Laporan Tugas Besar 2

Diajukan untuk memenuhi tugas mata kuliah Aljabar Linear dan Geometri (IF 2123)
pada Semester 3 Teknik Informatika tahun Akademik 2021-2022

oleh

I Gede Arya Raditya Parameswara (13520036)
Januar Budi Ghifari (13520132)
Rizky Ramadhana P. K. (13520151)

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.

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

BAB II

DASAR TEORI

2.1 Perkalian Matriks

I. Perkalian matriks dengan bilangan scalar

contoh :

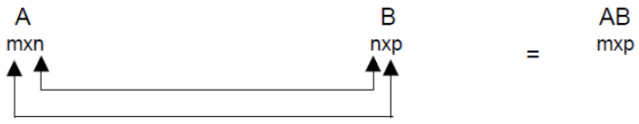
$$P = \begin{bmatrix} 3 & 8 \\ 5 & 1 \end{bmatrix} \text{ maka } 4P = 4 \begin{bmatrix} 3 & 8 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 12 & 32 \\ 20 & 4 \end{bmatrix}$$

Sifat-sifat perkalian matriks dengan scalar:

1)	$a(B+C)=aB+aC$	4)	$(a-b)C = aC-bC$
2)	$a(B-C) = aB-aC$	5)	$(ab)C = a(bC)$
3)	$(a+b)C = aC+bC$	6)	$(aB)^T = aB^T$

II. Perkalian dua matriks

Dua matriks AB dapat dikalikan apabila jumlah kolom matriks A sama dengan jumlah baris matriks B. jadi A_{mxn} B_{nxp} bisa didefinisikan, tapi B_{nxp}Amxn tidak dapat didefinisikan.



hasil kali dari matrik A dan B adalah matriks berordo mxp

contoh:

1.

$$B = \begin{bmatrix} 6 & 8 & 7 \end{bmatrix} \text{ dan } C = \begin{bmatrix} 4 \\ 7 \\ 2 \end{bmatrix}$$

$$B_{1x3}\, C_{3x1} = \begin{bmatrix} (6x4)+(8x7)+(7x2) \end{bmatrix} = \begin{bmatrix} 94 \end{bmatrix}$$

2.

$$A = \begin{bmatrix} 2 \\ 5 \\ 4 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 6 & 8 & 7 \end{bmatrix}$$

$$A_{3x1} B_{1x3} = \begin{bmatrix} 2x6 & 2x8 & 2x7 \\ 5x6 & 5x8 & 5x7 \\ 4x6 & 4x8 & 4x7 \end{bmatrix} = \begin{bmatrix} 12 & 16 & 14 \\ 30 & 40 & 35 \\ 24 & 32 & 28 \end{bmatrix}$$

3.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \end{bmatrix}$$

$$A_{2x2} B_{2x3} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \end{bmatrix}$$

$$AB = \begin{bmatrix} (1x1)+(2x0) & (1x0)+(2x2) & (1x1)+(2x0) \\ (3x1)+(4x0) & (3x0)+(4x2) & (3x1)+(4x0) \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 \\ 3 & 8 & 3 \end{bmatrix}$$

Sifat-sifat perkalian dua matriks:

1) A(BC) = (AB)C

2) A(B+C) = AB + AC

3) (B+C)A = BA + CA

4) A(B–C) = AB–AC

5) (B–C)A = BA–CA

6) a(BC) = (aB)C = B(aC)

7) AI = IA = A

2.2 Nilai Eigen dan Vektor Eigen

Jika A adalah sebuah matriks $n \times n$, maka sebuah vektor taknol $\boldsymbol{x}$ pada $\mathbb{R}^n$ disebut *vektor eigen* (vektor karakteristik) dari A jika $A\boldsymbol{x}$ adalah sebuah kelipatan skalar dari $\boldsymbol{x}$; jelasnya:

$A\boldsymbol{x} = \lambda\boldsymbol{x}$

untuk skalar sebarang λ. Skalar λ ini disebut *nilai eigen* (nilai karakteristik) dari A, dan x disebut sebagai vektor eigen (vektor karakteristik) dari A yang terkait dengan λ.

contoh:

Diberikan vektor $\boldsymbol{x} = \begin{bmatrix}1\\2\end{bmatrix}$ dan matriks $A = \begin{bmatrix}3 & 0\\8 & -1\end{bmatrix}$.

$$A\boldsymbol{x} = \begin{bmatrix}3 & 0\\8 & -1\end{bmatrix} \cdot \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}3\\6\end{bmatrix} = 3\begin{bmatrix}1\\2\end{bmatrix} = 3\boldsymbol{x}$$

Maka, vektor $\boldsymbol{x} = \begin{bmatrix}1\\2\end{bmatrix}$ disebut vektor eigen dari matriks $A = \begin{bmatrix}3 & 0\\8 & -1\end{bmatrix}$. yang terkait dengan nilai eigen $\lambda = 3$. Untuk memperoleh nilai eigen dari sebuah matriks A berukuran $n \times n$, persamaan $A\boldsymbol{x} = \lambda\boldsymbol{x}$ dapat dituliskan kembali menjadi

$$\begin{aligned} A\boldsymbol{x} &= \lambda I\boldsymbol{x} \\ A\boldsymbol{x} - \lambda I\boldsymbol{x} &= \boldsymbol{0} \\ (A - \lambda I)\boldsymbol{x} &= \boldsymbol{0} \end{aligned}$$

Agar λ dapat menjadi nilai eigen, harus terdapat satu solusi taknol dari persamaan ini. Persamaan ini memiliki solusi taknol jika dan hanya jika

$\det(A - \lambda I) = 0$

Persamaan di atas disebut sebagai persamaan karakteristik dari matriks A; skalar-skalar yang memenuhi persamaan ini adalah nilai-nilai eigen dari matriks A. Persamaan karakteristik di atas juga bisa dituliskan: $\det(\lambda I - A) = 0$

Apabila diperluas lagi, $\det(A - \lambda I)$ atau $\det(\lambda I - A)$ adalah sebuah polinomial p dalam variabel λ yang disebut sebagai polinomial karakteristik dari matriks A.

contoh:

Tentukan nilai-nilai eigen dari

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & -17 & 8 \end{bmatrix}$$

Pertama, cari dahulu matriks $A - \lambda I$.

$$A - \lambda I = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & -17 & 8 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & -17 & 8 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$
$$= \begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ 4 & -17 & 8 - \lambda \end{bmatrix}$$

Selanjutnya, cari $\det(A - \lambda I)$.

$$\begin{aligned} \det(A - \lambda I) &= (-\lambda)(-\lambda)(8-\lambda) + (1)(1)(4) + (0)(0)(-17) - (0)(-\lambda)(4) - (-\lambda)(1)(-17) - (1)(0)(8-\lambda) \\ &= (8\lambda^2 - \lambda^3) + 4 + 0 - 0 - 17\lambda - 0 \\ &= 8\lambda^2 - \lambda^3 + 4 - 17\lambda \\ &= -\lambda^3 + 8\lambda^2 - 17\lambda + 4 \end{aligned}$$

Dengan menggunakan persamaan karakteristik, diperoleh

$$\begin{aligned} \det(A - \lambda I) &= 0 \\ -\lambda^3 + 8\lambda^2 - 17\lambda + 4 &= 0 \\ \lambda^3 - 8\lambda^2 + 17\lambda - 4 &= 0 \\ (\lambda - 4)(\lambda^2 - 4\lambda + 1) &= 0 \end{aligned}$$

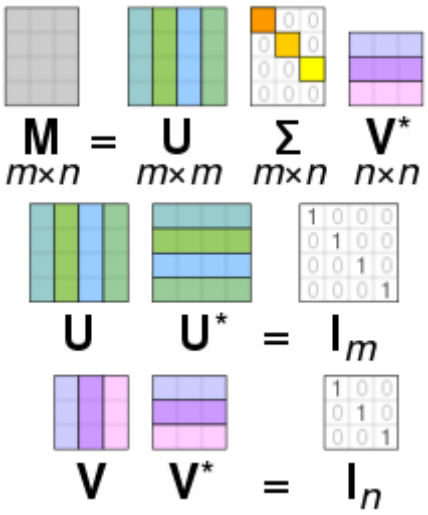
Dengan menggunakan rumus kuadratik, maka solusi untuk $(\lambda^2 - 4\lambda + 1) = 0$ adalah $2 + \sqrt{3}$ dan $2 - \sqrt{3}$, sehingga didapatlah nilai-nilai eigen dari matriks A, yaitu:

$$\lambda = 4, \qquad \lambda = 2 + \sqrt{3}, \qquad \lambda = 2 - \sqrt{3}$$

2.3 Matriks SVD

Singular Value Decomposition (SVD) adalah faktorisasi dari matriks real atau kompleks. Secara khusus, dekomposisi single value dari m x n matriks kompleks M adalah faktorisasi dari bentuk $U\Sigma V^T$, dimana U adalah m x m matriks, Σ adalah m x n demgam bilangan real non negatif pada diagonal , dan V adalah n x n matriks. Jika $\sigma_i = \Sigma_{ii}$ dari Σ M real, U dan V juga dapat dijamin matriks orthogonal real. Entri dari diagonal dikenal sebagai nilai-nilai singular dari M. Matriks U dan matriks V masing masing disebut dengan matriks singular kanan, dan matriks singular kiri.

Dengan nilai singular kiri merupakan normalisasi nilai vector eigen dari AA^T dan nilai singular kanan merupakan normalisasi nilai vector eigen dan $A^T A$.



Singular Value Decomposition (SVD) dapat diaplikasikan di matematika untuk menghitung pseudoinverse, aproksimasi matriks. Selain itu SVD juga berguna di bidang pengolahan gambar, khusunya kompresi gambar.

BAB III

IMPLEMENTASI PROGRAM

api.py

Nama	kembalian	Parameter	Deskripsi
compressImage	Hasil kompresi gambar berbentuk json dengan isi gambar dalam base64, waktu kompresi, rasio kompresi, ukuran akhir, dan status keberhasilan	-	Proses interaksi antara frontend dan backend dalam kompresi gambar

svd.py

Nama	kembalian	Parameter	Deskripsi
bagiTiapElemen	Vektor hasil pembagian	Dua buah vektor a dan b	Hasil pembagian *element-wise* dari dua buah vektor yang berukuran sama. Dengan penanganan khusus pada elemen 0, yaitu mengembalikan 0 pula
eigenDominan	Vektor eigen dan nilai eigen yang berkaitan dan yang paling besar	Matrix	Mencari vektor eigen dengan nilai eigen paling besar menggunakan metode power iteration
svd	Matrix dekomposisi u, s, dan v	Matrix	Melakukan dekomposisi SVD dan mengembalikan mat

			rix hasil dekomposisi u, s, dan v
kompresiSVD	Array	Array, banyaknya *singular values* yang akan digunakan	Mengaproksimasi array masukan menggunakan algoritma svd dengan *singular values* tertentu
kompresiGambarNLayer	Gambar dalam representasi array	Gambar dalam representasi array, banyaknya *channel*, banyaknya *singular values* yang akan digunakan	Melakukan kompresi menggunakan algoritma svd untuk masing-masing channel yang dimiliki
pertahankanTransparansi	Gambar dalam representasi array	Gambar dalam representasi array	Mempertahankan transparansi setelah melalui proses dekomposisi svd
finalisasi	Gambar dalam representasi array	Gambar dalam representa	Memastikan bahwa nilai pada array berupa bilangan bulat pada range

		si array, ekstensi file gambar	tertentu dan mengubahnya dalam bentuk base64

Website SVD Image Compressor ini menggunakan framework Flask untuk back-end nya dan React Js untuk front-endnya. Pada frontend kami menggunakan Chakra UI untuk mempermudah pembuatan design pada website kami sehingga website kami dapat terlihat user-friendly dan indah dipandang. Pengiriman dan pengambilan data kami menggunakan POST request dari FE ke BE menggunakan bantuan Axios POST. Ketika FE melakukan POST request, maka BE akan menerima requestnya lalu melakukan proses kompresi pada gambar yang dikirim. Pengiriman gambar dilakukan dengan mengubah gambar terlebih dahulu ke base64 lalu dikirim ke BE lalu diubah lagi oleh BE menjadi gambar dan dibentuk menjadi array of image menggunakan numpy. Setelah itu digunakanlah metodologi Singular Value Decomposition dari mata kuliah Aljabar Linear dan Geometri. Untuk menghandle gambar PNG, kami membuat fungsi untuk mempertahankan transparasi dengan mengecek layer transparan lalu mengubah semua layer menjadi transparan. Setelah gambar berhasil di proses, maka BE akan mengirimkan kembali gambar dalam bentuk base64 ke FE sebagai response dari request tersebut lalu ditampilkan pada layar.

Fungsi Yang Digunakan

```python
def compressImage():
    imageURL = request.get_json()
    imageBase64 = imageURL["data"]
    compressionRates = imageURL["rates"]
    mulai=perf_counter()
    metadata=imageBase64.split(",",maxsplit=1)[0]
    content=imageBase64.split(",",maxsplit=1)[1]
    tipe=re.split("[/;]",metadata)[1]
    sebelum_img=Image.open(io.BytesIO(base64.b64decode(content)))
    sebelum_array=array(sebelum_img)
    m=sebelum_array.shape[0]
    n=sebelum_array.shape[1]
    #Prasyarat ukuran gambar lebih dari 9 x 9 pixel
    if(min(m,n)>=50):
        if(compressionRates=='low'):
            k = 10
        elif(compressionRates=='med'):
            k = 15
        else:
            k = 20
    else:
        if(compressionRates=='low'):
            compressRate = 3
        elif(compressionRates=='med'):
            compressRate = 2
        else:
            compressRate = 1
        k = min(m,n)//(3*compressRate)
    print("Banyaknya singular values yang digunakan: ",k)
    print("Mode gambar: ",sebelum_img.mode)
    if(sebelum_img.mode=='RGB'):
        setelah_array=kompresiGambarNLayer(sebelum_array,3,k)
        im = finalisasi(setelah_array,tipe)
    elif(sebelum_img.mode=='L'):
        setelah_array=kompresiGambarNLayer(sebelum_array,1,k)
        im = finalisasi(setelah_array,tipe)
    elif(sebelum_img.mode=='RGBA'):
        a=sebelum_array[:,:,3]
        setelah_array=kompresiGambarNLayer(sebelum_array,3,k)
        setelah_array[:,:,3]=a
        setelah_array=pertahankanTransparansi(setelah_array)
        im = finalisasi(setelah_array,tipe)
    elif(sebelum_img.mode=='LA'):
        a=sebelum_array[:,:,1]
        setelah_array=kompresiGambarNLayer(sebelum_array,1,k)
        setelah_array[:,:,1]=a
        setelah_array=pertahankanTransparansi(setelah_array)
        im = finalisasi(setelah_array,tipe)
    elif(sebelum_img.mode=='P'):
        kodeTransparan=sebelum_img.info.get("transparency", None)
        if(kodeTransparan !=None):
            sebelum_img=sebelum_img.convert('RGBA')
            setelah_array=array(sebelum_img)
            a=setelah_array[:,:,3]
            setelah_array=kompresiGambarNLayer(setelah_array,3,k)
            setelah_array[:,:,3]=a
            setelah_array=pertahankanTransparansi(setelah_array)
        else:
            sebelum_img=sebelum_img.convert('RGB')
            sebelum_array=array(sebelum_img)
            setelah_array=kompresiGambarNLayer(sebelum_array,3,k)
        im = finalisasi(setelah_array,tipe)
```

```
    elif(sebelum_img.mode=='PA'):
        a=sebelum_array[:,:,1]
        setelah_array=kompresiGambarNLayer(sebelum_array,1,k)
        setelah_array[:,:,1]=a
        setelah_array=pertahankanTransparansi(setelah_array)
        im = finalisasi(setelah_array,tipe)
    else:
        try:
            setelah_array=kompresiGambarNLayer(sebelum_array,(1 if len(sebelum_array.shape)==
2 else sebelum_array.shape[2]),k)
            print(5)
            im = finalisasi(setelah_array,tipe)
        except:
            print('Proses gagal !')
            im=''

    selesai=perf_counter()
    lama=selesai-mulai
    lamaWaktu = "{:.2f}".format(lama)
    rasio=100*(k*(m+n)+k)/(m*n)
    rasioGambar="{:.2f}".format(rasio)
    print(f'Waktu pengerjaan : {lamaWaktu} detik')
    print(f'Rasio kompresi : {rasioGambar}%')
    im = str(im)
    berhasil = "yess"
    if im == '' :
        berhasil = "noo"
    sizeCompres = (len(im) * 3) / 4 - im.count('=', -2)
    im = str(metadata)+','+str(im)
    return jsonify(
        gambarKompres=str(im),
        waktuKompres=str(lamaWaktu),
        rasioKompres=str(rasioGambar),
        sizeKompres=str(sizeCompres),
        berhasilKompres=str(berhasil)
    )
```

Algoritma main atau compressImage

```
def bagiTiapElemen(a,b):
    return divide(a,b,out=zeros_like(a, dtype=float),where=b!=0)
```

Algoritma bagiTiapElemen

```
def eigenDominan(A, toleransi):
    m, n = A.shape
    k=min(m,n)
    v = ones(k) / sqrt(k)
    if m > n:
        A = matmul(transpose(A),A)
    elif m < n:
        A = matmul(A,transpose(A))

    nilaiEigen = bagiTiapElemen(matmul(A,v),v)[0]
    jumlahIterasi=0
    while( True):
        Av = matmul(A,v)
        vBaru = Av / norm(Av)
        nilaiEigenBaru = (bagiTiapElemen(matmul(A,vBaru),vBaru))[0]
        jumlahIterasi+=1
        if ((abs(nilaiEigen - nilaiEigenBaru) < toleransi) or (jumlahIterasi>=10000)):
            print("dilakukan ",jumlahIterasi," kali iterasi")
            break

        v = vBaru
        nilaiEigen = nilaiEigenBaru

    return nilaiEigenBaru, vBaru
```

Algoritma eigenDominan

```
def svd(A, k=None, toleransi=1):
    A = array(A, dtype=float)
    m, n = A.shape

    SVDKeN = []
    if k is None:
        k = min(m, n)

    for i in range(k):
        print("Proses pencarian singular value ke-",i+1)
        salinanA = A.copy()

        for u, nilaiSingular, v in SVDKeN[:i]:
            salinanA -= nilaiSingular * outer(u, v)

        if m > n:
            _, v = eigenDominan(salinanA, toleransi=toleransi)
            uAwal = matmul(A,v)
            sigma = norm(uAwal)
            u = uAwal / sigma
        else:
            _, u = eigenDominan(salinanA, toleransi=toleransi)
            vAwal = matmul(transpose(A),u)
            sigma = norm(vAwal)
            v = vAwal / sigma

        SVDKeN.append((u, sigma, v))

    us, S, vs = [array(x) for x in zip(*SVDKeN)]
    return transpose(us), S, vs
```

Algoritma svd

```
def kompresiSVD(arr,k):
  u,s,v=svd(arr,k)
  hasilKompresi=dot(u[:,:k],dot(diag(s[:k]),v[:k,:]))
  return hasilKompresi
```

Algoritma kompresiSVD

```
def kompresiGambarNLayer(arr,n,k):
  hasil=zeros(arr.shape)
  if(n==1):
    hasil=kompresiSVD(arr,k)
  else:
    for i in range(n):
      print("Mengolah layer ke-",i+1)
      hasil[:,:,i]=kompresiSVD(arr[:,:,i],k)
  return hasil
```

Algoritma kompresiGambarNLayer

```
def pertahankanTransparansi(arr):
  channel=arr.shape[2]
  layerTrpPixel = [0 for x in range(channel)]
  for i in range(arr.shape[0]):
    for j in range(arr.shape[1]):
      if arr[i,j,-1]==0:
        arr[i,j,:] = layerTrpPixel
  return arr
```

Algoritma pertahankanTransparansi

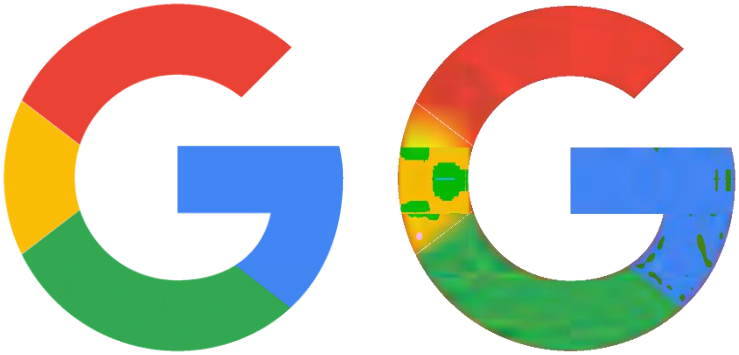
```
def finalisasi(arr,tipe):
  arr=(arr-arr.min())/(arr.max()-arr.min())*255
  im = Image.fromarray(arr.astype(uint8))
  buffered = io.BytesIO()
  im.save(buffered, format=tipe)
  img_str = base64.b64encode(buffered.getvalue())
  return img_str.decode('UTF-8')
```

Algoritma finalisasi

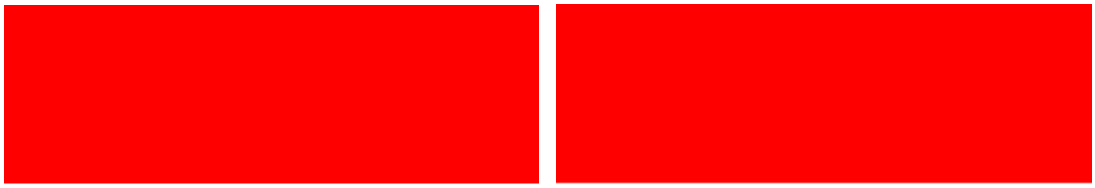
BAB IV

EKSPERIMEN

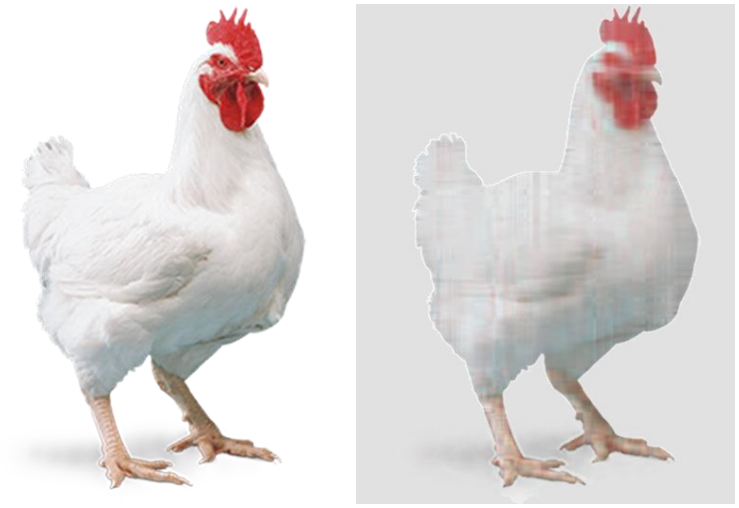
Pada bagian ini terdapat dokumentasi hasil testing yang telah kami lakukan terhadap test case yang terdapat pada spesifikasi tugas besar. Kami melampirkan beberapa uji kasus beberapa *mode image* yang didukung oleh Pillow. Pada pengujian ini, semua gambar dikompresi dalam waktu dibawah sepuluh detik. Program juga mampu mempertahankan transparansi baik di gambar dengan mode 'P' ataupun 'RGBA'. Program juga mampu mengompresi gambar *grayscale* (hanya satu *channel*) dengan baik. Hanya saja di beberapa kasus khusus yang sangat jarang, ukuran gambar setelah menjalani proses SVD justru meningkat.



Hasil kompresi gambar dengan mode 'P' dan latar transparan



Hasil kompresi gambar dengan mode 'P' tidak transparan



Hasil kompresi gambar dengan mode ‘RGBA’ (gambar di kanan sebetulnya juga memiliki latar transparan)

Hasil kompresi gambar dengan mode ‘RGB’

Hasil kompresi gambar dengan mode ‘L’

BAB V

SIMPULAN DAN SARAN

A. Kesimpulan

Hasil program kelompok kami berupa website yang dapat digunakan untuk melakukan kompresi gambar dengan tingkat kompresi sesuai kehendak pengguna melalui input berupa file gambar dan tingkat kompresi dan output berupa gambar awal, gambar yang sudah dikompresi, runtime algortima dan persentase hasil kompresi gambar. Gambar yang sudah dikompresi bisa diunduh oleh pengguna.

B. Saran

a. Proses pembuatan website sebaiknya diadakan seperti training terlebih dahulu agar masing-masing mahasiswa memiliki basic dan pengertian yang setara sebelum mengerjakan tubes ke 2 algeo ini.

REFERENSI

1. Anton, Howard, dan Chris Rorres. Elementary Linear Algebra: Applications Version.

2. Meyer, Carl D. Matrix Analysis and Applied Linear Algebra. Vol. 71. Siam, 2000.

3. Slide Bahan Kuliah IF2123 oleh Pak Rinaldi Munir

4. Dekomposisi Nilai Singular dan Aplikasinya, oleh Gregoria Ariyanti

5. "Understanding Singular Value Decomposition and its Application in Data Science" by Reza Bagheri,

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

6. "The Singular Value Decomposition"

https://math.mit.edu/~gs/linearalgebra/linearalgebra5_7-1.pdf