

Tugas Besar 2 IF2123 Aljabar Linier dan Geometri
Kompresi Gambar Menggunakan Algoritma *Singular Value Decomposition*



13520016 - Gagas Praharsa Bahar

13520045 - Addin Nabilal Huda

135200101 - Aira Thalca Avila Putra

DAFTAR ISI

HALAMAN COVER	1
DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
BAB 2 TEORI SINGKAT	5
Perkalian Matriks	5
Nilai Eigen	6
Vektor Eigen	6
Algoritma Singular Value Decomposition (SVD)	6
BAB 3 IMPLEMENTASI PROGRAM	8
Library yang digunakan	8
Fungsi yang dipakai:	8
Algoritma yang digunakan	11
BAB 4 EKSPERIMEN	15
User Interface	15
Studi Kasus	16
BAB 5 KESIMPULAN	24
DAFTAR PUSTAKA	25

BAB 1

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.

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.

Gambar 1 Ilustrasi algoritma SVD dengan rank k

Matriks U adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks $A^T A$. 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 $A A^T$. Matriks ini menyimpan informasi yang penting terkait kolom-kolom matriks awal, dengan informasi terpenting disimpan dalam baris pertama.

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 rekonstruksi dari SVD akan berupa informasi dari gambar yang terkompresi dengan ukuran yang lebih kecil dibanding gambar awal.

BAB 2

TEORI SINGKAT

2.1 Perkalian Matriks

Perkalian matriks adalah perkalian yang melibatkan suatu matriks atau susunan bilangan berupa kolom dan angka, serta memiliki sifat-sifat tertentu. Agar dua matriks dapat dikalikan, banyaknya kolom pada matriks pertama harus sama dengan banyak baris pada matriks kedua. Hasil perkalian matriks berupa matriks dengan baris sebanyak baris pada matriks pertama dan kolom sebanyak kolom pada matriks kedua. Perkalian matriks menjadi konsep dasar dalam aljabar linear dan memiliki banyak penerapan di bidang matematika terapan, statistika, ekonomi, dan teknik.

Jika A adalah matriks berukuran $m \times n$ dan B adalah baris berukuran $n \times p$ sebagai berikut,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

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

Gambar 2 Perkalian matriks

Maka hasil perkalian kedua matriks tersebut, $\mathbf{C}=\mathbf{AB}$ adalah sebuah matriks berukuran $m \times p$. Dengan setiap entri pada matriks C didefinisikan sebagai

$$c_{ij} a_{i1} b_{1j} + \dots + a_{in} b_{nj} = \sum_{k=1}^n c_{ij} a_{ik} b_{kj} \text{ untuk nilai } i = 1, \dots, m \text{ dan nilai } j = 1, \dots, p.$$

2.2 Nilai Eigen

“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 $n \times n$, maka sebuah vektor taknol x pada \mathbb{R}^n disebut vektor eigen dari A . Jika Ax sama dengan perkalian suatu skalar λ dari x , yaitu

$$Ax = \lambda x$$

untuk skalar sebarang λ . Skalar λ inilah yang disebut sebagai nilai eigen dari matriks A . λ adalah sebuah solusi yang memenuhi persamaan karakteristik $\det(\lambda I - A) = 0$ atau $\det(A - \lambda I) = 0$. Apabila diperluas lagi, $\det(\lambda I - A) = 0$ atau $\det(A - \lambda I) = 0$ adalah sebuah polinomial pdalam variabel λ yang disebut sebagai polinomial karakteristik dari matriks A . Pernyataan berikut yang ekuivalen dengan pernyataan “Jika A adalah matriks berukuran $n \times n$ ” di antaranya:

- a. λ adalah sebuah nilai eigen dari A
- b. λ adalah sebuah solusi dari persamaan karakteristik $\det(\lambda I - A) = 0$
- c. Sistem persamaan $(\lambda I - A) = 0$ mempunyai solusi tak trivial sehingga $Ax = \lambda x$
- d. Terdapat sebuah vektor x yang taknol sehingga

2.3 Vektor Eigen

Dari penjelasan pada 2.2, didapatkan x merupakan vektor eigen dari A . Jika Ax sama dengan perkalian suatu skalar λ dari x , yaitu

$$Ax = \lambda x$$

untuk skalar sebarang λ pada sebuah matriks A berukuran $n \times n$.

2.4 Algoritma Singular Value Decomposition (SVD)

Algoritma Singular Value Decomposition merupakan metode matematis yang digunakan untuk menguraikan matriks tunggal dengan mengompres menjadi tiga matriks. Algoritma ini pertama kali diusulkan oleh Eckartand Young dan termasuk algoritma teknik pengurangan dimensi yang paling cepat proses kerjanya bila dibandingkan dengan teknik pengurangan dimensi lainnya seperti PCA, ICA, dan fastICA(extensionICA). Beberapa penerapan algoritma ini di antaranya *clustering* pada data klinik, *image compressing*, *watermarking*, klasifikasi dokumen, *mapping gen*, dan pencarian data series. Algoritma SVD mempunyai kelebihan pada efisiensi waktu proses untuk digunakan pada data yang berskala besar. Algoritma SVD memfaktorkan matriks A berukuran $m \times n$ menjadi matriks U , Σ , dan V sedemikian sehingga

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

Dengan U merupakan matriks orthogonal berukuran $m \times m$, V adalah matriks orthogonal berukuran $n \times n$, dan Σ adalah matriks berukuran $m \times n$ yang elemen-elemen diagonal utamanya adalah nilai-nilai singular dari matriks A dan elemen-elemen lainnya adalah 0.

BAB 3

IMPLEMENTASI PROGRAM

Frontend: React

Backend: Flask

Program ini dibagi menjadi dua bagian, yaitu bagian backend untuk mengolah gambar dan bagian frontend untuk mengatur interaksi user dengan program. Bagian backend memanfaatkan framework Flask, yaitu suatu kerangka kerja mikro dalam pengembangan website berbasis bahasa Phyton. Sedangkan, bagian frontend menggunakan framework React, yaitu kerangka kerja yang digunakan untuk membangun User Interface dalam JavaScript.

3.1. Library yang digunakan

1. Python (Backend)

- a. Numpy
- b. Pillow
- c. OS
- d. Logging
- e. Flask, Flask_cors
- f. Werkzeug

2. Javascript (Frontend)

a. Reactstrap

Reactstrap merupakan library komponen untuk React yang menyediakan inbuild Bootstrap. Program ini memanfaatkan Reactstrap untuk membangun komponen-komponen *user interface* seperti Card, Container, Row, Col, Navbar, dan Button.

b. React-dropzone

React dropzone merupakan library yang menyediakan komponen untuk mengatur interaksi *drag and drop* file pada kerangka kerja React.

c. React-icons

React icons merupakan komponen-komponen bawaan React yang digunakan untuk menampilkan ikon-ikon pada *user interface*.

3.2. Fungsi yang dipakai:

1. app.py (Routing dengan Flask)

a. Route /upload: (method: POST)

i. handleFileUpload()

Fungsi ini akan menangani saat ada file gambar yang ter-upload ke dalam dropzone yang berada pada laman web beserta dengan k, konstanta kompresi.

b. Route /compress/<k>_<filename>: (method: GET)

i. compressHandler(filename, k)

Fungsi ini akan mengompres file sesuai dengan k lalu menyimpannya pada storage backend.

c. Route /download/<k>_<filename>:

i. download(filename)

Fungsi ini akan mengembalikan URL untuk mendownload gambar yang telah dikompres

d. Route /data (view function)

i. getcurrentData()

Fungsi ini akan mengembalikan data terakhir yang berisi data differential percentage dan waktu yang diperlukan untuk mengecilkan gambar

2. compress.py

Nama	Hasil return	Parameter	Deskripsi
SVD	Matrix dekomposisi U,S,V.T	Matrix compression rate,error	Mencari nilai U,S, dan V dari SVD matrix yang menjadi parameter
compress_image	Matrix hasil kompresi R,G,B	Image,compression rate	Melakukan dekompisasi SVD dari seluruh matrix dalam image
arrangeRGB	Matrix yang sudah dikompresi sebelum diubah menjadi image	Image, matrix R,matrix G,matrix B	Menyatukan matrix RGB menjadi satu matrix
compress_function	Pixel difference, runtime algoritma,	Compression rate, file	Melakukan keseluruhan proses kompresi mulai dari kompres, menyatukan hingga mereturn pixel difference dan runtime algoritma

Tabel 1 Fungsi yang digunakan pada compress.py

3. app.js

a. handleChange

Fungsi ini digunakan untuk mengupdate nilai K sesuai dengan input dari pengguna.

b. handleUploadImage

Fungsi ini digunakan untuk mengupload image yang sudah di input pengguna kepada server backend, dan memunculkan preview yang ada pada website.

c. handleCompress

Fungsi ini digunakan untuk mengompres image yang sudah ada lalu menampilkan compression rate, compression time, dan image yang telah dikompres. Selain itu, fungsi ini juga memberikan download link untuk pengguna.

3.3. Algoritma yang digunakan

Algoritma yang kami gunakan untuk mencari nilai SVD adalah dengan menggunakan simultaneous power iteration.

```
15 ~ def svd(A, k, epsilon=1e-3):
16     A = np.array(A,dtype=np.float64)
17     m = A.shape[0]
18     n = A.shape[1]
19     X=A.copy()
20 ~     if m>=n:
21         A = np.dot(A.T,A)
22         val = n
23         val2 = m
24 ~     else:
25         A = np.dot(A,A.T)
26         val = m
27         val2 = n
28     now = np.random.rand(val, k)
29     now , r = np.linalg.qr(now)
30 ~     for i in range(10):
31         temp = now
32         Z = np.dot(A,now)
33         now, R = np.linalg.qr(Z)
34         error = ((now-temp) ** 2).sum()
35 ~         if error < epsilon:
36             break
37     sing=np.sqrt(np.abs(np.diag(R)))
38 ~     if m<n:
39         u=now
40 ~         try:
41             v=np.dot(np.linalg.inv(np.diag(sing)),np.dot(u.T,X))
42 ~         except np.linalg.LinAlgError as err:
43 ~             if 'Singular matrix' in str(err):
44                 return X
45 ~         else:
46             v=now.T
47 ~             try:
48                 u=np.dot(X,np.dot(v.T,np.linalg.inv(np.diag(sing))))
49 ~             except np.linalg.LinAlgError as err:
50 ~                 if 'Singular matrix' in str(err):
51                     return X
52     return u, sing, v
```

Gambar 3 Algoritma SVD yang digunakan

Baris 16 hingga 36 digunakan untuk mencari k eigenvalue terbesar dan eigenvectornya. Algoritma ini memanfaatkan dekomposisi QR dari matrix kiri(A dot A.T jika m<n) dan sebaliknya. Matrix R dan now yang didapat pada akhir loop masing masing adalah matrix yang berisi eigenvector dan juga eigenvalue. Lalu eigenvector tersebut harus kita akarkan untuk mendapatkan matrix S, sementara eigenvalue itu sendiri bisa menjadi matrix U ataupun V

bergantung perhitungan yang kita lakukan(A dot A transpose atau sebaliknya). Matrix yang belum dicari kita cari dengan menggunakan nilai S dan satu matrix lain yang sudah didapat.

Selain fungsi SVD, kami juga memiliki fungsi untuk menyatukan kembali matrix dari image yang sudah di pisahkan sesuai warna (R,G,B).

```
def arrangeRGB(img,rr,rg,rb):
    logger.info("arranging...")
    result = np.zeros(img.shape)
    result[:, :, 0] = rr
    result[:, :, 1] = rg
    result[:, :, 2] = rb
    for i in range(np.shape(result)[0]):
        for j in range(np.shape(result)[1]):
            for k in range(np.shape(result)[2]):
                if result[i,j,k] < 0:
                    result[i,j,k] = abs(result[i,j,k])
                if result[i,j,k] > 255:
                    result[i,j,k] = 255
    result = result.astype(np.uint8)
    return result
```

Gambar 4 Algoritma untuk fungsi arrangeRGB

Fungsi ini menyatukan kembali matrix warna R,G,B yang sudah dikompresi dan melakukan pengecekan nilai masing-masing nilai pada matrix karena nilai dari R,G,B yang mungkin adalah 0-255.

Fungsi terakhir yang cukup menarik adalah fungsi main kami.

```
101  def compress_function(k, image_name):
102      # TODO : Support other filetypes
103      pth = os.path.abspath(os.getcwd())
104      logger.info(pth)
105      start = time()
106      path = UPLOAD_FOLDER + image_name
107      pic = Image.open(path)
108      format = pic.format
109      mode = pic.mode
110      if(mode == 'P'):
111          pic = pic.convert('RGBA')
112      elif(mode == 'L'):
113          pic = pic.convert('RGB')
114      elif(mode == 'LA'):
115          pic = pic.convert('RGBA')
116      img = np.asarray(pic)
117      m = img.shape[0]
118      n = img.shape[1]
119      rr,rg,rb = compress_image(img,percentage(img,k))
120      result = arrangeRGB(img,rr,rg,rb)
121      if(img.shape[2] == 4):
122          result[:, :, 3] = img[:, :, 3]
123      image = Image.fromarray(result)
124      if(mode == 'P'):
125          image = image.convert('P')
126      elif(mode == 'L'):
127          image = image.convert('L')
128      elif(mode == 'LA'):
129          image = image.convert('LA')
130      pixelDiff = (m+n+1)*percentage(img,k) / (m*m+n*n+min(m,n)) * 100
131      logger.info(f"Diff (percentage): {100-pixelDiff}")
132      path = DOWNLOAD_FOLDER + "compressed_" + image_name
133      image.save(path, format)
134      timeTaken = time() - start
135      logger.info(f'Time taken to run: {timeTaken} seconds')
136
137      return {'diff': 100-pixelDiff, 'time': timeTaken}
```

Gambar 5 Algoritma untuk fungsi compress_function

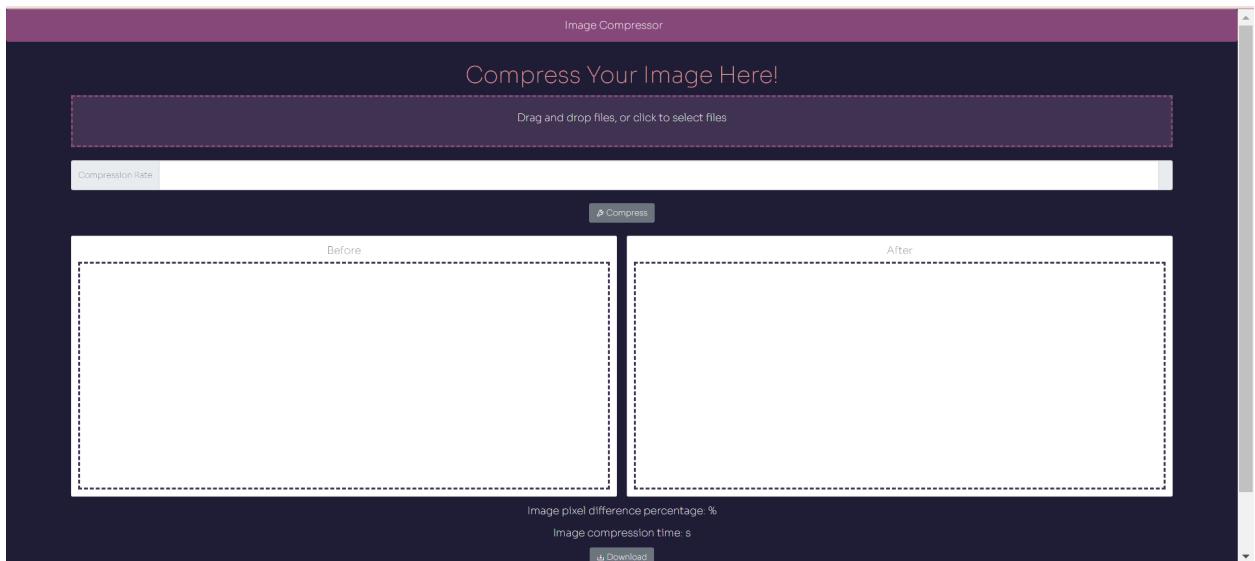
Fungsi ini menerima image yang akan di kompres, lalu jika image tersebut tidak memiliki mode warna RGB, maka akan diubah menjadi RGB atau RGBA jika ia memiliki transparensi. Image

itu lalu diolah dan dikompres. Setelah dikompresi, image yang didapat akan diconvert lagi menjadi mode sebelumnya(jika diubah). Fungsi mereturn nilai pixelDifference dan runtime algoritma.

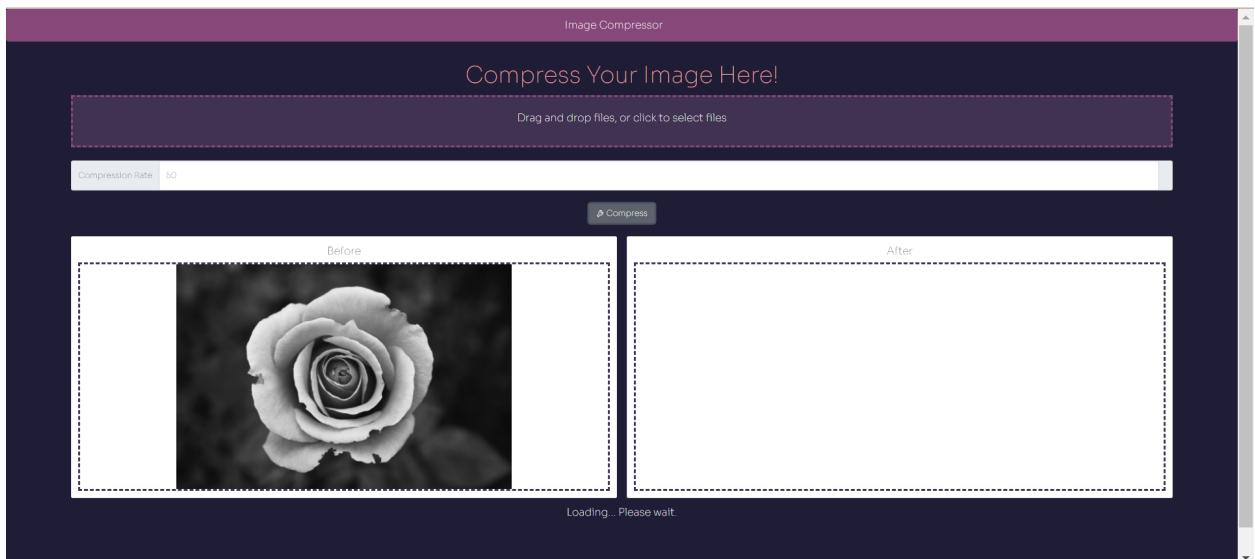
BAB 4

EKSPERIMEN

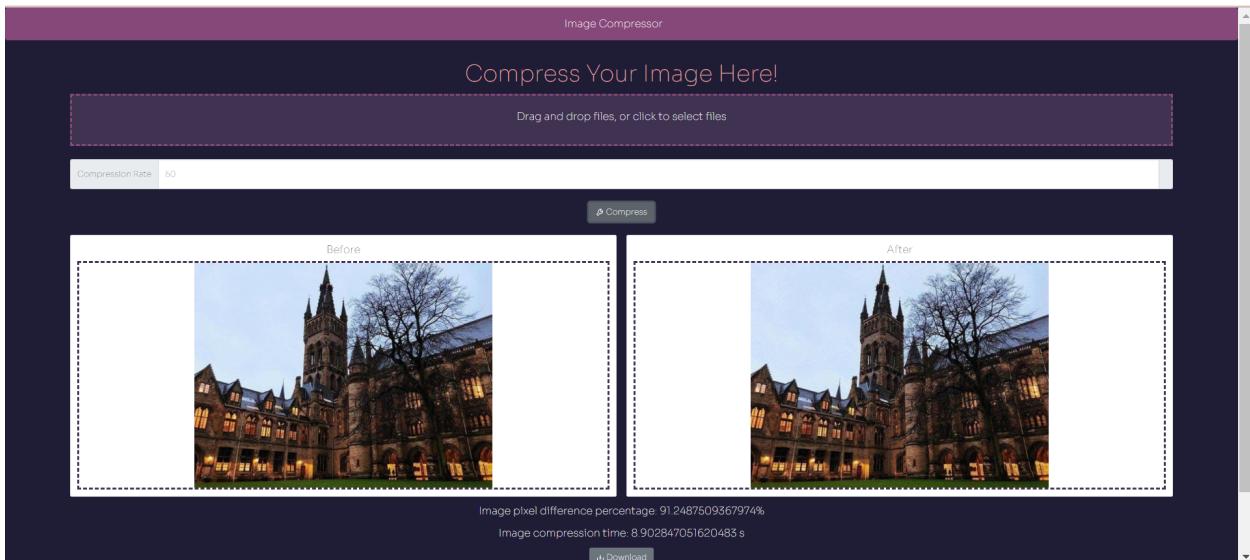
4.1 User Interface



Gambar 6 Tampilan website dengan zoom browser 67%



Gambar 7 Tampilan website ketika gambar sedang diproses



Gambar 8 Tampilan website ketika gambar sudah selesai diproses

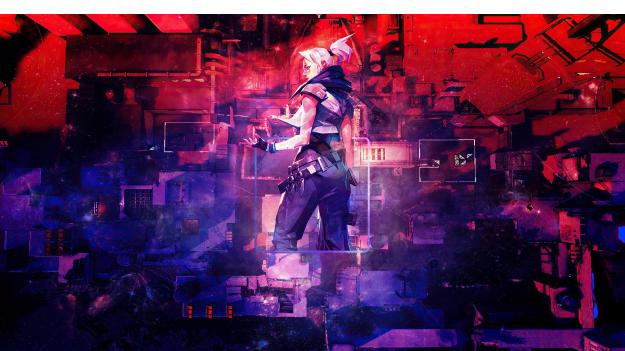
4.2 Studi Kasus

a. instagram.png

Gambar Asli	 The original Instagram logo, a white camera icon inside a rounded square with a colorful gradient background (purple, red, orange).	Ukuran asli: 166 kB Nama file: instagram.png
Compress dengan k = 15	 The same Instagram logo as above, but compressed with a quality setting of k=15. It appears slightly less vibrant but retains most of the original shape and icon.	Size setelah kompres: 95.7 kB Waktu yang dibutuhkan: 4.272430658340454 s <i>Pixel difference:</i> 98.5%

Compress dengan k = 30		Size setelah kompres: 112 kB Waktu yang dibutuhkan 5.399547338485718 s <i>Pixel difference: 97%</i>
Compress dengan k = 50		Size setelah kompres: 104 kB Waktu yang dibutuhkan: 6.815258026123047 s <i>Pixel difference: 95%</i>
Compress dengan k = 75		Size setelah kompres: 94.6 kB Waktu yang dibutuhkan: 8.302848815917969 s <i>Pixel difference: 92.5%</i>

b. 1095253.jpg

Gambar Asli		Ukuran asli: 6.58 MB Nama file: 1095253.jpg
Compress dengan k = 15		Size setelah kompres: 1.06 MB Waktu yang dibutuhkan: 25.91829776763916 s <i>Pixel difference:</i> 99.53632446933452%
Compress dengan k = 50		Size setelah kompres: 1.16 MB Waktu yang dibutuhkan 36.76667237281799 s <i>Pixel difference:</i> 98.45441489778173%
Compress dengan k = 100		Size setelah kompres: 1.18 MB Waktu yang dibutuhkan: 56.40375804901123 s <i>Pixel difference:</i> 96.90882979556346%

c. pexels-jack-hawley-57905.jpg

Gambar Asli		Ukuran asli: 635kB Nama file:pexels-jack-hawley-57905.jpg
Compress dengan k = 15		Size setelah kompres: 457 kB Waktu yang dibutuhkan: 38.18464946746826 s <i>Pixel difference:</i> 99.59484425997697%
Compress dengan k = 50		Size setelah kompres: 606 kB Waktu yang dibutuhkan 61.282291650772095 s <i>Pixel difference:</i> 98.64948086658987%

Compress dengan k = 100		Size setelah kompres: 605 kB Waktu yang dibutuhkan: 110.02710890769958 s <i>Pixel difference:</i> 97.29896173317975%
Compress dengan k = 10		Size setelah kompres: 433 kB Waktu yang dibutuhkan: 65.34303379058838 s <i>Pixel difference:</i> 7.670948677769511%

d. Cat03.jpg

Gambar Asli		Ukuran asli: 34.1kB Nama file:Cat03.jpg
-------------	---	--

Compress dengan k = 15		<p>Size setelah kompres: 29 kB Waktu yang dibutuhkan: 0.9750463962554932 s <i>Pixel difference:</i> 96.87825182101977%</p>
Compress dengan k = 30		<p>Size setelah kompres: 31.5 kB Waktu yang dibutuhkan 1.4742250442504883 s <i>Pixel difference:</i> 93.75650364203955%</p>

Compress dengan k = 1		<p>Size setelah kompres: 14.4 kB Waktu yang dibutuhkan: 0.5782768726348 s <i>Pixel difference:</i> 99.79188345473464%</p>
-----------------------	--	---

e. grayscale-image-api.png

Gambar Asli		<p>Ukuran asli: 83.8kB Nama file: 55b.png</p>
Compress dengan k = 15		<p>Size setelah kompres: 48.6 kB Waktu yang dibutuhkan: 1.3940722942352295 s <i>Pixel difference:</i> 97.52622162680075%</p>

Compress dengan k = 30		Size setelah kompres: 46.4 kB Waktu yang dibutuhkan 1.8003370761871338 s <i>Pixel difference:</i> 95.0524432536015%
Compress dengan k = 50		Size setelah kompres: 41.8 kB Waktu yang dibutuhkan: 2.1505370140075 684 s <i>Pixel difference:</i> 91.75407208933584%

BAB 5

KESIMPULAN

5.1 Kesimpulan

Dalam mengurangi ukuran file, dalam konteks ini adalah file gambar, dapat digunakan berbagai macam metode. Salah satunya adalah dengan algoritma SVD yang mengutilisasikan struktur data gambar yang bisa dibentuk dalam bentuk matriks.

Pada tugas besar ini, kami membuat sebuah aplikasi kompresi gambar yang didasarkan dengan algoritma SVD. Kami mengimplementasikan aplikasi ini dalam bentuk *website* lokal, yang dibuat menggunakan Flask dan React sebagai tech stack dari website ini. Website kami berhasil mengurangi ukuran sebagian besar gambar dengan tipe file yang didukung dalam waktu yang relatif singkat.

5.2 Saran

Saran untuk kelompok kami di antaranya :

1. Pembagian tugas dilakukan dengan lebih baik lagi
2. Kode program diberi komentar yang lebih jelas lagi. Agar kode yang digunakan lebih mudah dimengerti dan semua anggota bisa melakukan *debugging* pada algoritma yang bermasalah
3. Menggunakan hanya satu package manager untuk menghindari masalah dalam penggunaan library

5.3 Refleksi

Refleksi yang kami dapatkan dari tugas ini adalah kami bisa memperbaiki lagi kinerja kami dalam berbagai hal, contohnya pembuatan timeline kerja agar selalu ada progress dalam tugas tiap harinya. Kami juga merasa masih harus memperbaiki pembagian waktu karena penggerjaan tugas masih terlalu berdekatan dengan *deadline*. Hal lain yang kami dapatkan dari tugas ini adalah pengalaman dan pengetahuan membuat *webapp* berbasis Flask dan React.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>

<https://blog.miguelgrinberg.com/post/how-to-create-a-react--flask-project>

<https://medium.com/excited-developers/file-upload-with-react-flask-e115e6f2bf99>

https://repository.dinus.ac.id/docs/ajar/Nilai_Eigen_dan_Vektor_Eigen.pdf

Howard Anton & Chris Rores, Elementary Linear Algebra, 10th Edition

<https://docplayer.info/72939905-Penerapan-algoritma-singular-value-decomposition-svd-untuk-pengurangan-dimensi-pada-high-dimentional-biomedical-data-set.html>

https://dev.to/dev_elie/connecting-a-react-frontend-to-a-flask-backend-h1o

<https://flask.palletsprojects.com/en/2.0.x/logging/>

http://mlwiki.org/index.php/Power_Iteration

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>