

TUGAS BESAR 2 IF 2123
ALJABAR LINIER DAN GEOMETRI

oleh

Rozan Fadhil Al Hafidz	13520039
Andreas Indra Kurniawan	13520091
Zayd Muhammad Kawakibi Zuhri	13520144



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

DAFTAR ISI

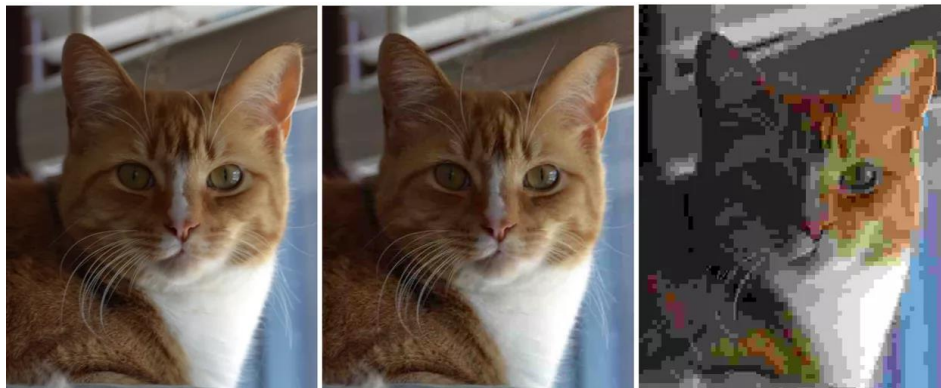
DAFTAR ISI.....	2
DESKRIPSI MASALAH.....	3
BAB II : DASAR TEORI	5
2.1. Perkalian Matriks	5
2.2. Nilai-Nilai dan Vektor-Vektor Eigen.....	5
2.3. Singular Value Decomposition	6
2.4. Kompresi Matriks Menggunakan Metode SVD	7
2.5. Kompresi Gambar Menggunakan Metode SVD	8
BAB III : IMPLEMENTASI.....	9
3.1. eigen.py	9
3.2. compressImage.py	10
3.3. Vue	11
3.4. Flask	12
BAB IV : EKSPERIMEN.....	13
Hasil Eksekusi Program	13
BAB V : PENUTUPAN	17
5.1. Kesimpulan.....	17
5.2. Saran	17
5.3. Refleksi.....	17
DAFTAR PUSTAKA.....	18

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 \(lifewire.com\)](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.

BAB II

DASAR TEORI

2.1. Perkalian Matriks

Perkalian dua buah matriks $C_{m \times n} = A_{m \times r} \times B_{r \times n}$

Misal $A = [a_{ij}]$ dan $B = [b_{ij}]$ maka $C = A \times B = [c_{ij}]$, $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$

Syarat: jumlah kolom A sama dengan jumlah baris B

2.2. Nilai-Nilai dan Vektor-Vektor Eigen

Jika terdapat matriks A berukuran $n \times n$ maka vektor tidak-nol di \mathbb{R}^n disebut **vektor eigen** dari A jika Ax sama dengan perkalian suatu skalar λ dengan x , yaitu:

$$Ax = \lambda x$$

Dengan skalar λ disebut **nilai eigen** dari A, dan x dinamakan **vektor eigen** yang berkoresponden dengan nilai eigen λ .

Cara paling umum untuk mencari nilai dan vektor eigen adalah untuk menggunakan persamaan $(\lambda I - A)x = 0$ untuk mencari vektor eigen, dimana untuk mendapat solusi nontrivial dari nilai eigen λ haruslah $\det(\lambda I - A) = 0$. Namun, cara ini meliputi banyak perhitungan simbolis atau penggunaan resolusi polinomial untuk mencari nilai dan vektor eigen. Jika kita memasukkan matriks kecil, tidak masalah. Namun, untuk kompresi gambar, gambar akan berukuran ratusan bahkan ribuan pixel yang ekuivalen dengan matriks berukuran ratusan baris dan ratusan kolom. Metode klasik ini sangat lambat, bahkan kadang tidak bisa mencarinya. Karena itu, diperlukan metode yang iteratif, walaupun tidak 100% mutlak akurat, namun jauh lebih cepat dalam mencari nilai dan vektor eigen.

Walaupun hanya satu metode yang memang digunakan untuk kompresi, terdapat 3 metode berbeda yang diimplementasikan dalam eigen.py.

Yang pertama (`get_eigen_values()`) adalah algoritma QR, yang dapat digunakan untuk mencari semua nilai eigen suatu vektor. Setiap iterasi meliputi langkah berikut:

$$A_k = Q_k R_k$$
$$A_{k+1} = R_k Q_k$$

Metode ini memanfaatkan dekomposisi QR dari matriks A dengan memecahnya menjadi matriks ortogonal Q dan matriks triangular atas R. Untuk mendapatkan iterasi A selanjutnya, R iterasi sebelumnya dikalikan Q iterasi sebelumnya. Dua langkah ini diulang sejumlah 10000/(jumlah baris matriks) kali dalam implementasi kami. Untuk mempercepat iterasi, pada awal fungsi matriks A diubah terlebih dahulu menjadi bentuk hessenberg. Saat iterasi beres, nilai-nilai eigen matriks A akan terletak pada diagonal matriks A_{k+n} .

Metode kedua (`get_eigen_vectors()`) memanfaatkan rumus Shifted Inverse Power Method untuk mendapatkan vektor eigen tiap nilai eigen matriks A dalam satu iterasi. Rumusnya sebagai berikut:

$$x = \frac{(A - \lambda I)^{-1} b}{\|(A - \lambda I)^{-1} b\|}$$

Dengan menginisiasi vektor b sesuai ukuran vektor eigen x dan berisi angka random (dalam

kasus kami dengan angka 1), rumus tersebut akan menghasilkan vektor eigen x yang bersesuaian dengan nilai eigen λ dari matriks A . Namun, metode ini memiliki satu masalah besar: perhitungan inverse matriks. Jika hasil $A - \lambda I$ merupakan matriks singular, maka akan terjadi error karena tidak dapat dihitung inversenya.

Karena kekurangan tersebut, kami membuat fungsi ketiga yang dapat mengembalikan nilai dan vektor eigen sekaligus, dengan iterasi yang sama. Fungsi `get_eigen()` memanfaatkan simultaneous power method atau juga disebut Orthogonal Iteration. Metode ini merupakan versi extended dari Power Method, dengan memanfaatkan QR decomposition untuk mengortogonalnkan vektor-vektor eigen. Berikut algoritmanya:

$$\begin{aligned} B &= Q_0 R_0 \\ A Q_{k-1} &= Z_k \\ Z_k &= Q_k R_k \end{aligned}$$

Pada awalnya, diinisialisasi dengan matriks B random, tetapi seukuran dengan matriks asal A . Matriks B ini didekomposisi menjadi matriks ortogonal Q dan matriks triangular atas R pada awal algoritma. Untuk iterasi ke-1 dan seterusnya, matriks Q dikalikan dengan matriks asal A menjadi matriks Z . Matriks Z tersebut kembali didekomposisi menjadi matriks Q dan R iterasi selanjutnya. Pada akhir iterasi, nilai-nilai eigen akan terletak pada diagonal R dan vektor-vektor eigen merupakan kolom-kolom matriks Q . Metode ini merupakan variasi power method karena sekaligus menginisialisasi matriks semua vektor eigen dalam dekomposisi QR, yang kebetulan matriks Q -nya bersifat ortogonal sehingga terbentuklah vektor eigen, dan juga nilai eigen dalam R .

2.3.Singular Value Decomposition

Singular value decomposition adalah sebuah metode untuk mendekomposisikan matriks non-bujursangkar menjadi tiga buah matriks. Misalkan kita akan mendekomposisikan matriks A menggunakan metode SVD, maka hasilnya adalah sebagai berikut

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

Keterangan :

U = matriks ortogonal $m \times m$

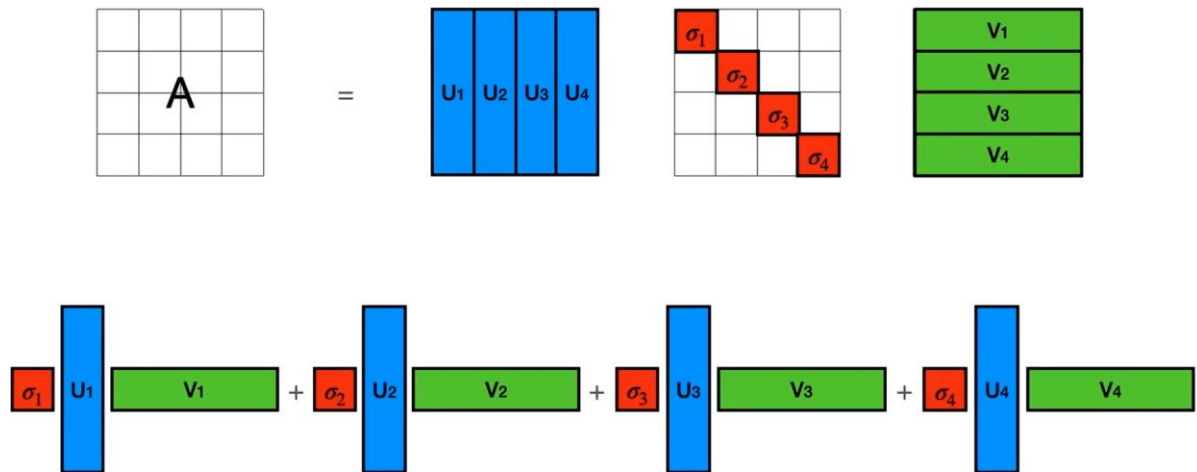
V = matriks ortogonal $n \times n$

Σ = matriks berukuran $m \times n$

Matriks U adalah matriks yang berisi vektor-vektor eigen dari matriks $A \times A^T$ yang telah dinormalisasi

Matriks V adalah matriks yang berisi vektor-vektor eigen dari matriks $A^T \times A$ yang telah dinormalisasi

Matriks Σ adalah matriks diagonal yang elemen diagonalnya yang berisi akar pangkat dua dari nilai eigen tidak nol dari matriks $A^T \times A$ yang terurut mengecil.

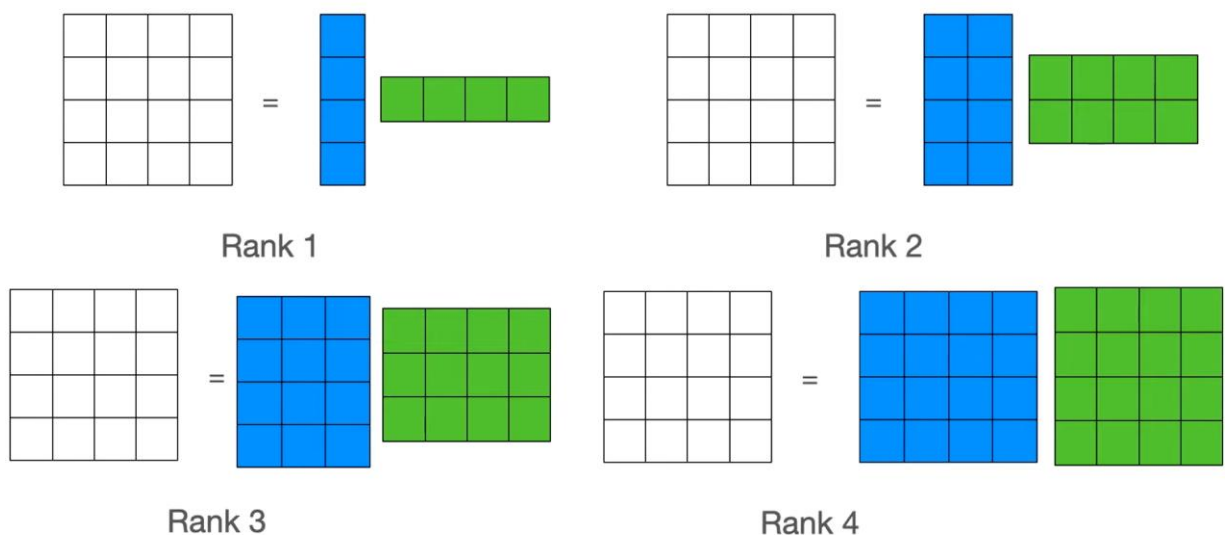


Gambar 3. Ilustrasi dekomposisi matriks A

2.4. Kompresi Matriks Menggunakan Metode SVD

Kita bisa mengonstruksi kembali matriks SVD dengan cara mengalikan matriks U , Σ , dan V^T . Pada matriks SVD, informasi terpenting dimuat pada kolom pertama. Oleh karena itu, kita bisa mengompres matriks tersebut menjadi matriks baru yang mirip dengan matriks awal dengan cara mengalikan matriks U , Σ , dan V^T namun hanya sebagian baris saja. Akibatnya, kita hanya mendapatkan informasi-informasi penting dari matriks tersebut dan membuang informasi yang tidak terlalu penting.

Rank of a matrix



Gambar 3. Ilustrasi mengonstruksi kembali matriks A

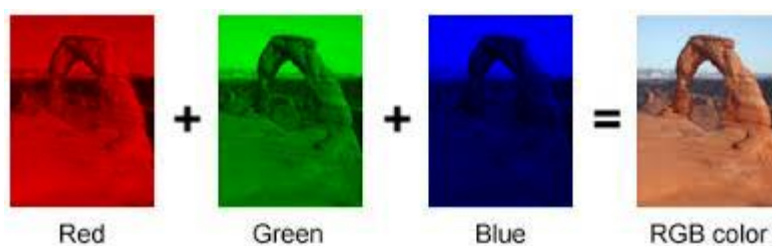
2.5. Kompresi Gambar Menggunakan Metode SVD

Gambar terdiri dari kumpulan *pixel* (*picture element*) yang merupakan elemen terkecil yang memuat informasi pada gambar. *Pixel* biasanya berbentuk kodak yang tersusun di dalam bidang dua dimensi.



Gambar 4. Pixel yang menyusun gambar.

Selain itu, gambar bisa terdiri dari beberapa channel. Channel adalah komponen warna pada suatu gambar. Misalnya, suatu gambar bisa terdiri dari komponen RGB (*Red, Green, Blue*), RGBA (*Red, Green, Blue, Alpha*), atau CYMK (*Cyan, Yellow, Magenta, Key*). Untuk mengompres suatu gambar, kita perlu membagi komponen-komponen tersebut, mengompresnya satu per satu, kemudian menyatukannya kembali.



Gambar 5. Komponen-komponen dari gambar RGB.

Dari komponen-komponen tersebut, kita bisa mengubah setiap *pixel*-nya menjadi integer dengan elemen antara 0 sampai 255. Jadi, komponen tersebut dapat kita ubah menjadi matriks yang berelemen integer. Oleh karena itu, kita bisa melakukan kompresi gambar dengan cara mengompres matriks-matriks dari tiap komponen tersebut.

BAB III

IMPLEMENTASI

Pada program yang kami buat, kami menggunakan beberapa library python untuk membantu program kami dalam melakukan proses kompresi gambar. Berikut ini adalah library-library yang kami gunakan.

a. Library PIL

Library ini digunakan untuk membaca gambar dan memecah channel-channelnya. Setelah itu, library ini digunakan untuk menggabungkan kembali channel-channel yang telah dikompres menjadi satu gambar yang baru.

b. Library numpy

Library ini digunakan sebagai penampung variabel matriks yang berisi data dari gambar. Selain itu, library ini juga digunakan untuk melakukan operasi-operasi pada matriks tersebut yang bertujuan untuk mengompres gambar

3.1. eigen.py

File ini berisi fungsi-fungsi untuk kebutuhan perhitungan nilai-nilai maupun vektor-vektor eigen. Terdapat 3 fungsi yang dapat digunakan, dan ketiganya memanfaatkan metode-metode berbeda. Fungsi yang digunakan untuk SVD yaitu `get_eigen()`, dengan fungsi lainnya tidak digunakan pada produk akhir namun dibiarkan karena sempat digunakan untuk testing sebelumnya, dan cukup berguna untuk kebutuhan lain.

1. `def get_eigen_values(matrix)`

Fungsi ini digunakan untuk mendapatkan semua nilai-nilai eigen suatu matriks. Hal ini dilakukan dengan menggunakan QR Algorithm yang sudah dijelaskan dalam bagian teori. Fungsi ini mengubah matriks asal 'matrix' menjadi bentuk hessenberg, lalu diiterasi sesuai QR Algorithm dengan jumlah iterasi sebanyak 10000 dibagi ukuran matrix.

Fungsi ini menerima satu argumen, yaitu 'matrix' berupa `numpy.array` berdimensi dua.

Fungsi mengembalikan nilai-nilai eigen terurut membesar berupa variable 'eig_vals' dalam bentuk `numpy.array`.

2. `def get_eigen_vectors(matrix, eig_vals)`

Fungsi ini digunakan untuk mendapatkan semua vektor eigen dari tiap nilai eigen yang diberikan. Fungsi ini memanfaatkan rumus Shifted Inverse Power Method yang dijelaskan dalam bagian teori.

Fungsi ini menerima dua argumen, yaitu:

- `matrix` : Variabel bertipe `numpy array` dua dimensi atau matriks
- `eig_vals` : Variabel bertipe `numpy array` berisi nilai-nilai eigen dari variabel matriks

Dan mengembalikan satu variabel bertipe `numpy array` dua dimensi dengan tiap baris merupakan suatu vektor eigen dari matrix.

3. `def get_eigen(matrix)`

Fungsi ini digunakan untuk mendapatkan seluruh nilai-nilai eigen dan juga vektor-vektor eigen dari matrix. Hal ini dilakukan dengan menggunakan Orthogonal Iteration, yang sudah dirincikan dalam bagian teori.

Fungsi ini menerima satu argumen, yaitu 'matrix' berupa numpy.array berdimensi dua.

Fungsi mengembalikan hasil berupa tuple yang berisi:

- eig_vals : Variabel bertipe numpy array berisi nilai-nilai eigen dari variabel matriks
- eig_vecs : Variabel bertipe numpy array berdimensi dua dengan tiap kolom merupakan vektor eigen dari variabel matriks

3.2. compressImage.py

File compressImage.py berfungsi untuk melakukan kompresi gambar menggunakan metode SVD. Untuk menjalankan tugasnya, file ini terdiri dari beberapa fungsi, yaitu :

1. Fungsi pembuka gambar

def openImage(img)

- Fungsi ini berfungsi untuk memecah sebuah gambar menjadi channel-channel gambar.
- Fungsi ini menerima argumen img, yaitu variabel bertipe gambar yang akan dipecah
- Fungsi ini mengembalikan 5 variabel, yaitu:
 - imgChannels : List yang berisi semua channel dari gambar yang diterima selain channel alpha. Setiap elemennya bertipe Image
 - imgAlpha : Variabel bertipe Image yang berisi Channel alpha dari gambar yang diterima (jika ada). Jika tidak ada, akan bernilai nan.
 - hasAlphaValue : Boolean yang bernilai true jika gambar memiliki channel alpha atau false jika tidak
 - bands : Berisi tuple dari band, yaitu jenis-jenis channel yang dimiliki gambar yang diterima fungsi

2. Fungsi pemecah matriks menjadi SVD

def getSVDMatrices(m, rank)

- Fungsi ini berfungsi untuk mendekomposisi channel gambar menjadi matriks SVD.
- Fungsi ini menerima dua argumen, yaitu :
 - m : Variabel bertipe Image yang hanya mengandung satu channel. Variabel ini yang akan didekomposisi menjadi matriks SVD
 - rank : Variabel bertipe integer yang akan menentukan sampai rank ke-berapa matriks akan didekomposisi
- Fungsi ini mengembalikan 3 variabel, yaitu :
 - u : Variabel bertipe numpy matriks yang memuat vektor-vektor eigen dari matriks $m \times m^T$
 - s : Variabel bertipe numpy matriks yang berisi akar-akar dari nilai nilai eigen dari matriks m.

- `vt` : Variabel bertipe numpy matriks yang memuat transpose dari vektor-vektor eigen dari matriks $m^T \times m$.
3. Fungsi pengompres sebuah channel gambar
- ```
def compressSingleChannel(channel, rank)
```
- Fungsi ini berfungsi untuk mengompres sebuah channel dari gambar. Untuk menjalankan tugasnya, fungsi ini memanggil fungsi pemecah matriks SVD (`getSVDMatrices(m, rank)`)
  - Fungsi ini menerima dua argumen, yaitu :
    - `channel` : Variabel bertipe Image yang hanya mengandung satu channel. Variabel ini yang akan dikompres agar ukurannya lebih kecil.
    - `rank` : Variabel bertipe integer yang akan menentukan sampai rank ke-berapa matriks akan dibentuk kembali. Semakin kecil rank, semakin kecil ukuran hasilnya.
  - Fungsi ini mengembalikan satu variabel, yaitu variabel bertipe numpy matriks yang berisi channel yang telah dikompres
4. Fungsi pengompres sebuah gambar
- ```
def compressImage(image, percentage, imageName)
```
- Ini adalah fungsi utama dari file `compressImage.py`. Fungsi ini berfungsi untuk mengompres sebuah gambar utuh. Untuk menjalankan tugasnya, fungsi ini memanggil fungsi-fungsi lain yang ada di dalam file ini.
 - Fungsi ini menerima tiga argumen, yaitu :
 - `image` : Variabel bertipe Image yang merupakan gambar yang akan dikompres.
 - `percentage` : Variabel bertipe integer yang persentase kompresi gambar. Variabel ini yang akan menentukan sejauh mana gambar akan dikompres.
 - `imageName` : Variabel bertipe string yang memuat informasi nama gambar yang akan dikompres beserta ekstensinya. Variabel ini digunakan untuk menentukan ekstensi gambar hasil kompresi.
 - Fungsi ini mengembalikan satu variabel, yaitu variabel bertipe string yang memuat ekstensi gambar.

3.3. Vue

Vue memiliki beberapa folder di dalamnya. Namun folder yang perlu diperhatikan adalah folder Data dan src. Folder data ini berisi file json yang berfungsi sebagai database pada server website. Folder Src berisi komponen-komponen utama dari vue.

`input_image.vue`

File ini merupakan bagian front-end berisi form yang dapat di submit oleh user dan berisi berbagai fungsi yang dapat digunakan yaitu:

1. Reset yang berguna untuk mengosongkan database dan mengembalikan ke state semula

2. fileSelected yang berfungsi untuk memasukan file yang diinput ke database lalu mengubah state form sehingga ditampilkan image yang di submit
3. compress yang akan dipanggil ketika button submit ditekan, berfungsi untuk meminta flask menjalankan fungsi kompresi lalu save gambar ke file server dan me return gambar yang telah di compress ke vue.
4. download untuk melakukan download ke folder download di komputer.
5. startTimer untuk menandai kapan waktu compress dimulai
6. stopTimer untuk menghentikan timer

3.4. Flask

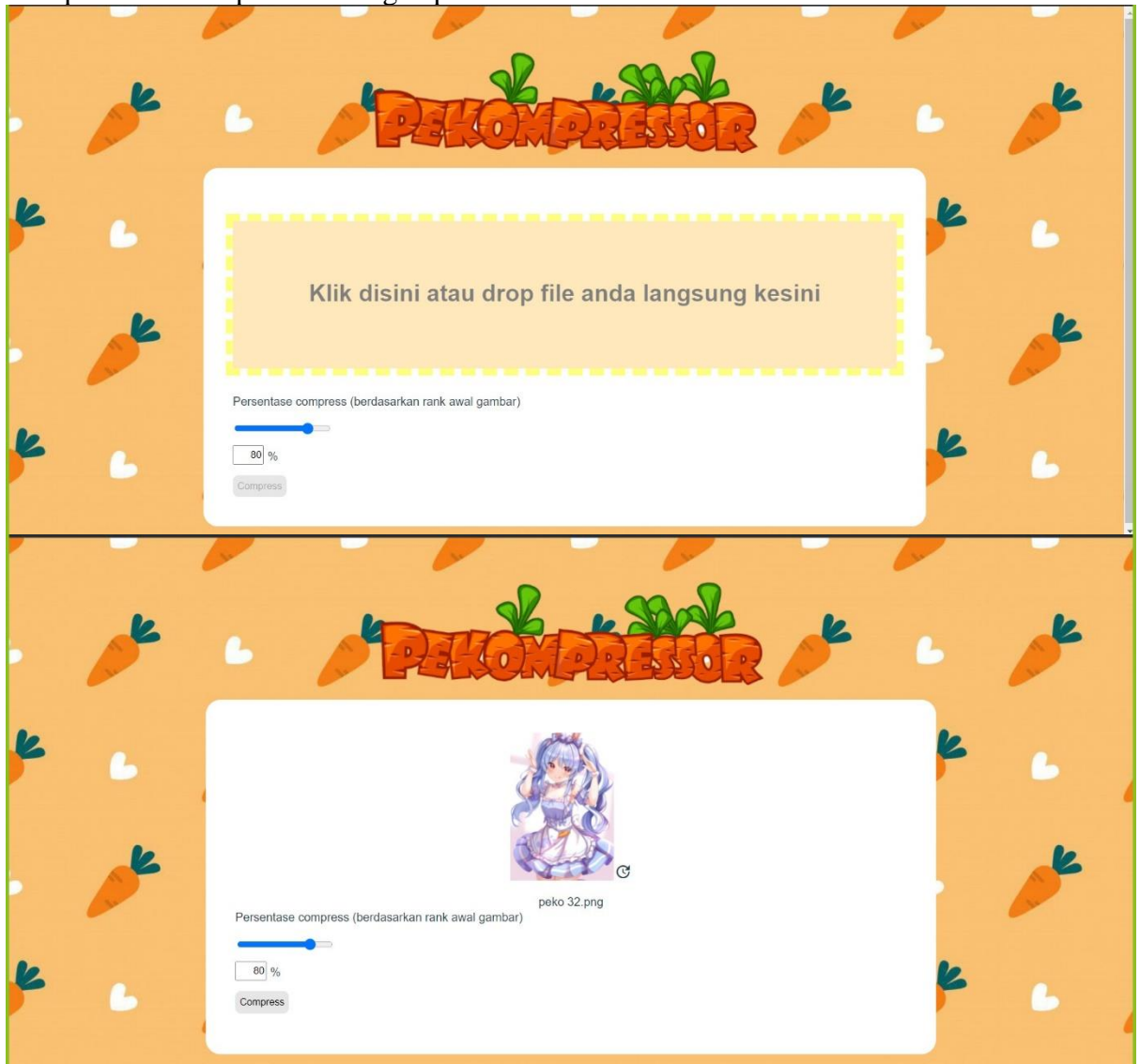
Flask ini berisi banyak folder, namun yang perlu diperhatikan adalah folder env/Scripts yang digunakan untuk mengaktifkan virtual environment python lalu file app.py yang dapat menyambungkan python ke server flask sehingga fungsi-fungsi python dapat dipanggil oleh front-end menggunakan router.

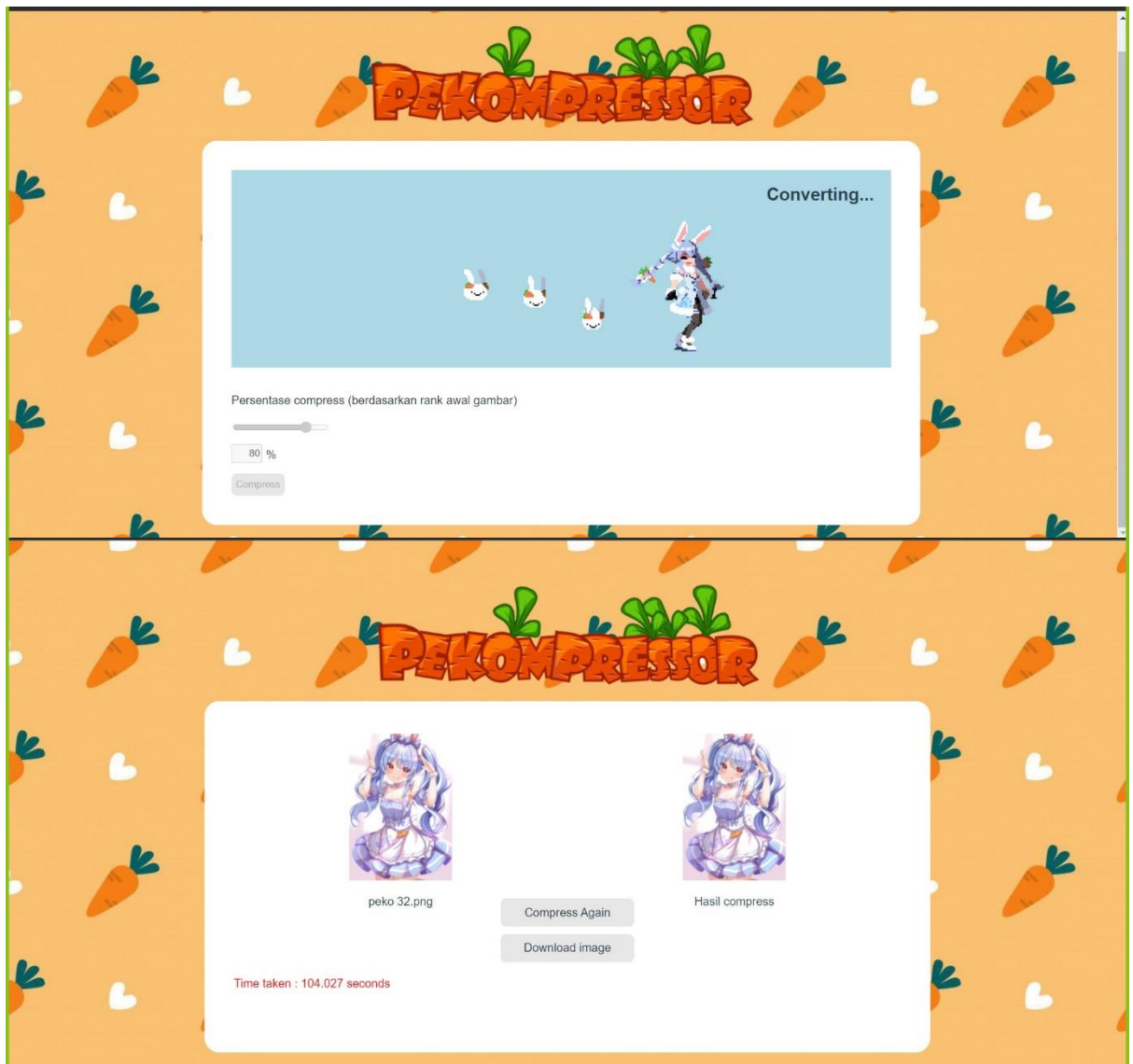
BAB IV



EKSPERIMEN

Hasil Eksekusi Program

1. Compress file bertipe PNG dengan persentase 80%







 peko 32	9/29/2021 5:15 PM	PNG File	3,588 KB
 peko 32_Compressed	11/15/2021 12:16 AM	PNG File	480 KB

- Kompresi gambar jpeg dengan resolusi 2400 x 3500 dengan persentase 80%





 gurawr	11/14/2021 10:51 PM	JPG File	1,086 KB
 gurawr_Compressed	11/15/2021 12:05 AM	JPG File	370 KB

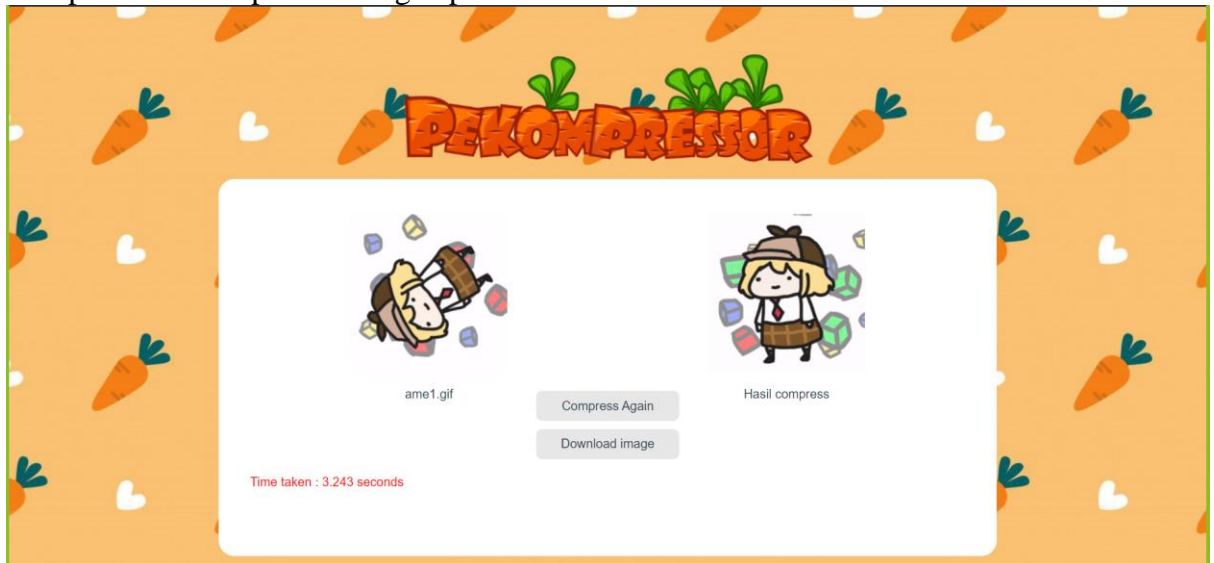
- Compress image png yang tidak persegi dengan persentase 100%





Gambar kiri merupakan image asli dan image kanan merupakan hasil compress dengan persentase 100% di website. Walaupun persentasenya 100% tetapi tetap terjadi penurunan ukuran file karena karena 100% yang diambil merupakan persentase rank dari matrix. Karena gambar tersebut tidak persegi, maka pasti ada komponen yang hilang dari gambar tersebut.

 ame_brain_damage	11/14/2021 10:51 PM	PNG File	418 KB
 ame_brain_damage_Compressed	11/14/2021 11:15 PM	PNG File	50 KB

4. Compress file bertipe GIF dengan persentase 0%





File bertipe GIF akan otomatis berubah menjadi file image yang static dengan extension GIF

 ame1	12/26/2020 2:37 AM	GIF File	664 KB
 ame1_Compressed	11/14/2021 11:30 PM	GIF File	12 KB

5. File bertipe png yang mempertahankan transparansi dan kompresi dengan persentase 80%



 shibakiage punch	4/26/2021 11:26 PM	PNG File	63 KB
 shibakiage punch_Compressed	11/15/2021 12:19 AM	PNG File	51 KB

BAB V

PENUTUPAN

5.1. Kesimpulan

Kesimpulan dari tugas besar ini adalah pencarian nilai eigen yang di ajarkan di kelas aljabar linear dan geometri dapat digunakan untuk aplikasi nyata seperti kompresi gambar jika fungsi pencarian nilai eigen dikembangkan.

5.2. Saran

Untuk pengembangan lebih lanjut, program kami dapat dikembangkan untuk bagian back-end bisa dilakukan penanganan lebih lanjut sehingga Ketika terjadi error server tidak perlu di restart ulang.

5.3. Refleksi

-13520091 Andreas

Selama pengerjaan tugas ini saya mengalami kesulitan dalam menyambungkan front-end dan back-end karena saya baru mempelajari struktur dari back-end. Selain itu saya juga mengalami kesulitan saat belajar vue.js dan routernya.

-13520144 Zayd

Pada awal pengerjaan tubes ini saya mengira kalau tugas menghitung nilai dan vektor eigen suatu adalah hal yang cukup mudah. Namun, setelah memulai tubes, perlahan-lahan terlihat betapa susahya menghitung eigen dari suatu matriks yang besar, seukuran gambar. Setelah melalui banyak website, paper, dan rumus, saya menemukan *respect* dan kagum yang lebih terhadap orang di bidang matematika yang mengurus hal seperti ini, dan bagaimana berbagai metode dan teorema mereka menjadi berguna di dunia nyata.

-13520039 Rozan

Selama mengerjakan tugas ini, saya mengalami kesulitan dalam memanfaatkan kemudahan bahasa python sehingga saya harus sering mencari di google cara melakukan suatu operasi dengan memanfaatkan library yang sudah tersedia di bahasa python.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2021-2022/algeo21-22.htm>

<http://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html>

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

https://youtu.be/LHlg_lfihiA

<https://testdriven.io/blog/developing-a-single-page-app-with-flask-and-vuejs/#bootstrap-setup>

<https://www.udemy.com/course/build-web-apps-with-vuejs-firebase/>

<https://www.ultimate-photo-tips.com/what-is-a-pixel.html>

<https://youtu.be/DG7YTIGnCEo>

<https://youtu.be/H7qMMudo3e8>