TUGAS BESAR 2

IF 2123 ALJABAR LINIER DAN GEOMETRI

APLIKASI NILAI EIGEN DAN VEKTOR EIGEN DALAM KOMPRESI GAMBAR

Oleh

13520153 Vito Ghifari

13520161 M Syahrul Surya Putra

13520165 Ghazian Tsabit Alkamil

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER 1 2021/2022

DAFTAR ISI

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

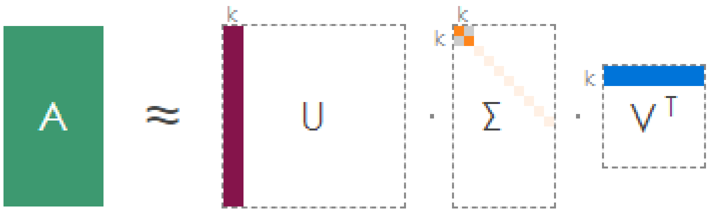
Sumber : Understanding Compression in Digital Photography (lifewire.com)

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.

$$A_{m \times n} = U_{m \times m} \; S_{m \times n} \; V^{T}_{nxn}$$

Gambar 1. Algoritma SVD

Matriks U adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks $A \times A^T$. 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 ATA. 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 rekonstruksi dari SVD akan berupa informasi dari gambar yang terkompresi dengan ukuran yang lebih kecil dibanding gambar awal.

BAB II

TEORI SINGKAT

2.1. Perkalian Matriks

Dalam matematika, perkalian matriks adalah suatu operasi biner dari dua matriks yang menghasilkan sebuah matriks. Agar dua matriks dapat dikalikan, banyaknya kolom pada matriks pertama harus sama dengan banyaknya baris pada matriks kedua. Matriks hasil perkalian keduanya, akan memiliki baris sebanyak baris matriks pertama, dan kolom sebanyak kolom matriks kedua. Perkalian matriks A dan B dinyatakan sebagai AB.

Jika A adalah matriks berukuran **m** × **n** dan B adalah matriks berukuran **n** × **p**, dengan elemen-elemen sebagai berikut:

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

Hasil perkalian kedua matriks tersebut, C = AB (dinyatakan tanpa menggunakan tanda kali atau titik), adalah sebuah matriks berukuran **m** × **p**.

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

2.2. Nilai Eigen dan Vektor Eigen

Dalam aljabar linear, vektor eigen (eigenvector) atau vektor karakteristik dari suatu matriks berukuran ***n*** × ***n*** adalah vektor tak nol yang hanya mengalami perubahan panjang ketika dikali dengan matriks tersebut. Nilai eigen (eigenvalue) yang berasosiasi dengan vektor tersebut, umumnya dilambangkan dengan λ, menyatakan besar perubahan panjang vektor yang terjadi. Secara umum dalam ruang vektor multidimensi, vektor eigen tidak mengalami rotasi ketika ditransformasikan oleh matriks.

Hal ini berlaku untuk matriks dengan elemen bilangan real, dan akan mengalami rotasi ketika elemen berupa bilangan kompleks. Nilai eigen dan vektor eigen berguna dalam proses kalkulasi matriks, yang keduanya diterapkan dalam bidang matematika murni dan matematika terapan, contohnya pada transformasi linear. Ruang eigen dari λ merupakan ruang vektor yang dibentuk dari gabungan vektor nol dan kumpulan vektor eigen yang berasosiasi dengan λ.

Istilah eigen sering kali dipadankan dengan istilah karakteristik, karena kata "eigen" yang berasal dari bahasa Jerman memiliki arti "asli", dalam konteks menjadi ciri khas atau karakteristik dari suatu sifat.

Jika ***T*** adalah suatu pemetaan linear dari suatu ruang vektor ***V*** atas lapangan ***F*** ke dirinya sendiri, dan ***v*** adalah vektor tak nol di ***V***, maka ***v*** adalah vektor eigen dari ***T*** jika ***T(v)*** adalah suatu kelipatan skalar dari ***v***. Hal ini dapat ditulis sebagai

$$T(\mathbf{v}) = \lambda\mathbf{v},$$

dengan λ adalah suatu skalar di ***F***, yang disebut dengan nilai eigen (eigenvalue), nilai karakteristik, atau akar karakteristik, yang berasosiasi dengan ***v***.

Ada hubungan erat antara matriks persegi berukuran ***n*** × ***n*** dengan pemetaan linear dari ruang vektor berdimensi ***n*** ke dirinya sendiri. Hal ini menyebabkan, dalam suatu ruang vektor berdimensi hingga, nilai eigen dan vektor eigen dapat didefinisikan menggunakan bahasa matriks, maupun menggunakan bahasa pemetaan linear. Jika ***V*** berdimensi hingga, persamaan di atas setara dengan

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

dengan ***A*** adalah representasi matriks dari ***T*** dan **x** adalah vektor koordinat **v**.

2.3. Singular Value Decomposition

Pada aljabar linear, dikenal cara-cara penguraian dekomposisi matriks. Beberapa di antaranya adalah faktorisasi LU, dekomposisi QR, dan Singular Value Decomposition. Singular Value Decomposition adalah sebuah suatu metode penguraian matriks yang membentuk tiga matriks hasil dekomposisi. Sebuah matriks **A** berukuran $m \times n$ setelah dilakukan pemfaktoran dengan SVD akan membentuk persamaan

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

dengan **U** adalah matriks ortogonal berukuran $m \times m$ yang merupakan vektor eigen dari $A \cdot A^T$. **U** dapat disebut sebagai vektor singular kiri (*left singular vectors*). Matriks berikutnya, yaitu **Σ,** merupakan matriks diagonal tak negatif berukuran $m \times n$ sebagai nilai singular yang terurut mengecil dari matriks **A**. Nilai singular (σ) dari matriks **A** diperloleh dari akar kuadrat nilai eigen (λ) matriks $A \cdot A^T$. Nilai singular ini menyimpan informasi tentang data matriks yang bersangkutan dengan nilai yang paling berkontribusi adalah nilai singular yang paling besar. Matriks yang ketiga sebagai hasil pemfaktoran dari **A** adalah $\mathbf{V}^T$, yaitu vektor eigen $A^T \cdot A$. $\mathbf{V}^T$ dapat disebut sebagai vektor singular kanan (*right singular vectors*).

BAB III

IMPLEMENTASI PROGRAM

3.1. *Tech Stack*

Tech Stack yang kelompok kami gunakan pada website *image compression* adalah kerangka kerja flask untuk bagian *back-end* dan kerangka kerja ReactJS untuk bagian *front-end*.

Flask adalah kerangka kerja aplikasi web bersifat kerangka kerja mikro yang ditulis dalam bahasa pemrograman Python dan menggunakan dependensi Werkzeug dan Jinja2. Flask disebut kerangka kerja mikro karena tidak membutuhkan alat-alat tertentu atau pustaka. Flask mendukung ekstensi yang dapat menambahkan fitur aplikasi seolah-olah mereka diimplementasikan dalam Flask itu sendiri.

ReactJS adalah pustaka JavaScript untuk *front-end* yang *open-source* dan gratis untuk membangun antarmuka pengguna atau komponen UI. ReactJS dapat digunakan sebagai basis dalam pengembangan aplikasi *single-page* atau *mobile*. Namun, ReactJS hanya memperhatikan manajemen status dan merender status tersebut ke DOM, jadi pembuat aplikasi ReactJS biasanya memerlukan penggunaan pustaka tambahan untuk perutean, serta fungsionalitas sisi klien tertentu.

3.2. Algoritma Kompresi

Perhitungan nilai eigen dan vektor eigen dengan metode standar untuk mendapatkan matriks SVD dinilai kurang efektif. Ini disebabkan oleh matriks yang terbentuk dari gambar yang akan diproses membentuk matriks dengan ukuran cukup besar, yaitu di angka ratusan hingga ribuan baris dan kolom. Sementara itu, untuk mendapat nilai eigen dan vektor eigen, determinan dan solusi dari persamaan polinomial yang terbentuk dari matriks $A \cdot A^T$ dan $A^T \cdot A$ juga harus ditentukan. Oleh karena itu, kami menggunakan metode alternatif dengan pendekatan numerik dan memanfaatkan *processing power* pada komputer.

Salah satu metode kalkulasi nilai eigen dan vektor eigen dengan cara iteratif adalah menggunakan *simultaneous power iteration.* Metode ini didasarkan oleh metode *power iteration* sebagai perhitungan satu vektor eigen dan satu nilai eigen yang paling dominan. Langkah *power iteration* secara singkat adalah sebagai berikut.

1. Matriks awal yang dicari nilai eigen dan vektor eigen adalah matriks **A** berukuran $m \times m$.
2. Inisialisasi suatu matriks x_0 berisi angka acak.
3. Operasikan $x_1 = Ax_0$.
4. Normalisasikan x_1.
5. Berikutnya hitung $x_2 = Ax_1$.
6. Iterasikan hingga matriks **x** konvergen, sehingga **x** adalah vektor eigen dari **A**.
7. Untuk mendapatkan nilai eigen, dapat digunakan persamaan berikut

$$\Lambda = \frac{x^T A x}{x^T x}$$

Metode di atas hanya dapat digunakan untuk mendapatkan vektor eigen dan nilai eigen yang paling dominan. Karena itu, digunakan *simultaneous power iteration* untuk mendapatkan sebanyak *k* nilai eigen dan *k* vektor eigen. Sedikit tambahan, pada *simultaneous power iteration*, digunakan dekomposisi QR untuk menormalisasikan vektor-vektor hasil iterasi matriks **A**. Langkah-langkah kalkulasi nilai eigen dan vektor-vektor eigen pada sebuah matriks **A** berukuran $m \times m$ adalah sebagai berikut.

1. Misalkan Q_0 adalah sebuah matriks ortogonal berukuran $k \times m$.
2. Kalkulasikan $Z_1 = AQ_0$.
3. Uraikan Z_1 dengan dekomposisi QR, yaitu $Q_1 R_1 = Z_1$.
4. Hitung nilai $Z_2 = AQ_1$.
5. Ulangi langkah 3-4 hingga **Z** konvergen.
6. Matriks **Z** adalah vektor-vektor eigen.
7. Hal yang dapat diperhatikan adalah nilai-nilai diagonal dari matriks **R** merupakan nilai eigen dari matriks **A**.
8. Akar kuadrat elemen-elemen **R** menjadi nilai-nilai singular pada SVD, yaitu **Σ**.

Dengan cara tersebut, matriks hasil dekomposisi dari suatu matriks **B** dapat diperoleh nilainya. Namun, cara iteratif seperti ini tidak menjamin hasil nilai-nilai eigen dan vektor-vektor eigen-nya akan konvergen. Dengan kata lain, **U** dan $\mathbf{V}^{\mathrm{T}}$ tidak akan selalu menjadi hasil eksak untuk dekomposisi dari matriks **B**. Jika sebuah matriks **B** berukuran $n \times m$ diuraikan dengan SVD, matriks $\mathbf{V}^{\mathrm{T}}$ diperoleh dengan melakukan operasi invers seperti pada persamaan di bawah ini.

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

$$\Sigma^{-1}U^{-1}B = V^T$$

Karena **U** merupakan matriks ortogonal,

$$U^{-1} = U^T$$

sehingga didapatkan

$$\Sigma^{-1}U^T B = V^T.$$

Pada langkah akhir, didapatkan masing-masing matriks yang merupakan hasil dekomposisi **B** dengan mengambil nilai-nilai singular sebanyak *k*.

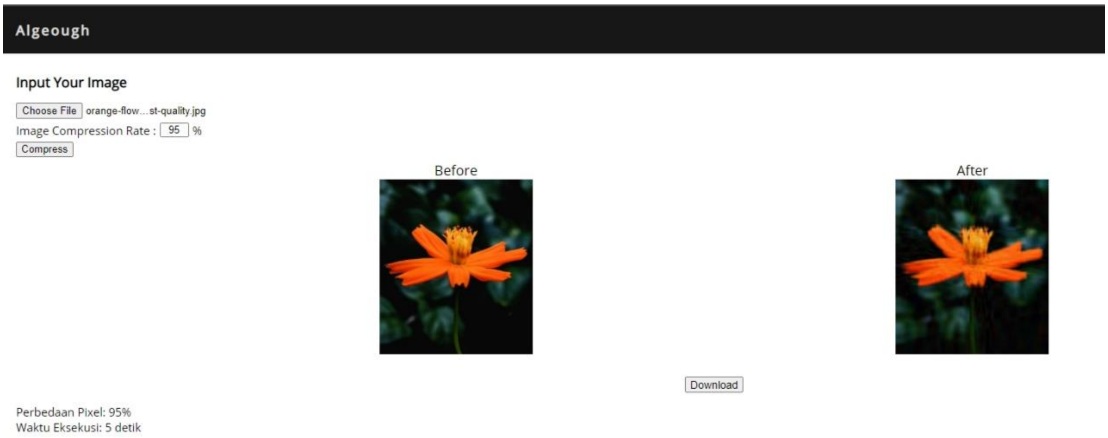
$$B = U\Sigma V^T$$

BAB IV

EKSPERIMEN

4.1. Eksperimen Pertama

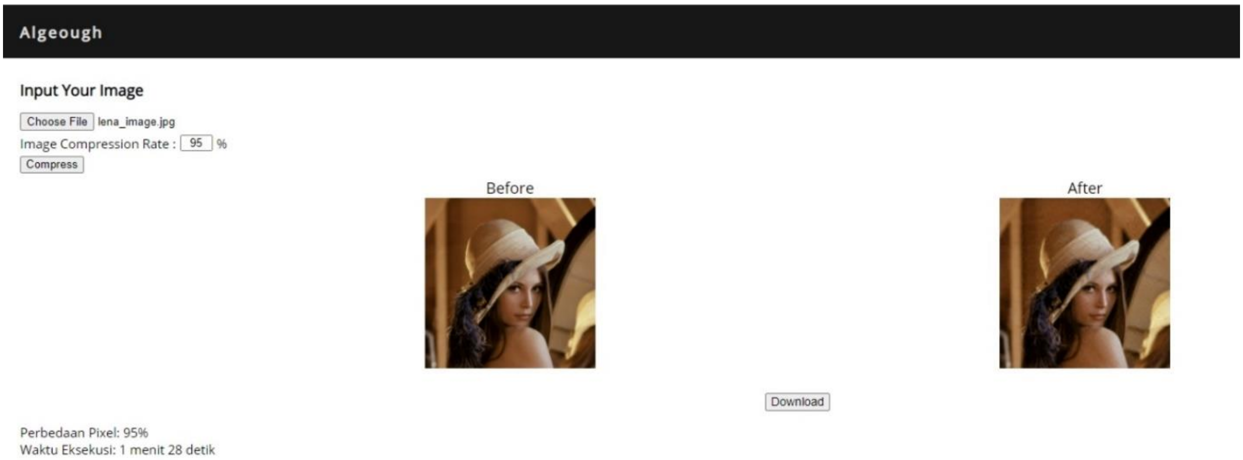
Eksperimen pertama adalah melakukan kompresi terhadap gambar *orange-flower-the-highest-quality.jpg* dengan masukan tingkat kompresi sebesar 95 %. Dengan menggunakan algoritma SVD yang telah di implementasikan sebelumnya, didapatkan hasil dari kompresi terhadap gambar tesebut dengan waktu eksekusi lima detik.



Gambar 4.1. Eksperimen Pertama

4.2. Eksperimen Kedua

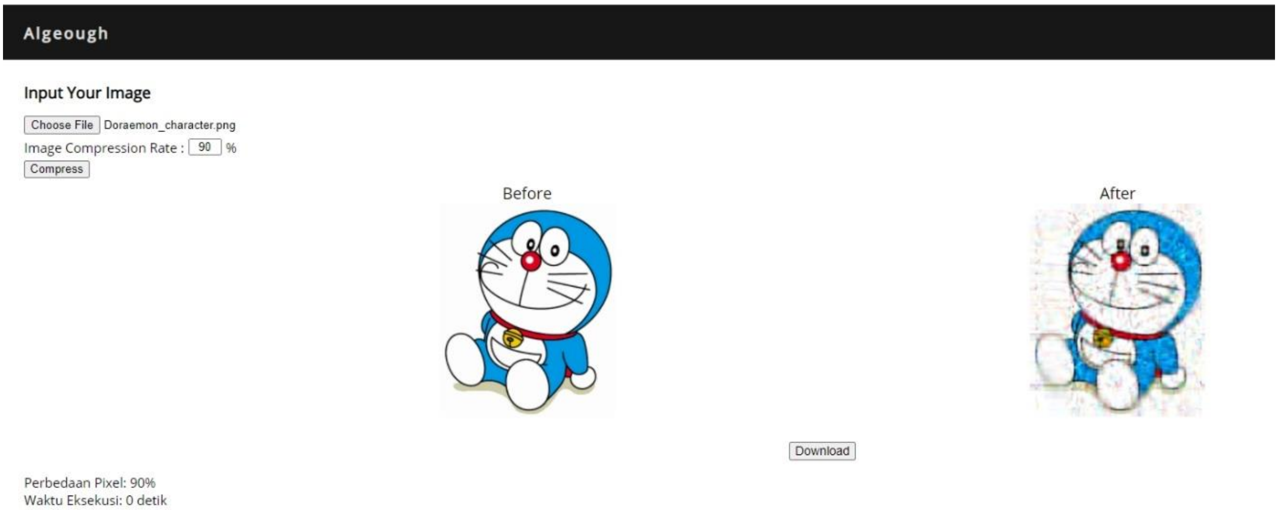
Eksperimen kedua adalah melakukan kompresi terhadap gambar lena_image.*jpg* dengan masukan tingkat kompresi sebesar 95 %. Dengan menggunakan algoritma SVD yang telah di implementasikan sebelumnya, didapatkan hasil dari kompresi terhadap gambar tesebut dengan waktu eksekusi 1 menit 28 detik.



Gambar 4.2. Eksperimen Kedua

4.3. Eksperimen Ketiga

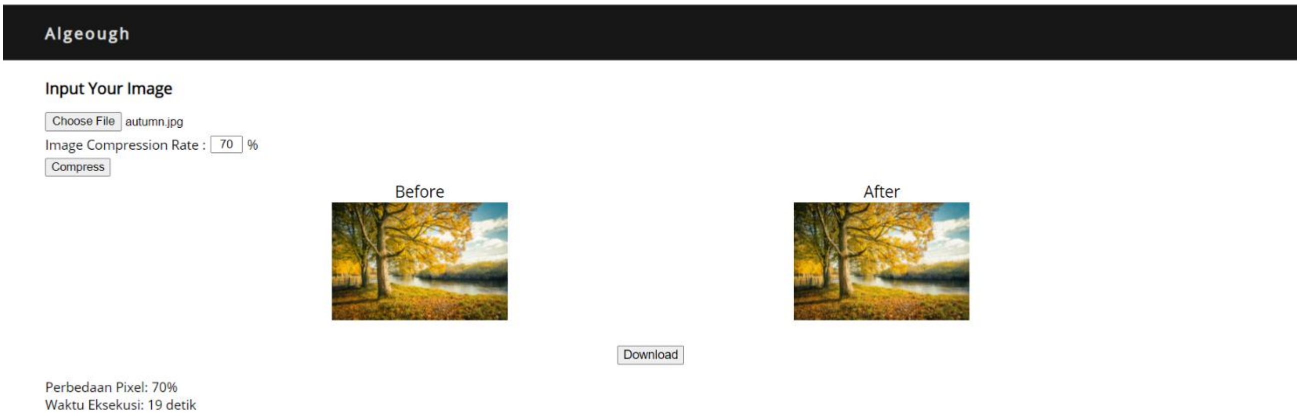
Eksperimen kedua adalah melakukan kompresi terhadap gambar Doraemon_character.*png* dengan masukan tingkat kompresi sebesar 90 %. Dengan menggunakan algoritma SVD yang telah di implementasikan sebelumnya, didapatkan hasil dari kompresi terhadap gambar tesebut dengan waktu eksekusi 0 detik.



Gambar 4.3. Eksperimen Ketiga

4.4. Eksperimen Kempat

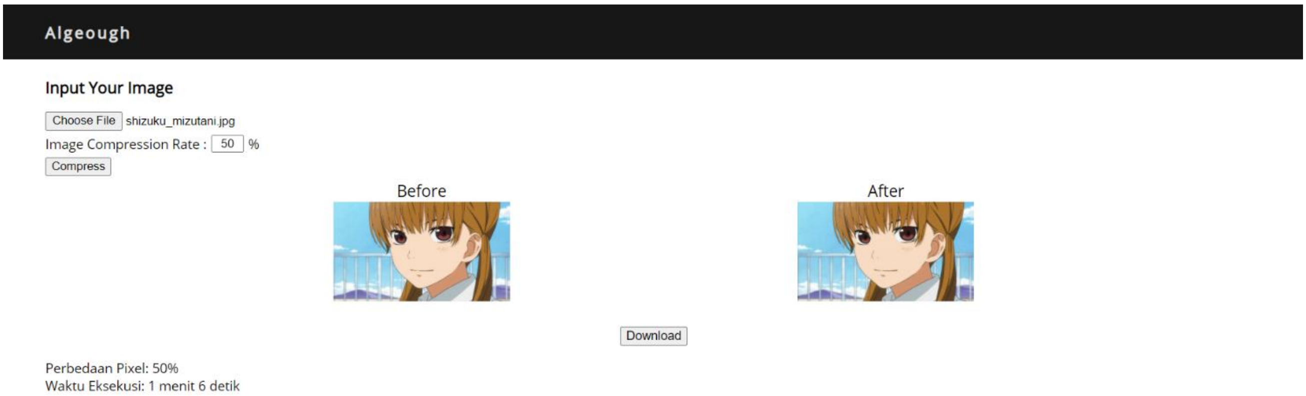
Eksperimen keempat menggunakan gambar autumn.jpg dengan masukan tingkat kompresi sebesar 70%. Dengan menggunakan algoritma SVD yang telah diimplementasikan sebelumnya, didapatkan hasil dari kompresi terhadap gambar tersebut dengan waktu eksekusi 19 detik.



Gambar 4.4. Eksperimen Keempat

4.5. Eksperimen Kelima

Eksperimen kelima menggunakan gambar shizuku_mizutani.jpg dengan masukan tingkat kompresi sebesar 50%. Dengan menggunakan algoritma SVD yang telah diimplementasikan sebelumnya, didapatkan hasil dari kompresi terhadap gambar tersebut dengan waktu eksekusi 1 menit 6 detik.



Gambar 4.5. Eksperimen Kelima

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1. Kesimpulan

Berdasarkan implementasi program dan eksperimen yang telah dilakukan didapatkan kesimpulan sebagai berikut.

1. Ditemukan program kompresi gambar dengan menggunakan algoritma Singular Value Decomposition.
2. Ditemukan solusi pada pembuatan algoritma S*ingular Value Decomposition* dengan menggunakan metode S*imultaneous Iteration* untuk memperoleh nilai eigen dan vektor eigen.
3. Dengan menggunakan S*imultaneous Iteration* pemrosesan terhadap matriks gambar lebih cepat dilakukan.
4. Perlu dilakukan penanganan kasus untuk matriks gambar yang bertipe RGB dan untuk matriks yang bertipe selain RGB.

5.2. Saran

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, terdapat beberapa saran dari penulis sebagai berikut.

1. Gunakanlah metode *Simultaneous Iteration* untuk memperoleh nilai eigen dan vektor eigen, karena metode tersebut lebih cepat dan efisien untuk menangani matriks yang berukuran besar seperti matriks gambar.
2. Perlu dianalisis lebih lanjut untuk jenis-jenis komponen warna gambar yang berbeda seperti RGB, RGBA, dan P.
3. Lebih baik untuk membuat *environment* Python dari awal agar tidak ada *dependencies* yang tidak terpakai pada *project* ini.

5.3. Refleksi

Dalam proses pengerjaan tugas besar ini tentu kami menemukan beberapa kendala, namun dari kendala tersebutlah kami belajar. Hal-hal yang menjadi refleksi pada saat kami mengerjakan tugas besar ini adalah sebagai berikut.

1. Pada awalnya kelompok kami mengalami kesulitan pada saat menerapkan algoritma SVD tanpa menggunakan library python yang sudah tersedia.
2. Pada awalnya algoritma SVD membutuhkan waktu yang lama ketika memproses matriks yang berasal dari gambar karena ukurannya terlalu besar.

3. Matriks yang berasal dari gambar memiliki beberapa jenis dan perlu dilakukan penanganan kasus untuk masing masing jenis matriks dari gambar tersebut.
4. Pada tugas besar kali ini kami banyak belajar tentang bagaimana cara membuat website aplikasi mulai dari bagian *back-end* sampai *front-end*.

REFERENSI

"Developing Web Applications with Flask"

https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/Python3_Flask.html

"How To Create a React + Flask Project"

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

Understanding Singular Value Decomposition and its Application in Data Science

https://towardsdatascience.com/understanding-singular-value-decomposition-and-its-application-in-data-science-388a54be95d

"Nilai Eigen dan Vektor Eigen Bagian 1" by Rinaldi Munir

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

Power Iteration - ML Wiki

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