

ALJABAR LINIER DAN GEOMETRI
APLIKASI NILAI EIGEN DAN VEKTOR EIGEN DALAM KOMPRESI GAMBAR

LAPORAN TUGAS BESAR 2

Diajukan sebagai salah satu tugas mata kuliah Aljabar Linier dan Geometri pada Semester 3
Tahun Akademik 2020-2021



Oleh

Arik Rayi Arkananta	13520048
Ubaidillah Ariq Prathama	13520085
Azka Syauqy Irsyad	13520107

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

BAB I

DESKRIPSI MASALAH

A. Abstraksi

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 \(lifewire.com\)](http://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 $A^T A$. 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.

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

B. Penggunaan 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. Anda dapat mengubah *layout* selama *layout* masih terdiri dari komponen yang sama.

Image Compression

Input Your Image

[Choose File..](#) No File Chosen


Image Compression Rate : %

Before

After

Image pixel difference percentage: 50 %

Image compression time: 0.5 seconds

[Download](#) 

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.

C. Saran Pengerjaan

Anda disarankan untuk membuat program *testing* pada **backend** terlebih dahulu untuk menguji keberhasilan dari proses kompresi dan rekonstruksi matriks gambar sebelum mengerjakan tampilan website.

D. 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 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 II TEORI SINGKAT

A. Perkalian Matriks

Perkalian matriks adalah perkalian yang melibatkan suatu matriks atau susunan bilangan berupa kolom dan angka, serta memiliki memiliki sifat-sifat tertentu. Misalkan matriks A (a, b, c, d) berukuran 2X2 dikalikan dengan matriks B (e, f, g, h) berukuran 2X2, sehingga rumusnya akan menjadi:

$$A \times B$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} j & k \\ l & m \end{pmatrix}$$

$$A \times B = \begin{pmatrix} aj + bl & ak + bm \\ cj + dl & ck + dm \end{pmatrix}$$

Syarat dua matriks dapat dioperasikan perkalian yaitu banyak kolom matriks pertama harus sama dengan banyak baris matriks kedua, sebagai berikut:

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

B. Nilai Eigen

Dalam Aljabar Linear, nilai eigen adalah nilai karakteristik dari suatu matriks berukuran $n \times n$. Nilai eigen sangat berhubungan dengan vektor eigen yang akan dijelaskan dibagian berikut. Untuk mencari nilai eigen pada suatu matriks, kita harus menemukan nilai λ yang memenuhi persamaan karakteristik dari matriks A, yaitu nilai-nilai λ yang padanya $\det(A - \lambda I) = 0$. dengan I adalah matriks identitas.

Contoh $A - \lambda I$ untuk matriks A yang berukuran 3x3 adalah:

$$A - \lambda I = \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix} - \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix} = \begin{pmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{pmatrix}$$

Menghitung $\det(A - \lambda I)$:

$$\begin{aligned} \det(A - \lambda I) &= (1-\lambda) \begin{vmatrix} -5-\lambda & 3 \\ -6 & 4-\lambda \end{vmatrix} - (-3) \begin{vmatrix} 3 & 3 \\ 6 & 4-\lambda \end{vmatrix} + 3 \begin{vmatrix} 3 & -5-\lambda \\ 6 & -6 \end{vmatrix} \\ &= (1-\lambda)((-5-\lambda)(4-\lambda) - (3)(-6)) + 3(3(4-\lambda) - 3 \times 6) + 3(3 \times (-6) - (-5-\lambda)6) \\ &= (1-\lambda)(-20 + 5\lambda - 4\lambda + \lambda^2 + 18) + 3(12 - 3\lambda - 18) + 3(-18 + 30 + 6\lambda) \\ &= (1-\lambda)(-2 + \lambda + \lambda^2) + 3(-6 - 3\lambda) + 3(12 + 6\lambda) \\ &= -2 + \lambda + \lambda^2 + 2\lambda - \lambda^2 - \lambda^3 - 18 - 9\lambda + 36 + 18\lambda \\ &= 16 + 12\lambda - \lambda^3. \end{aligned}$$

Didapat bahwa $\det(A - \lambda I) = 16 + 12\lambda - \lambda^3$, sehingga dengan cara pemfaktoran untuk persamaan $\det(A - \lambda I) = 0$, didapat akar-akar yang merupakan nilai-nilai eigen yaitu $\lambda = 4$ dan $\lambda = -2$.

C. Vektor Eigen

Jika nilai eigen adalah nilai karakteristik dari suatu matriks berukuran $n \times n$, vektor eigen adalah vektor kolom bukan nol yang bila dikalikan dengan suatu matriks berukuran $n \times n$ akan menghasilkan vektor lain yang memiliki nilai kelipatan dari vektor Eigen itu sendiri. Secara geometris, vektor eigen, yang bersesuaian dengan nilai eigen bukan nol nyata, menunjuk ke arah di mana ia diregangkan oleh transformasi dan nilai eigen adalah faktor yang dengannya ia diregangkan.

Untuk mencari vektor eigen, kita pertama-tama harus mempunyai nilai eigen dari sebuah matriks. Setelah nilai eigen dari matriks ditemukan, kita dapat menemukan vektor eigen oleh Gaussian Elimination. Untuk setiap nilai eigen λ , yang kita miliki $(A - \lambda)x = 0$, dimana x adalah vektor eigen yang terkait dengan nilai eigen λ . Lalu, temukan x dengan eliminasi Gaussian. Artinya, konversikan matriks yang ditambah $(A - \lambda = 0)$ untuk membentuk baris eselon dan memecahkan sistem linier yang dihasilkan oleh substitusi kembali.

Contoh perhitungan matriks pada bagian teori nilai eigen untuk nilai eigen $\lambda = 4$:

$$A - 4I = \begin{pmatrix} -3 & -3 & 3 \\ 3 & -9 & 3 \\ 6 & -6 & 0 \end{pmatrix}$$

Setelah itu diubah menjadi bentuk eselon baris:

$$\left(\begin{array}{ccc|c} -3 & -3 & 3 & 0 \\ 3 & -9 & 3 & 0 \\ 6 & -6 & 0 & 0 \end{array} \right)$$

Lalu dilakukan eliminasi Gauss-Jordan dan akan didapatkan persamaan:

$$\begin{aligned} x_1 - 1/2x_3 &= 0 \\ x_2 - 1/2x_3 &= 0 \end{aligned}$$

Sehingga didapatkan vektor eigen x yaitu:

$$\mathbf{x} = \begin{pmatrix} x_1 = \frac{x_3}{2} \\ x_2 = \frac{x_3}{2} \\ x_3 \end{pmatrix} = x_3 \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}$$

Lalu, perhitungan dilanjutkan untuk nilai eigen λ selanjutnya dan semua hasil vektor eigen x akan menjadi hasil vektor eigen.

D. Singular Value Decomposition (SVD)

SVD atau dekomposisi nilai singular adalah suatu pemfaktoran matriks dengan mengurai suatu matriks ke dalam dua matriks uniter U dan V , dan sebuah matriks diagonal Σ yang berisi faktor skala yang disebut dengan nilai singular. Dekomposisi nilai singular dari matriks A dinyatakan sebagai:

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

Vektor-vektor singular kiri Nilai-nilai singular Vektor-vektor singular kanan

Berikut adalah langkah-langkah SVD mendekomposisi $A_{2 \times 3}$ menjadi U , Σ , dan V dengan contoh matriks $A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$:

1. Untuk vektor singular kiri, hitung nilai-nilai eigen dari AA^T . $Rank(A) = k$ adalah banyaknya nilai-nilai eigen tidak nol dari AA^T .

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

2. Tentukan vektor-vektor eigen $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m$ yang berkoresponden dengan nilai-nilai eigen dari AA^T . Normalisasi $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m$ dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks U .

- a. Dihitung vektor-vektor eigen yang bersesuaian dengan semua nilai eigen tersebut adalah:

$$u1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ dan } u1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

- b. Normalisasi kedua vektor:

$$\hat{u}1 = \frac{u1}{\|u1\|} = \frac{(1,1)}{\sqrt{2}} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \text{ dan } \hat{u}2 = \frac{u2}{\|u2\|} = \frac{(1,-1)}{\sqrt{2}} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

- c. Sehingga diperoleh matriks U :

$$U = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

3. Untuk vektor singular kanan, hitung nilai-nilai eigen dari $A^T A$ lalu tentukan nilai-nilai singularnya.

$$A^T A = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

4. Tentukan vektor-vektor eigen $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ yang berkoresponden dengan nilai-nilai eigen dari $A^T A$. Normalisasi $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks V . Transpose-kan matriks V sehingga menjadi V^T .

- a. Dihitung vektor-vektor eigen yang bersesuaian dengan semua nilai eigen tersebut adalah:

$$v1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, v2 = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \text{ dan } v3 = \begin{bmatrix} 1 \\ 2 \\ -5 \end{bmatrix}$$

- b. Normalisasi kedua vektor:

$$\hat{v}1 = \frac{v1}{\|v1\|} = \frac{(1,2,1)}{\sqrt{6}} = \begin{bmatrix} 1/\sqrt{6} \\ 2/\sqrt{6} \\ 1/\sqrt{6} \end{bmatrix}, \hat{v}2 = \frac{v2}{\|v2\|} = \frac{(2,-1,0)}{\sqrt{5}} = \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \\ 0 \end{bmatrix}, \hat{v}3 = \frac{v3}{\|v3\|} = \frac{(1,2,-5)}{\sqrt{30}} = \begin{bmatrix} 1/\sqrt{30} \\ 2/\sqrt{30} \\ -5/\sqrt{30} \end{bmatrix}$$

- c. Didapat matriks V dan V^T :

$$V = \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{5} & 1/\sqrt{30} \\ 2/\sqrt{6} & -1/\sqrt{5} & 2/\sqrt{30} \\ 1/\sqrt{6} & 0 & -5/\sqrt{30} \end{bmatrix} \text{ dan } V^T = \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6} \\ 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{30} & 2/\sqrt{30} & -5/\sqrt{30} \end{bmatrix}$$

5. Bentuklah matriks Σ berukuran $m \times n$ dengan elemen-elemen diagonalnya adalah nilai-nilai singular dari matriks A dengan susunan dari besar ke kecil. Nilai singular di dalam Σ adalah akar pangkat dua dari nilai-nilai eigen yang tidak nol dari $A^T A$.

$$\Sigma = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

6. Maka, $A = U\Sigma V^T$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} * \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} * \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6} \\ 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{30} & 2/\sqrt{30} & -5/\sqrt{30} \end{bmatrix}$$

BAB III

IMPLEMENTASI PROGRAM

Program ini dibuat dengan framework Flask untuk menggabungkan front end dan backend yang berbahasa python. Adapun fungsi-fungsi yang digunakan dalam program kompresi gambar ini antara lain:

A. Back End

1. Fungsi eigen

Input : matriks a, integer k

Output : array eigen value, matriks eigen vector

Deskripsi : menghitung eigen value dan eigen vector dengan menggunakan metode QR decomposition. QR decomposition yang dilakukan secara berulang kali akan menyebabkan matriks konvergen ke suatu matriks yang berisi eigen value dan eigen vector.

2. Fungsi svd

Input : matriks a, integer ratio

Output : matriks reconimage, float diff

Deskripsi : Mencari nilai eigenvalues dan eigenvectors dari A^tA untuk mencari matriks Sigma dan matriks V untuk SVD. Matriks U didapat dari matriks Sigma dan matriks V. Setelah itu dilakukan kompresi pada matriks U, Sigma, dan Vt berdasarkan nilai ratio pada input. Nilai ratio menandakan persentase rank matriks yang dipakai.

3. Fungsi compress

Input : matriks U, matriks Sigma, matriks Vt

Output : matriks UNow, matriks SigmaNow, matriks VtNow, float diff

Deskripsi : Fungsi ini dipanggil dalam fungsi svd untuk mengubah ukuran matriks sesuai dengan ratio yang diberikan input. Fungsi ini juga mengembalikan persentase pixel dibandingkan gambar awal.

4. Fungsi compressImage

Input : string filename, string fileExt, integer ratio

Output : image yang sudah dikompres, integer diff

Deskripsi : Membaca string fileExt dari masukan yang merupakan extension dari file filename. Jika fileExt adalah '.png', maka akan dimasukkan ke fungsi compressImagePNG untuk dikompresi sehingga menghasilkan image yang sudah dikompresi dengan transparansinya yang bertahan dan integer diff. Jika fileExt adalah selain '.png', maka akan dimasukkan ke fungsi compressImagenonPNG untuk dikompresi juga sehingga menghasilkan image yang sudah dikompres dan integer diff.

5. Fungsi compressImagePNG

Input : string filename, integer ratio
Output : image yang sudah dikompres, integer diff
Deskripsi : Mengubah image yang bertipe PNG, yang berarti mempunyai channel alpha, menjadi RGBa matriks, kemudian RGBa matriks dipisah menjadi empat. Setiap matriks ini dilakukan kompresi dengan memanggil fungsi svd kecuali channel alphanya. Setelah dikompres, keempat matriks digabung Kembali dan direkonstruksi gambar.

6. Fungsi compressImagenonPNG

Input : string filename, integer ratio
Output : image yang sudah dikompres, integer diff
Deskripsi : Mengubah image yang bertipe JPEG dan JPG menjadi RGB matriks, kemudian RGB matriks dipisah menjadi tiga. Setiap matriks ini dilakukan kompresi dengan memanggil fungsi svd. Setelah dikompres, ketiga matriks digabung Kembali dan direkonstruksi gambar.

B. Front End

Untuk pembuatan frontend, secara garis besar hanya menggunakan HTML dan CSS saja. HTML yang secara umum sama dengan HTML yang digunakan seperti biasanya, namun juga terintegrasi dengan Flask yang dibuat pada backend, atau dapat disebut bahwa program yang kami buat merupakan Flask Web Application. Pada bagian HTML, *user* dapat memberikan input file gambar dan juga *rank* kompresi yang akan dilakukan selanjutnya di dalam backend, lebih tepatnya di fungsi kompresinya. Pada bagian Flask, di jalan suatu fungsi yang dapat memperlihatkan gambar di frontend yang dibuat dan mengupload gambar input dari frontend ke dalam fungsi compressing. Pada frontend juga sudah dibuat fitur untuk dapat menyimpan gambar yang telah dikompres. CSS di sini digunakan sebagai *styling* yang implementasinya dapat dilihat ketika aplikasi Flask dijalankan di localhost.

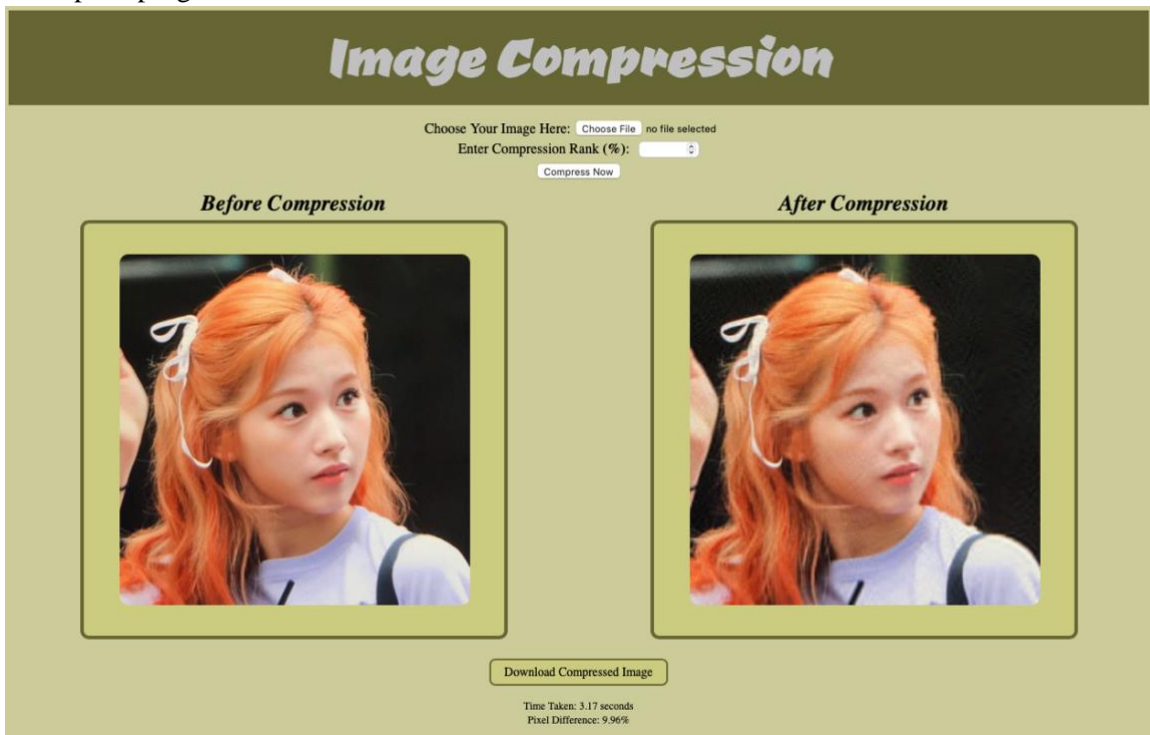
BAB IV EKSPERIMEN

A. Gambar JPG

Masukan:



Hasil pada program web:



Analisis: Untuk kompresi dari file masukan gambar sana.jpg yang memiliki ukuran 150 KB dengan Compression Rank 5%, termakan waktu sebesar 3.17 detik dengan perbandingan sebesar pixel 9.96%. Dapat dilihat gambar setelah kompresi jauh lebih buram dari gambar awal.

B. Gambar JPG dengan Ukuran 4K

Masukan:



Hasil pada program web:



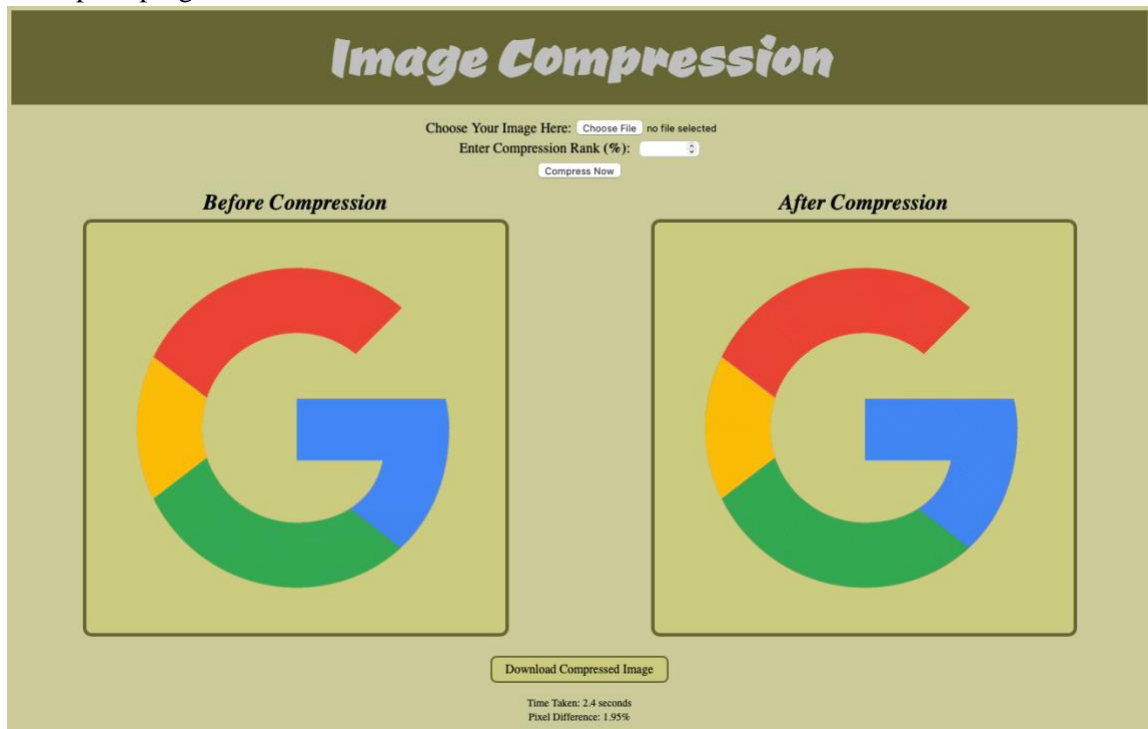
Analisis: Untuk file masukan gambar wallpaper 4K yang berukuran 4 MB dengan Compression Rank 5%, termakan waktu sebesar 515.11 detik atau sekitar 8.5 menit dengan perbandingan pixel sebesar 12,5%. Waktu yang termakan terbilang lama, namun hal ini wajar karena file sangatlah besar dan server dari web ini hanyalah laptop biasa.

C. Gambar PNG dengan Transparansi

Masukan:



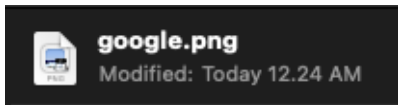
Hasil pada program web:



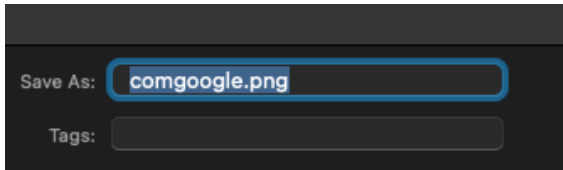
Analisis: Untuk file PNG yang memiliki channel transparansi dengan ukuran file 54 KB dengan Compression Rate sebesar 1%, termakan waktu sebesar 2.4 detik dengan perbandingan pixel sebesar 1.95%. Transparansi tetap bertahan karena channel alpha tidak dibuang, tetapi dikembalikan lagi pada file hasil kompresi.

D. Mengunduh Hasil dari Kompresi

Masukan:



Keluaran:



Analisis: Nama dari file hasil kompresi akan secara otomatis diberi nama 'com{nama file awal}.ext'. Extension dari file juga tetap bertahan tidak berubah.

BAB V

KESIMPULAN

A. Kesimpulan

Algoritma SVD dapat digunakan untuk membuat kompresi sebuah gambar. Algoritma ini cukup sederhana dibandingkan beberapa algoritma lain. Kekurangan dari algoritma ini adalah kompleksitas yang cukup besar menyebabkan runtime menjadi lambat. Hal ini dikarenakan harus menghitung eigen value dan eigen vector dari gambar itu sendiri. Akan tetapi, algoritma ini cukup baik untuk menghemat memori pixel yang digunakan tanpa terlalu merusak kualitas gambar.

Aplikasi sederhana kompresi gambar ini juga dapat lebih berguna karena dideploy ke sebuah website. Hal ini memudahkan user untuk melakukan kompresi gambar. Menggunakan HTML, CSS sederhana dan bantuan framework flask dapat membuat sebuah website sederhana.

B. Saran

Penjadwalan tugas besar mungkin dapat diperbaiki karena deadline nya cukup singkat dan banyak tugas besar lain yang sedang berjalan. Kami juga harus mempelajari dasar membuat web selain dari membuat algoritma image compression ini sendiri jadi cukup berat.

C. Refleksi

Dalam pembuatan algoritma image compression awalnya kami cukup kesulitan karena selalu menemukan bug dalam merekonstruksi urutan eigenvectors sehingga gambarnya hanya berwarna hitam, tetapi hal ini dapat diselesaikan dengan baik. Masalah utama terdapat dalam pembuatan web karena kami masi belum punya dasarnya sehingga perlu banyak research. Kami juga sempat kesulitan dalam menghubungkan frontend dan backend.

DAFTAR REFERENSI

<https://saintif.com/perkalian-matriks/>

<https://risalandi.com/nilai-eigen-dan-vektor-eigen/>

https://id.wikipedia.org/wiki/Penguraian_nilai_singular

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

<https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3>