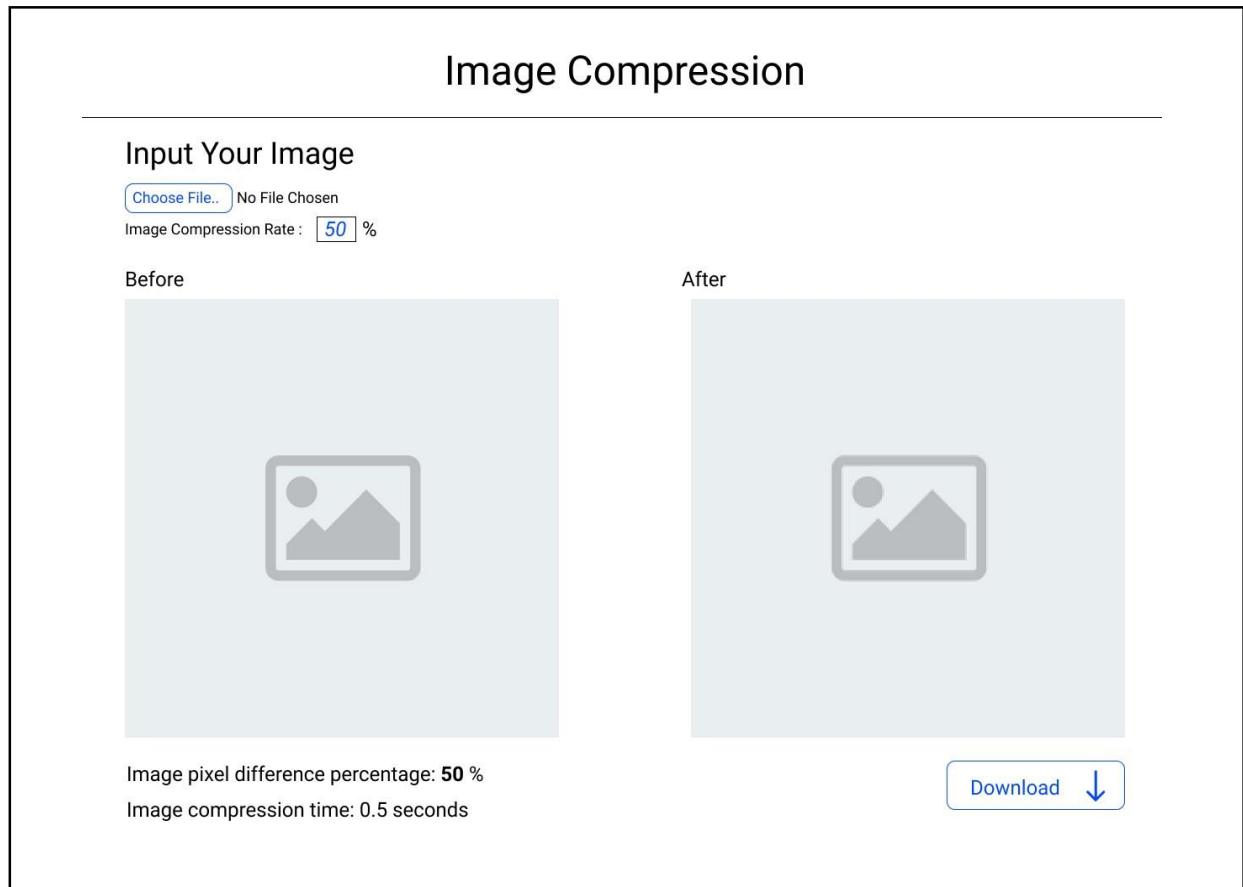
Pada kesempatan kali ini, kami mendapatkan tantangan untuk membuat website kompresi gambar sederhana dengan menggunakan algoritma SVD.

PENGUNAAN PROGRAM

Berikut ini adalah input yang akan dimasukkan pengguna untuk eksekusi program.

1. **File gambar**, berisi *file* gambar input yang ingin dikompresi dengan format *file* yang bebas selama merupakan format untuk gambar.
2. **Tingkat kompresi**, berisi tingkat kompresi dari gambar (formatnya dibebaskan, cth: Jumlah *singular value* yang digunakan)

Tampilan *layout* dari aplikasi web yang akan dibangun kurang lebih adalah sebagai berikut. Kami dapat mengubah *layout* selama *layout* masih terdiri dari komponen yang sama.



Gambar 3. Contoh tampilan layout dari aplikasi web yang dibangun.

Catatan: Warna biru menunjukkan komponen yang dapat di klik.

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Tampilan *front end* dari *website* dibuat semenarik mungkin selama mencakup seluruh informasi pada layout yang diberikan di atas. Tampilan program merupakan bagian dari penilaian.

SPESIFIKASI TUGAS

Buatlah program kompresi gambar dengan memanfaatkan algoritma SVD dalam bentuk website lokal sederhana. Spesifikasi website adalah sebagai berikut:

1. Website mampu menerima *file* gambar beserta *input* tingkat kompresi gambar (dibebaskan formatnya).
2. Website mampu menampilkan gambar *input*, *output*, *runtime* algoritma, dan persentase hasil kompresi gambar (perubahan jumlah pixel gambar).
3. File *output* hasil kompresi dapat diunduh melalui website.
4. Kompresi gambar tetap mempertahankan warna dari gambar asli.
5. **(Bonus)** Kompresi gambar tetap mempertahankan transparansi dari gambar asli, misal untuk gambar png dengan *background* transparan.

6. Bahasa pemrograman yang boleh digunakan adalah Python, Javascript, dan Go.
7. Penggunaan *framework* untuk *back end* dan *front end website* dibebaskan. Contoh *framework* website yang bisa dipakai adalah Flask, Django, React, Vue, dan Svelte.
8. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang andabuat (unsur kreativitas diperbolehkan/dianjurkan).
9. Program harus modular dan mengandung komentar yang jelas.
10. Diperbolehkan menggunakan *library* pengolahan citra seperti OpenCV2, PIL, atau image dari Go.
11. **Dilarang** menggunakan *library* perhitungan SVD dan *library* pengolahan eigen yang sudah jadi.

BAB II

TEORI SINGKAT

1. Perkalian Matriks

Perkalian matriks adalah suatu operasi yang menghasilkan matriks baru yang memiliki nilai berupa hasil perkalian dari elemen tiap baris pada matriks pertama dengan elemen tiap kolom pada matriks kedua. Oleh karena itu, untuk melakukan perkalian sebuah matriks, maka jumlah baris dari matriks pertama harus sama dengan jumlah kolom pada matriks kedua sehingga apabila dihitung akan menghasilkan matriks baru dengan rincian seperti ini :

$$A_{m \times n} \times B_{n \times p} = C_{m \times p}$$

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj},$$

Terdapat beberapa sifat yang dapat diterapkan pada perkalian matriks, antara lain :

- a) $A \times 0 = 0 \times A = 0$
- b) $(A \times B) \times C = A \times (B \times C)$
- c) $A \times (B + C) = A \times B + A \times C$
- d) $c(A \times B) = (c \times A) \times B$
- e) $A \times I = I \times A = A$

Contoh dari perkalian matriks adalah sebagai berikut :

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \times 10 + 2 \times 20 + 3 \times 30 & 1 \times 11 + 2 \times 21 + 3 \times 31 \\ 4 \times 10 + 5 \times 20 + 6 \times 30 & 4 \times 11 + 5 \times 21 + 6 \times 31 \end{bmatrix}$$

$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$

2. Nilai Eigen

Nilai eigen (λ) atau *eigenvalues* adalah set nilai skalar yang diperoleh dari sebuah matriks dengan ukuran $n \times n$ dan dapat disebut pula sebagai nilai karakteristiknya. Pada sebuah matriks A , harus berlaku perhitungan berikut :

$$Ax = \lambda x$$

$$IAx = \lambda Ix$$

$$Ax = \lambda Ix$$

$$(\lambda I - A)x = 0$$

$x = 0$ sehingga agar persamaan memiliki solusi tidak-nol, maka determinan dari $(\lambda I - A)$ harus sama dengan 0. Persamaan determinan tersebut dapat dikatakan sebagai persamaan karakteristik dari matriks A , dan akar-akar persamaannya (λ) disebut sebagai nilai-nilai eigen. Contoh pencarian nilai eigen adalah sebagai berikut :

$$\text{Matriks } A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

$$(\lambda I - A) = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix} = \begin{bmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{bmatrix}$$

$$\det(\lambda I - A) = 0$$

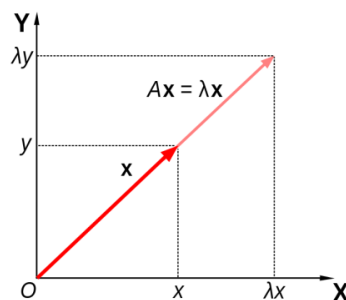
$$\begin{vmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{vmatrix} = 0$$

$$(\lambda - 3)(\lambda + 1) - (0)(-8) = 0$$

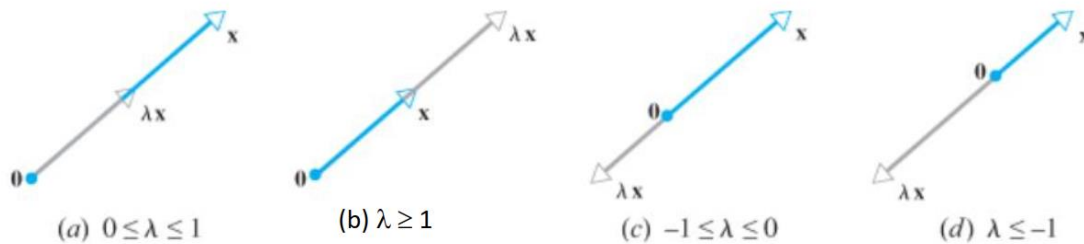
$$\lambda_1 = 3 \text{ dan } \lambda_2 = -1$$

3. Vektor Eigen

Vektor eigen (x) adalah vektor kolom yang ketika dikalikan dengan sebuah matriks $n \times n$ akan membentuk vektor baru yang merupakan kelipatan dari vektor itu sendiri.



Operasi $Ax = \lambda x$ menyebabkan vektor x menyusut atau memanjang dengan faktor λ pada arah yang sama (λ positif) ataupun berkebalikan arah (λ negatif).



Vektor eigen sendiri dapat diketahui apabila nilai eigen telah diketahui. Vektor eigen dapat dicari dengan memasukkan nilai eigen (λ) ke dalam rumus $(\lambda I - A)x = 0$. Contoh penyelesaian soal dari vektor eigen adalah sebagai berikut :

$$\text{Matriks } A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

$$(\lambda I - A)x = 0$$

$$\begin{bmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Untuk $\lambda = 3$, maka :

$$\begin{bmatrix} 0 & 0 \\ -8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Didapatkan persamaan berupa : $-8x_1 + 4x_2 = 0$

$$8x_1 = 4x_2 \rightarrow x_1 = \frac{1}{2} x_2$$

Solusi : $x_1 = \frac{1}{2} s$, $x_2 = s$, s bilangan real

$$\text{Vektor eigen : } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}s \\ s \end{bmatrix} = s \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

4. Matriks SVD

Singular value decomposition (SVD) adalah metode dekomposisi matriks untuk mereduksi sebuah matriks menjadi faktor-faktor penyusunnya agar perhitungan matriks berikutnya dapat menjadi lebih sederhana. Metode ini digunakan untuk matriks yang tidak berbentuk bujur sangkar (berukuran $m \times n$). SVD memfaktorkan matriks A berukuran $m \times n$ tersebut menjadi matriks U, Σ , dan V sehingga $A = U\Sigma V^T$.

$$\begin{matrix} \begin{matrix} \text{[Grid of 4x4 squares]} \end{matrix} & = & \begin{matrix} \text{[Grid of 4x4 squares with colored columns]} \end{matrix} & \begin{matrix} \text{[Grid of 4x4 squares with colored cells]} \end{matrix} & \begin{matrix} \text{[Grid of 4x4 squares with colored rows]} \end{matrix} \\ \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Terdapat beberapa tahapan yang harus dilalui untuk memfaktorkan suatu matriks menggunakan metode SVD. Untuk mencari matriks U dari matriks A , maka tahapan yang dilalui adalah sebagai berikut :

- Menghitung nilai-nilai eigen (λ) dari AA^T
 - Menentukan vektor eigen yang berkoresponden dengan nilai-nilai eigen dari AA^T sehingga didapatkan matriks u_1, u_2, \dots, u_n .
 - Melakukan normalisasi untuk matriks u_1, u_2, \dots, u_n dengan cara membagi setiap komponen vektornya dengan panjang vektor.
 - Menggabungkan seluruh hasil normalisasi sehingga diperoleh matriks U .
- Kemudian, untuk mencari matriks Σ dan V^T , tahapan yang dilalui adalah sebagai berikut :
- Menghitung nilai-nilai eigen (λ) dari $A^T A$
 - Menentukan nilai-nilai singular (σ) dengan rumus $\sigma = \sqrt{\lambda}$.
 - Membentuk matriks Σ berukuran $m \times n$ dengan elemen diagonalnya berupa nilai singular tidak nol yang didapatkan dari tahapan b diurut mengecil.
 - Menentukan vektor eigen yang berkoresponden dengan nilai-nilai eigen dari AA^T sehingga didapatkan matriks u_1, u_2, \dots, u_n .
 - Melakukan normalisasi untuk matriks u_1, u_2, \dots, u_n dengan cara membagi setiap komponen vektornya dengan panjang vektor.
 - Menggabungkan seluruh hasil normalisasi sehingga diperoleh matriks V .
 - Melakukan *transpose* terhadap matriks V sehingga diperoleh matriks V^T .

5. Perhitungan SVD menggunakan The Power Method

Salah satu algoritma sederhana untuk mengkalkulasikan *Singular value decomposition* (SVD) dari sebuah matriks adalah The Power Method. Metode ini mengaproksimasi nilai vektor singular kanan (v_i) dari matriks terlebih dahulu, kemudian menggunakannya untuk mendapatkan vektor singular kiri (u_i) dan (s_i).

Tahapan untuk mendekomposisi suatu matriks A dengan metode ini adalah sebagai berikut :

- Membuat sebuah vektor x_0 sehingga tiap elemen pada vektor tersebut memiliki range antara 0 dan 1. Banyak elemen pada vektor tersebut adalah sama dengan banyaknya kolom pada matriks awal
- Mengalikan matriks $A^T A$ dengan vektor x_0 untuk mendapatkan vektor x_1 . Langkah ini diulang sebanyak i kali hingga mendapatkan vektor x_i yang selisihnya relatif kecil dengan x_{i-1} atau dihentikan saat dirasa aproksimasi dari x_i sudah cukup akurat.
- v_1 didapatkan dengan menormalisasikan x_i menjadi vektor satuan
- s_1 didapatkan dengan menghitung $\|Av_1\|$
- u_1 didapatkan dengan menghitung Av_1/s_1
- Masukkan u_1, s_1 , dan v_1 ke matriks nya masing - masing dengan kolom yang bersesuaian dengan indeksnya.
- Kurangi matriks A awal dengan $u_1 v_1^T s_1$ sehingga matriks tersebut dapat diproses kembali untuk mendapatkan vektor u, v , dan s selanjutnya
- Ulangi langkah a) hingga g) untuk mendapatkan elemen dari dekomposisi matriks hingga sebanyak k kolom pertama.

BAB III

IMPLEMENTASI PROGRAM

Struktur dari program kami adalah sebagai berikut :

```
Algeo02-20094/
├── src/
│   ├── compress/
│   │   ├── compress.py
│   │   └── compresssvdver2.py
│   ├── static/
│   │   ├── assets/
│   │   │   ├── 3ver4.svg
│   │   │   ├── logoSARAP.png
│   │   │   ├── index.css
│   │   │   ├── patrick.jpg
│   │   │   ├── rania.jpg
│   │   │   └── saul.jpg
│   │   ├── FileSaver.js
│   │   └── index.js
│   ├── templates/
│   │   └── index.html
│   └── app.py
├── doc/
│   ├── Algeo02-20094.pdf
│   ├── aboutus.png
│   ├── home.png
│   └── results.png
├── test/
│   ├── birds.png
│   ├── jokowi.jpeg
│   ├── ngupil.png
│   ├── temp.png
│   ├── tigers.jpg
│   └── transparan.png
└── README.md
```

Bahasa Pemrograman : Python, JavaScript, HTML, CSS

Framework :

- a) Frontend : -
- b) Backend : Flask (Python)

1. Penjelasan Tech Stack

Frontend : Menggunakan scripting language HTML dan CSS untuk keperluan struktur website, dan JavaScript untuk keperluan *scripting response* dan dinamika website. Bagian ini tidak menggunakan *framework* khusus.

Backend : Menggunakan bahasa pemrograman Python dibarengi dengan *framework* Flask untuk menghubungkan pengiriman data antara *Frontend* (Form input gambar dan compression rate) dengan *Backend* (algoritma pemrosesan gambar dan output waktu + pixel difference). Penggunaan *framework* Flask ditujukan untuk mempermudah GET data dari HTML dan POST data ke HTML.

2. Algoritma Kompresi

a) **function** svd

```
baris ← len(matriksawal)
kolom ← len(matriksawal[0])
```

Pada bagian ini, baris dan kolom digunakan sebagai istilah baru untuk memanggil jumlah baris dan kolom dari sebuah matriks dengan mengakses panjangnya.

```
kiri ← numpy.zeros((baris, 1))
tengah ← []
kanan ← numpy.zeros((kolom, 1))
```

Kemudian, diinisialisasikan matriks kiri (yaitu U), tengah (yaitu sigma atau kumpulan nilai singular) dan kanan (yaitu V transpose).

Setelah itu, akan dilakukan *looping* sebanyak k kali. K sendiri merupakan modifikasi pemanggilan rasio yang digunakan untuk mengompres suatu gambar. Pada *looping* tersebut, terdapat algoritma berikut :

```
matriksawaltranspos ← numpy.transpose(matriksawal)
```

Algoritma ini digunakan untuk *men-transpose* suatu matriks dimana baris akan menjadi kolom dan kolom menjadi baris.

```
matriksgabungan ← numpy.dot(matriksawaltranspos, matriksawal)
```

Algoritma ini akan melakukan perkalian dot untuk matriksawaltranspos dengan matriksawal.

```
x ← random.normal(0, 1, size=kolom)
for j in range(10):
    x ← numpy.dot(matriksgabungan, x)
```

Pada bagian ini, kita akan mencari nilai x_i dengan cara mengiterasikannya sebanyak 10 kali agar selisihnya relatif kecil dengan x_{i-1} .

```
normx ← numpy.linalg.norm(x)
v ← numpy.divide(x, normx, where=normx!=0)
```

Disini, kita mencari matriks v_i dengan cara menormalisasikan x_i menjadi vektor satuan, namun algoritma ini tidak akan bekerja pada bagian dimana hasil nilai distribusi gaussnya adalah 0.

```
nilaisingular ← linalg.norm(numpy.dot(matriksawal, v))
matriksawalv ← numpy.dot(matriksawal, v)
tengah.append(nilaisingular)
```

Matriks tengah, atau s_i didapatkan dengan cara menghitung $\|Av_i\|$. Fungsi *append* sendiri berfungsi untuk menggabungkan nilai singular ke dalam matriks tengah.

```
u ← numpy.reshape(numpy.divide(matriksawalv, nilaisingular, where=nilaisingular!=0),  
(baris, 1))  
kiri ← numpy.concatenate((kiri,u), axis = 1)
```

Disini, matriks u akan melewati operasi pembagian dan akan diubah bentuknya. Kemudian, matriks u ini akan digabungkan dengan matriks kiri secara horizontal.

```
v ← numpy.reshape(v, (kolom, 1))  
kanan ← numpy.concatenate((kanan,v), axis = 1)
```

Matriks v akan diubah bentuknya menjadi 1 kolom dan kemudian digabungkan secara horizontal ke matriks kanan.

```
matriksawal ← matriksawal - numpy.dot(numpy.dot(u, numpy.transpose(v)),  
nilaisingular)
```

Pada proses ini, matriks A akan dikurangi dengan $u_1 v_1^T s_1$ sehingga matriks tersebut dapat diproses kembali untuk mendapatkan vektor u, v, dan s selanjutnya pada *looping*.

```
return kiri[:, 1:], tengah, numpy.transpose(kanan[:, 1:])
```

Terakhir, fungsi ini mengembalikan *return* berupa matriks kiri yang di potong 1 kolomnya, matriks tengah, dan transpose dari matriks kanan yang dipotong 1 kolomnya.

b) **function** banyaknyaKdigunakan

```
def banyaknyaKdigunakan(matriksawal,rasio)  
    baris, kolom = matriksawal.shape[0], matriksawal.shape[1],  
    if baris < kolom :  
        total = baris  
    else :  
        total = kolom  
    digunakan = round((rasio/100)*total)  
    return digunakan
```

Fungsi tersebut menerima 2 parameter yakni matriksawal berupa matriks dan rasio berupa float dalam persen. Fungsi ini mencari banyaknya singular values yang ingin kita gunakan pada algoritma SVD dengan cara mencari terlebih dahulu banyaknya singular values total. Kemudian, fungsi mengalikan banyaknya singular values total tersebut dengan rasio tadi sehingga didapat banyaknya singular values yang ingin kita gunakan. Terakhir, fungsi mengembalikan *return* berupa singular values yang digunakan tersebut.

c) **function** gambartomatriks

```
def gambartomatriks(gambarawal):  
    modePA = False  
    modeP = False  
    if gambarawal.mode == 'P' :  
        gambarawal = gambarawal.convert('RGBA')  
        modeP = True  
    if gambarawal.mode == 'PA':  
        gambarawal = gambarawal.convert('RGBA')  
        modePA = True  
    matriksawal = numpy.array(gambarawal) # convert gambarnya jadi matriks  
    return modeP, modePA, matriksawal
```

Fungsi tersebut menerima parameter sebuah objek gambar awal yang sudah dibuka menggunakan library pengolahan citra PIL. Kemudian, fungsi mengecek dan mencatat mode awal dari gambar tersebut apakah 'P' atau 'PA' karena akan mengalami proses yang berbeda dalam SVD, sehingga

diconvert terlebih dahulu menjadi RGBA. Fungsi juga mengkonversikan gambar tersebut menjadi sebuah matriks menggunakan library Numpy. Terakhir, fungsi mengembalikan *return* berupa boolean modeP, modePA yang mencatat mode awal mereka, dan matriks dari gambar awal tersebut.

d) **function** matrikstogambar

```
def matrikstogambar(matrikshasil):  
    numpy.clip(matrikshasil,0,255,matrikshasil)  
    matriksunsigned = matrikshasil.astype('uint8')  
    hasilgambar = Image.fromarray(matriksunsigned)  
    return hasilgambar
```

Fungsi menerima parameter sebuah matriks hasil setelah pemrosesan algoritma SVD. Pertama, fungsi membatasi nilai yang diluar range nilai 0 – 255 menggunakan fungsi `numpy.clip` sehingga tidak error saat matriks dikonversi kembali menjadi gambar. Kemudian, fungsi mengkonversikan juga tiap elemen menjadi *unsigned* integer 8 bit sehingga tiap elemennya berupa bilangan bulat dari 0 – 255. Terakhir, fungsi mengkonversikan matriks yang elemennya sudah *unsigned* tersebut menjadi hasil gambar menggunakan fungsi `Image.fromarray` dan mengembalikan *return* berupa hasil gambar tersebut.

e) **function** buangpixelsisa

```
def buangpixelsisa(matrikshasil, berwarna) :  
    if (berwarna):  
        indekstransparansi = 3  
    else :  
        indekstransparansi = 1  
    for baris in range(matrikshasil.shape[0]) :  
        for kolom in range (matrikshasil.shape[1]):  
            if matrikshasil[baris,kolom,indekstransparansi] == 0 :  
                matrikshasil[baris,kolom,0] = 0  
                if (berwarna) :  
                    matrikshasil[baris,kolom,1] = 0  
                    matrikshasil[baris,kolom,2] = 0  
    return matrikshasil
```

Fungsi tersebut menerima parameter sebuah matriks hasil setelah pemrosesan SVD dan berwarna yang berupa boolean. Fungsi tersebut dibuat dengan tujuan menghemat memori bagi gambar yang memiliki transparansi, dengan cara membuat semua pixel pada bagian transparan gambar menjadi 0 sehingga tidak menyia-nyiakan bit apapun. Untuk matriks gambar yang berwarna, indeks transparansinya berada di layer ketiga matriks, sementara matriks gambar yang greyscale, indeks transparansinya berada di layer kesatu matriks. Setelah selesai proses pengenalan semua pixel transparan, maka fungsi akan mengembalikan *return* sebuah matriks hasil yang sudah dibuang pixelnya.

f) **function** kompresgambarwarna

```
def kompresgambarwarna(matriksawal, rasio,transparan):  
    k= banyaknyaKdigunakan(matriksawal,rasio)  
    if (transparan):
```

```
matrikshasil = numpy.zeros((matriksawal.shape[0],
matriksawal.shape[1], 4))
else :
    matrikshasil = numpy.zeros((matriksawal.shape[0],
matriksawal.shape[1], 3))
    for warna in range(3):
        kiri, tengah, kanan = svd(matriksawal[:, :, warna], k)
        tengah = numpy.diag(tengah)
        matrikshasil[:, :, warna] = kiri[:, 0:k] @ tengah[0:k, 0:k] @
kanan[0:k, :]
    if (transparan):
        matrikshasil[:, :, 3] = matriksawal[:, :, 3]
        matrikshasil = buangpixelsisa(matrikshasil, True)
    hasilgambar = matrikstogambar(matrikshasil)
return hasilgambar
```

Fungsi tersebut menerima parameter matriks awal yakni matriks dari sebuah gambar berwarna yang ingin dikompres dengan SVD, rasio kompresi yang merupakan float dengan persentase banyaknya singular values yang ingin digunakan, dan transparan yang merupakan boolean. Pertama, fungsi menggunakan fungsi `banyaknyaKdigunakan` untuk menghitung banyaknya singular values yang ingin digunakan. Kemudian, fungsi menginisialisasi sebuah matriks kosong dengan baris dan kolom yang sama dengan matriks awal dinamakan matriks hasil. Apabila transparan, maka matriks memiliki 4 layer (RGBA), berlaku juga untuk (CMYK) dan apabila tidak maka matriks memiliki 3 layer (RGB). Kemudian, fungsi akan melakukan proses dekomposisi SVD untuk tiap layer warna menjadi matriks kiri (U), matriks kanan (V^T), dan array tengah yang berisi singular values (Σ). Kemudian, fungsi mengubah array Σ menjadi matriks diagonal menggunakan fungsi `numpy.diag`. Lalu, fungsi akan mengalikan kembali matriks kiri tengah kanan tiap layer tersebut sampai ke rank-k nya, kemudian di-assign ke layer matriks hasil yang bersesuaian. Apabila matriksnya memiliki layer transparan, maka layer transparan awal akan diassign ke layer transparan matriks hasil dan dilanjut pengenalan pixel sisa menggunakan fungsi `buangpixelsisa`. Terakhir, fungsi mengkonversikan matriks tersebut menjadi gambar hasil dengan mode yang sesuai dengan mode awal menggunakan fungsi `matrikstogambar`, kemudian mengembalikan *return* berupa gambar hasil tersebut.

g) **function** kompresgambargrey

```
def kompresgambargrey(matriksawal, rasio, transparan):
    k = banyaknyaKdigunakan(matriksawal, rasio)
    if (transparan):
        matrikshasil = numpy.zeros((matriksawal.shape[0],
matriksawal.shape[1], 2))
        kiri, tengah, kanan = svd(matriksawal[:, :, 0], k)
    else :
        matrikshasil = numpy.zeros((matriksawal.shape[0],
matriksawal.shape[1]))
        kiri, tengah, kanan = svd(matriksawal, k)
    singular values
    if (transparan) :
        matrikshasil[:, :, 0] = kiri[:, 0:k] @ tengah[0:k, 0:k] @ kanan[0:k, :]
        matrikshasil[:, :, 1] = matriksawal[:, :, 1]
        matrikshasil = buangpixelsisa(matrikshasil, False)
```

```
else :  
    matrikshasil = kiri[:, 0:k] @ tengah[0:k,0:k] @ kanan[0:k,:]   
    hasilgambar = matrikstogambar(matrikshasil)  
    return hasilgambar
```

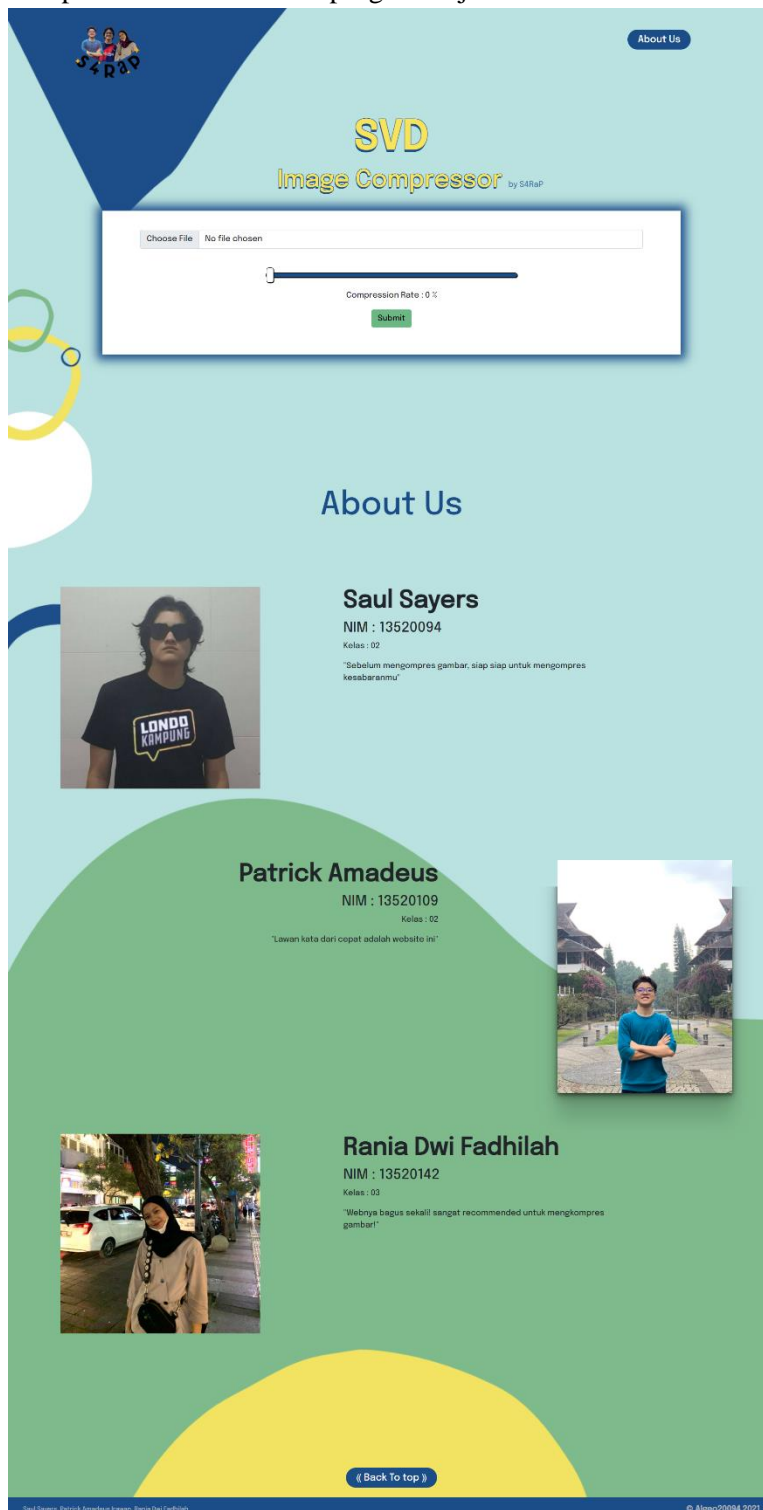
Fungsi tersebut menerima parameter yang sama dengan fungsi kompresgambarwarna, namun matriks awalnya berasal dari gambar greyscale. Pertama, fungsi menggunakan fungsi `banyaknyaK` digunakan untuk menghitung banyaknya singular values yang ingin digunakan. Kemudian, fungsi menginisialisasi sebuah matriks kosong dengan baris dan kolom yang sama dengan matriks awal dinamakan matriks hasil. Apabila transparan, maka matriks memiliki 2 layer (LA) dan apabila tidak maka matriks memiliki 1 layer (L) sehingga dimensi matriksnya cukup 2 (baris x kolom). Kemudian, fungsi akan melakukan proses dekomposisi SVD untuk tiap layer warna menjadi matriks kiri (U), matriks kanan (V^T), dan array tengah yang berisi singular values (Σ). Kemudian, fungsi mengubah array Σ menjadi matriks diagonal menggunakan fungsi `numpy.diag`. Lalu, fungsi akan mengalikan kembali matriks kiri tengah kanan tersebut sampai ke rank-k nya, kemudian di-assign ke matriks hasil. Apabila matriksnya memiliki layer transparan, maka layer transparan awal akan diassign ke layer transparan matriks hasil dan dilanjut pengenalan pixel sisa menggunakan fungsi `uangpixelsisa`. Terakhir, fungsi mengkonversikan matriks tersebut menjadi gambar hasil dengan mode yang sesuai dengan mode awal menggunakan fungsi `matrikstogambar`, kemudian mengembalikan `return` berupa gambar hasil tersebut.

BAB IV

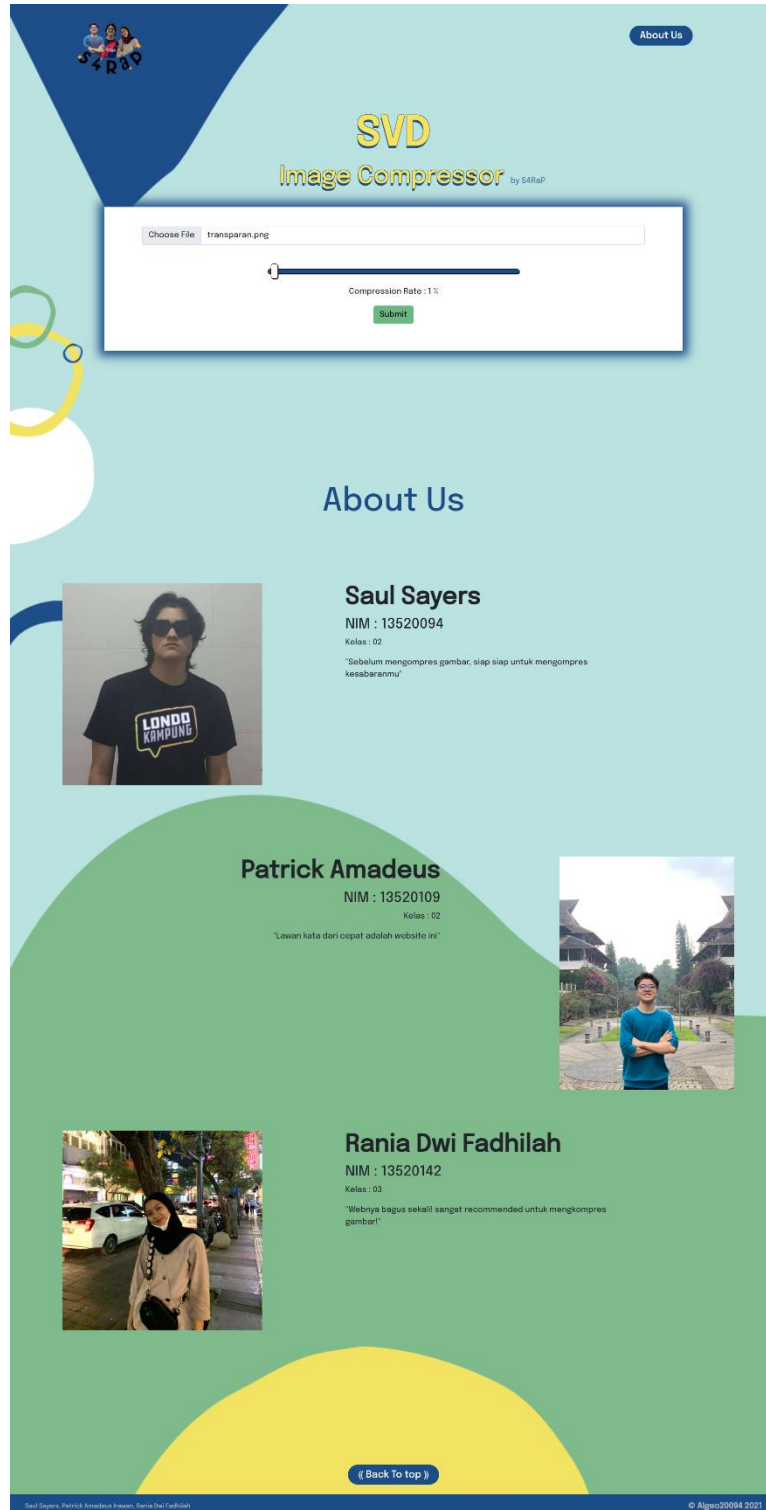
EKSPERIMEN

1. Interface

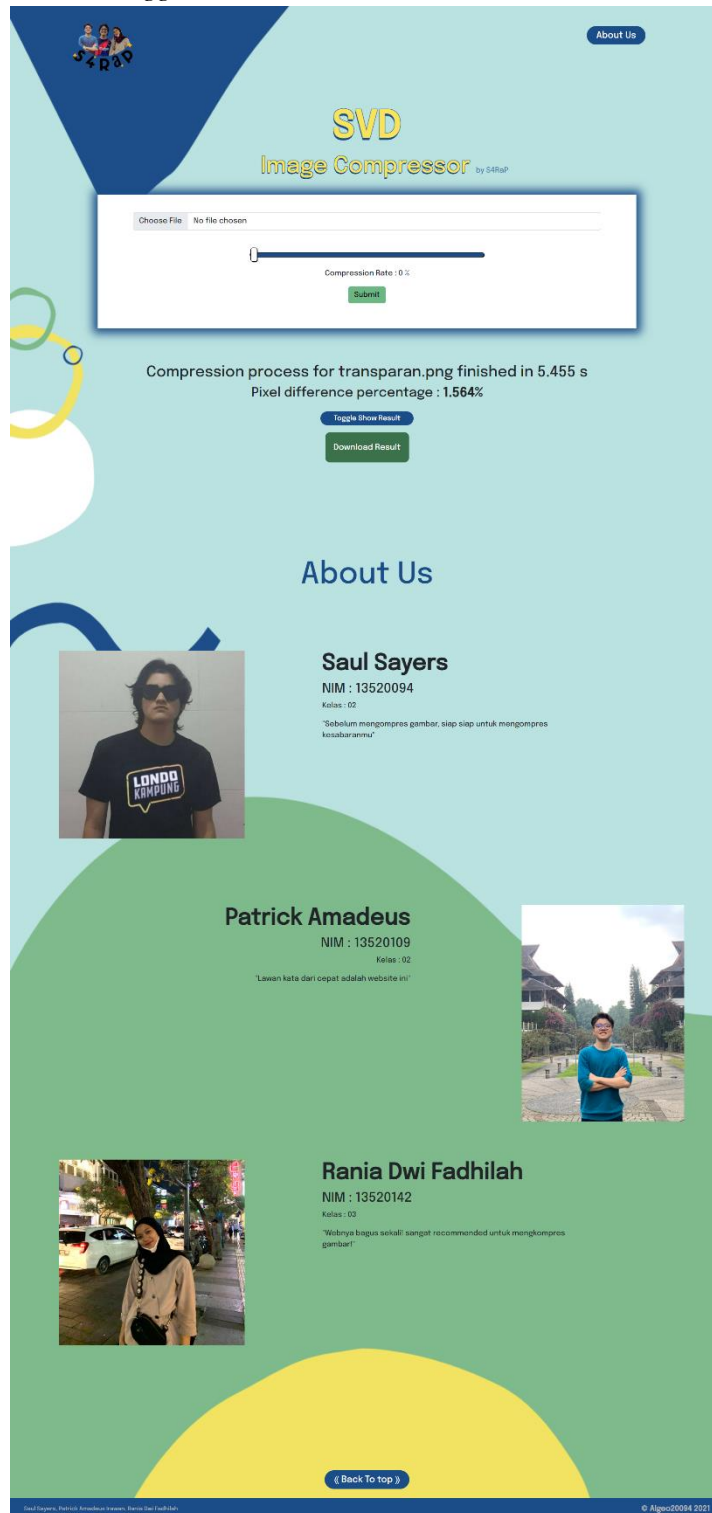
- a) Tampilan website sebelum program dijalankan



- b) Tampilan website setelah program dijalankan
- o Setelah *input* file



- Sebelum *toggle*



- Setelah *toggle*

The screenshot displays the SVD Image Compressor web application. At the top, there is a logo with the text 'S4R0D' and an 'About Us' button. The main heading is 'SVD Image Compressor' by s4r0d. Below this is a file selection area with a 'Choose File' button and a 'No file chosen' status. A slider for 'Compression Rate' is set to 0%, with a 'Submit' button. A message states: 'Compression process for transparan.png finished in 5.455 s Pixel difference percentage : 1.564%'. Below this, there is a 'Toggle Show Result' button. Two images of a beagle are shown side-by-side: 'Original' and '1% compressed'. The compressed image is visibly blurred. A 'Download Result' button is located below the images. The 'About Us' section follows, featuring three team members: Saul Sayers (NIM: 13520094, Kelas: 02), Patrick Amadeus (NIM: 13520109, Kelas: 02), and Rania Dwi Fadhillah (NIM: 13520142, Kelas: 03). Each member has a profile picture and a short testimonial. A 'Back To top' button is at the bottom, and a footer contains the names of the team members and the copyright notice '© Algen/2020/4 2021'.

SVD Image Compressor by s4r0d

Choose File No file chosen

Compression Rate : 0 %

Submit

Compression process for transparan.png finished in 5.455 s
Pixel difference percentage : 1.564%

Toggle Show Result

Original 1% compressed

Download Result

About Us

Saul Sayers
NIM : 13520094
Kelas : 02
"Tabelum mengompres gambar, siap siap untuk mengompres keabadianmu"

Patrick Amadeus
NIM : 13520109
Kelas : 02
"Laman kata dari cepat adalah website ini"

Rania Dwi Fadhillah
NIM : 13520142
Kelas : 03
"Webnya bagus sekali sangat recommended untuk mengompres gambar"

(Back To top)


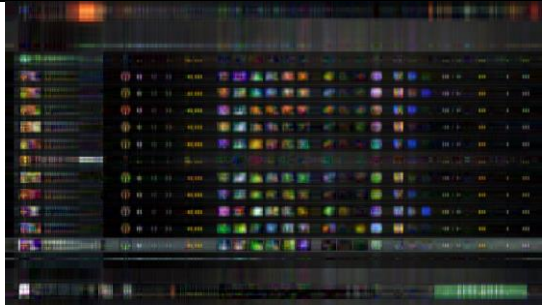


Saul Sayers, Patrick Amadeus, Rania Dwi Fadhillah © Algen/2020/4 2021



2. Studi Kasus

Catatan:




- Compression rate dihitung berdasarkan banyaknya singular values digunakan / total
- Perubahan pixels dihitung dengan rumus yang disediakan pada sheet QnA no 6.




a) temp.png

| Ukuran file : 2.357 kb | | Total <i>singular values</i> : 1100 | Kasus gambar : RGBA |
|----------------------------------|---|--|-------------------------------|
| Gambar Asli |  | | |
| Compress <u>1%</u> |  | <p><i>Singular values</i> yang digunakan : <u>11</u> Waktu : <u>59,789</u> detik Size : <u>2299</u> kb Perubahan <i>pixels</i> : <u>1,592%</u></p> | |
| Compress <u>3%</u> |  | <p><i>Singular values</i> yang digunakan : <u>32</u> Waktu : <u>79,577</u> detik Size : <u>2680</u> kb Perubahan <i>pixels</i> : <u>4,631%</u></p> | |
| Compress <u>6%</u> |  | <p><i>Singular values</i> yang digunakan : <u>65</u> Waktu : <u>101,385</u> detik Size : <u>2951</u> kb Perubahan <i>pixels</i> : <u>9,407%</u></p> | |


| | | |
|------------------------|---|--|
| Compress <u>12%</u> |  | Singular values yang digunakan : <u>130</u> Waktu : <u>139,431</u> detik Size : <u>3211</u> kb Perubahan pixels : <u>18,814</u> % |
| Compress <u>25%</u> |  | Singular values yang digunakan : <u>270</u> Waktu : <u>237,65</u> detik Size : <u>3390</u> kb Perubahan pixels : <u>39,076</u> % |






b) transparan.png

| Ukuran file : 59 kb | Total singular values : 530 | Kasus Gambar : P |
|-------------------------------|--|--|
| Gambar Asli |  | |
| Compress <u>1%</u> |  | Singular values yang digunakan : 5 Waktu : <u>2,223</u> detik Size : <u>34</u> kb Perubahan pixels : <u>1,564</u> % |
| Compress <u>5%</u> |  | Singular values yang digunakan : <u>27</u> Waktu : <u>3,388</u> detik Size : <u>48</u> kb Perubahan pixels : <u>8,447</u> % |





| | | |
|----------------------------|--|--|
| Compress <u>10%</u> |  | <i>Singular values</i> yang digunakan : <u>53</u> Waktu : <u>5,278</u> detik Size : <u>51</u> kb Perubahan <i>pixels</i> : <u>16,581%</u> |
| Compress <u>25%</u> |  | <i>Singular values</i> yang digunakan : <u>133</u> Waktu : <u>12,04</u> detik Size : <u>53</u> kb Perubahan <i>pixels</i> : <u>41,609%</u> |
| Compress <u>50%</u> |  | <i>Singular values</i> yang digunakan : <u>266</u> Waktu : <u>21,74</u> detik Size : <u>49</u> kb Perubahan <i>pixels</i> : <u>83,219%</u> |


c) jokowi.jpeg

| | | |
|--------------------------------------|--|-------------------------------------|
| Ukuran file : <i>33 kb</i> | Total <i>singular values</i> : <i>500</i> | Kasus Gambar : <i>RGB</i> |
| Gambar Asli |  | |



| | | |
|---------------------|---|--|
| Compress <u>1%</u> |  | <p><i>Singular values</i> yang digunakan : <u>5</u> Waktu : <u>1,023</u> detik Size : <u>21</u> kb Perubahan <i>pixels</i> : <u>1,763</u>%</p> |
| Compress <u>5%</u> |  | <p><i>Singular values</i> yang digunakan : <u>25</u> Waktu : <u>1,515</u> detik Size : <u>28</u> kb Perubahan <i>pixels</i> : <u>8,814</u>%</p> |
| Compress <u>10%</u> |  | <p><i>Singular values</i> yang digunakan : <u>50</u> Waktu : <u>2,502</u> detik Size : <u>30</u> kb Perubahan <i>pixels</i> : <u>17,628</u>%</p> |
| Compress <u>25%</u> |  | <p><i>Singular values</i> yang digunakan : <u>126</u> Waktu : <u>5,273</u> detik Size : <u>31</u> kb Perubahan <i>pixels</i> : <u>44,423</u>%</p> |
| Compress <u>50%</u> |  | <p><i>Singular values</i> yang digunakan : <u>252</u> Waktu : <u>7,128</u> detik Size : <u>32</u> kb Perubahan <i>pixels</i> : <u>88,846</u>%</p> |





d) ngupil.png

| Ukuran file : <i>155 kb</i> | Total <i>singular values</i> : 1000 | Kasus Gambar : <i>RGB</i> |
|---------------------------------------|---|---|
| Gambar Asli |  | |
| <i>Compress</i> <u>1%</u> |  | <i>Singular values</i> yang digunakan : <u>10</u> Waktu : <u>8,57</u> detik Size : <u>90</u> kb Perubahan <i>pixels</i> : <u>1,759%</u> |
| <i>Compress</i> <u>5%</u> |  | <i>Singular values</i> yang digunakan : <u>50</u> Waktu : <u>12,763</u> detik Size : <u>132</u> kb Perubahan <i>pixels</i> : <u>8,969%</u> |
| <i>Compress</i> <u>10%</u> |  | <i>Singular values</i> yang digunakan : <u>100</u> Waktu : <u>28,528</u> detik Size : <u>138</u> kb Perubahan <i>pixels</i> : <u>17,937%</u> |





| | | |
|------------------------|--|---|
| Compress <u>25%</u> |  | Singular values yang digunakan : <u>250</u> Waktu : <u>77,5</u> detik Size : <u>119</u> kb Perubahan pixels : <u>45,02</u> % |
| Compress <u>50%</u> |  | Singular values yang digunakan : <u>500</u> Waktu : <u>110,8</u> detik Size : <u>117</u> kb Perubahan pixels : <u>90,039</u> % |



e) birds.png

| Ukuran file : <i>203 kb</i> | | Total singular values : <i>512</i> | Kasus Gambar : <i>L</i> |
|--------------------------------|--|---|----------------------------|
| Gambar Asli |  | | |
| Compress <u>1%</u> |  | Singular values yang digunakan : <u>5</u> Waktu : <u>1,113</u> detik Size : <u>89</u> kb Perubahan pixels : <u>1,629</u> % | |

| | | |
|---------------------------------------|--|--|
| <p><i>Compress</i> <u>5%</u></p> |  | <p><i>Singular values</i> yang digunakan : <u>26</u> Waktu : <u>1,44</u> detik Size : <u>130</u> kb Perubahan <i>pixels</i> : <u>8,47</u>%</p> |
| <p><i>Compress</i> <u>10%</u></p> |  | <p><i>Singular values</i> yang digunakan : <u>51</u> Waktu : <u>2,373</u> detik Size : <u>146</u> kb Perubahan <i>pixels</i> : <u>16,615</u>%</p> |
| <p><i>Compress</i> <u>25%</u></p> |  | <p><i>Singular values</i> yang digunakan : <u>128</u> Waktu : <u>3,402</u> detik Size : <u>171</u> kb Perubahan <i>pixels</i> : <u>41,699</u>%</p> |
| <p><i>Compress</i> <u>50%</u></p> |  | <p><i>Singular values</i> yang digunakan : <u>256</u> Waktu : <u>7,458</u> detik Size : <u>185</u> kb Perubahan <i>pixels</i> : <u>83,398</u>%</p> |

f) tigers.jpg

| Ukuran file : 465 kb | | Total <i>singular values</i> : 1200 | Kasus Gambar : RGB |
|--------------------------------|---|---|---|
| Gambar Asli |  | | |
| Compress 1% |  | | <i>Singular values</i> yang digunakan : 12 Waktu : 22,421 detik Size : 137 kb Perubahan <i>pixels</i> : 1,751% |
| Compress 5% |  | | <i>Singular values</i> yang digunakan : 60 Waktu : 39,336 detik Size : 187 kb Perubahan <i>pixels</i> : 8,753% |
| Compress 10% |  | | <i>Singular values</i> yang digunakan : 120 Waktu : 66,502 detik Size : 216 kb Perubahan <i>pixels</i> : 17,506% |

| | | |
|--------------------------------------|---|---|
| <i>Compress</i> <u>25%</u> |  | <i>Singular values</i> yang digunakan : <u>300</u> Waktu : <u>148,163</u> detik <i>Size</i> : <u>238</u> kb Perubahan <i>pixels</i> : <u>43,766</u> % |
| <i>Compress</i> <u>50%</u> |  | <i>Singular values</i> yang digunakan : <u>600</u> Waktu : <u>202,69</u> detik <i>Size</i> : <u>255</u> kb Perubahan <i>pixels</i> : <u>87,531</u> % |

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

I. Kesimpulan

Pada tugas besar mata kuliah IF2123 Aljabar Linear dan Geometri yang ke-2 ini, telah berhasil diimplementasikan hasil pembelajaran nilai eigen, vektor eigen, dan *singular value decomposition* dalam bentuk situs web yang berfungsi untuk mengompres suatu gambar.

Pendekatan SVD ini dilakukan dengan mengubah gambar menjadi suatu matriks, kemudian matriks tersebut diolah menggunakan k tertentu, dimana k sendiri berasal dari *input* yang dimasukkan oleh *user*. Pengolahan matriks sendiri disesuaikan dengan jenis gambarnya. Jenis gambar sendiri dapat berupa RGB, RGBA, CMYK, P, L, dan LA.

Situs web yang kelompok buat menerima *input* berupa gambar dan juga rasio kompresi. Kemudian, *output* yang akan diberikan oleh situs web adalah hasil gambar setelah di kompres, *runtime* algoritma, dan persentase hasil kompresi gambar. Hasil gambar yang sudah di kompres juga dapat diunduh oleh *user*.

Dalam membuat situs web ini sendiri, kelompok menggunakan beberapa bahasa, antara lain Python, JavaScript, dan juga HTML. Dalam proses implementasi program kompresi, digunakan beberapa *library* untuk mempercepat dan mempermudah proses pembuatan algoritma, yaitu numpy, PIL, time, base64, dan juga io. Selain *library* diatas, program juga menggunakan *framework* Flask untuk pembuatan *website*. Penggunaan *framework* ini sangat membantu proses sinkronisasi antara *frontend* dengan *backend*.

Dengan demikian, kelompok dapat menyimpulkan bahwa dengan mengerjakan Tugas Besar II IF2123 Aljabar Linier dan Geometri Semester 1 Tahun 2021/2022 ini, dapat diketahui bahwa kegiatan mengompres suatu gambar dapat dilakukan dengan membuat sebuah program yang mengimplementasikan salah satu materi pembelajaran yaitu pendekatan *singular value decomposition* (SVD).

II. Saran

Tugas Besar II IF2123 Aljabar Linier dan Geometri Semester 1 Tahun 2021/2022 menjadi salah satu proses pembelajaran bagi kelompok dalam menerapkan ilmu-ilmu yang dipelajari pada perkuliahan ataupun dengan melakukan eksplorasi materi secara mandiri. Berikut ini adalah saran dari kelompok untuk pihak-pihak yang ingin melakukan atau mengerjakan hal serupa.

- Situs web dibuat dengan membuat *source code frontend* dan juga *backend*. Pembuatan *source code* ini belum terlalu dikuasai oleh ketiga anggota kelompok yang terlibat dalam pengerjaan tugas besar ini. Oleh karena itu, kelompok merekomendasikan pihak-pihak terkait untuk meluangkan waktu yang cukup untuk melakukan eksplorasi terkait *frontend* dan juga *backend* ini untuk memudahkan pembuatan situs web.
- Algoritma pemrograman dari situs web kompres adalah SVD. Untuk dapat mengimplementasikan SVD dengan baik dan efisien, sebaiknya pihak-pihak terkait meluangkan waktu untuk memahami materi ini hingga tuntas agar mempermudah proses pembuatan kode.
- Dalam mengerjakan suatu tugas secara berkelompok, penting untuk memiliki strategi serta distribusi tugas yang baik dan efisien. Cara penulisan kode dan kemampuan menulis komentar menjadi hal yang sangat penting dalam

mengerjakan kode pemrograman secara berkelompok. Dengan adanya komentar pada kode, anggota lain pada kelompok dapat memahami cara kerja suatu kode dengan lebih cepat. Kemampuan tersebut juga didukung dengan adanya *version control system* (VCS) yang baik untuk digunakan oleh *programmer* dalam membuat sebuah kode pemrograman secara bersamaan. Kelompok sangat menyarankan penggunaan 'Github' untuk digunakan sebagai *version control system* (VCS) dalam pengerjaan tugas besar, maupun pada pembuatan program yang lainnya.

III. Refleksi

Dalam membuat situs web yang berfungsi untuk mengompres suatu gambar ini, kelompok harus membuat *source code frontend* dan juga *backend*, berikut algoritma kompres itu sendiri. Pada bagian algoritma kompres, kelompok mengimplementasikan pendekatan *singular value decomposition*. Seperti yang diajarkan di kelas, awalnya kelompok menggunakan cara runtut manual yaitu dengan mencari nilai eigennya terlebih dahulu, menentukan vektor eigen, kemudian mulai melakukan pendekatan SVD. Namun, terdapat banyak sekali kendala selama proses penerapan yang satu ini. Akhirnya, kelompok mencoba untuk menggunakan pendekatan iterasi QR untuk mendapatkan nilai eigennya. Ketika berhasil diimplementasikan, ternyata waktu jalannya program sangatlah lambat karena besarnya matriks gambar. Oleh karena itu, kelompok harus mencari cara lain agar kompres dapat dilakukan secara lebih cepat. Pada akhirnya, kelompok memilih untuk menyelesaikan persoalan ini dengan menggunakan *power iteration* untuk melakukan pendekatan SVD terhadap matriks gambar. Dengan metode ini, algoritma kompres kelompok dapat berjalan secara efisien dan lebih cepat dibanding sebelumnya. Melalui tahap ini, kelompok menyadari bahwa harus dilakukan riset dan *trial and error* terlebih dahulu untuk membuat kode yang paling efisien.

Selain dari algoritma kompresi sendiri, kelompok menyadari bahwa ilmu *web development* sangat dibutuhkan dan harus lebih dikembangkan lagi agar dapat membuat sebuah situs web yang lebih menarik, fungsional, dan juga *user-friendly* dari segi UI maupun UX. Pada pengerjaan tugas besar ini, kelompok sempat berulang kali mengganti *framework* karena ditemukannya beberapa kesulitan yang tidak memungkinkan untuk dipecahkan dalam jangka waktu yang cukup singkat.

REFERENSI

- Towardsdatascience.com. (2019, 5 Agustus). Singular Value Decomposition Example in Python. Diakses pada 2 November 2021, dari <https://towardsdatascience.com/singular-value-decomposition-example-in-python-dab2507d85a0>
- Informatika.stei.itb.ac.id/~rinaldi.munir. (2021). Algeo #18 Nilai Eigen dan Vektor Eigen (Bagian 1). Diakses pada 3 November 2021, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
- Informatika.stei.itb.ac.id/~rinaldi.munir. (2021). Algeo #19 Nilai Eigen dan Vektor Eigen (Bagian 2). Diakses pada 3 November 2021, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian2.pdf>
- Informatika.stei.itb.ac.id/~rinaldi.munir. (2021). Algeo #19b Singular Value Decomposition (SVD). Diakses pada 3 November 2021, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf>
- cmi.ac.in. (2013, November 15). Image Compression and Linear Algebra. Diakses pada 1 November 2021, dari <https://www.cmi.ac.in/~ksutar/NLA2013/imagecompression.pdf>
- machinelearningmastery.com. (2018, 26 Februari). How to Calculate the SVD from Scratch with Python. Diakses pada 2 November 2021, dari <https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/>
- towardsdatascience.com. (2021, 31 Januari). Simple SVD algorithms. Diakses pada 1 November 2021, dari <https://towardsdatascience.com/simple-svd-algorithms-13291ad2eef2>
- pythonnumericalmethods.berkeley.edu. (2020). Eigenvalues and Eigenvectors in Python. Diakses pada 3 November 2021, dari <https://pythonnumericalmethods.berkeley.edu/notebooks/chapter15.04-Eigenvalues-and-Eigenvectors-in-Python.html>
- pythonnumericalmethods.berkeley.edu. (2020). The QR Method. Diakses pada 3 November 2021, dari <https://pythonnumericalmethods.berkeley.edu/notebooks/chapter15.03-The-QR-Method.html>
- jeremykun.com. (2016). Singular Value Decomposition Part 2: Theorem, Proof, Algorithm. Diakses pada 5 November 2021, dari <https://jeremykun.com/2016/05/16/singular-value-decomposition-part-2-theorem-proof-algorithm/>
- johnfoster.pge.utexas.edu. (2020). Singular Value Decomposition. Diakses pada 5 November 2021, dari https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_SVD.html
- core.ac.uk. (2016). Algorithms for Large Scale Problems in Eigenvalue and Svd Computations and in Big Data Applications. Diakses pada 6 November 2021, dari <https://core.ac.uk/download/pdf/235413726.pdf>
- danielkhashabi.com. (2015, 2 Maret). Singular Value Decomposition: Theory and Applications. Diakses pada 8 November 2021, dari <http://danielkhashabi.com/learn/svd.pdf>
- github.com. (2020, 7 Agustus). FileSaver.js. Diakses pada 10 November 2021, dari <https://github.com/eligrey/FileSaver.js/>

