

# Laporan Tugas Besar 2 IF2123

## Aljabar Linier dan Geometri

### Aplikasi Nilai Eigen dan Vektor Eigen dalam Kompresi Gambar



oleh  
**Kelompok 49 (FRS)**

Febryola Kurnia Putri (13520140)  
Steven Gianmarg H. Siahaan (13520145)  
Raden Rifqi Rahman (13520166)

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

## BAB I

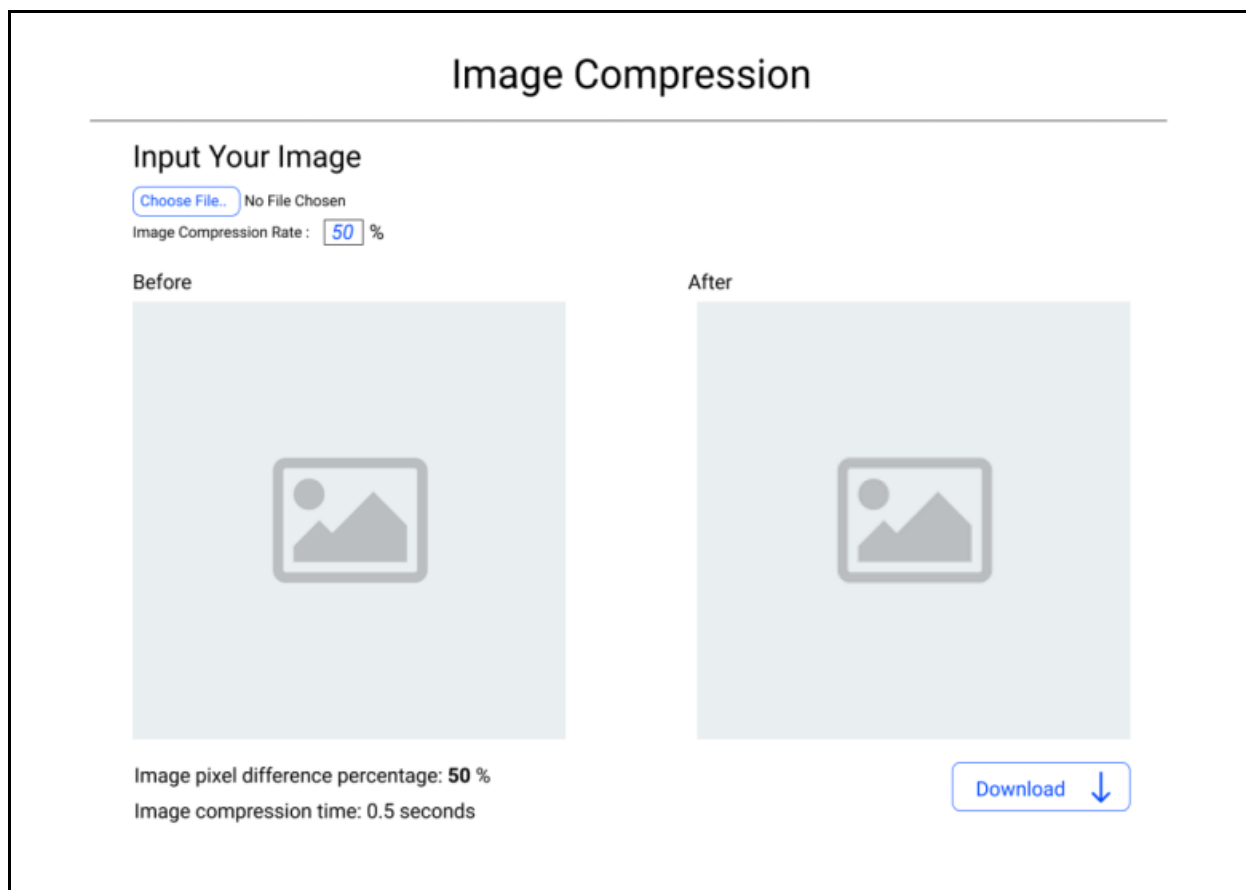
### Deskripsi Masalah

Pada tugas besar dua mata kuliah Aljabar Linier dan Geometri ini, kami diminta untuk membuat sebuah website yang bertujuan untuk mengompresi gambar. Detail penjelasannya sebagai berikut:

Berikut ini deskripsi penggunaan program berupa 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.



Gambar 1. Contoh tampilan layout dari aplikasi web yang dibangun.

Berikut spesifikasi dari program:

Membuat 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. Dekomposisi Matriks

Mendekomposisi matriks artinya memfaktorkan sebuah matriks, misalnya A, menjadi hasil kali dari sejumlah matriks lain,  $P_1, P_2, \dots, P_k$ .

$$A = P_1 P_2 \dots P_k$$

Hal ini dapat diselesaikan dengan menggunakan metode Singular Value Decomposition (SVD) yang akan dijelaskan pada bagian berikut ini.

#### B. Singular Value Decomposition (SVD)

seperti yang telah diketahui bahwa matriks bujursangkar A berukuran  $n \times n$  dapat difaktorkan menjadi:

$$A = EDE^{-1}, \text{ dengan}$$

E adalah matriks yang kolom-kolomnya adalah basis ruang eigen dari matriks A,

$$E = (e_1 | e_2 | \dots | e_n)$$

D adalah matriks diagonal sedemikian sehingga  $D = E^{-1}AE$

SVD memfaktorkan matriks A berukuran  $m \times n$  menjadi matriks U,  $\Sigma$ , dan V sedemikian sehingga

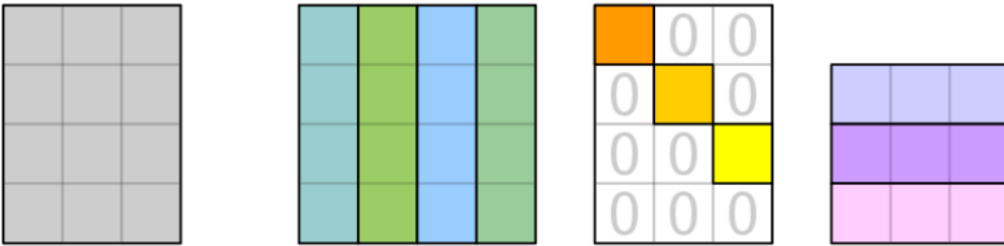
$$A = U\Sigma V^T, \text{ dengan}$$

U = matriks ortogonal  $m \times m$ ,

V = matriks orthogonal  $n \times n$

$\Sigma$  = matriks berukuran  $m \times n$  yang elemen-elemen diagonal utamanya adalah nilai-nilai singular dari matriks A dan elemen-elemen lainnya 0

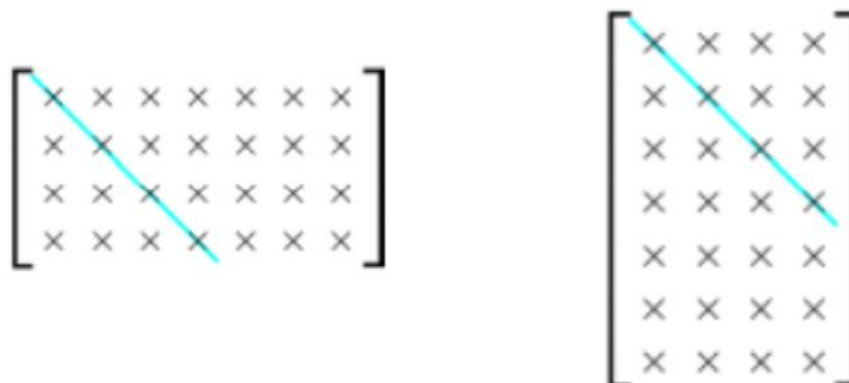
Visualisasinya dapat dilihat sebagai berikut.



$$\begin{matrix} \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

#### C. Diagonal Matriks

Diagonal utama sebuah matriks didefinisikan pada matriks persegi (matriks bujursangkar) berukuran  $n \times n$ . Untuk matriks bukan bujursangkar, yaitu matriks  $m \times n$ , diagonal utama matriks didefinisikan pada garis yang dimulai dari sudut kiri atas terus ke bawah matriks sejauh mungkin. Visualisasinya dapat dilihat sebagai berikut.



#### D. Matriks Ortogonal

Matriks ortogonal adalah matriks yang kolom-kolomnya adalah vektor yang saling orthogonal satu sama lain (hasil kali titik sama dengan 0). Jika vektor-vektor kolom tersebut merupakan vektor satuan, maka matriks ortogonal tersebut dinamakan juga matriks ortonormal. Vektor satuan adalah vektor yang dinormalisasi dengan panjang atau magnitudenya sehingga memiliki panjang atau magnitude = 1. Jika  $Q$  adalah matriks ortogonal  $m \times n$ , dan kolom-kolom matriks  $Q$  adalah vektor-vektor satuan  $v_1, v_2, \dots, v_m$ , maka  $v_i \cdot v_j = 0$  untuk  $i \neq j$ . Atau, dapat juga dikatakan bahwa  $Q$  adalah matriks ortogonal jika  $Q^T Q = I$ , dalam hal ini  $I$  adalah matriks identitas.

#### E. Nilai-Nilai Singular Matriks

Misalkan  $A$  adalah matriks  $m \times n$ . jika  $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$  adalah nilai eigen dari  $A^T A$ , maka nilai-nilai singularnya adalah  $\sigma_1 = \lambda_1^{1/2}, \sigma_2 = \lambda_2^{1/2}, \sigma_3 = \lambda_3^{1/2}, \dots, \sigma_n = \lambda_n$ .

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

Vektor-vektor  
singular kiri

Nilai-nilai  
singular

Vektor-vektor  
singular kanan

#### F. Langkah-Langkah Melakukan SVD

1. Untuk vektor singular kiri, hitung nilai-nilai eigen dari  $AA^T$ .  $\text{Rank}(A) = k =$  banyaknya nilai-nilai eigen tidak nol dari  $AA^T$ .
2. Tentukan vektor-vektor eigen  $u_1, u_2, \dots, u_m$  yang berkoresponden dengan nilai-nilai eigen dari  $AA^T$ . Normalisasi  $u_1, u_2, \dots, u_m$  dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks  $U$ .
3. Untuk vektor singular kanan, hitung nilai-nilai eigen dari  $A^T A$  lalu tentukan nilai nilai-singularnya.
4. Tentukan vektor-vektor eigen  $v_1, v_2, \dots, v_n$  yang berkoresponden dengan nilai-nilai eigen dari  $A^T A$ . Normalisasi  $v_1, v_2, \dots, v_n$  dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks  $V$ . Transpose-kan matriks  $V$  sehingga menjadi  $V^T$ .
5. Bentuklah matriks 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 adalah akar pangkat dua dari nilai-nilai eigen yang tidak nol dari  $A^T A$ .
6. maka terbentuklah  $A = U \Sigma V^T$

## BAB III

### Implementasi Program

#### A. Tech Stack

Berikut adalah kakas-kakas (*tools*) yang digunakan dalam implementasi program yang dibuat.

##### 1. Node.js

Node.js adalah *runtime* javascript asinkronus dan *event-driven* yang dibangun dari mesin javascript V8 Chrome. Node.js didesain untuk membangun aplikasi jaringan yang *scalable*. Node.js memungkinkan pengembang untuk menggunakan javascript untuk melakukan *server-side scripting*, yakni menjalankan javascript pada server untuk membangun halaman web yang dinamis sebelum halaman tersebut dikirim ke browser web pengguna.

##### 2. Express.js

Express.js adalah *framework* aplikasi web minimalis dan fleksibel pada Node.js yang menyediakan fitur-fitur yang tangguh untuk aplikasi web dan mobile. Dengan segudang utilitas *HyperText Transfer Protocol* (HTTP), Express.js memungkinkan pembuatan *Application Programming Interface* (API) yang mudah dan cepat.

##### 3. Python

Python adalah bahasa pemrograman yang memungkinkan kita untuk bekerja dengan cepat dan mengintegrasikan sistem secara lebih efektif. Python adalah bahasa pemrograman yang *powerful*, dapat dijalankan di mana saja, mudah untuk dipelajari, dan terbuka.

##### 4. PIL dan NumPy

PIL (Python Imaging Library) adalah salah satu pustaka dalam bahasa Python yang memberikan kapabilitas untuk melakukan pemrosesan gambar dalam program Python. Pustaka ini menyediakan dukungan format file yang ekstensif, representasi internal yang efisien, dan kapabilitas pemrosesan gambar yang cukup kuat.

NumPy adalah paket fundamental untuk komputasi ilmiah dalam bahasa Python. NumPy adalah sebuah pustaka Python yang menyediakan objek array multidimensi, berbagai macam variasi objek turunan (seperti matriks), dan koleksi rutin program untuk operasi array dengan cepat, termasuk operasi matematis, operasi logis, manipulasi bentuk dan ukuran, pengurutan, pemilihan, input/output, aljabar linier dasar, operasi dasar statistik, simulasi acak, dll.

#### B. Garis Besar Program

Secara umum, alur kerja aplikasi yang dibuat terdiri atas beberapa bagian sebagai berikut.

##### 1. Setup server

Pada saat aplikasi dijalankan, aplikasi melakukan setup server yang dibangun menggunakan Express yang akan dijalankan pada *runtime* Node.js untuk menerima *HTTP request* dari klien. Setup yang dimaksud termasuk menyiapkan file yang digunakan sebagai tampilan web dan fungsi untuk menangani *request* yang masuk.

Selanjutnya, server dihubungkan ke *port* tertentu (8080 pada aplikasi yang dibuat) dan kemudian server siap mendengarkan semua *request* yang masuk.

## 2. Klien mengirim request

Setelah server disiapkan, klien (pengguna) dapat mengakses situs web pada alamat dan *port* yang didengar oleh server. Apabila aplikasi dijalankan secara lokal, alamat situs web yang dimaksud adalah *localhost* sehingga untuk *port* 8080, pengguna dapat mengakses situs web pada alamat <http://localhost:8080>.

Setelah mengakses alamat tersebut, klien akan diberikan dan ditampilkan halaman utama dari situs web yang telah dibuat. Klien dapat berpindah ke halaman */compression.html* untuk melakukan kompresi gambar. Pada laman ini, klien diberikan opsi untuk mengunggah (*upload*) file gambar dan memilih mode kompresi gambar yang diinginkan. Mode yang dimaksud adalah ukuran partisi gambar (16x16, 32x32, atau 64x64) dan *rank* dari SVD matriks gambar yang diinginkan. Setelah klien mengunggah gambar dan memilih mode, klien dapat menekan tombol *SUBMIT* untuk mengirim *request* ke server untuk melakukan kompresi gambar sesuai dengan gambar yang diunggah dan mode yang dipilih.

## 3. Kompresi gambar

*Request* dari klien untuk kompresi gambar akan diterima oleh server. *Request* ini kemudian ditangani oleh fungsi yang telah disiapkan pada server untuk melakukan kompresi. Untuk *request* kompresi gambar, server memanggil fungsi tersebut dengan argumen file gambar dan mode yang sesuai dengan *request* dari klien. Fungsi ini kemudian melanjutkan input (argumen) yang diberikan oleh server ke dalam program Python.

Fungsi tersebut membuat proses baru yang mengeksekusi program Python untuk mengkompresi gambar dengan argumen (input) gambar dan mode yang diberikan. Selanjutnya, program Python menerima gambar, ukuran partisi dan *rank* (mode) yang diminta untuk dikompres. Dengan bantuan pustaka PIL, program mengkonversi gambar menjadi data dalam representasi matriks pixel yang disediakan oleh pustaka NumPy. Jika terdapat beberapa *channel* pada gambar (misal gambar RGB memiliki tiga *channel*, yaitu merah, hijau, dan biru), program memisahkan masing-masing *channel* dari gambar menjadi representasi matriks masing-masing. Jika gambar memiliki *channel alpha*, yaitu *channel* yang menyatakan data transparansi gambar, *channel* ini disimpan pada variabel tertentu untuk sementara dan tidak akan diproses.

Untuk setiap *channel* yang ada pada gambar (selain *alpha*), matriks *channel* dipartisi ke dalam beberapa bagian/potongan/*chunk* sesuai dengan mode yang diminta. Untuk setiap *chunk*, matriks *chunk*–submatriks dari matriks *channel*–didekomposisi dengan SVD menggunakan algoritma *power iteration*. Hasil dari SVD matriks *chunk* akan dipangkas dan diambil sebesar *rank* tertentu sesuai dengan mode. Hasil dekomposisi yang telah dipangkas kemudian direkonstruksi, sesuai dengan persamaan yang berlaku, menjadi matriks *chunk* yang telah dikompres. Selanjutnya, submatriks dari matriks *channel* yang berkorespondensi dengan *chunk* tersebut diganti dengan matriks *chunk* yang telah dikompres.

Setelah setiap *chunk* pada setiap *channel* dikompres, data pada matriks *channel* baru dikoreksi sehingga berada pada rentang nilai dan tipe data yang valid, yaitu *UINT8* atau



*Unsigned 8-bit Integer* dengan rentang nilai [0, 255]. Untuk setiap *channel*, dihitung letak perbedaan nilai matriks baru dengan matriks aslinya. Selanjutnya, dilakukan operasi union terhadap perbedaan dari setiap *channel* sehingga didapat banyaknya pixel yang berbeda antara kedua gambar.

Dengan selesainya rekonstruksi matriks *channel*, program dilanjutkan dengan menggabungkan setiap matriks *channel* menjadi sebuah gambar baru yang telah dikompresi. Penggabungan matriks *channel* tersebut termasuk dengan *channel alpha* yang dipertahankan sejak awal, jika ada. Selain itu, program juga menghitung lamanya waktu kompresi sejak program dieksekusi hingga gambar baru didapatkan. Setelah gambar dikompresi, program menyimpan gambar serta menuliskan banyaknya perbedaan pixel (dalam persentase) dan lama waktu kompresi pada dua buah file sementara.

#### **4. Server mengembalikan respons**

Hasil dari program kompresi gambar yang ditulis dalam bahasa Python adalah dua buah file, yakni file gambar hasil kompresi dan file yang berisi perbedaan pixel dan lama waktu kompresi. Aplikasi (dalam Node.js) membaca file gambar dengan *encoding base64* dan membaca file lainnya sebagai teks. Hasil dari pembacaan ini adalah tiga buah teks yang masing-masing merepresentasikan gambar hasil kompresi, perbedaan pixel, dan lama waktu kompresi. Selanjutnya, server membentuk sebuah objek javascript yang memuat ketiga data tersebut dan mengirimkannya sebagai respons kepada klien dalam bentuk *Javascript Object Notation* (JSON). Langkah akhir dari menangani *request* yang diterima adalah menutup respons kepada klien dan menghapus file sementara yang tersimpan pada server, yaitu file yang diunggah oleh klien dan dua file hasil kompresi yang ditulis oleh program Python.

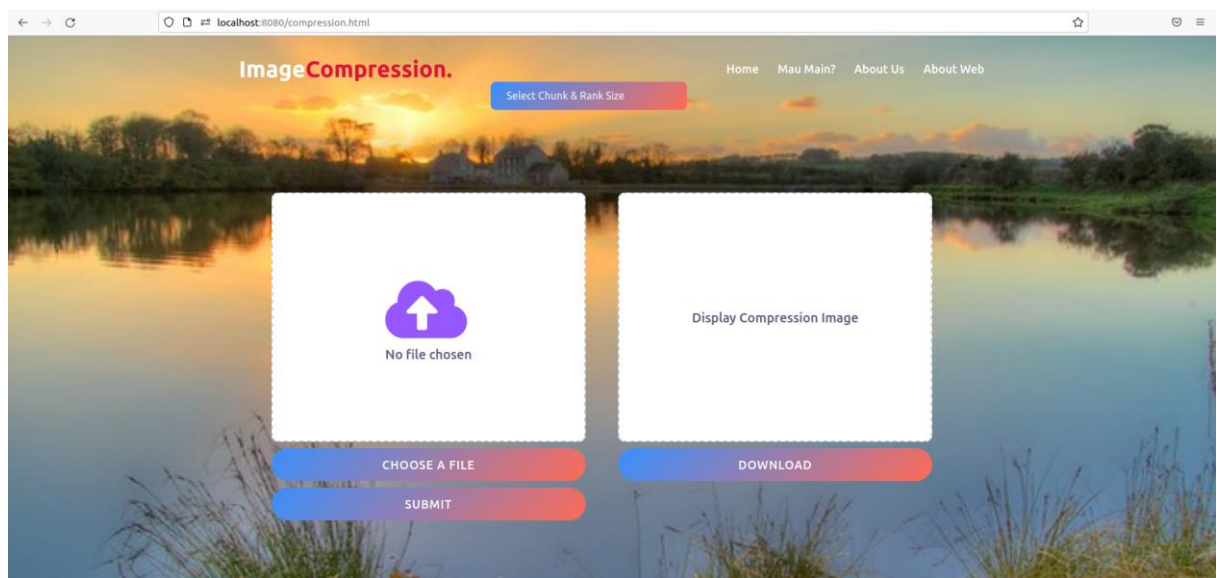
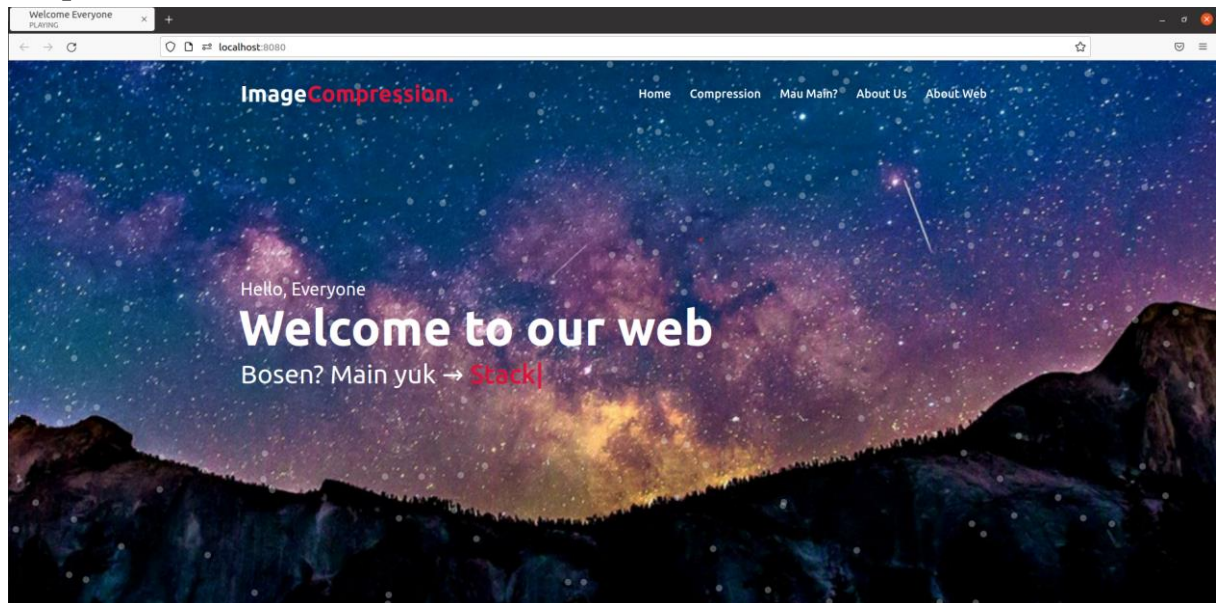
#### **5. Klien menerima hasil kompresi**

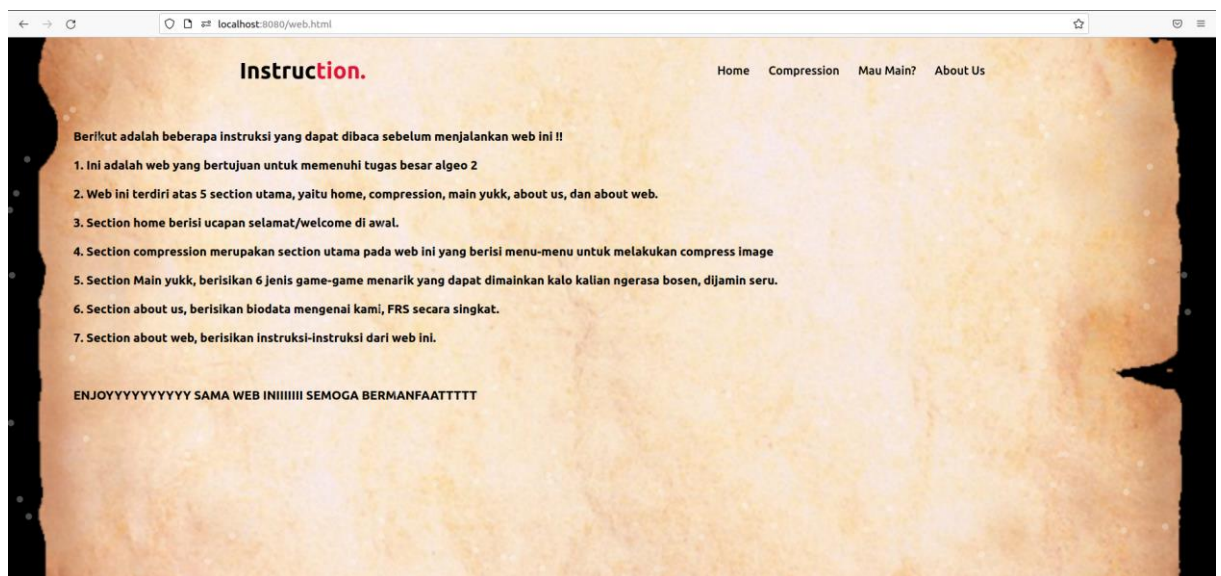
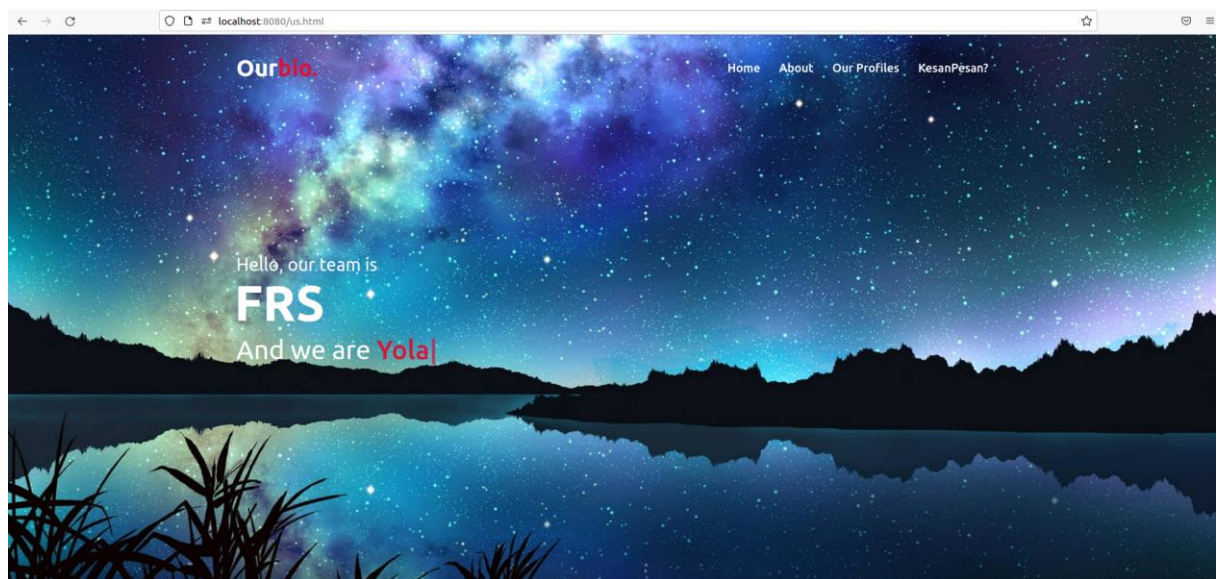
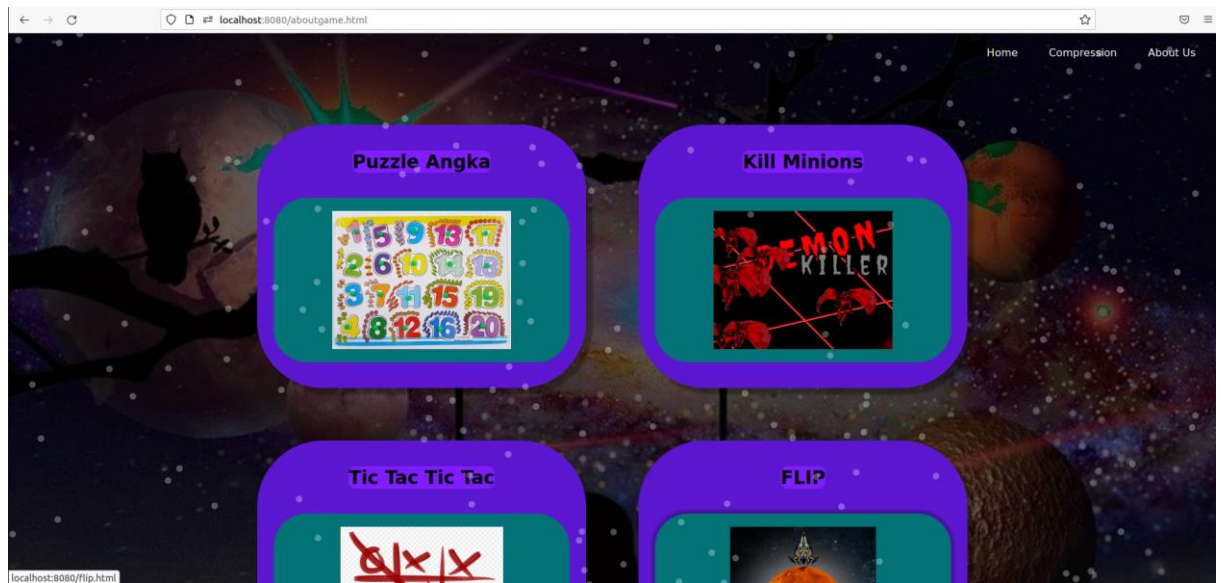
Respons berbentuk JSON diterima oleh klien sebagai balasan dari *request* sebelumnya. Pada mesin klien, data-data dari objek yang diterima kemudian ditampilkan di layar mesin klien pada tempat yang bersangkutan, yaitu data gambar ditampilkan sebagai gambar pada kotak untuk menampilkan hasil kompresi gambar dan data perbedaan pixel & lama waktu kompresi ditampilkan sebagai teks pada kotak di bawah gambar hasil kompresi. Dengan ditampilkannya gambar hasil kompresi dan informasi lainnya, tombol *DOWNLOAD* yang ada sejak klien pertama kali mengakses laman kompresi menjadi aktif dan dapat ditekan untuk mengunduh gambar hasil kompresi yang ditampilkan.

## BAB IV

### Eksperimen

#### 1. tampilan web secara umum



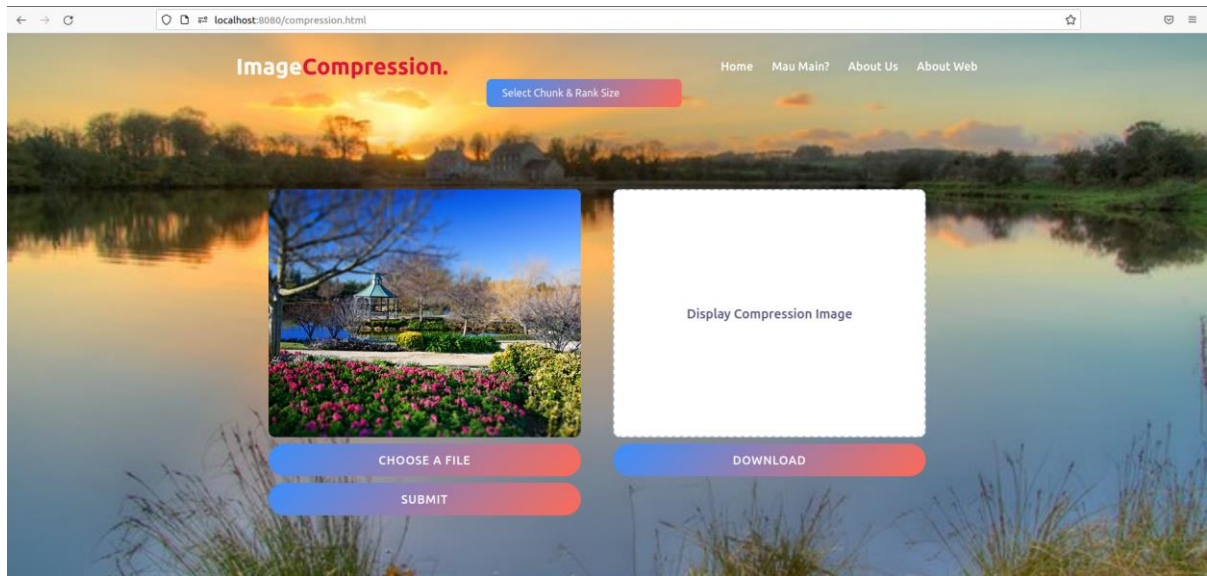


## 2. Pengujian program utama

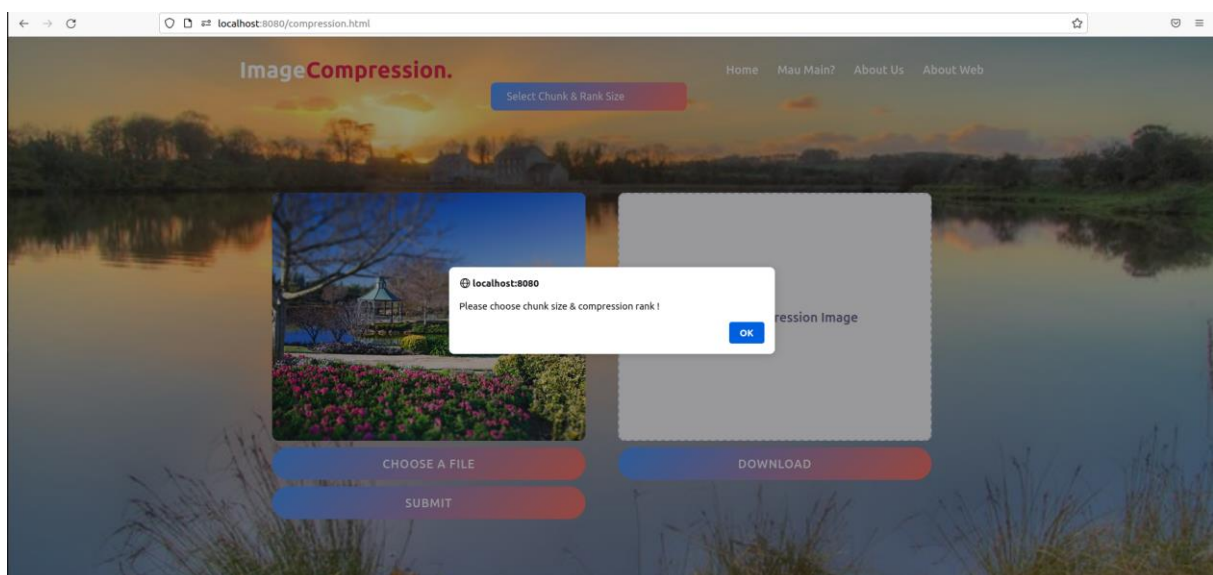


Proses Pengujian ini dilakukan pada page compression yang menjadi program utama pada website ini. Pengujian akan dilakukan untuk semua tingkat kompresi gambar yang tersedia pada web dan dilakukan juga pengujian semua button yang ada pada page ini. Alur program ini adalah:

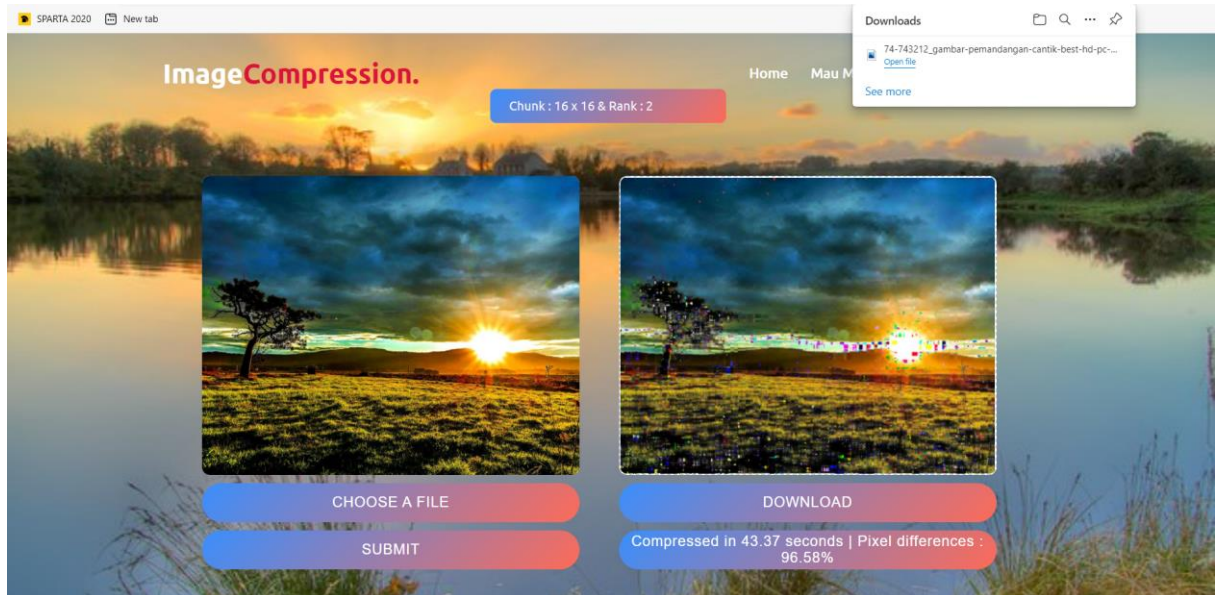
1. Memilih file gambar dengan mengklik button choose a file
2. file akan ditampilkan pada box yang tersedia di website seperti pada gambar di bawah



3. Memilih chunk dan rank size terlebih dahulu, jika melakukan submit terlebih dahulu maka akan tampil pesan sebagai berikut



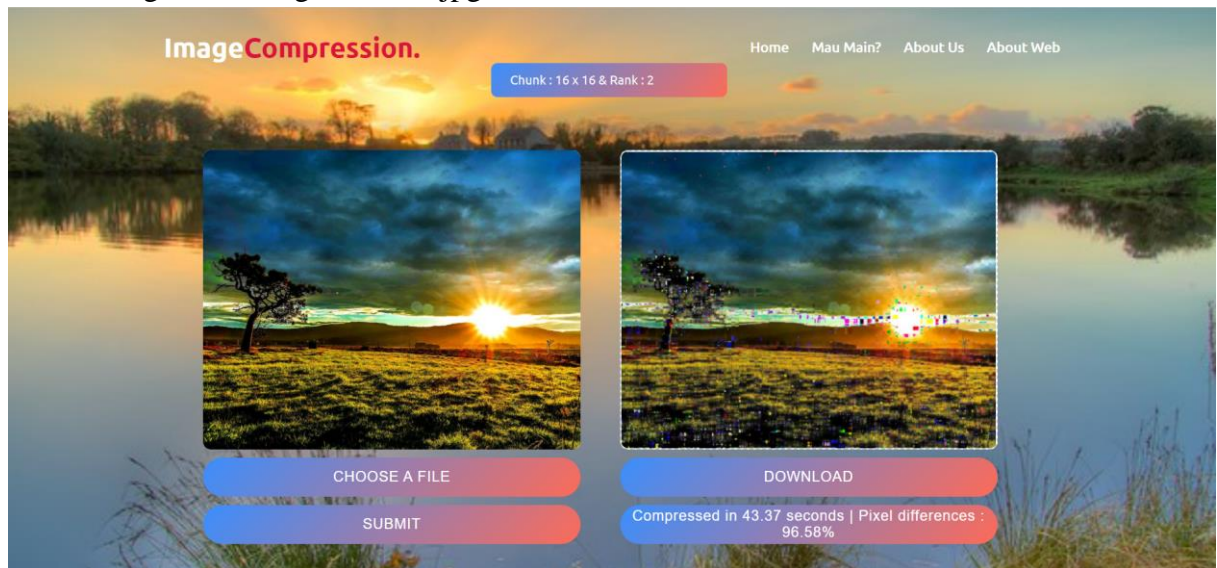
4. Kompresi image dilakukan dengan menganalisis masing-masing kasusnya
5. Download file



### 3. Pengujian Program berdasarkan rate compression

a. Chunk ukuran 16 x 16 dan rank : 2

1. gambar dengan format jpg

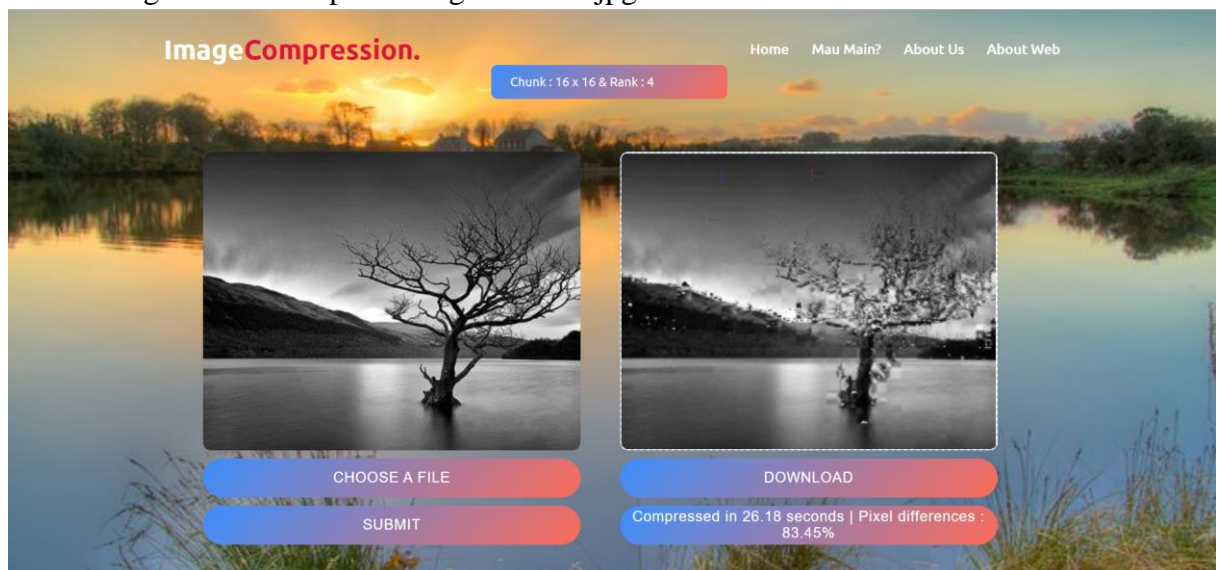


2. gambar dengan format png

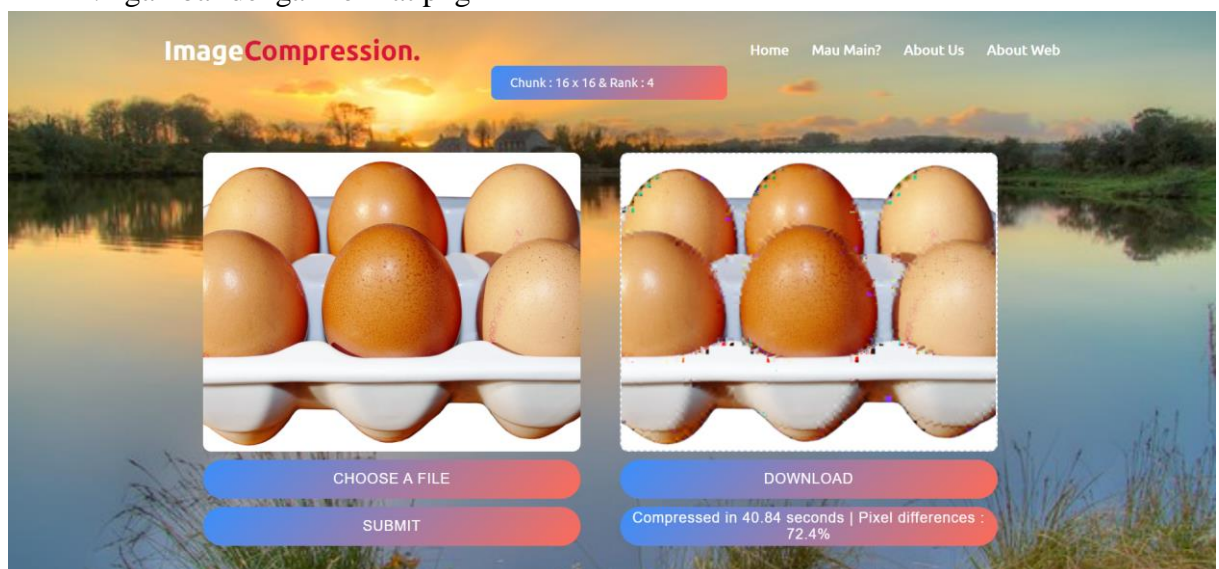




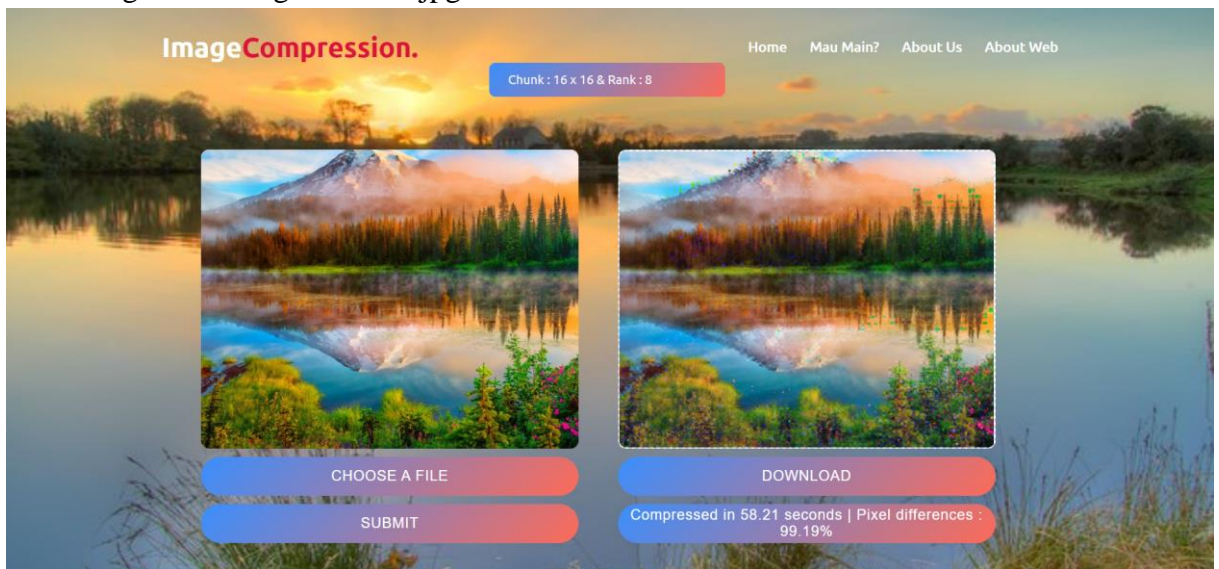
- b. Chunk ukuran 16 x 16 dan rank : 4
1. gambar hitam putih dengan format jpg



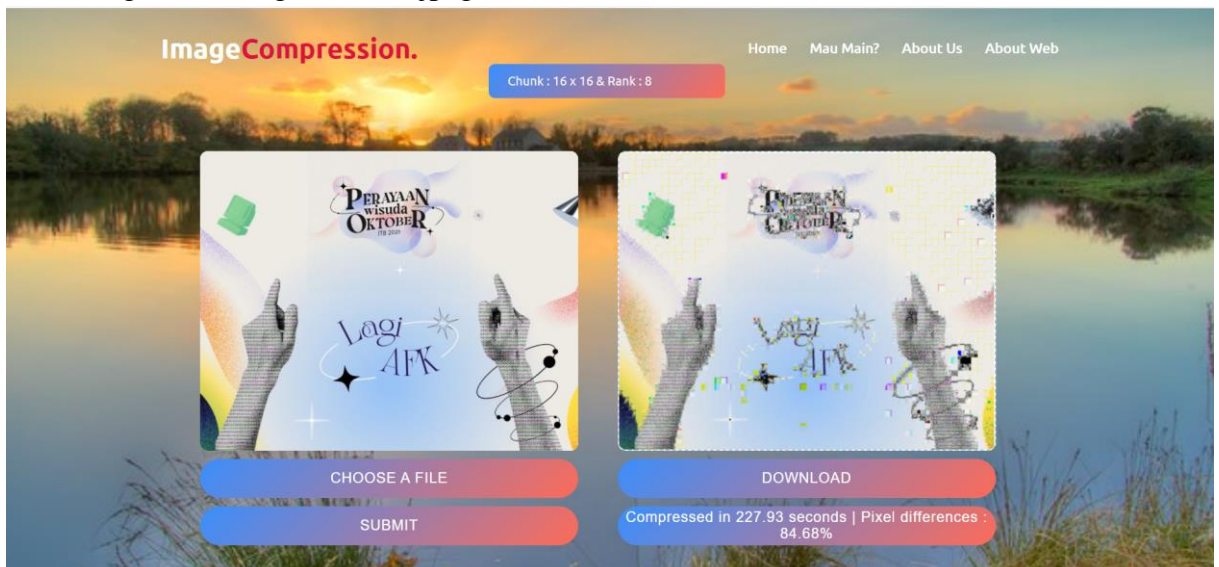
2. gambar dengan format png



- c. Chunk ukuran 16 x 16 dan rank : 8
1. gambar dengan format jpg

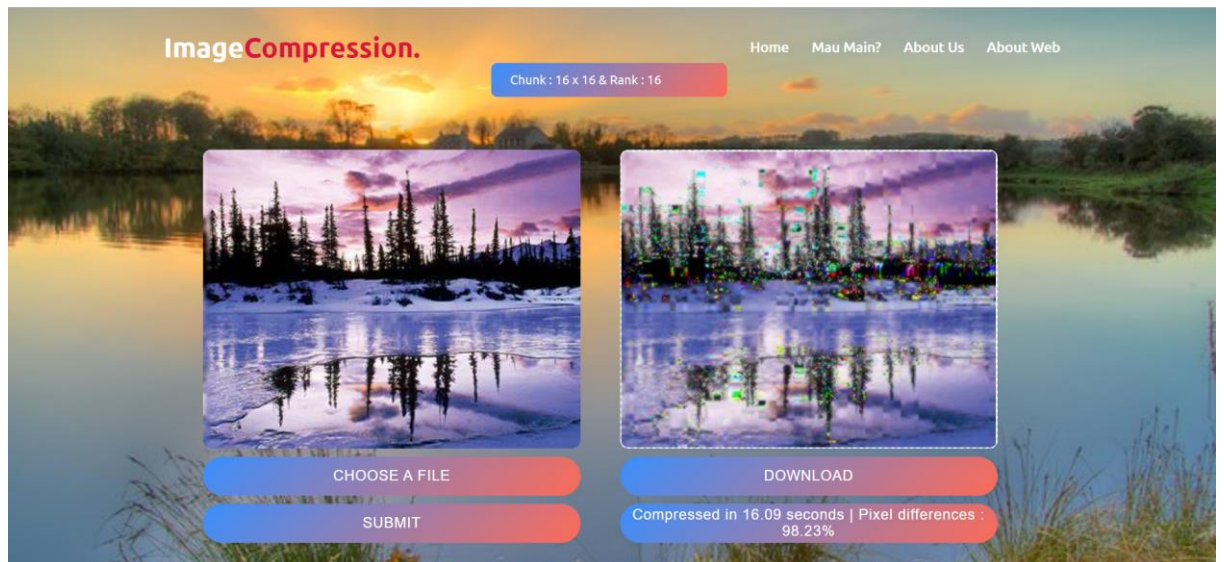


2. gambar dengan format jpeg

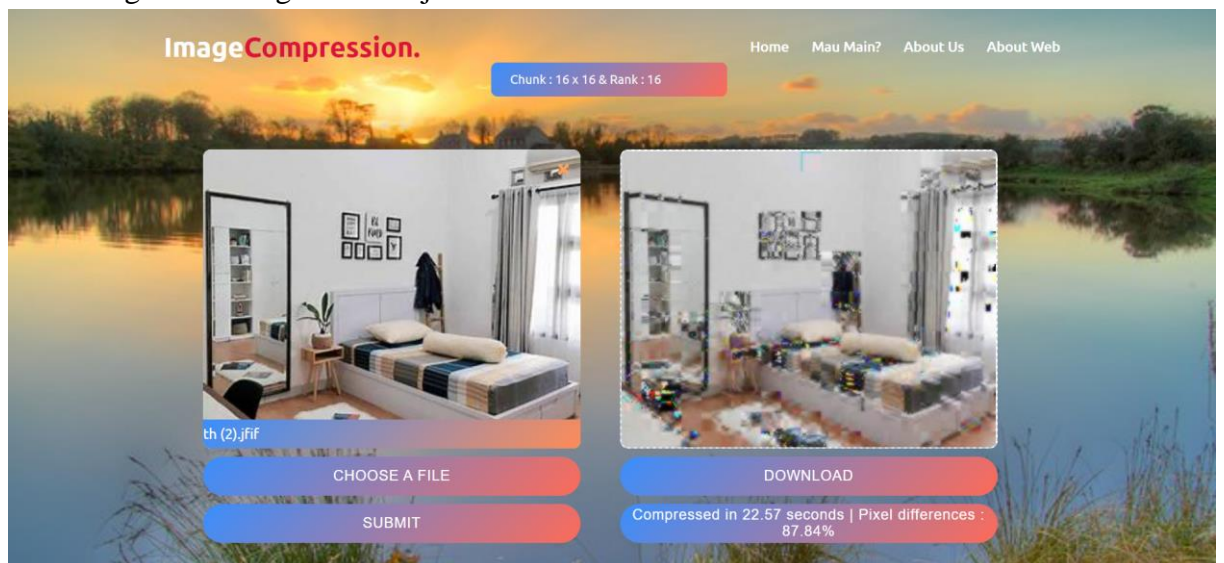


- d. Chunk ukuran 16 x 16 dan rank : 16
1. gambar dengan format jpg



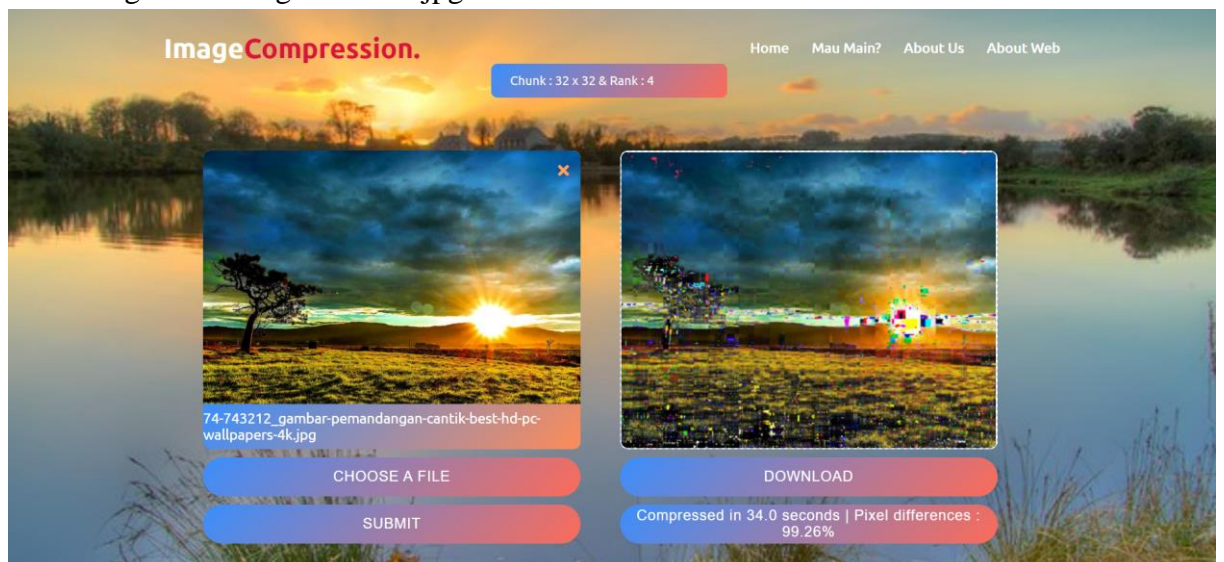


2. gambar dengan format jfif



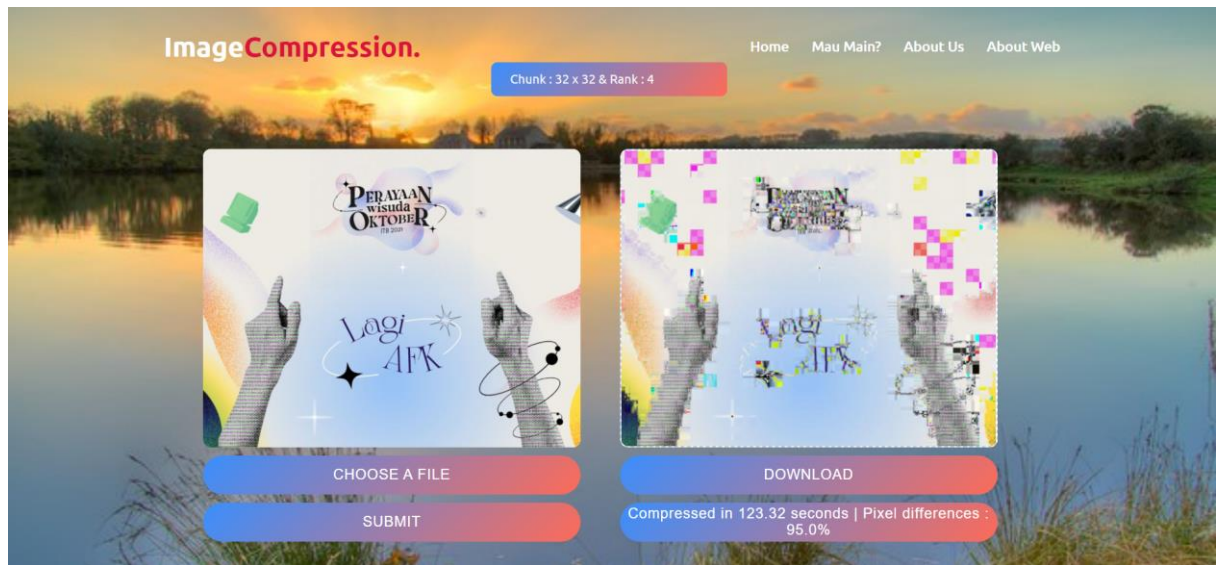
e. Chunk ukuran 32 x 32 dan rank : 4

1. gambar dengan format jpg

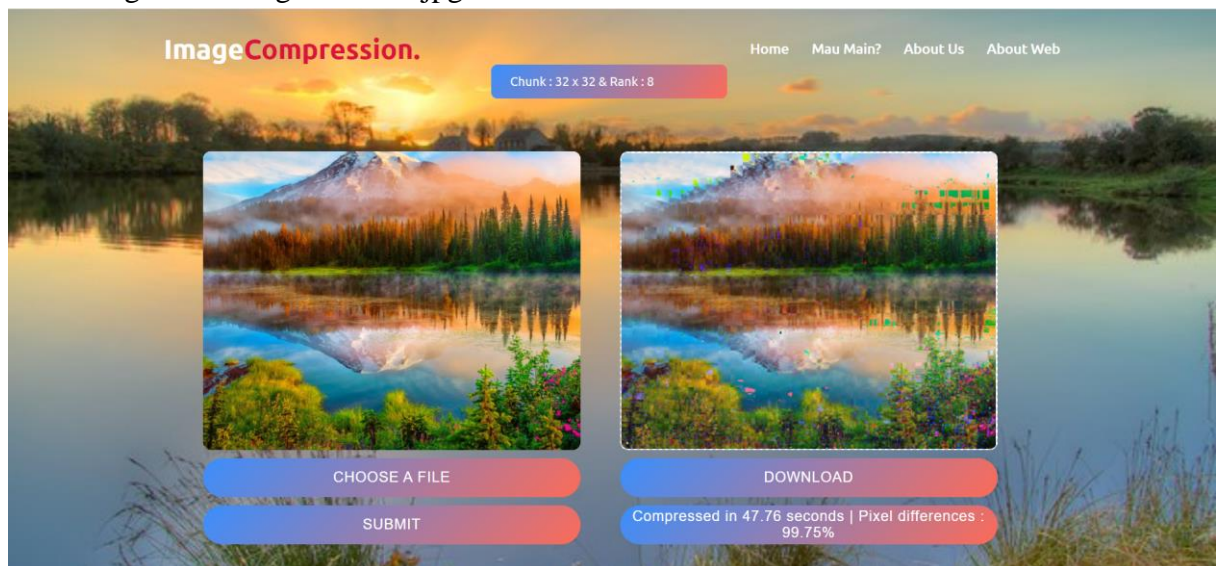


2. gambar dengan format jpeg

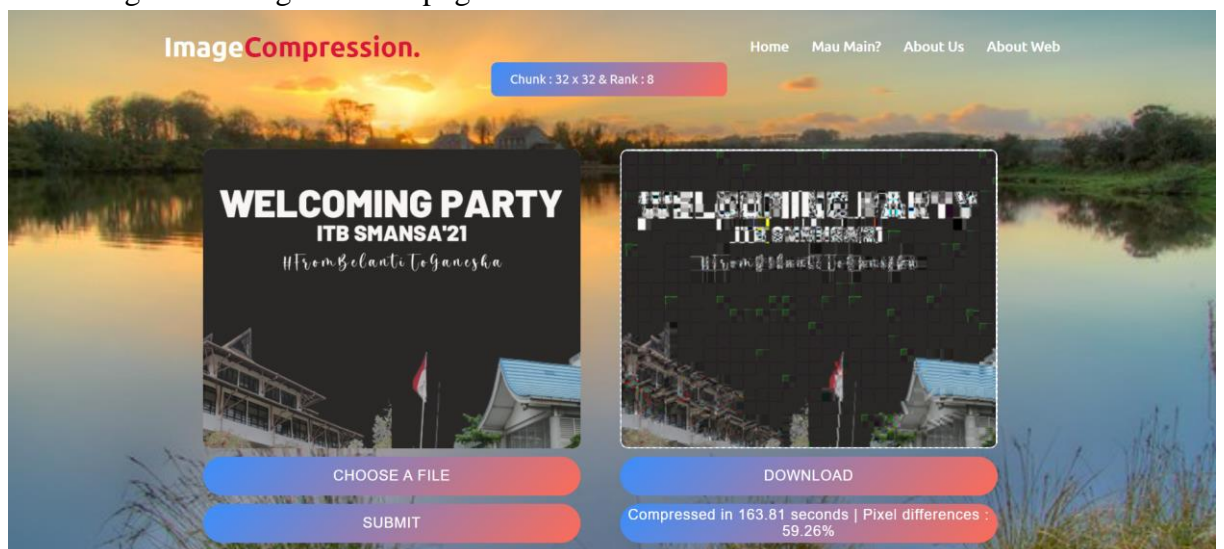




- f. Chunk ukuran 32 x 32 dan rank : 8
1. gambar dengan format jpg



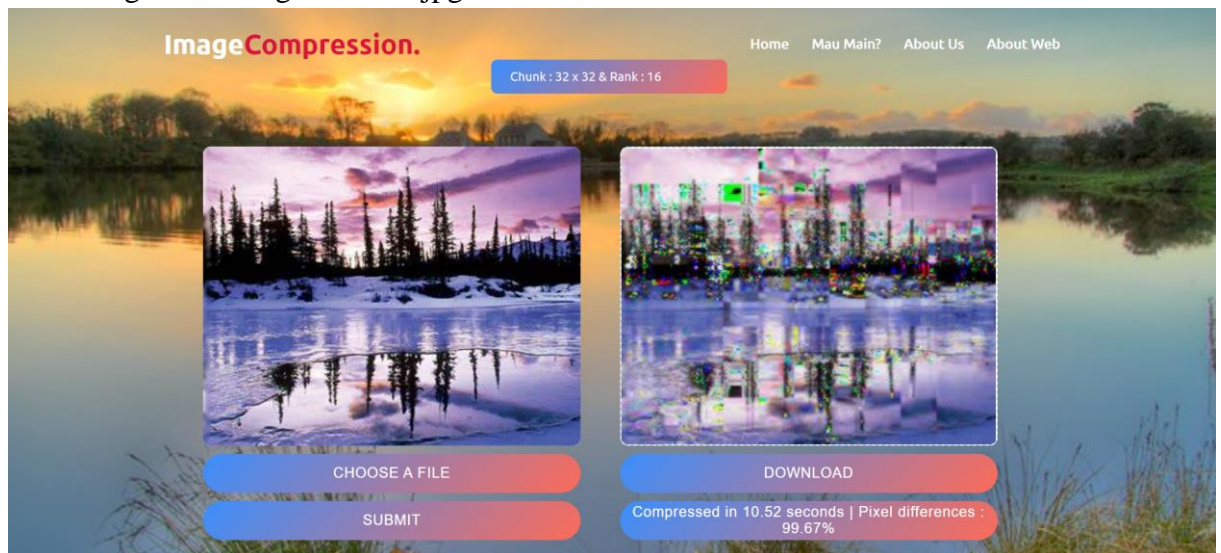
2. gambar dengan format png



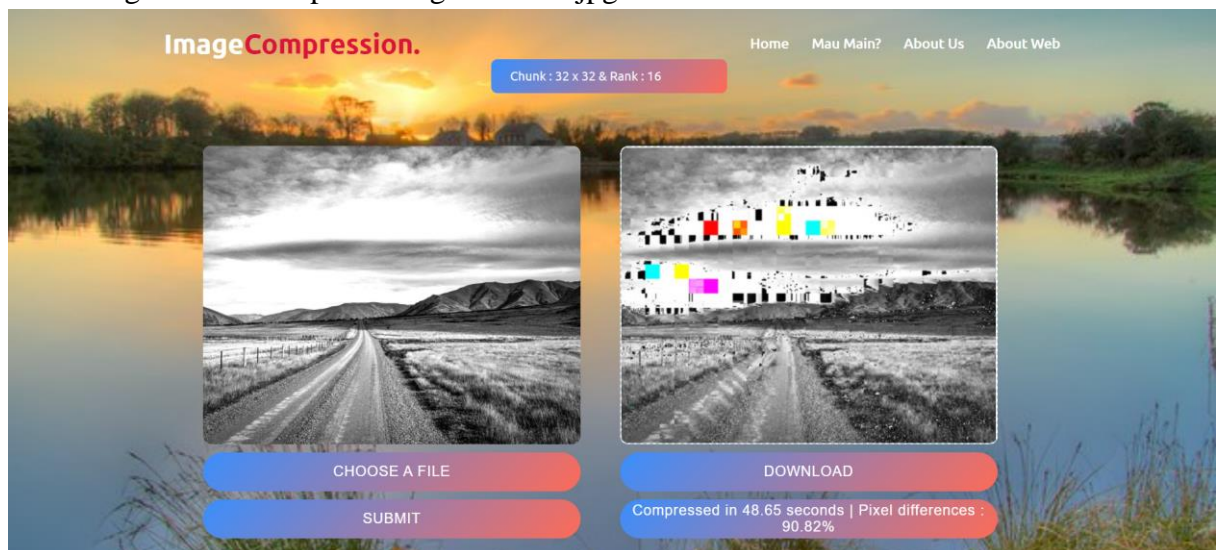


g. Chunk ukuran 32 x 32 dan rank : 16

1. gambar dengan format jpg

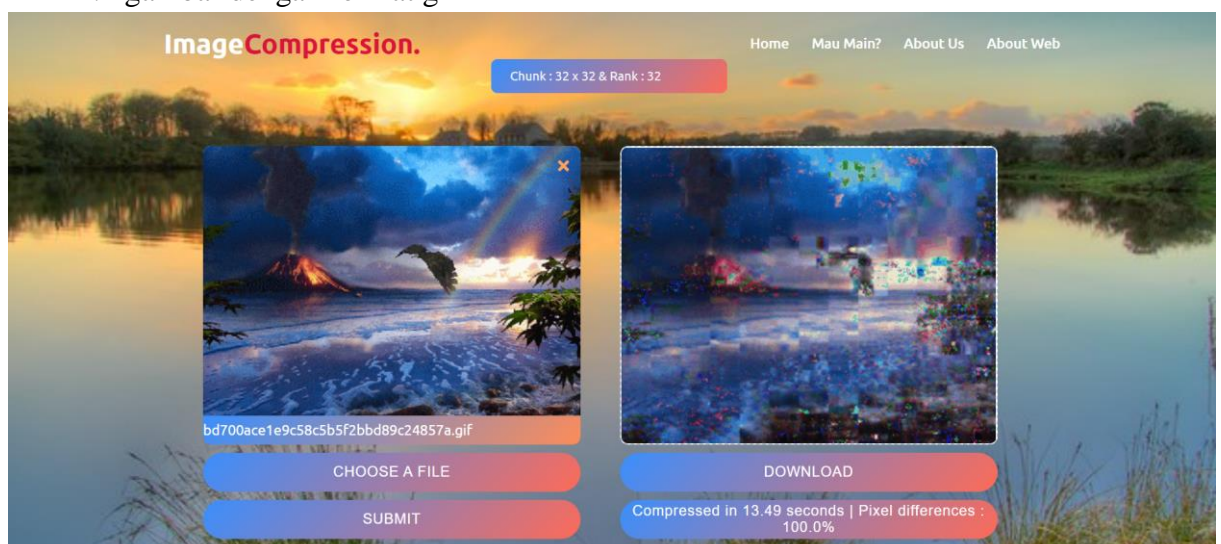


2. gambar hitam putih dengan format jpg



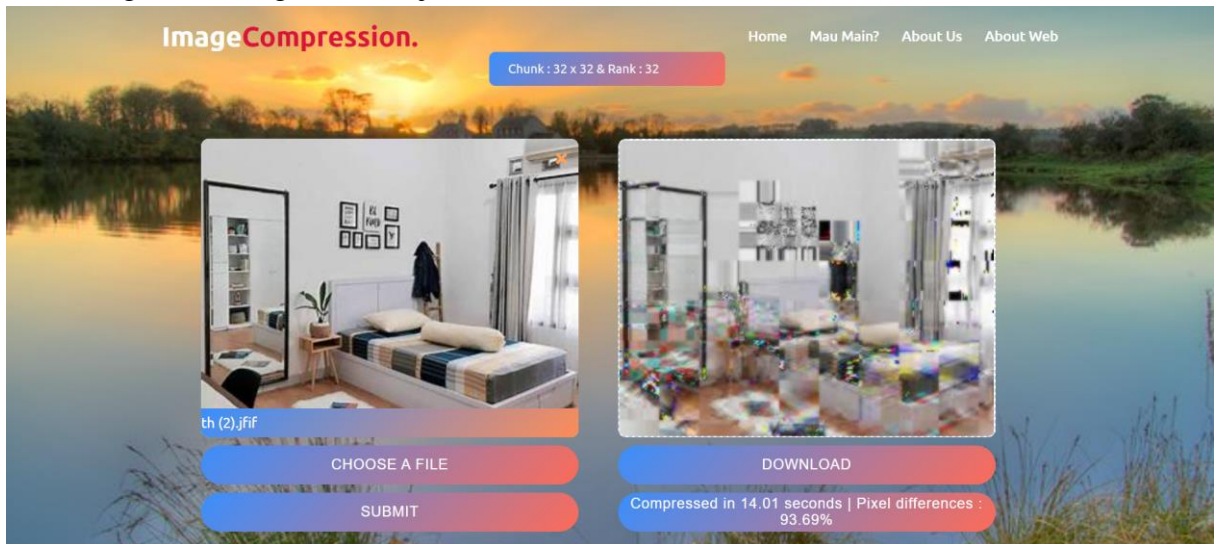
h. Chunk ukuran 32 x 32 dan rank : 32

1. gambar dengan format gif



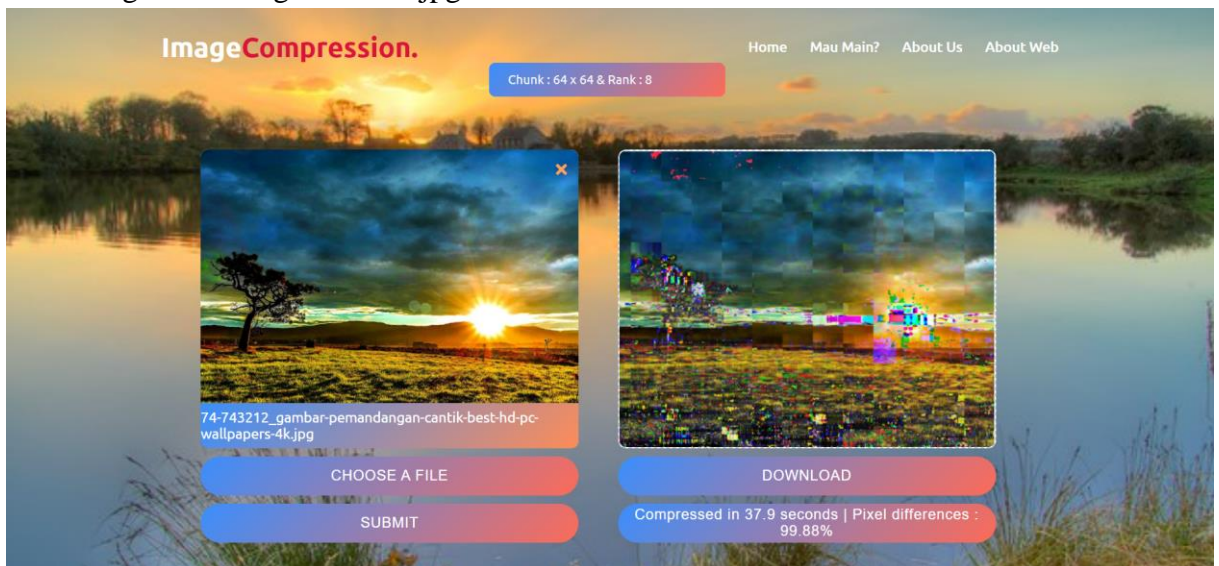


2. gambar dengan format jfif

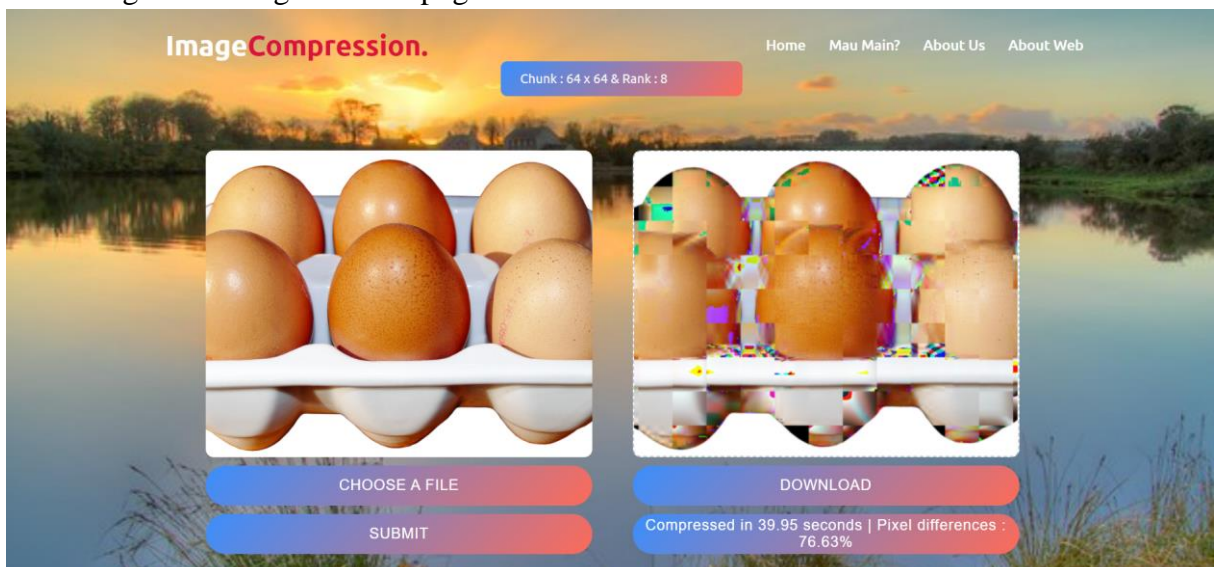


i. Chunk ukuran 64 x 64 dan rank : 8

1. gambar dengan format jpg

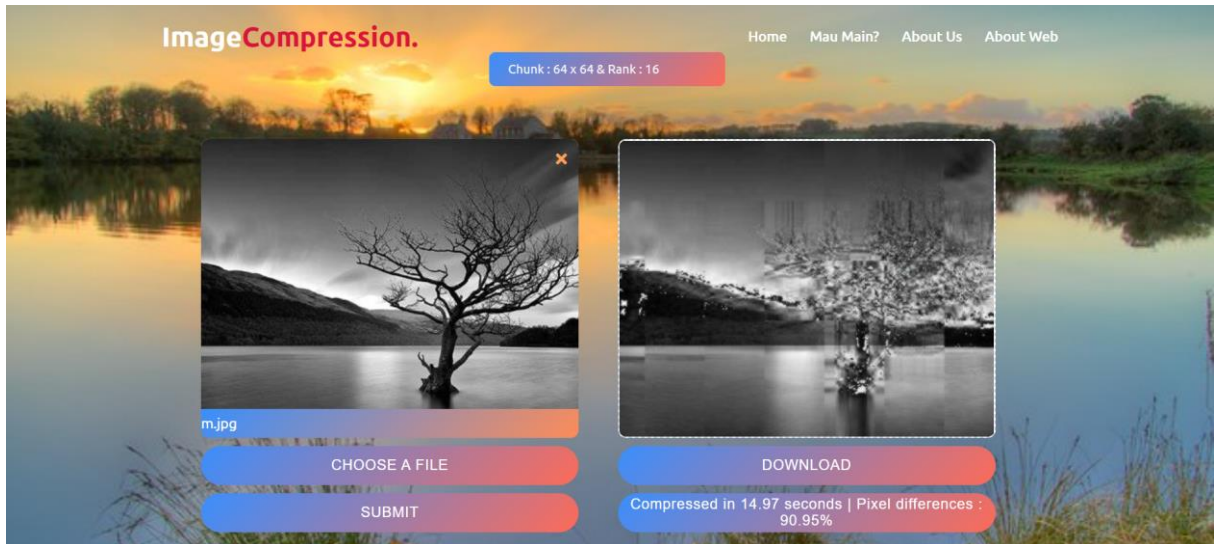


2. gambar dengan format png

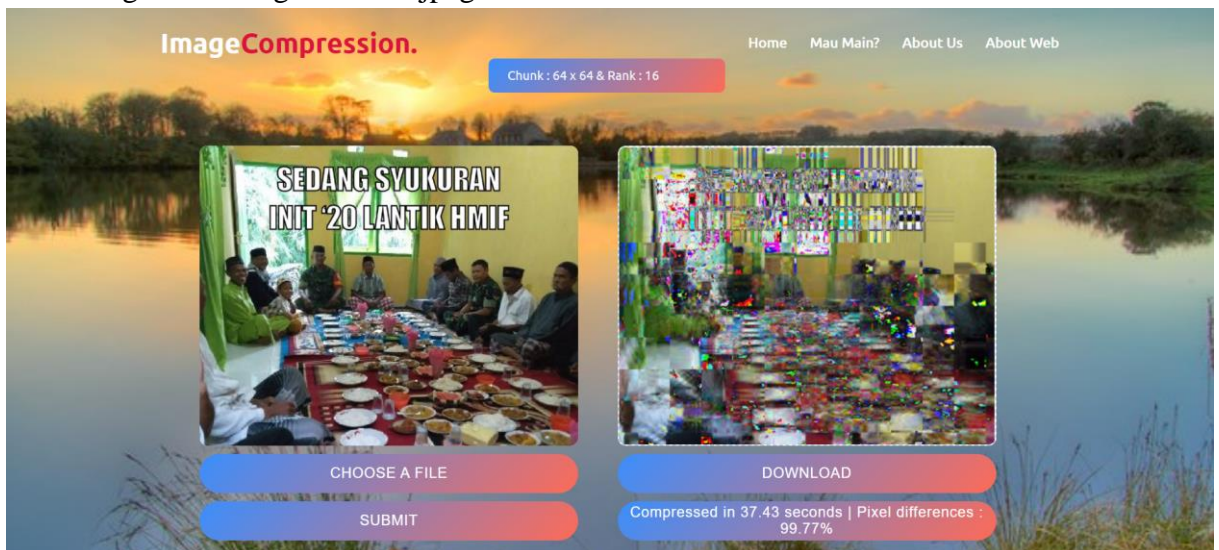




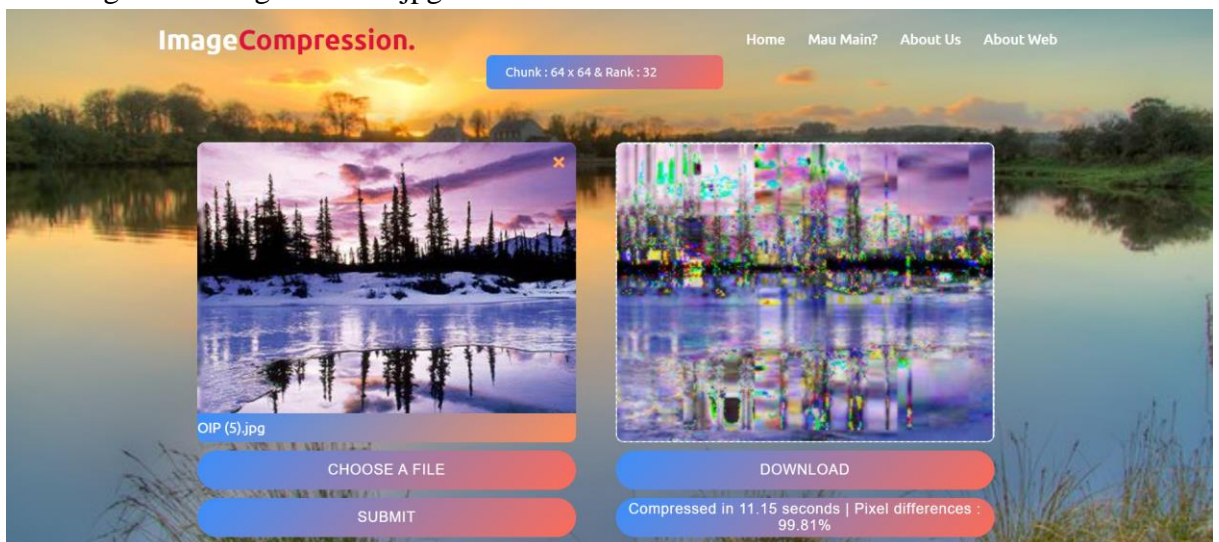
- j. Chunk ukuran 64 x 64 dan rank : 16
1. gambar hitam putih dengan format jpg



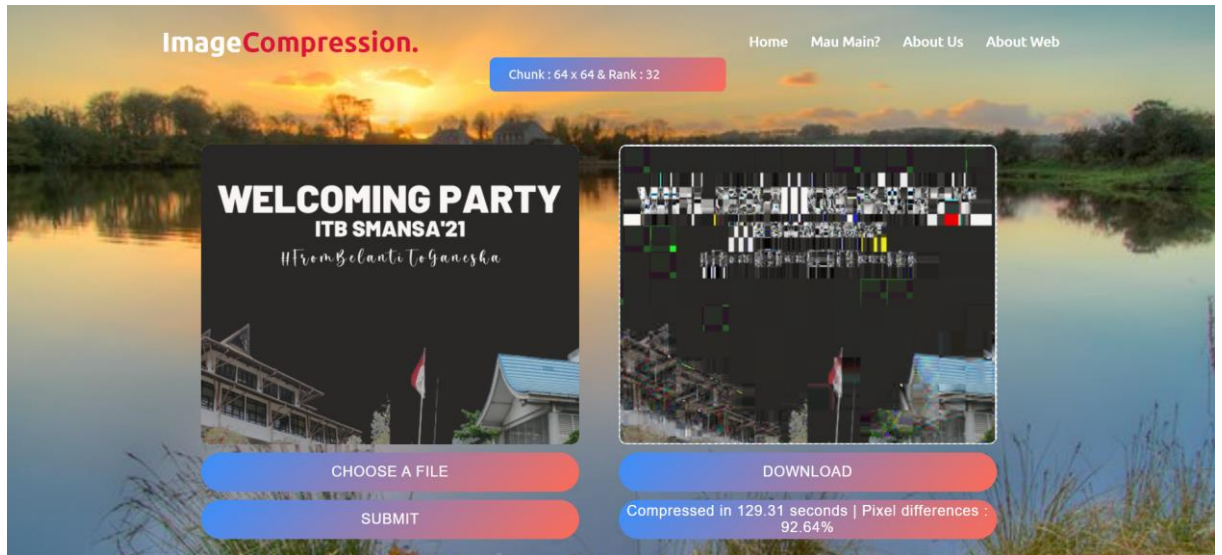
2. gambar dengan format jpeg



- k. Chunk ukuran 64 x 64 dan rank : 32
1. gambar dengan format jpg



2. gambar dengan format png



## **BAB V**

### **Kesimpulan**

#### **1. Kesimpulan**

Program kompresi gambar telah selesai dan berhasil diimplementasikan dengan menggunakan algoritma svd yang sesuai sebagaimana yang telah dipelajari pada mata kuliah Aljabar Linear dan Geometri. Hal-hal yang diimplementasikan pada program ini adalah sebagai berikut:

a. Aplikasi SVD pada matriks

Bagian ini berisikan implementasi dari svd (*singular value decomposition*) yang telah dipelajari di kelas sebelumnya yang melibatkan aplikasi dan penerapan nilai eigen, basis eigen, dan vektor eigen.

b. Nilai eigen, basis eigen, dan vektor eigen

Tiga hal ini menjadi komponen utama yang diperlukan untuk menentukan algoritma svd yang digunakan pada program.

c. Library numpy dan PIL pada python

Library numpy digunakan untuk melakukan pengolahan pada matriks sedangkan PIL digunakan untuk melakukan pengolahan citra pada gambar.

d. Backend dan Frontend

Backend pada web menggunakan nodejs dan Expressjs, sedangkan pada bagian frontend menggunakan html, css, dan javascript.

Semua implementasi yang digunakan pada program ini kemudian berhasil digunakan untuk menyelesaikan kompresi gambar dan fungsionalitas web yang sesuai dengan spesifikasi pada tugas besar.

#### **2. Saran**

Mengimplementasikan algoritma svd bukanlah hal yang mudah karena diperlukan operasi-operasi lainnya yang harus diimplementasikan terlebih dahulu. Ada banyak algoritma yang digunakan pada svd ini, tetapi penulis menyarankan untuk menggunakan algoritma power iteration yang akan membuat svd bekerja lebih cepat walaupun belum bisa menyamai kecepatan algoritma svd yang sudah tersedia pada library python. Akan tetapi, pada implementasinya algoritma jenis ini dapat digunakan dalam algoritma kompresi gambar sebagai langkah selanjutnya dalam pembuatan tugas ini. Selain itu, dalam pembuatan website dapat digunakan bahasa pemrograman yang mungkin menurut kalian lebih mudah untuk diterapkan, terutama dalam memilih backend dan frontend yang digunakan agar algoritma svd dan kompresi gambar yang telah diimplementasikan dapat terealisasi juga ke dalam website.

### 3. Refleksi

Belajar mengimplementasikan algoritma svd dalam suatu program merupakan hal yang menarik karena algoritma svd sebenarnya sudah ada pada library python atau bahasa pemrograman lain akan tetapi dengan menerapkannya, maka tingkat eksplorasi diri juga akan bertambah. Setelah mengerjakan tugas ini, timbul apresiasi dari kami untuk semua yang telah terlibat dalam pembagian materi svd ataupun tugas besar ini karena melalui tugas ini kami mempelajari banyak hal, salah satunya dalam pembuatan website. Selain itu, dalam mengerjakan tugas ini, kami menyadari banyak kegunaan dan manfaat dari algoritma svd dalam kehidupan sehari-hari yang dapat diterapkan. Hal yang paling penting adalah dalam pengerjaan tugas ini, kami merasa dapat memperbaiki kinerja kami dalam berkelompok mulai dari pembagian tugas ataupun *timeline* untuk mengejar *deadline*. Dengan *timeline* dan pembagian tugas yang jelas ini kami dapat berkontribusi bersama dengan baik dan mengumpulkan tugas dengan baik dan tepat waktu.

## DAFTAR REFERENSI

Slide kuliah IF2123 Aljabar Linear dan Geometri tahun ajaran 2021/2022.

“Image Compression using Singular Value Decomposition (SVD)”

[http://www.math.utah.edu/~goller/F15\\_M2270/BradyMathews\\_SVDImage.pdf](http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf)

“Understanding Singular Value Decomposition and its Application in Data Science” by Reza Bagheri

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

“The Singular Value Decomposition”

[https://math.mit.edu/~gs/linearalgebra/linearalgebra5\\_7-1.pdf](https://math.mit.edu/~gs/linearalgebra/linearalgebra5_7-1.pdf)

“Image Compression with Singular Value Decomposition”

<http://timbaumann.info/svd-image-compression-demo/>

“Developing Web Applications with Flask”

[https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/Python3\\_Flask.html](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>