

**LAPORAN TUGAS BESAR 2**  
**IF2123 ALJABAR LINIER DAN GEOMETRI**



Disusun oleh:

Kelompok 51 SVDed

13520051 – Flavia Beatrix Leoni A. S.

13520073 – Lyora Felicya

13520097 – Angelica Winasta Sinisuka

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

**2021**

## Daftar Isi

Daftar Isi .....	i
Daftar Gambar .....	ii
BAB 1 Deskripsi Masalah .....	1
BAB 2 Teori Singkat .....	2
2.1    Perkalian Matriks .....	2
2.2    Nilai Eigen.....	2
2.3    Vektor Eigen .....	2
2.4    Matriks SVD .....	3
BAB 3 Implementasi Program.....	4
BAB 4 Eksperimen .....	8
BAB 5 Kesimpulan.....	13
5.1 Kesimpulan .....	13
5.2 Saran .....	13
5.3 Refleksi .....	13
Daftar Referensi .....	15

## Daftar Gambar

Gambar 1 Tampilan halaman utama website .....	8
Gambar 2 Tampilan halaman hasil kompresi gambar .....	8
Gambar 3 Gambar asli 1 .....	9
Gambar 4 Hasil kompresi gambar 1 dengan $k = 15$ .....	9
Gambar 5 Hasil kompresi gambar 1 dengan $k = 30$ .....	9
Gambar 6 Hasil kompresi gambar 1 dengan $k = 50$ .....	9
Gambar 7 Hasil kompresi gambar 1 dengan $k = 100$ .....	9
Gambar 8 Gambar asli 2 .....	10
Gambar 9 Hasil kompresi gambar 2 dengan $k = 15$ .....	10
Gambar 10 Hasil kompresi gambar 2 dengan $k = 25$ .....	10
Gambar 11 Hasil kompresi gambar 2 dengan $k = 35$ .....	10
Gambar 12 Hasil kompresi gambar 2 dengan $k = 45$ .....	10
Gambar 13 Gambar asli 3 .....	11
Gambar 14 Hasil kompresi gambar 3 dengan $k = 15$ .....	11
Gambar 15 Hasil kompresi gambar 3 dengan $k = 30$ .....	11
Gambar 16 Hasil kompresi gambar 3 dengan $k = 50$ .....	11
Gambar 17 Hasil kompresi gambar 3 dengan $k = 100$ .....	11

## BAB 1

### Deskripsi Masalah

Membuat 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 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 anda buat (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 2**

### **Teori Singkat**

#### **2.1 Perkalian Matriks**

Sebuah matriks dapat dikalikan dengan sebuah bilangan skalar ataupun dengan matriks lainnya. Pada operasi perkalian skalar, sebuah matriks dikalikan dengan bilangan skalar. Jika diketahui  $A$  merupakan suatu matriks dan  $K$  merupakan bilangan real, maka hasil perkalian  $K$  dengan matriks  $A$  adalah matriks yang diperoleh dengan mengalikan setiap elemen  $A$  dengan  $K$ . Namun, untuk perkalian dua matriks terdapat aturan tersendiri. Syarat dua buah matriks, misal matriks  $A$  dan matriks  $B$ , dapat dikalikan adalah jika banyaknya kolom matriks  $A$  sama dengan banyaknya baris matriks  $B$ .

Untuk mencari hasil kali matriks  $A$  dengan matriks  $B$  ialah dengan mengalikan elemen pada baris-baris matriks  $A$  dengan elemen pada kolom-kolom matriks  $B$ , kemudian jumlahkan hasil perkalian antara baris dan kolom tersebut.

#### **2.2 Nilai Eigen**

Kata “eigen” berasal dari Bahasa Jerman yang artinya “asli” atau “karakteristik”. Dengan kata lain, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran  $n \times n$ . Jika  $A$  adalah sebuah matriks berukuran  $n \times n$ , maka  $\lambda$  adalah nilai eigen dari  $A$  jika dan hanya jika ia memenuhi persamaan

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

Persamaan tersebut disebut persamaan karakteristik dari  $A$ .

#### **2.3 Vektor Eigen**

Apabila  $A$  merupakan matriks  $n \times n$  maka vektor tidak nol  $\mathbf{x}$  di  $R^n$  disebut vektor eigen dari  $A$  apabila  $A\mathbf{x}$  sama dengan perkalian skalar  $\lambda$  dengan  $\mathbf{x}$  yaitu

$$A\mathbf{x} = \lambda\mathbf{x}$$

$\mathbf{x}$  disebut sebagai vektor eigen yang berhubungan atau berkoresponden dengan  $\lambda$ . Selain itu, vektor eigen  $\mathbf{x}$  menyatakan vektor kolom yang jika dikalikan dengan sebuah matriks  $n \times n$  menghasilkan vektor lain yang merupakan kelipatan dari vektor itu.

## 2.4 Matriks SVD

Suatu matriks  $A$  berukuran  $m \times n$  dapat didekomposisi menjadi matriks  $U$ ,  $\Sigma$ , dan  $V$  sedemikian sehingga

$$A = U\Sigma V^T$$

dimana  $U$  adalah matriks ortogonal berukuran  $m \times m$ ,  $V$  adalah matriks ortogonal berukuran  $n \times n$ , dan  $\Sigma$  adalah matriks berukuran  $m \times n$  yang elemen-elemen diagonal utamanya adalah nilai-nilai singular dari matriks  $A$  dan elemen-elemen lainnya adalah nol.

Diagonal utama sebuah matriks biasanya didefinisikan pada matriks persegi berukuran  $n \times n$ . Untuk matriks non-persegi, diagonal utama untuk matriks berukuran  $m \times n$  didefinisikan pada garis yang dimulai dari sudut kiri atas terus ke bawah matriks sejauh mungkin.

Matriks ortogonal adalah matriks yang kolom-kolomnya adalah vektor yang saling ortogonal satu sama lain (hasil kali titik sama dengan nol). Jika vektor-vektor tersebut merupakan vektor satuan, maka matriks ortogonal tersebut dinamakan juga matriks ortonormal. Vektor satuan adalah vektor yang dinormalisasi dengan panjang atau *magnitude*-nya sehingga memiliki panjang atau *magnitude* sama dengan satu. Jika  $Q$  adalah matriks ortogonal  $m \times n$  dan kolom-kolom matriks  $Q$  adalah vektor-vektor satuan  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$  maka  $\mathbf{v}_i \cdot \mathbf{v}_j = 0$  untuk  $i \neq j$ . Matriks  $Q$  juga dapat dikatakan sebagai matriks ortogonal jika  $Q^T Q = I$ , dalam hal ini  $I$  adalah matriks identitas.

Misalkan  $A$  adalah matriks  $m \times n$ . Jika  $\lambda_1, \lambda_2, \dots, \lambda_n$  adalah nilai-nilai eigen dari  $A^T A$ , maka

$$\sigma_1 = \sqrt{\lambda_1}, \sigma_2 = \sqrt{\lambda_2}, \dots, \sigma_n = \sqrt{\lambda_n}$$

disebut nilai-nilai singular dari matriks  $A$ . Diasumsikan  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$  sehingga  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ .

## BAB 3

### Implementasi Program

Program ini menggunakan bahasa pemrograman Python. *Website* kompresi gambar yang kami buat memanfaatkan *framework* Flask untuk *back-end* serta HTML dan CSS untuk *front-end*. Adapun *library* yang digunakan yaitu Numpy untuk mengolah matriks dan PIL untuk mengolah gambar. *Website* mampu menerima *file* gambar beserta *input* tingkat kompresi gambar, dalam hal ini yaitu banyak nilai singular ( $k$ ) yang ingin digunakan, kemudian menampilkan gambar *input*, *output*, *runtime* algoritma, serta persentase hasil kompresi gambar. File *output* hasil kompresi juga dapat diunduh melalui *website*.

Terdapat banyak algoritma untuk mendapatkan nilai eigen dan vektor eigen yang dapat digunakan untuk mendekomposisi matriks. Pada program ini, kami memanfaatkan QR *factorization* untuk mendapatkan nilai eigen dan vektor eigen. Schur *decomposition* menyatakan bahwa untuk setiap matriks  $A$  berukuran  $n \times n$ , terdapat matriks uniter  $Q$  dan matriks segitiga atas  $T$  sehingga matriks  $A$  dapat ditulis sebagai

$$A = Q T Q^*$$

$Q$  adalah matriks uniter sehingga

$$Q^* = Q^{-1}$$

Kalikan persamaan matriks  $A$  dengan  $Q^*$  di sebelah kiri dan  $Q$  di sebelah kanan sehingga

$$A = Q T Q^*$$

$$Q^* A Q = T$$

Dekomposisi persamaan ini menjadi

$$(Q_k^* \dots Q_1^*) A (Q_1 \dots Q_k) = T$$

dengan  $k$  menuju tak hingga ( $\infty$ ).

Kelompokkan persamaan tersebut menjadi

$$A_1 = Q_1^* A Q_1$$

$$A_2 = Q_2^* Q_1^* A Q_1 Q_2 = Q_2^* A_1 Q_2$$

...

$$A_k = (Q_k^* \dots Q_1^*) A (Q_1 \dots Q_k) = Q_k^* Q_{k-1} Q_k = T$$

Diperoleh bahwa

$$Q_x^* A_{x-1} = R_x$$

$$A_{x-1} = Q_x R_x$$

dimana  $U_x$  merupakan matriks ortogonal dan  $R_x$  merupakan matriks segitiga atas yang dapat diperoleh dengan melakukan QR *factorization* pada A serta x merupakan bilangan bulat antara 1 dan k. Substitusi persamaan ini pada persamaan sebelumnya sehingga

$$A_x = Q_x^* A_{x-1} Q_x = R_x Q_x$$

Oleh karena itu, dengan melakukan QR *factorization* secara terus menerus pada  $A_x$ , kita dapat memperoleh suatu matriks  $A_k$  yang menyerupai matriks diagonal. Elemen diagonal dari matriks  $A_k$  merupakan nilai-nilai eigen dari A dan perkalian antara setiap Q yang diperoleh dari setiap QR *factorization* menghasilkan suatu matriks vektor eigen dari A.

Dengan metode dekomposisi SVD, suatu matriks A dapat didekomposisi menjadi 3 matriks, yaitu matriks ortogonal U, matriks diagonal S, dan transpos dari matriks ortogonal V. Gunakan algoritma yang telah dijelaskan sebelumnya untuk mencari U yaitu matriks vektor eigen dari  $AA^T$  dan V yaitu matriks vektor eigen dari  $A^T A$ . Sebuah matriks persegi A dikatakan ortogonal jika transposnya sama dengan inversnya sehingga matriks S dapat diperoleh dengan

$$S = (U^T U) S (V^T V) = U^T (USV^T) V = U^T A V$$

Algoritma kompresi diimplementasikan dengan memanfaatkan konsep SVD yang sudah dijelaskan sebelumnya. Pada dasarnya, gambar berwarna terdiri dari array 3 dimensi yang berukuran  $n \times m \times 3$ , dimana n dan m menyatakan jumlah pixel dari gambar. Proses kompresi dimulai dari mengubah *input* gambar menjadi array dengan menggunakan Numpy.array. Selanjutnya array gambar tersebut akan dibagi menjadi 3 komponen yaitu R, G, B dan masing-masing array tersebut akan didekomposisi menjadi matriks U, S, dan VT dan direkonstruksi kembali berdasarkan *input* jumlah nilai singularnya yaitu k dengan mengambil kolom dan baris sebanyak k dari U dan V serta nilai singular sebanyak k dari S. Terakhir, ketiga komponen tersebut akan disatukan kembali untuk membentuk gambar dengan ukuran yang telah dikompres, menghasilkan ukuran gambar yang lebih kecil dari sebelumnya.



Nama program utama secara garis besar adalah app.py yang berisi program utama dan route yang ada serta templates yang berisi HTML dan tampilan CSS. Berikut merupakan fungsi dan spesifikasi yang ada dalam program kami.

**a. svd.py**

Nama Fungsi	Spesifikasi
isDiagonal(A)	Fungsi untuk mengecek apakah suatu matriks adalah matriks diagonal A merupakan tipe data matriks
eigen(A)	Menerima parameter A berupa matriks Menghasilkan suatu matriks vektor eigen dari A (proses lebih lanjut telah dijelaskan sebelumnya)
svd(A)	Menerima parameter A berupa matriks Mengembalikan matriks U dan V dari hasil vektor eigen, serta matriks S dari rumus $S = U^T A V$

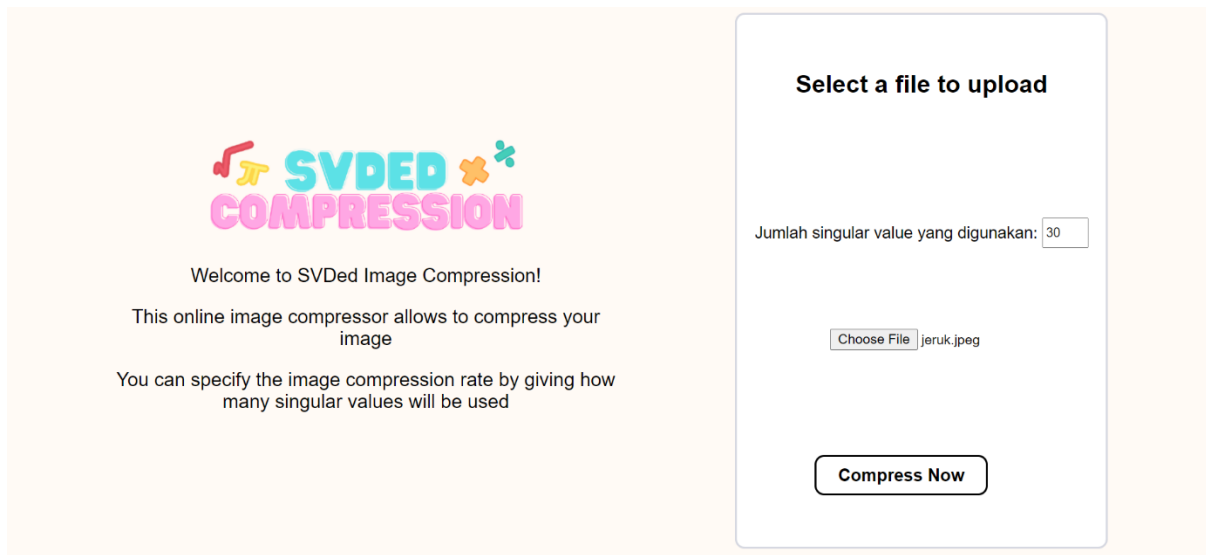
**b. app.py**

Jenis Fungsi	Nama Fungsi	Spesifikasi
Fungsi Tanpa Routing	allowed_file(filename)	Menerima parameter string filename kemudian Mengembalikan true jika <i>extension file</i> termasuk <i>extension</i> yang diperbolehkan
	openImage(imagePath)	Menerima parameter string dari pathimage yang ingin dibuka Mengembalikan array yang terdiri dari r,g,dan b serta object image
	compression(A,k)	Menerima parameter matriks A Fungsi ini mengompres setiap matriks r,g,b dengan dengan acuan <i>singular value</i> (k) Mengembalikan <i>object image</i> yang sudah dikompres

Fungsi dengan routing	home('/')	Merender template image.html (halaman pertama)
	uploads('/')	Menggunakan <i>method post</i> untuk menerima <i>input file</i> dari <i>user</i> dan <i>singular value</i> yang dimasukkan. Kemudian fungsi ini akan mengecek jika input user benar atau tidak, jika benar maka <i>file</i> pengguna akan disimpan dan <i>redirect</i> ke route <i>‘/compressed/&lt;filename&gt;/&lt;k_value&gt;’</i>
	compressing('/compressed/<filename>/<k_value>')	Menerima parameter filename dan k_value. Fungsi ini akan mengompres <i>image</i> kemudian mengembalikan hasil dari pemrosesan ke <i>render_template('image2.html)</i> atau halaman kedua <i>website</i>
	downloading('/download')	Fungsi ini memberikan <i>attachment file</i> pada <i>button download</i> di halaman kedua <i>website</i> agar gambar dapat diunduh.

## BAB 4

### Eksperimen



The screenshot shows the main interface of the SVDed Image Compression website. On the left, there is a welcome message and a brief description of the tool. On the right, there is a form for uploading a file and specifying the number of singular values to use for compression.

**SVDed COMPRESSION**

Welcome to SVDed Image Compression!

This online image compressor allows to compress your image

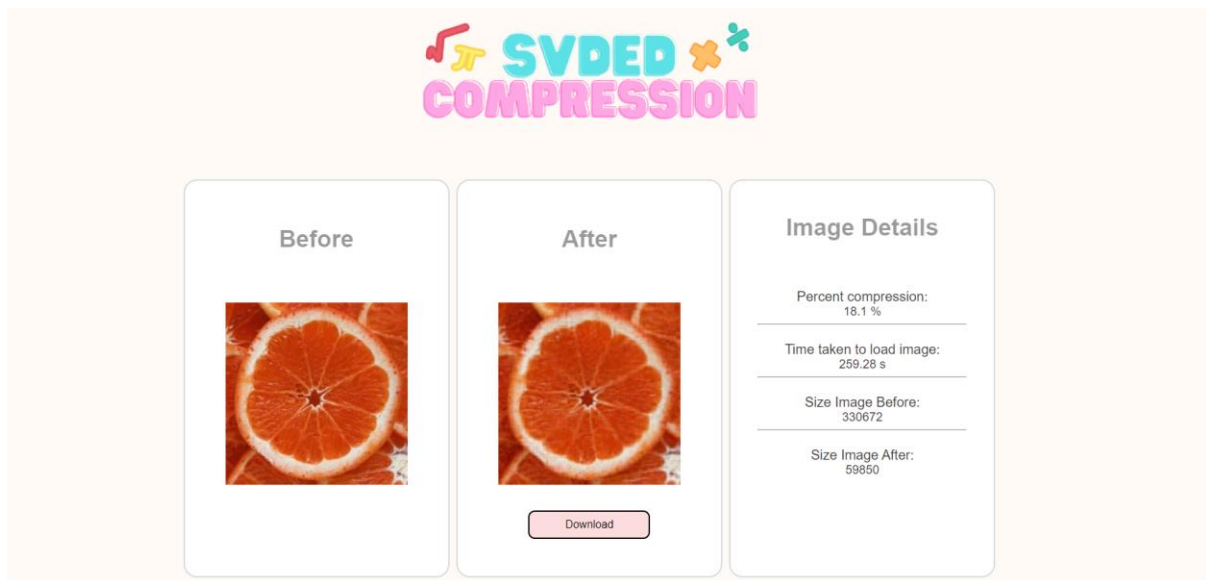
You can specify the image compression rate by giving how many singular values will be used

**Select a file to upload**

Jumlah singular value yang digunakan:

jeruk.jpeg

*Gambar 1 Tampilan halaman utama website*



The screenshot shows the result of the image compression process. It displays the original image (Before) and the compressed image (After) side-by-side. A 'Download' button is provided for the compressed image. To the right, there is a section for 'Image Details' showing the compression statistics.

**SVDed COMPRESSION**

**Before**

**After**

**Image Details**

Percent compression: 18.1 %

Time taken to load image: 259.28 s

Size Image Before: 330672

Size Image After: 59850

*Gambar 2 Tampilan halaman hasil kompresi gambar*

## 1. Percobaan 1



*Gambar 3 Gambar asli 1*

Gambar Asli

332 x 332 pixels

Ukuran sebelum kompresi :  
330677



*Gambar 4 Hasil kompresi gambar 1 dengan k = 15*



*Gambar 5 Hasil kompresi gambar 1 dengan k = 30*



*Gambar 6 Hasil kompresi gambar 1 dengan k = 50*



*Gambar 7 Hasil kompresi gambar 1 dengan k = 100*

	k = 15	k = 30	k = 50	k = 100
<b>Ukuran setelah kompresi</b>	29925	59850	99750	199500
<b>Runtime Algoritma</b>	80.94 s	82.45 s	82.74 s	83.26 s
<b>Persentase hasil kompresi</b>	9.05 %	18.1 %	30.17 %	60.33 %

Pada percobaan gambar pertama, dapat dilihat bahwa gambar berukuran 332 x 332 pixels membutuhkan ruang penyimpanan sebesar 330677 untuk menyimpan gambar. Setelah dilakukan kompresi dengan mengambil nilai singular sebanyak 15, 30, 50, dan 100, dapat dilihat bahwa ruang penyimpanan yang dibutuhkan menjadi jauh lebih kecil, dengan kualitas gambar yang cukup baik. Untuk  $k = 15$ , *runtime* nya yang paling singkat yaitu 80.94 s serta ukuran setelah kompresi juga menjadi jauh lebih kecil, namun dapat dilihat bahwa kualitas gambarnya sangat menurun. Berbeda dengan percobaan untuk  $k = 50$  dan  $k = 100$ , kualitas gambar hasil kompresi sudah cukup mirip dengan gambar asli, dengan ruang penyimpanan

yang lebih kecil. Namun, perlu diperhatikan bahwa semakin banyak jumlah nilai singular yang digunakan, semakin lama pula *runtime* algoritma yang dibutuhkan untuk mengompres gambar tersebut.

## 2. Percobaan 2



Gambar 8 Gambar asli 2

Gambar Asli

100 x 100 pixels

Ukuran sebelum kompresi :  
30000



Gambar 9 Hasil kompresi gambar 2 dengan  $k = 15$



Gambar 10 Hasil kompresi gambar 2 dengan  $k = 25$



Gambar 11 Hasil kompresi gambar 2 dengan  $k = 35$



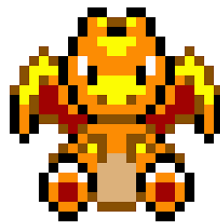
Gambar 12 Hasil kompresi gambar 2 dengan  $k = 45$

	k = 15	k = 25	k = 35	k = 45
Ukuran setelah kompresi	9045	15075	21105	27135
Runtime Algoritma	2.60 s	2.60 s	2.58 s	2.57 s
Persentase hasil kompresi	30.15 %	50.25 %	70.35 %	90.45 %

Pada percobaan gambar kedua, dapat dilihat bahwa gambar berukuran 100 x 100 pixels membutuhkan ruang penyimpanan sebesar 30000 untuk menyimpan gambar. Setelah dilakukan kompresi dengan mengambil nilai singular sebanyak 15, 25, 35, dan 45, dapat dilihat

bahwa ruang penyimpanan yang dibutuhkan menjadi jauh lebih kecil, dengan kualitas gambar yang cukup baik. Untuk setiap  $k$ , *runtime* kurang lebih membutuhkan antara  $\pm 2.57$ s serta ukuran setelah kompresi juga menjadi jauh lebih kecil, namun dapat dilihat bahwa kualitas gambarnya sangat menurun. Berbeda dengan percobaan untuk  $k = 35$  dan  $k = 45$ , kualitas gambar hasil kompresi sudah cukup mirip dengan gambar asli, dengan ruang penyimpanan yang lebih kecil.

### 3. Percobaan 3



*Gambar 13 Gambar asli 3*

Gambar Asli

225 x 225 pixels

Ukuran sebelum kompresi :  
151875



*Gambar 14 Hasil kompresi gambar 3 dengan  $k = 15$*



*Gambar 15 Hasil kompresi gambar 3 dengan  $k = 30$*



*Gambar 16 Hasil kompresi gambar 3 dengan  $k = 50$*



*Gambar 17 Hasil kompresi gambar 3 dengan  $k = 100$*

	k = 15	k = 30	k = 50	k = 100
<b>Ukuran setelah kompresi</b>	20295	40590	67650	135300
<b>Runtime Algoritma</b>	35.83 s	37.50 s	30.94 s	38.32 s
<b>Persentase hasil kompresi</b>	13.36 %	26.73 %	44.54 %	89.09 %

Pada percobaan gambar ketiga, dapat dilihat bahwa gambar berukuran  $225 \times 225$  pixels membutuhkan ruang penyimpanan sebesar 151875 untuk menyimpan gambar. Setelah dilakukan kompresi dengan mengambil nilai singular sebanyak 15, 30, 50, dan 100, dapat dilihat bahwa ruang penyimpanan yang dibutuhkan menjadi jauh lebih kecil, dengan kualitas gambar yang cukup baik. Secara umum, *runtime* algoritma meningkat seiring  $k$  yang bertambah, tetapi pada  $k = 50$ , *runtime* nya paling singkat, yaitu 30.94 s. Sementara itu, *runtime* dengan  $k = 15$  yaitu 35.83 s dan *runtime* dengan  $k = 100$  yaitu 38.32 s. Pengambilan nilai  $k$  yang semakin kecil juga menyebabkan ukuran gambar setelah kompresi yang lebih kecil. Pada gambar ini, dapat dilihat bahwa dengan nilai  $k$  berapapun, kualitas gambar hasil kompresi tetap menyerupai dengan gambar asli.

Dari percobaan di atas, dapat dilihat bahwa sebuah gambar dapat direkonstruksi dengan banyak *singular values*  $k$  dengan mengambil kolom dan baris sebanyak  $k$  dari  $U$  dan  $V$  serta singular value sebanyak  $k$  dari  $S$  atau  $\Sigma$  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 values* karena kebanyakan informasi disimpan di *singular values* awal karena *singular values* terurut mengecil. Dengan mengambil beberapa nilai singular pertama dari sebuah gambar, dapat dihasilkan ukuran gambar yang jauh lebih kecil namun tetap mempertahankan kualitas dari gambar tersebut.

## **BAB 5**

### **Kesimpulan**

#### **5.1 Kesimpulan**

- 5.1.1 *Website* dapat digunakan untuk melakukan kompresi gambar sesuai dengan jumlah *singular value* yang diinginkan. Setelah menerima *input file* gambar dan tingkat kompresi berupa jumlah *singular value* yang digunakan dari pengguna, *website* akan menampilkan gambar sebelum dan setelah kompresi, persentase hasil kompresi gambar, *runtime* algoritma, serta ukuran gambar sebelum dan setelah kompresi. Pengguna juga dapat mengunduh *file* gambar hasil kompresi.
- 5.1.2 Terdapat banyak *library* yang disediakan Python yang sangat membantu dalam proses pengerjaan *website* kompresi gambar ini.
- 5.1.3 Pada percobaan eksperimen kompres gambar, waktu yang dibutuhkan bervariasi. Dengan gambar dan  $k$  yang sama, *runtime* yang diperlukan dapat berbeda pula. Hal ini dapat terjadi karena pengaruh variasi arsitektur komputer seperti pengaruh cache atau komputer yang mengolah berbagai hal.

#### **5.2 Saran**

- 5.2.1 Dilakukan pengembangan program agar dapat mempertahankan transparansi dari gambar asli.
- 5.2.2 Dilakukan pengembangan program agar gambar hasil kompresi dapat diunduh dalam berbagai format.

#### **5.3 Refleksi**

- 5.3.1 Pembuatan tugas besar ini menambah wawasan baru mengenai pembuatan *website* terutama dengan *framework* Flask.
- 5.3.2 Pembuatan tugas besar ini memperluas wawasan kami mengenai cara-cara yang dapat dilakukan untuk mencari nilai eigen, vektor eigen, dan mendekomposisi matriks dengan metode SVD.
- 5.3.3 Diperlukan adanya komunikasi yang baik antaranggota agar tugas besar ini dapat diselesaikan dengan baik.



5.3.4 Diperlukan pemahaman yang lebih agar dapat menggunakan SVD untuk mengompres gambar serta mencari solusi lain agar program dapat berjalan dengan efektif.

## Daftar Referensi

- Driscoll, Toby. *QR algorithm for eigenvalues*. Youtube, diunggah oleh Toby Driscoll, 20 April 2016, [https://youtu.be/\\_neGVEBjLJA](https://youtu.be/_neGVEBjLJA).
- Dyori, Abdelhadi, *How To Make a Web Application Using Flask in Python 3*, 16 April 2020, <https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3>.
- geeksforgeeks.org. Python: Pillow (a fork of PIL). 3 Juli 2021. diakses dari <https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/>.
- Mathews, Brady. 2014. *Image Compression using Singular Value Decomposition (SVD)*. The University of Utah.
- Munir, Rinaldi. 2021. *Nilai Eigen dan Vektor Eigen (Bagian 1)*. Bandung: Program Studi Teknik Informatika STEI-ITB.
- Munir, Rinaldi. 2021. *Singular Value Decomposition (SVD)*. Bandung: Program Studi Teknik Informatika STEI-ITB.
- Tutorials, Easy. *How To Make Website Using HTML & CSS/Full Responsive Multi Page Website Design Step by Step*. Youtube, diunggah oleh Easy Tutorials, 15 Maret 2021, <https://youtu.be/oYRda7UtuhA>.