14/11/2021LAPORAN TUGAS BESAR 2 IF 2123

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:

Rayhan Kinan Muhannad (K2)	**13520065**
Andhika Arta Aryanto (K2)	**13520081**
Sarah Azka Arief (K2)	**13520083**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

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.

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.

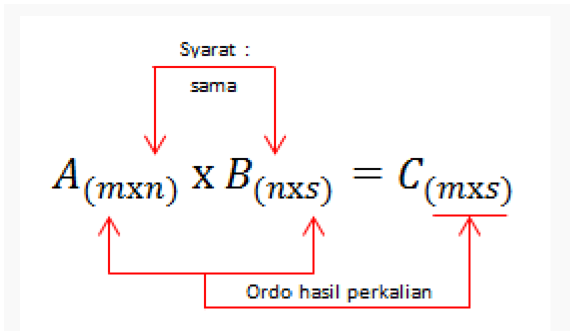
Pada Tugas Besar 2 IF2123 Aljabar Linier dan Geometri, para penulis diminta untuk membuat *website* kompresi gambar sederhana dengan menggunakan algoritma SVD. *Website* ini dapat menerima file gambar dengan format file dibebaskan selama merupakan format untuk gambar dan juga tingkat kompresi yang formatnya dibebaskan. Selain itu, *website* dapat menampilkan gambar *input, output, runtime* algoritma, dan persentase hasil kompresi gambar (perubahan jumlah pixel gambar). File *output* berupa gambar yang sudah dipampatkan harus tetap mempertahankan warna serta transparansi dari gambar asli dan dapat diunduh melalui *website*.

BAB II

TEORI SINGKAT

2.1. Perkalian Matriks

Perkalian antara dua matriks misalnya matriks $\boldsymbol{A}$ dan $\boldsymbol{B}$, hanya bisa dilakukan jika jumlah kolom $\boldsymbol{A}$ sama dengan jumlah baris $\boldsymbol{B}$. Perkalian tersebut akan menghasilkan suatu matriks dengan jumlah baris sama dengan baris matriks $\boldsymbol{A}$ dan jumlah kolom sama dengan kolom matriks $\boldsymbol{B}$.



Gambar 2.1 Syarat Perkalian Matriks

Misalkan matriks $\boldsymbol{A}$ memiliki ordo $(\mathbf{3} \times \mathbf{4})$ dan matriks $\boldsymbol{B}$ memiliki ordo $(\mathbf{4} \times \mathbf{2})$, maka matriks $\boldsymbol{C}$ memiliki ordo $(\mathbf{3} \times \mathbf{2})$. Elemen $\boldsymbol{C}$ pada baris ke-2 dan kolom ke-2 atau $\boldsymbol{C}_{\mathbf{22}}$ diperoleh dari jumlah hasil perkalian elemen-elemen baris ke-2 matriks $\boldsymbol{A}$ dan kolom ke-2 matriks $\boldsymbol{B}$.

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{2} & \mathbf{1} & \mathbf{4} & \mathbf{3} \\ \mathbf{2} & \mathbf{5} & \mathbf{1} & \mathbf{2} \\ \mathbf{1} & \mathbf{3} & \mathbf{2} & \mathbf{2} \end{pmatrix} \text{ dan } \boldsymbol{B} = \begin{pmatrix} \mathbf{1} & \mathbf{3} \\ \mathbf{3} & \mathbf{2} \\ \mathbf{2} & \mathbf{5} \\ \mathbf{1} & \mathbf{4} \end{pmatrix}$$

$$\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{B} = \begin{pmatrix} \mathbf{2} & \mathbf{1} & \mathbf{4} & \mathbf{3} \\ \mathbf{2} & \mathbf{5} & \mathbf{1} & \mathbf{2} \\ \mathbf{1} & \mathbf{3} & \mathbf{2} & \mathbf{2} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{1} & \mathbf{3} \\ \mathbf{3} & \mathbf{2} \\ \mathbf{2} & \mathbf{5} \\ \mathbf{1} & \mathbf{4} \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} \mathbf{2 \cdot 1 + 1 \cdot 3 + 4 \cdot 2 + 3 \cdot 1} & \mathbf{2 \cdot 3 + 1 \cdot 2 + 4 \cdot 5 + 3 \cdot 4} \\ \mathbf{2 \cdot 1 + 5 \cdot 3 + 1 \cdot 2 + 2 \cdot 1} & \mathbf{2 \cdot 3 + 5 \cdot 2 + 1 \cdot 5 + 2 \cdot 4} \\ \mathbf{1 \cdot 1 + 3 \cdot 3 + 2 \cdot 2 + 2 \cdot 1} & \mathbf{1 \cdot 3 + 3 \cdot 2 + 2 \cdot 5 + 2 \cdot 4} \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} \mathbf{16} & \mathbf{40} \\ \mathbf{21} & \mathbf{29} \\ \mathbf{16} & \mathbf{27} \end{pmatrix}$$

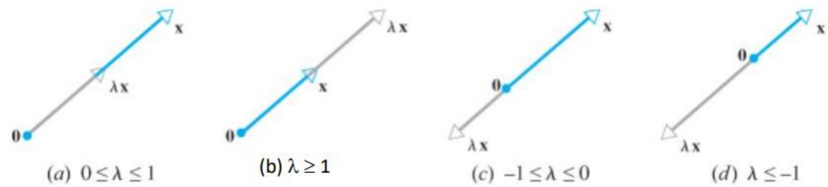
Persamaan 2.1 Contoh Perkalian Matriks

Perlu dicatat juga bahwa hasil perkalian $\boldsymbol{A} \times \boldsymbol{B}$ akan berbeda dengan hasil perkalian $\boldsymbol{B} \times \boldsymbol{A}$ (tidak komutatif).

2.2. Nilai Eigen dan Vektor Eigen

Jika $\boldsymbol{A}$ adalah matriks $\boldsymbol{n} \times \boldsymbol{n}$, maka vektor tidak nol $\boldsymbol{x}$ di $\mathbb{R}^{\boldsymbol{n}}$ disebut vektor eigen dari $\boldsymbol{A}$ jika $\boldsymbol{A} \cdot \boldsymbol{x}$ sama dengan perkalian suatu skalar dengan $\boldsymbol{\lambda} \cdot \boldsymbol{x}$, yaitu $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$. Skalar $\boldsymbol{\lambda}$ disebut nilai eigen dari $\boldsymbol{A}$, dan $\boldsymbol{x}$ dinamakan vektor eigen yang berkoresponden dengan $\boldsymbol{\lambda}$.

Kata "*eigen*" berasal dari Bahasa Jerman yang artinya "asli" atau "karakteristik". Jadi, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran $\boldsymbol{n} \times \boldsymbol{n}$. Vektor eigen $\boldsymbol{x}$ menyatakan vektor kolom yang apabila dikalikan dengan sebuah matriks $\boldsymbol{n} \times \boldsymbol{n}$ menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri, jadi operasi $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$ menyebabkan vektor $\boldsymbol{x}$ menyusut atau memanjang dengan faktor $\boldsymbol{\lambda}$ dengan arah yang sama jika $\boldsymbol{\lambda}$ positif dan arah berkebalikan jika $\boldsymbol{\lambda}$ negatif.



Gambar 2.2 Ilustrasi Vektor Eigen

Untuk mencari nilai eigen dan vektor eigen, anggap ada matriks $\boldsymbol{A}$ berukuran $\boldsymbol{n} \times \boldsymbol{n}$. Vektor eigen dan nilai eigen dihitung dengan:

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$$
$$\boldsymbol{I} \cdot \boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{I} \cdot \boldsymbol{x}$$
$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{I} \cdot \boldsymbol{x}$$
$$(\boldsymbol{\lambda} \cdot \boldsymbol{I} - \boldsymbol{A}) \cdot \boldsymbol{x} = \boldsymbol{0}$$

Persamaan 2.2 Penurunan Rumus Nilai Eigen

Agar $(\boldsymbol{\lambda} \cdot \boldsymbol{I} - \boldsymbol{A})$ memiliki solusi tidak nol, maka haruslah $\mathbf{det}(\boldsymbol{\lambda} \cdot \boldsymbol{I} - \boldsymbol{A}) = \boldsymbol{0}$. Persamaan dari $\mathbf{det}(\boldsymbol{\lambda} \cdot \boldsymbol{I} - \boldsymbol{A})$ disebut persamaan karakteristik dari matriks $\boldsymbol{A}$, dan akar akar dari persamaan ini, yaitu $\boldsymbol{\lambda}$, merupakan nilai eigennya.

Setelah mendapat nilai eigen, kita masukkan kembali nilai eigen tersebut ke persamaan $(\boldsymbol{\lambda} \cdot \boldsymbol{I} - \boldsymbol{A}) \cdot \boldsymbol{x} = \boldsymbol{0}$ dan persamaan parametrik dan basis dari $\boldsymbol{x_1}$ dan $\boldsymbol{x_2}$ tersebut merupakan vektor eigen yang berkorespondensi dengan nilai eigen yang disubsitusi tadi, lalu lakukan lagi hal ini dengan nilai eigen lainnya untuk mendapatkan semua vektor eigen. Berikut diberikan contoh untuk menggambarkan lebih baik:

$$(\lambda I - A)\mathbf{x} = 0 \rightarrow \begin{bmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{Untuk } \lambda = 3 \rightarrow \begin{bmatrix} 0 & 0 \\ -8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow -8x_1 + 4x_2 = 0 \rightarrow 8x_1 = 4x_2 \rightarrow x_1 = \tfrac{1}{2} x_2$$

$$\rightarrow \text{ Solusi: } x_1 = \tfrac{1}{2} t,\ x_2 = t,\ t \in \mathbf{R}$$

$$\text{Vektor eigen: } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} t \\ t \end{bmatrix} = t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \rightarrow \text{membentuk } \textbf{ruang eigen} \ (\textit{eigenspace})$$

Gambar 2.3 Contoh Vektor Eigen

Selain cara ini, terdapat cara lain, diantaranya adalah cara – cara yang menghasilkan *upper Hessenberg matrix* dan *tridiagonal matrix*, dimana matriks tersebut merupakan matriks yang digunakan untuk mencari nilai eigen, matriks-matriks tersebut akan mempermudah perhitungan karena nilai eigen untuk triangular matriks adalah elemen diagonalnya. Selain itu, ada juga cara iterative untuk menghasilkan nilai eigen dengan melakukan manipulasi pada matriks dengan urutan-urutan tertentu yang akan lama-lama membuat matriks yang memuat nilai eigen. Bahkan, beberapa algoritma juga langsung menghasilkan vektor-vektor yang merupakan vektor eigen. Beberapa contoh algoritma tersebut adalah sebagai berikut:

a. Lanczoz Algorithm
b. Power Iteration
c. Inverse Iteration
d. Rayleigh Quotient Iteration
e. QR Algorithm

Pada tugas besar kali ini, kami menggunakan algoritma *power iteration* untuk mencari nilai eigen dan juga vektor eigen.

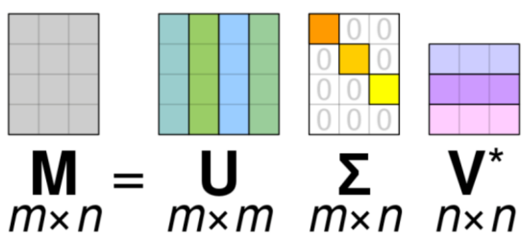
2.3. Singular Value Decomposition

SVD atau *Singular Value Decomposition* adalah suatu cara pemfaktoran yang bisa dilakukan pada matriks nonpersegi. SVD memfaktorkan matriks $\boldsymbol{A}$ berukuran $\boldsymbol{m} \times \boldsymbol{n}$ menjadi matriks $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, dan $\boldsymbol{V}$ sedemikian sehingga $\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^{\boldsymbol{T}}$ adalah sebagai berikut:

$\boldsymbol{U}$ = matriks ortogonal $\boldsymbol{m} \times \boldsymbol{m}$,

$\boldsymbol{V}$ = matriks orthogonal $\boldsymbol{n} \times \boldsymbol{n}$,

$\boldsymbol{\Sigma}$ = matriks berukuran $\boldsymbol{m} \times \boldsymbol{n}$ yang elemen – elemen diagonal utamanya adalah nilai – nilai singular dari matriks $\boldsymbol{A}$ dan elemen – elemen lainnya 0.



Gambar 2.4 Dekomposisi Matriks SVD

Sebelumnya sering disinggung mengenai matriks orthogonal. Matriks orthogonal sendiri merupakan matriks yang kolom-kolomnya adalah vektor yang saling orthogonal satu sama lain alias hasil kali titiknya sama dengan 0. Jika $\boldsymbol{Q}$ adalah matriks orthogonal dengan ukuran $\boldsymbol{m} \times \boldsymbol{n}$, dan kolom-kolom matriks $\boldsymbol{Q}$ adalah $\boldsymbol{V_1}, \boldsymbol{V_2}, \dots, \boldsymbol{V_m}$ maka $\boldsymbol{V_i} \cdot \boldsymbol{V_j} = \boldsymbol{0}$ untuk $\boldsymbol{i} \neq \boldsymbol{j}$.

Selain itu, ada juga istilah nilai singular atau *singular value* yang adalah akar-akar dari nilai eigen yang tidak nol. Untuk mencari nilai singular, anggap terdapat matriks $\boldsymbol{A}$ yang merupakan matriks $\boldsymbol{m} \times \boldsymbol{n}$ dengan nilai eigen berupa $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \boldsymbol{\lambda_3} \dots, \boldsymbol{\lambda_n}$. Maka, nilai singular dari matriks $\boldsymbol{A}$ adalah $\boldsymbol{\sigma_1} = \sqrt{\boldsymbol{\lambda_1}}, \boldsymbol{\sigma_2} = \sqrt{\boldsymbol{\lambda_2}}$, dan seterusnya sampai $\boldsymbol{\sigma_n} = \sqrt{\boldsymbol{\lambda_n}}$. Umumnya, nilai singular diurutkan dari yang terbesar hingga terkecil.

Terdapat beberapa algoritma yang dapat digunakan dalam mencari SVD. Berikut algoritma yang digunakan pada program kami yang diambil dari buku Howard–Anton:

THEOREM 9.5.4 Singular Value Decomposition (Expanded Form)

If A is an $m \times n$ matrix of rank k, then A can be factored as

$$A = U\Sigma V^T = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_k | \mathbf{u}_{k+1} \quad \cdots \quad \mathbf{u}_m] \left[\begin{array}{ccccc|c} \sigma_1 & 0 & \cdots & 0 & & \\ 0 & \sigma_2 & \cdots & 0 & & 0_{k\times(n-k)} \\ \vdots & \vdots & \ddots & \vdots & & \\ 0 & 0 & \cdots & \sigma_k & & 0_{(m-k)\times(n-k)} \\ & 0_{(m-k)\times k} & & & & \end{array}\right] \left[\begin{array}{c} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \\ \hline \mathbf{v}_{k+1}^T \\ \vdots \\ \mathbf{v}_n^T \end{array}\right]$$

in which U, Σ, and V have sizes $m \times m$, $m \times n$, and $n \times n$, respectively, and in which

(a) $V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_n]$ orthogonally diagonalizes A^TA.

(b) The nonzero diagonal entries of Σ are $\sigma_1 = \sqrt{\lambda_1}, \sigma_2 = \sqrt{\lambda_2}, \dots, \sigma_k = \sqrt{\lambda_k}$, where $\lambda_1, \lambda_2, \dots, \lambda_k$ are the nonzero eigenvalues of A^TA corresponding to the column vectors of V.

(c) The column vectors of V are ordered so that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > 0$.

(d) $\mathbf{u}_i = \dfrac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|} = \dfrac{1}{\sigma_i} A\mathbf{v}_i \quad (i = 1, 2, \dots, k)$

(e) $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$ *is an orthonormal basis for* col(A)}.

(f) $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k, \mathbf{u}_{k+1}, \dots, \mathbf{u}_m\}$ *is an extension of* $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$ to an ortho-normal basis for R^m.

The vectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$ are called the ***left singular vectors*** of A, and the vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ are called the ***right singular vectors*** of A.

Gambar 2.5 Algoritma Pencarian Matriks SVD

BAB III
IMPLEMENTASI PROGRAM

3.1 Penjelasan Tech Stack

Framework yang digunakan di dalam bagian website pada program ini adalah *Flask*. *Flask* adalah *web application framework* yang ditulis menggunakan bahasa Python dan dapat diimplementasikan menggunakan bahasa Python pula. *Flask* pertama kali dikembangkan oleh Armin Ronacher pada tahun 2010 yang merupakan gabungan dari beberapa *library* Python, yaitu *Werkzeug* (*response and request handling*), WSGI (*interface* antara user dengan server), *Jinja* (*Webpage Templating Engine*), *MarkupSafe* (*string handling*), dan *ItsDangerous* (*data serialization*).

Selain Flask, digunakan pula library *OpenCV* yang digunakan untuk membaca file gambar, mengubahnya menjadi matriks BGR/BGRA, menampilkannya sebagai element HTML, kemudian menyimpannya kembali dalam bentuk binary. Selain *OpenCV*, penulis juga menggunakan library PIL. Penggunaan dua modul pengolahan citra didasarkan pada fungsi PIL yang dapat mengolah dan menggabung gambar dengan fungsi *merge* serta kompatibilitas *OpenCV* dengan flask. Terakhir, penulis menggunakan library *NumPy* untuk melakukan operasi dan pemrosesan matriks.

3.2 Penjelasan Garis Besar Algoritma Kompresi

Algoritma kompresi pada program kami terletak pada folder 'src'. Struktur pohon dari program kami adalah sebagai berikut:

```
src
│   app.py
│   Eigen.py
│   SVD.py
│
├───static
│   ├───images
│   │       imageNotAvailable.png
│   │       logo.png
│   │       logoSmall.png
│   │
│   └───styles
│           frontpage.css
│
└───templates
        frontpage.html
```

Berdasarkan implementasi dan fungsionalitasnya, penulis membagi program menjadi 3 bagian, yaitu bagian *website* (*frontend* dan *backend*), bagian pencarian nilai eigen dan vektor eigen, serta bagian pemrosesan dan pemampatan matriks gambar menggunakan SVD (*Singular Value Decomposition*).

a. Bagian *Website* (*Frontend* dan *Backend*)

Bagian *Website* pada program ini dibagi kembali menjadi dua subbagian, yaitu *frontend* (HTML sebagai inisialisasi elemen pada *webpage*, CSS sebagai pemercantik elemen HTML, dan Jinja sebagai *Webpage Templating Engine* yang berfungsi sebagai *control flow* dan *display* data yang dikirim oleh *backend*) dan *backend* (*Werkzeug* sebagai penampung *response* serta *request handling* yang dikirim dari *user* ke *server* dan *Flask* sebagai framework yang menggabungkan semua *module* tersebut).

Frontend adalah bagian di dalam *web development* yang menangani *user side* dari suatu *website* seperti pembuatan GUI (*Graphical User Interface*) yang *user friendly*. Agar *user* dapat berinteraksi dengan *website*, diperlukan tampilan *website* yang tertata, rapih, dan yang terpenting mudah dimengerti oleh *user*. Pada program ini, penulis menggunakan bahasa HTML dan CSS serta menggunakan *library* Python yang bernama Jinja untuk membangun *frontend* dari website ini. Bahasa HTML adalah bahasa markup yang digunakan untuk menginisialisasi elemen pada *website*. Banyak sekali elemen yang dipakai di program ini, seperti *input* dan *form* untuk menerima masukkan dari *user* berupa *file* dan *text*, *button* untuk menerima masukan berupa *click* dari *user*, serta *img* untuk menampilakan *file image* masukan user dan hasil kompresi dari bagian program lainnya. Bahasa CSS adalah bahasa *style sheet* yang digunakan untuk mempercantik elemen HTML yang sudah diinisialisasi sebelumnya. Ukuran elemen, posisi elemen, serta *font text* dapat dipercantik dengan CSS. Terakhir, *module* Python Jinja digunakan untuk menampilkan *response* yang dikirim oleh *backend* dengan menciptakan semacam *control flow* (*if-else*, *for loop*, dan *while loop*) atau mengubah nilai intrinsik elemen HTML agar *website* yang ditampilkan sesuai dengan kemauan kita.

Backend adalah bagian di dalam *web development* yang menangani *server side* dari suatu *website* seperti *response and request handling*, *URL building*, *file handling*, dan *data processing*. Pada program ini, penulis menggunakan *framework* Flask untuk menghubungkan semua *module* Python yang dibutuhkan untuk membangun *server side* dari suatu *website*.

Dengan menggunakan Flask, penulis dapat membuat *website* yang responsif. *User* dapat mengirimkan data lewat *website*, kemudian *Flask* akan membaca dan menyimpannya dengan bantuan *Werkzeug*. Tidak hanya itu, dengan adanya *Flask*, *server* dapat mengirim data hasil pemrosesan kepada *user* dalam bentuk *binary* yang kompatibel dengan elemen HTML yang penulis gunakan sebagai bagian dari *frontend*. *Flask* juga dapat mengintegrasikan algoritma lainnya seperti *image processing* menggunakan SVD di dalam pembuatan fungsi *routing*.

b. Bagian Pencarian Nilai Eigen dan Vektor Eigen

Untuk mencari nilai Eigen dan vektor Eigen, digunakan metode aljabar linier dengan nama *Power Iteration*, teorinya sebagai berikut:

- Misal A adalah suatu matriks simetris, dekomposisi eigen dari A adalah $A = Q \wedge Q^T$, apabila q_i merupakan kolom dari Q, adalah nilai eigen dominan apabila $\lambda_1 > \lambda_i$ untuk semua $i = 2, 3, 4, \ldots$ dan eigen vektor yang berkoresponden juga berarti merupakan eigen vektor dominan, dominan disini maksudnya dengan nilai terbesar.

Untuk melakukan hal ini, digunakan *Power Method* yaitu:

Pertama-tama, diambil unit vector random x_0, sehingga

- $x_1 = A \cdot x_0$
- $x_2 = A \cdot A \cdot x_0$

dan seterusnya sampai dia mendekati nilai sebenarnya.

Metode diatas hanya menemukan vektor eigen paling besar, namun karena kita mengetahui bahwa vektor eigen lain ortogonal kepada eigen vektor dominan, bisa digunakan lagi *power method* untuk memaksa vektor kedua yang dihasilkan merupakan suatu vektor yang ortogonal dari vektor petama, hal ini dilakukan terus menerus sampai ditemukan semua vektor eigen dari suatu matriks. Untuk algoritma yang digunakan pada kode ini:

- Ambil Q_0 sehingga $Q_0^T . Q^T = I$, lalu dilakukan iterasi berupa
- $Z_k = A \cdot Q_{k-1} \cdot Z_{k-1}$
- $Q_k \cdot R_k = Z_k$ (dalam hal ini digunakan library numpy untuk melakukan *QR decomposition*)

Pada program kami, digunakan k yang kecil agar program lebih efisien dan *runtime* cepat, kami menggunakan SVD yang dapat membenarkan nilai eigen yang kurang mendekati, sehingga matriks akhir yang dihasilkan setelah dikomposisi tetap sesuai dengan matriks awalnya.

c. Bagian Pemrosesan dan Pemampatan Matriks Gambar dengan SVD

Bagian ini terdapat pada file SVD.py dan secara umum terdiri atas dua fungsi yakni ‘compress’ dan ‘svd’. Fungsi ‘compress’ menerima dua argumen berupa gambar yang berbentuk 3D array serta persentase kompresi gambar dan mengembalikan gambar yang telah dipampatkan, sementara fungsi ‘svd’ menerima dua argumen yakni matriks yang akan diproses serta skala k dan mengembalikan matriks U, Σ, dan V^T yang sudah dipotong sesuai k.

Pada fungsi ‘svd’, anggap matriks yang diproses sebagai A. Tahap pertama yang dilakukan adalah mencari nilai eigen dan vektor eigen dari $A^T \cdot A$ dengan memanfaatkan fungsi *simultaneous_power_iteration* dari Eigen.py. Akar dari nilai-nilai tidak nol dari nilai absolut eigen kemudian digunakan untuk mencari *singular value* sementara vektor eigen digunakan sebagai matriks V. Kemudian, matriks sigma dicari dengan cara mendiagonalisasi *singular values*, matriks V^Tdicari dengan cara mentranspos matriks V, dan matriks U dicari melalui persamaan $U_i = \frac{1}{\sigma_i} AV_i$. Terakhir, U, Σ, dan V^T dipotong sesuai skala k, sehingga fungsi ‘svd’ dengan argumen berupa matriks A $(m \times n)$ dengan skala k akan mengembalikan tiga matriks yakni matriks U $(m \times k)$, matriks Σ $(k \times k)$, dan matriks V^T $(k \times n)$.

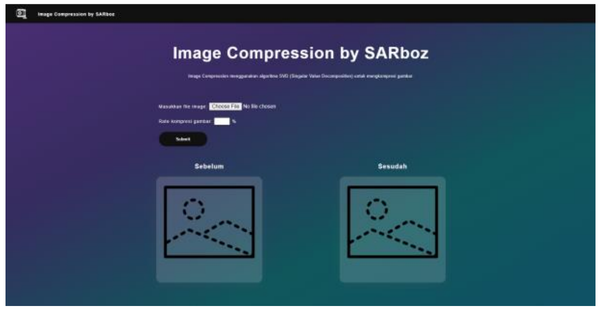
Pada fungsi ‘compress’, nilai pada 3D array dikonversi menjadi float32 untuk meningkatkan presisi dari hasil kompresi. Setelah itu, skala k ditentukan melalui input persentase yang diberikan oleh pengguna, dengan k adalah persentase kompresi dikali dengan $^{mn}/_{m+n+1}$ dengan m adalah jumlah baris matriks dan n adalah jumlah kolom matriks. Selanjutnya, 3D array tersebut dipecah menjadi matriks dari masing-masing saluran (RGB/RGBA). Matriks-matriks tersebut kemudian didekomposisi dengan menggunakan fungsi SVD buatan kami. Perlu dicatat bahwa *try-except* digunakan pada tahap ini demi menangani kasus gambar dengan 4 saluran serta menangani kasus dimana terdapat saluran yang hanya berisi nilai kosong sehingga tidak memiliki *singular value*. Setelah melakukan SVD, hasil dekomposisi dikalikan sehingga didapat aproksimasi dari matriks setiap saluran. Selanjutnya, matriks setiap saluran dikonversi menjadi L *image* dengan menggunakan library PIL dan hasilnya digabung dengan fungsi *merge* dari PIL sehingga dihasilkan gambar

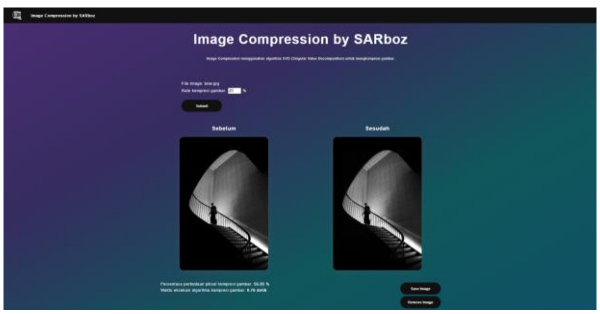
RGB/RGBA. Karena *framework Flask* lebih kompatibel dengan OpenCV, gambar dikonversi menjadi bentuk BGR/BGRA dengan menggunakan *library* OpenCV dan dikembalikan.

BAB IV

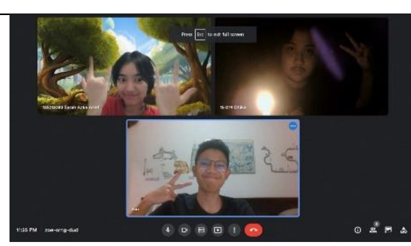
EKSPERIMEN

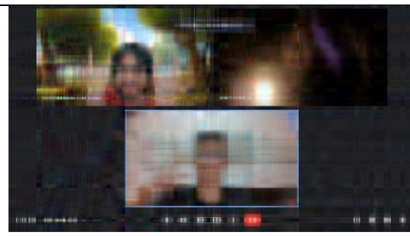
5.1 Tampilan Website





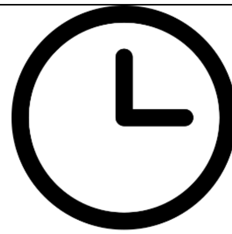
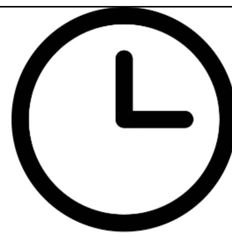
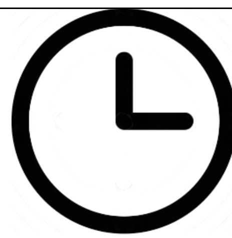
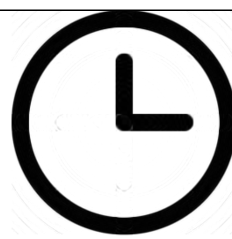
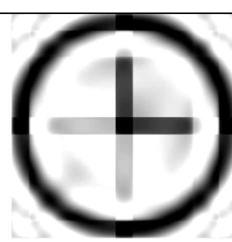
5.2. Hasil Kompresi Gambar

No.	Persentase Kompresi	Gambar	Deskripsi
1.	100% (Asli)		Ukuran File: 133 KB Ukuran Gambar: 867 x 1553 File type: JPG
	80%		Ukuran File: 210 KB Skala singular value: 445 Runtime: 3.13 detik Pixel compression percentage: 49.41%

	40%		Ukuran File: 225 KB Skala singular value: 222 Runtime: 1,77 detik Pixel compression percentage: 58.61%
	20%		Ukuran File: 239 KB Skala singular value: 111 Runtime: 1.08 detik Pixel compression percentage: 72.2%
	2%		Ukuran File: 218 KB Skala singular value: 11 Runtime: 0.37 detik Pixel compression percentage: 92.19%%
2.	100% (Asli)		Ukuran File: 4.018 KB Ukuran Gambar: 4320 x 7680 File type: JPG
	80%		Ukuran File: 4.171 KB Skala singular value: 2212 Runtime: 181.04 detik Pixel compression percentage: 41.38%
	40%		Ukuran File: 4.592 KB Skala singular value: 1106 Runtime: 79.1 detik Pixel compression percentage: 50.82%

	20%		Ukuran File: 4.724 KB Skala singular value: 553 Runtime: 39.24 detik Pixel compression percentage: 63.64%
	2%		Ukuran File: 3.671 KB Skala singular value: 55 Runtime: 17.67 detik Pixel compression percentage: 83.62%
3.	100% (Asli)		Ukuran File: 76 KB Ukuran Gambar: 385 x 576 File type: JPG
	80%		Ukuran File: 91 KB Skala singular value: 184 Runtime: 0.93 detik Pixel compression percentage: 56.81%
	40%		Ukuran File: 91 KB Skala singular value: 92 Runtime: 0.48 detik Pixel compression percentage: 88.06%
	20%		Ukuran File: 88 KB Skala singular value: 46 Runtime: 0.17 detik Pixel compression percentage: 94.5%

	2%		Ukuran File: 60 KB Skala singular value: 5 Runtime: 0.05 detik Pixel compression percentage: 98.41%
4.	100% (Asli)		Ukuran File: 25 KB Ukuran Gambar: 750 x 500 File type: JPG
	80%		Ukuran File: 64 KB Skala singular value: 240 Runtime: 0.97 detik Pixel compression percentage: 40.8%
	40%		Ukuran File: 81 KB Skala singular value: 120 Runtime: 0.66 detik Pixel compression percentage: 53.77%
	20%		Ukuran File: 81 KB Skala singular value: 60 Runtime: 0.67 detik Pixel compression percentage: 56.94%

	2%		Ukuran File: 52 KB Skala singular value: 6 Runtime: 0.05 detik Pixel compression percentage: 64.29%
5.	100% (Asli)		Ukuran File: 11 KB Ukuran Gambar: 512 x 512 File type: PNG
	80%		Ukuran File: 56 KB Skala singular value: 205 Runtime: 0.57 detik Pixel compression percentage: 3.69%
	40%		Ukuran File: 122 KB Skala singular value: 102 Runtime: 0.32 detik Pixel compression percentage: 7.0%
	20%		Ukuran File: 145 KB Skala singular value: 51 Runtime: 0.13 detik Pixel compression percentage: 9.87%
	2%		Ukuran File: 139 KB Skala singular value: 5 Runtime: 0.06 detik Pixel compression percentage: 14.57%

6.	100% (Asli)		Ukuran File: 35 KB Ukuran Gambar: 512 x 512 File type: PNG
	80%		Ukuran File: 204 KB Skala singular value: 205 Runtime: 1.17 detik Pixel compression percentage: 27.41%
	40%		Ukuran File: 306 KB Skala singular value: 102 Runtime: 0.72 detik Pixel compression percentage: 38.66%
	20%		Ukuran File: 321 KB Skala singular value: 51 Runtime: 0.25 detik Pixel compression percentage: 43.95%
	2%		Ukuran File: 347 KB Skala singular value: 5 Runtime: 0.06 detik Pixel compression percentage: 60.96%

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

5.1. Kesimpulan

Program kompresi gambar yang mengaplikasikan materi yang berhubungan dengan matriks yang dipelajari dalam mata kuliah IF2123 Aljabar Linier dan Geometri telah berhasil dirancang, dibuat, dan dijalankan. Adapun materi-materi yang diimplementasikan pada program ini berupa:

1. Nilai Eigen dan Vektor Eigen
2. Singular Value Decomposition
3. Operasi Matriks dan Sifat-sifatnya

Melalui penerapan materi-materi tersebut, program ini berhasil diselesaikan sesuai dengan ketentuan yang tertera pada spesifikasi Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri dengan mencari nilai dan vektor eigen suatu matriks dan kemudian menggunakannya pada metode *singular value decomposition* sehingga dapat dilakukan kompresi dengan cara memotong ukuran matriks-matriks dekomposisi sedemikian rupa sehingga bagian yang tersisa merupakan nilai terpentingnya saja. Hal ini menyebabkan ukuran informasi yang ditampung oleh matriks untuk mengecil tanpa mengubah bentuk dan nilai awal matriks.

Vektor eigen sendiri merupakan vektor yang berada pada suatu ruang eigen (*eigenspace*) sehingga terdapat beragam kemungkinan nilai dari suatu vektor eigen. Nilai yang inkonsisten ini dapat berpengaruh pada hasil perkalian matriks-matriks dekomposisi, karena vektor eigen sendiri digunakan untuk mencari matriks $\boldsymbol{U}$ dan $\boldsymbol{V}$ pada *singular value decomposition*. Oleh karena itu, agar hasil dari perkalian $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, dan $\boldsymbol{V^T}$ secara konsisten menghasilkan matriks orisinil, algoritma *singular value decomposition* kami mencari matriks $\boldsymbol{U}$ melalui hasil dari matriks $\boldsymbol{V}$. Dengan begini, nilai dari matriks $\boldsymbol{U}$ akan menyesuaikan diri dengan nilai dari vektor eigen yang digunakan untuk mencari matriks $\boldsymbol{V}$.

5.2. Saran

Pengerjaan Tugas Besar 2 Aljabar Linier dan Geometri tentunya tidak luput dari hambatan dan juga kendala. Demi pelaksanaan yang lancar, penulis telah mencatat beberapa saran serta poin yang dapat diperhatikan bagi pihak yang berminat untuk membuat program kompresi gambar berdasarkan nilai eigen, vektor eigen, dan SVD:

a. Guna memahami hal apa saja yang dibutuhkan dalam proses pengerjaan serta mengetahui keunggulan dan kekurangan dari setiap *framework* dan modul yang dapat digunakan, penulis menyarankan untuk mengalokasi waktu demi mendalami serta memahami masalah yang akan diselesaikan serta penyelesaian yang dirasa sesuai sebelum memulai tahap pengerjaan.
b. Perencanaan yang matang dapat menunjang keberlangsungan program serta kelancaran dari proses pengerjaan. Oleh karena itu, penulis sangat menitikberatkan pentingnya keberadaan rancangan program yang matang dan rinci demi menghindari terjadinya pemilihan *framework* yang kurang tepat guna, modul yang tidak kompatibel, hingga struktur atau bentuk program yang inefisien.
c. Apabila pengerjaan dilakukan dalam bentuk kelompok seperti yang dilakukan oleh para penulis, sangat direkomendasikan untuk melakukan pembagian tugas dengan pembobotan yang rata dan sesuai dengan kemampuan dan kapasitas dari masing-masing anggota kelompok. Disarankan juga agar pembagian tugas ini dilakukan setelah setiap anggota sudah melakukan eksplorasi terhadap permasalahan yang akan dipecahkan dan gambaran umum program sudah ada agar tugas yang dibagi sudah jelas dan pengerjaannya dilakukan secara terarah walaupun terpisah.
d. Seharusnya kami lebih baik dalam melakukan pengambilan keputusan mengenai modul pengolahan citra yang digunakan, karena untuk sekarang kami menggunakan 2 modul berbeda, dengan alasan salah satu modul yang digunakan tidak kompatibel dengan *framework* web yang kami pakai.

5.3. Refleksi

Tugas Besar 2 Aljabar Linier dan Geometri telah menjadi proses pembelajaran yang sangat berharga bagi para penulis. Aplikasi maupun layanan kompresi gambar telah tersedia di internet secara gratis dan sering digunakan oleh banyak orang. Walau terlihat sederhana, para penulis menyadari bahwa proses pembuatannya tidaklah semudah yang dibayangkan. Oleh karena itu, timbul rasa apresiatif dari lubuk hati penulis bagi pihak-pihak yang berkutat dalam pembuatan algoritma dan juga program yang digunakan sehari-hari, khususnya algoritma dan juga program yang berhubungan dengan kompresi gambar. Tak hanya itu, para penulis juga semakin menghargai keberadaan dan juga usaha yang dituangkan dalam penciptaan modul dan *framework* seperti *NumPy, OpenCV, Flask*, dan lain sebagainya.

Tentunya kelancaran proses pengerjaan tugas besar ini jauh dari ideal. Beragam kendala dan juga hambatan ditemui oleh para penulis selama proses penyelesaian tugas besar. Sebagai contoh, pengambilan keputusan modul pengolahan citra berdasarkan kompatibilitas dengan *framework* dari *backend* telah membuat para penulis menemui hambatan lainnya yakni ukuran *file* kompresi yang kurang ideal sehingga mengakibatkan para penulis untuk menggunakan dua modul pengolahan citra untuk mengimbangi kekurangan dan keunggulan satu sama lain pada akhirnya. Namun, dari kejadian tersebut dan kejadian lainnya, para penulis menyadari pentingnya perencanaan yang matang serta pemahaman yang jelas dalam proses pemecahan suatu masalah.

Para penulis juga menyadari pentingnya aspek ketelitian guna menghindari masalah-masalah kecil yang diakibatkan oleh kelalaian. Penggunaan komentar yang singkat, padat, dan jelas dapat membantu menghindari miskonsepsi dalam suatu kode. Selain itu, komunikasi yang lancar antaranggota kelompok dapat membantu melancarkan proses integrasi kode antaranggota. Terakhir, penting bagi setiap anggota kelompok untuk memiliki tidur yang cukup demi mempertahankan fokus dan kesehatan jasmani maupun rohani, karena apalah arti nilai yang bagus apabila kondisi jiwa dan raga tidak bagus.

REFERENSI

Youtube.com (2019). Using SVD for image compression in Python (Singular Value Decomposition). Diakses pada 3 November 2021, dari https://www.youtube.com/watch?v=SU851ljMIZ8

Web.mit.edu (2015). Singular Value Decomposition (SVD) tutorial. Diakses pada 5 November 2021, dari https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

Math.utah.edu (2016). Linear Algebra in Image Compression: SVD and DCT. Diakses pada 6 November 2021, dari http://www.math.utah.edu/~gustafso/s2019/2270/projects-2019/presented/fraser/Linear%20Algebra%20in%20Image%20Compression_%20SVD%20and%20DCT.pdf

Mse.redwoods.edu (2006). Image Compression Using Singular Value Decomposition. Diakses pada 8 November 2021, dari https://mse.redwoods.edu/darnold/math45/laproj/fall2006/iancraig/SVD_paper.pdf

Math.utah.edu (2014). Image Compression using Singular Value Decomposition (SVD). Diakses pada 10 November 2021, dari http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf

Cfm.brown.edu (2017). Linear Systems of Algebraic Equations: SVD factorization. Diakses pada 10 November 2021, dari https://www.cfm.brown.edu/people/dobrush/am34/sage/svd.html