Kompresi Citra Menggunakan *Truncated Singular Value Decomposition* (TSVD), Sebuah Eksplorasi Numerik

Grace Reni Agustina, AD. Garnadi, Sri Nurdianti

Abstrak

Citra atau *image* memiliki representasi sebagai sebuah matriks untuk citra grayscale, sementara citra berwarna dipandang sebagai tensor 3 tingkat dari matriks berukuran sama. Dikaji dalam tulisan ini masalah kompresi citra menggunakan dekomposisi singular. Pada tulisan ini, dilakukan eksplorasi numerik kompresi citra sebagai masalah TSVD atau Dekomposisi Nilai Singular Terpuncung.

Kata kunci: SVD, TSVD

1. Pendahuluan

Singular Value Decomposition (SVD) merupakan teknik komputasi numerik yang melakukan faktorisasi terhadap sebuah matriks tak nol sehingga diperoleh tiga matriks tak nol. Salah satu matriks yang diperoleh dari proses SVD akan memuat nilai-nilai singular dari matriks asal. Istilah “nilai singular” menyatakan jarak antara sebuah matriks dan himpunan matriks-matriks singular. Nilai-nilai singular berguna untuk suatu matriks yang merupakan transformasi dari sebuah ruang vektor ke ruang vektor yang lain atau dimensi berbeda. Sistem persamaan aljabar yang *overdetermined* atau *undetermined* adalah contoh transformasi ruang vektor berbeda. (Maulidiya 2010)

Sebuah citra berwarna dapat direpresentasikan secara numerik sebagai sebuah tensor berukuran $m \times n \times 3$. Dengan metode SVD kita dapat membuat tensor baru dengan rank setiap lapisan yang lebih rendah dibanding tensor representasi citra yang asli. Setelah membuat tensor baru tersebut kita dapat merekonstruksi kembali citra baru dengan ukuran yang lebih kecil, namun tetap mirip dengan citra aslinya.

2. Landasan Teori

2.1 Matriks dan Tensor

Matriks adalah susunan segi empat siku-siku dari bilangan yang dibatasi dengan tanda kurung. Suatu matriks tersusun atas baris dan kolom, jika matriks tersusun atas m baris dan n kolom maka dikatakan matriks tersebut berukuran (berordo) $m \times n$. Penulisan matriks biasanya menggunakan huruf besar A, B, C dan seterusnya, sedangkan penulisan matriks beserta ukurannya (matriks dengan m baris dan n kolom) adalah $A_{m \times n}$, $B_{m \times n}$ dan seterusnya. (Sibaroni 2002)

Pada umumnya, a_{ij} akan menyatakan entri matriks A yang berada pada baris i dan kolom j . Jadi jika A adalah matriks $m \times n$, maka

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

kita sekali-sekali akan memendekkan ini menjadi $A = (a_{ij})$. Dengan cara serupa matriks B dituliskan sebagai (b_{ij}) , matriks C dituliskan (c_{ij}) , dan seterusnya. (Leon 1998)

Tensor dapat dipandang sebagai sebuah balok yang terdiri atas beberapa lapis segi empat siku-siku yaitu matriks. Misalkan T adalah sebuah tensor berukuran $m \times n \times l$ (tensor terdiri atas m baris, n kolom dan terdapat l lapisan). Tensor ini dapat ditulis secara singkat seperti matriks diatas yaitu $T = (t_{ijk})$.

2.2 SVD Matriks dan Tensor

Jika A adalah matriks $m \times n$, maka A mempunyai suatu dekomposisi nilai singular. Misalkan A adalah matriks $m \times n$ dengan dekomposisi nilai singular $U\Sigma V^T$.

- 1) Nilai-nilai singular $\sigma_1, \dots, \sigma_n$ dari matriks A adalah tunggal; akan tetapi, matriks-matriks U dan V tidak tunggal.
- 2) Karena V mendiagonalkan $A^T A$, maka vektor-vektor v_j menjadi vektor-vektor eigen $A^T A$.
- 3) Karena $AA^T = U\Sigma\Sigma^T U^T$, maka U mendiagonalkan AA^T dan vektor-vektor u_j menjadi vektor eigen AA^T .
- 4) Dengan membandingkan kolom ke- j dari setiap ruas persamaan

$$AV = U\Sigma$$

kita dapatkan

$$Av_j = \sigma_j u_j, j=1, \dots, n$$

Dengan cara yang serupa,

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

dan oleh karena itu

$$A^T u_j = \sigma_j v_j \text{ untuk } j=1, \dots, n$$

$$A^T u_j = \mathbf{0} \text{ untuk } j=n+1, \dots, m$$

Vektor-vektor v_j disebut *vektor singular kanan (right singular vector)* dari matriks A dan vektor-vektor u_j disebut *vektor singular kiri (left singular vector)* dari matriks A .

- 5) Jika A mempunyai rank r , maka

(i) $v_1, \dots, v_r$ membentuk basis ortonormal untuk $R(A^T)$.

- (ii) $\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ membentuk basis ortonormal untuk $N(A)$.
 - (iii) $\mathbf{u}_1, \dots, \mathbf{u}_r$ membentuk basis ortonormal untuk $R(A)$
 - (iv) $\mathbf{u}_{r+1}, \dots, \mathbf{u}_m$ membentuk basis ortonormal untuk $N(A^T)$.
- 6) Rank dari matriks Asama dengan jumlah nilai singular taknol-nya (di mana nilai-nilai singular tersebut dihitung sesuai dengan kelipatannya).(Leon 1998)

Contoh: Misalkan $A = \begin{pmatrix} 2 & -2 \\ 1 & 2 \end{pmatrix}$, akan di cari nilai dekomposisi singular dari A.

$$AA^T = \begin{pmatrix} 2 & -2 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ -2 & 2 \end{pmatrix} = \begin{pmatrix} 8 & -2 \\ -2 & 5 \end{pmatrix}$$

Nilai eigen dari AA^T adalah $\lambda_1 = 9$ dan $\lambda_2 = 4$, sehingga diperoleh nilai singular dari A yaitu $\sigma_1 = \sqrt{9} = 3$ dan $\sigma_2 = \sqrt{4} = 2$. Untuk $\lambda_1 = 9$ diperoleh vektor eigen $\mathbf{u}_1 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$ dan dari $\lambda_2 = 4$ diperoleh vektor eigen $\mathbf{u}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, dengan menormalisasi $\mathbf{u}_1$ dan $\mathbf{u}_2$ kita peroleh $U = \frac{1}{\sqrt{5}} \begin{pmatrix} -2 & 1 \\ 1 & 2 \end{pmatrix}$.

Selanjutnya nilai eigen dari $A^T A = \begin{pmatrix} 2 & 1 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ -2 & 8 \end{pmatrix}$ adalah $\lambda_1 = 9$ dan $\lambda_2 = 4$. Seperti sebelumnya kita mencari kembali vektor eigen yang bersesuaian dengan kedua nilai eigen tersebut, yaitu $\mathbf{v}_1 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ dan $\mathbf{v}_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ yang setelah dinormalisasi kita peroleh $V = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 & 2 \\ 2 & 1 \end{pmatrix}$. Sehingga kita peroleh dekomposisi singular dari A sebagai berikut:

$$A = U \Sigma V^T = \frac{1}{\sqrt{5}} \begin{pmatrix} -2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \frac{1}{\sqrt{5}} \begin{pmatrix} -1 & 2 \\ 2 & 1 \end{pmatrix}$$

SVD merupakan metode faktorisasi matriks, oleh karena itu kita harus mengubah tensor yang merepresentasikan citra berwarna menjadi sebuah matriks, proses ini biasa disebut *matricisation*. Ada berbagai macam bentuk *matricisation* yang dapat kita lakukan, namun pada percobaan ini tensor akan dibentuk menjadi matriks berukuran $m \times (n \times 3)$ dimana lapisan dari tensor akan disusun kesamping dimulai dari lapisan pertama, kedua dan ketiga membentuk sebuah matriks. Setelah matriks baru dibentuk barulah kita dapat menerapkan metode SVD untuk memperoleh matriks representasi citra dengan rank yang lebih rendah lalu mengembalikan matriks tersebut menjadi sebuah tensor. Selain dengan *matricisation*, kita dapat juga dapat menerapkan SVD ke tiap lapisan dari tensor satu persatu, namun penggunaan *matricisation* akan lebih efisien jika rank baru yang diinginkan untuk setiap lapisan tensor adalah sama.

Contoh: Misalkan sebuah tensor T dengan $T(:, :, 1) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, T(:, :, 2) = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}, T(:, :, 3) = \begin{pmatrix} 9 & 10 \\ 11 & 12 \end{pmatrix}$, dimana T berukuran $2 \times 2 \times 3$, *matricisation* tensor tersebut menjadi matriks berukuran $2 \times (2 \times 3)$ misalkan $T' = \begin{pmatrix} 1 & 2 & 5 & 6 & 9 & 10 \\ 3 & 4 & 7 & 8 & 11 & 12 \end{pmatrix}$.

2.3 Gambar sebagai matriks dan tensor

Suatu gambar video atau foto dapat didigitkan dengan memecahnya ke dalam jajaran sel segiempat (atau *pixels*) dan mengukur tingkat keabuan dari masing-masing sel. Informasi ini dapat disimpan dan ditransmisikan dengan menggunakan suatu matriks A berorde $n \times n$. Entri-entri dari A adalah bilangan-bilangan taknegatif yang bersesuaian dengan ukuran tingkat-tingkat keabuan tersebut. Karena tingkat keabuan dari suatu sel ternyata pada umumnya berdekatan dengan tingkat keabuan dari sel-sel di sebelahnya, maka kita dapat mereduksi jumlah penyimpanan yang diperlukan dari n^2 sampai suatu kelipatan dari n . Biasanya, matriks A tersebut akan mempunyai banyak nilai singular kecil. Akibatnya matriks A dapat didekati oleh suatu matriks dengan rank yang jauh lebih rendah. (Leon 1998)

Jika A mempunyai dekomposisi nilai singular $U\Sigma V^T$, maka A dapat diwakili oleh suatu ekspansi hasil kali luar

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_n u_n v_n^T$$

Matriks terdekat ber-rank k diperoleh dengan memotong penjumlahan ini setelah k suku yang pertama:

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_k u_k v_k^T$$

Penyimpanan total untuk A_k adalah $k(2n + 1)$. Kita dapat memilih k hingga jauh lebih kecil dari n dan tetap mempunyai citra digital untuk A_k yang sangat dekat dengan citra asalnya. Untuk pilihan-pilihan k yang sejenis, tempat penyimpanan yang dibutuhkan A_k akan kurang dari 20 persen dari jumlah tempat penyimpanan yang diperlukan untuk keseluruhan matriks A . (Leon 1998)

2.4 Norma Matriks

Misalkan $A \in \mathbb{R}^{m \times n}$, norma Frobenius dari matriks A didefinisikan sebagai berikut

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}$$

atau dapat dituliskan

$$\|A\|_F^2 = \text{tr}(A^T A),$$

dimana *trace* dari matriks $B \in \mathbb{R}^{n \times n}$ adalah jumlah dari elemen diaagonalnya,

$$\text{tr}(B) = \sum_{i=1}^n b_{ii}$$

2.5 Taksiran Matriks

Misalkan A adalah matriks *rank* rendah dengan *noise* : $A = A_0 + N$, dimana *noise* N lebih kecil jika dibandingkan dengan A . Jika kita mengetahui rank dari A_0 , atau dapat memperkirakannya dengan memeriksa nilai singular dari A , maka kita dapat menghapus *noise* dengan menaksir A dengan matriks yang memiliki *rank* yang benar atau yang kita prediksi. Misalkan *rank* sebenarnya yang kita duga adalah k , maka kita menaksir

$$A = \sum_{i=1}^n \sigma_i u_i v_i^T \approx \sum_{i=1}^k \sigma_i u_i v_i^T =: A_k$$

Misalkan $A \in \mathbb{R}^{m \times n}$ memiliki *rank* $r > k$. Norma Frobenius dari masalah taksiran matriks ini adalah

$$\min_{\text{rank}(Z) = k} \|A - Z\|_F$$

memiliki solusi

$$Z = A_k = U_k \Sigma_k V_k^T,$$

dimana $U_k = (u_1, \dots, u_k)$, $V_k = (v_1, \dots, v_k)$ dan $\Sigma_k = \text{diag}(\sigma_1, \dots, \sigma_k)$. Nilai minimumnya adalah

$$\|A - A_k\|_F = \left(\sum_{i=k+1}^p \sigma_i^2 \right)^{1/2}$$

dimana $p = \min(m, n)$

3. Eksperimen Numerik

I. Bagian 1

Pada bagian ini akan dilakukan kompresi citra menggunakan metode SVD dengan mengubah rank dengan rank baru yang sama untuk setiap lapisan dari tensor sebuah citra .

Gambar asli:

Gambar 1

Gambar 1.1

Gambar 1.3

Gambar 1.2

Gambar 1.4

Keterangan: gambar di atas secara berurutan dimulai dari Gambar 1.1 merupakan gambar yang dikonstruksi kembali dari pendekatan matriks Gambar 1 dengan rank 50, 100, 150 dan 200.

Berdasarkan beberapa percobaan kompresi citra menggunakan metode SVD dengan mengubah rank matriks diatas, dapat dilihat bahwa semakin tinggi rank matriks yang digunakan maka kualitas citra yang dihasilkan akan semakin mirip dengan citra aslinya. Rank dari matriks dari citra asli yang digunakan adalah 683, dan pada percobaan dengan rank 200 diperoleh citra yang secara kasat mata sudah cukup baik atau mirip dengan citra aslinya. Script untuk memperoleh setiap gambar di atas diberikan di Lampiran 1.

Error dari tiap pendekatan Gambar 1 untuk setiap k (E_k) didefinisikan sebagai berikut

$$E_k = A - A_k$$

Dimana A_k adalah pendekatan dari matriks A dengan rank k , kemudian dapat ditulis

$$e_k = \|E_k\|_F = \left(\sum_{i=k+1}^n \sigma_i^2 \right)^{\frac{1}{2}}$$

Berikut adalah grafik dari nilai singular Gambar 1 dan grafik $\log e_k$ terhadap $\|A_k\|$.

II. Bagian 2

Pada bagian ini akan dilakukan kompresi citra menggunakan metode SVD dengan mengubah rank dengan k (rank) yang berbeda untuk tiap lapisan dari tensor sebuah citra.

Citra asli:

Gambar 2

Gambar 2.1

Gambar 2.3

Gambar 2.2

Gambar 2.5

Gambar 2.4

Gambar 2.6

Keterangan:

- i. Gambar 2.1 : Red=100, Green=150, Blue=200
- ii. Gambar 2.2 : Red=150, Green=200, Blue=100
- iii. Gambar 2.3 : Red=200, Green=100, Blue=150
- iv. Gambar 2.4 : Red=50, Green=Blue=200
- v. Gambar 2.5 : Red=200, Green=50, Blue=200
- vi. Gambar 2.6 : Red=Green=200, Blue=50

Pada percobaan bagian ke-2 ini ingin diketahui bagaimana pengaruh penggunaan rank yang berbeda untuk tiap lapisan dari tensor citra (Red, Green, dan Blue) terhadap citra yang dihasilkan. Pada Gambar 2.1, Gambar 2.2 dan Gambar 2.3 rank tiap tensor di buat berbeda dengan selisih yang tidak terlalu jauh antara satu lapisan dengan lapisan lainnya, yaitu 100,150,200. Pada poin ini diperoleh citra yang cukup baik, dan tidak tampak perbedaan berarti antarsetiap citra walaupun menggunakan kombinasi rank yang berbeda. Apabila digunakan rank yang cukup jauh berbeda antara satu lapisan dengan lapisan lainnya seperti pada Gambar 2.4, Gambar 2.5, dan Gambar 2.6, citra yang dihasilkan juga cukup berbeda. Pada Gambar 2.4 dan Gambar 2.6 memang dihasilkan citra yang baik, namun tidak demikian pada gambar 2.5 dimana rank dari lapisan ke-2 hanya 50 dan rank lapisan lainnya 200, citra yang dihasilkan tidak sebaik Gambar 2.4 dan Gambar 2.6 . Hal ini kemungkinan disebabkan lapisan ke-2 adalah lapisan yang paling banyak menyimpan informasi dari citra aslinya, sehingga saat lapisan ini dikompresi dengan rank yang cukup kecil diperoleh citra yang kurang baik. Script untuk memperoleh setiap gambar diatas diberikan di Lampiran 2.

Error dari tiap pendekatan Gambar 2 untuk setiap k dan lapisan ke j (E_k^j) didefinisikan sebagai berikut

$$E_k^j = A^j - A_k^j, \text{ dimana } j = 1,2,3$$

Dimana A_k^j adalah pendekatan dari matriks A dengan rank k untuk lapisan ke j, kemudian dapat ditulis

$$e_k^j = \|E_k^j\|_F = \left(\sum_{i=k+1}^n (\sigma_i^j)^2 \right)^{\frac{1}{2}}$$

Berikut adalah grafik dari nilai singular Gambar 1 dan grafik $\log e_k^j$ terhadap $\|A_k^j\|$.

a) Lapisan pertama (*red*)

b) Lapisan kedua (*green*)

c) Lapisan ketiga (*blue*)

4. Daftar Pustaka

- Eldén L.1944. Matrix Methods in Data Mining and Pettern Recognition. Philadelphia (US): Society for Industrial and Applied Mathematics.
- Maulidiya D. 2010. Interpretasi Singular Value Decomposition (SVD) pada pengolahan citra digital. Bengkulu (ID) : Universitas Bengkulu.
- Leon SJ.2001. Aljabar Linear dan Aplikasinya. Bondan A, penerjemah. Jakarta (ID): Erlangga. Terjemahan dari: Linear Algebra with Applications.
- Sibaroni Y. 2002. Buku Ajar Aljabar Linear. STT Telkom.

LAMPIRAN 1.

1. Gambar 1.1 (Rank 50)

Input:

```
%Rank k=50
data = imread('IMG_4593.JPG');
img = im2double(data);
[m, n, d] = size(img);
img_matrix = reshape(img, [m, n*d]);
rank(img_matrix)
ans =
    683
[U, D, V] = svd(img_matrix);
k = 50;
img_matrix_rank_k = U(:, 1:k) * D(1:k, 1:k) * V(:, 1:k)';
img_rank_k = reshape(img_matrix_rank_k, [m, n, d]);
figure;
imwrite(img_rank_k, 'rank50g.jpg');
```

2. Gambar 1.2 (Rank 100)

Input:

```
%Rank k=100
data=imread('IMG_4593.JPG');
img=im2double(data);
[m,n,d]=size(img);
img_matrix = reshape(img, [m, n*d]);
[U, D, V] = svd(img_matrix);
k = 100;
img_matrix_rank_k = U(:, 1:k) * D(1:k, 1:k) * V(:, 1:k)';
img_rank_k = reshape(img_matrix_rank_k, [m, n, d]);
figure;
imwrite(img_rank_k, 'rank100j.jpg');
```

3. Gambar 1.3 (Rank 150)

Input:

```
%Rank k=150
data=imread('IMG_4593.JPG');
img=im2double(data);
[m, n, d] = size(img);
img_matrix = reshape(img, [m, n*d]);
[U, D, V] = svd(img_matrix);
k=150;
img_matrix_rank_k = U(:, 1:k) * D(1:k, 1:k) * V(:, 1:k)';
img_rank_k = reshape(img_matrix_rank_k, [m, n, d]);
figure;
imwrite(img_rank_k, 'rank150j.jpg');
```

4. Gambar 1.4 (Rank 200)

Input:

```
%Rank k=200
data=imread('IMG_4593.JPG');
img=im2double(data);
[m, n, d] = size(img);
img_matrix = reshape(img, [m, n*d]);
[U, D, V] = svd(img_matrix);
k=200;
img_matrix_rank_k = U(:, 1:k) * D(1:k, 1:k) * V(:, 1:k)';
img_rank_k = reshape(img_matrix_rank_k, [m, n, d]);
figure;
imwrite(img_rank_k, 'rank200j.jpg');
```


1. Gambar 2.1 (Red=100, Green=150, Blue=200)

Input:

```
data=imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);
green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=100 untuk lapisan pertama
redk=rU(:,1:100)*rD(1:100,1:100)*rV(:,1:100)';
%k=150 untuk lapisan kedua
greenk=gU(:,1:150)*gD(1:150,1:150)*gV(:,1:150)';
%k=200 untuk lapisan ketiga
bluek=bU(:,1:200)*bD(1:200,1:200)*bV(:,1:200)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb100_150_200.jpg')
```

2. Gambar 2.2 (Red=150, Green=200, Blue=100)

Input:

```
% 150-200-100
data = imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);
green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=150 untuk lapisan pertama
redk=rU(:,1:150)*rD(1:150,1:150)*rV(:,1:150)';
%k=200 untuk lapisan kedua
greenk=gU(:,1:200)*gD(1:200,1:200)*gV(:,1:200)';
%k=100 untuk lapisan ketiga
bluek=bU(:,1:100)*bD(1:100,1:100)*bV(:,1:100)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb150_200_100.jpg')
```

3. Gambar 2.3 (Red=200, Green=100, Blue=150)

Input:

```
%200-100-150
```

```

data = imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);
green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=200 untuk lapisan pertama
redk=rU(:,1:200)*rD(1:200,1:200)*rV(:,1:200)';
%k=100 untuk lapisan kedua
greenk=gU(:,1:100)*gD(1:100,1:100)*gV(:,1:100)';
%k=150 untuk lapisan ketiga
bluek=bU(:,1:150)*bD(1:150,1:150)*bV(:,1:150)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb200_100_150.jpg')

```

4. Gambar 2.4 (Red=50,Green=Blue=200)

Input:

```

%50-200-200
data = imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);
green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=50 untuk lapisan pertama
redk=rU(:,1:50)*rD(1:50,1:50)*rV(:,1:50)';
%k=200 untuk lapisan kedua
greenk=gU(:,1:200)*gD(1:200,1:200)*gV(:,1:200)';
%k=200 untuk lapisan ketiga
bluek=bU(:,1:200)*bD(1:200,1:200)*bV(:,1:200)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb50_200_200.jpg')

```

5. Gambar 2.5 (Red=200, Green=50, Blue=200)

Input:

```

%200-50-200
data = imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);

```

```

green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=200 untuk lapisan pertama
redk=rU(:,1:200)*rD(1:200,1:200)*rV(:,1:200)';
%k=50 untuk lapisan kedua
greenk=gU(:,1:50)*gD(1:50,1:50)*gV(:,1:50)';
%k=200 untuk lapisan ketiga
bluek=bU(:,1:200)*bD(1:200,1:200)*bV(:,1:200)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb200_50_200.jpg')

```

6. Gambar 2.6 (Red=Green=200, Blue=50)

Input:

```

%200-200-50
data = imread('rektoratipb.jpg');
img = im2double(data);
red=img(:,:,1);
green=img(:,:,2);
blue=img(:,:,3);
[rU,rD, rV] = svd(red);
[gU, gD, gV] = svd(green);
[bU, bD, bV] = svd(blue);
%k=200 untuk lapisan pertama
redk=rU(:,1:200)*rD(1:200,1:200)*rV(:,1:200)';
%k=200 untuk lapisan kedua
greenk=gU(:,1:200)*gD(1:200,1:200)*gV(:,1:200)';
%k=50 untuk lapisan ketiga
bluek=bU(:,1:50)*bD(1:50,1:50)*bV(:,1:50)';
ipbbaru(:,:,1)=redk;
ipbbaru(:,:,2)=greenk;
ipbbaru(:,:,3)=bluek;
imwrite(ipbbaru,'ipbb200_200_50.jpg')

```