Loughborough University

Cover Sheet for Assessed Coursework

Name: **Sayali Avinash Chavan**

ID Number: **B812081**

Module Title: **Advanced Programming**

Module Code: **18COP501**

Degree Programme: **Advanced Computer Science** Year/Part: **2018/T**

Coursework Title/Description: **JPEG Compression Simulation**

Staff Member responsible: **Prof Eran Edirisinghe**

Deadline for Submission: **19-10-2018**

If this coursework was part of a group activity, list the names of the other group members:

NA

DECLARATION:

I certify that the attached coursework is my own work and that anything taken from or based upon the work of others has its source clearly and explicitly cited.

Signature: 

Coursework received by: ..... (Signature)

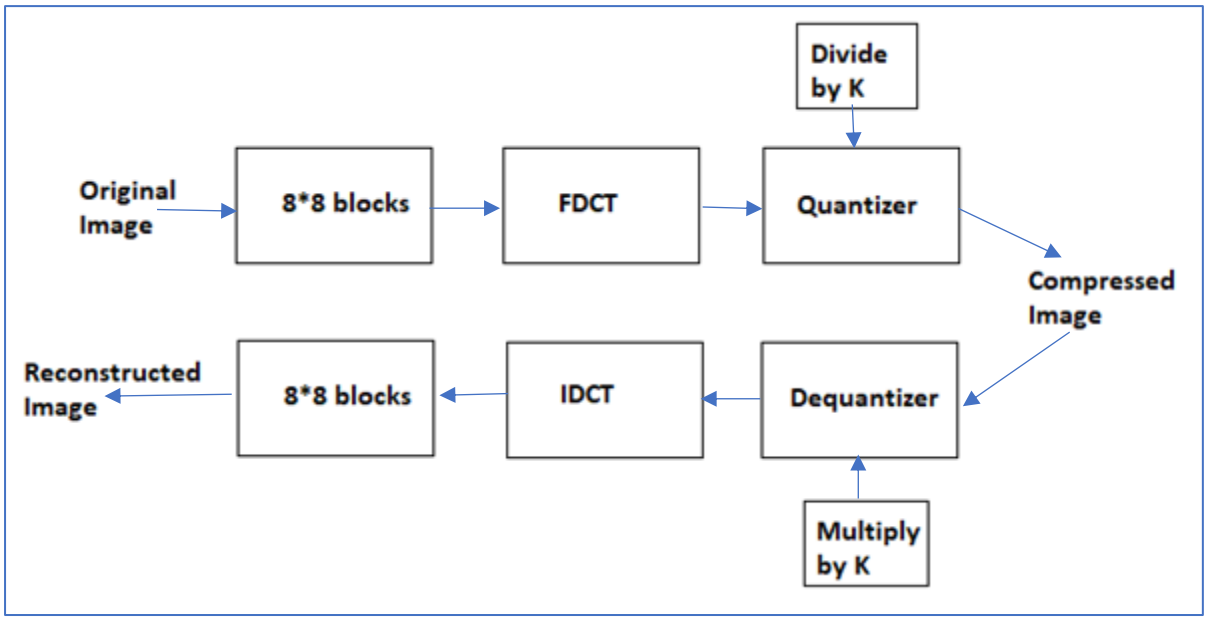
Date received:

Introduction

To view any digital image nowadays we have wide range of devices. The cost of this devices increases with the increase in quality of image. Hence to reduce this cost JPEG (Joint Photographic Experts Group) committee had come up with the general-purpose compression technique to delete some pixels in image which reduces the resolution without visibly affecting the image quality which depends on its compression/quality tradeoff factors.

This report gives overview of the JPEG standard with DCT-based Lossy Compression technique also called as “Baseline Method”, block diagram, requirements, specification, design and implementation and result with explanation.

Image compression process has two main parts upper part shows the Encoder processing steps and the lower part shows Decoding processing steps the Encoding and Decoding block is called as “CODEC”.



Block Diagram of JPEG–DCT Simulation.

- **Encoding:**

1. **8*8 blocks**
   Original Image is given as input, each component of image is divided into 8*8 blocks which are input to next block.

2. **FDCT- Forward Discrete Cosine Transform**
   DCT function works by differentiating the images into lower spatial frequencies which is the transformation into encoded format. The 8*8 block input is converted by FDCT into unique 64 discrete signal format called as “DCT Coefficients”.

3. **Quantizer**

   This block has two inputs one is DCT Coefficient and Divide by Factor K First DCT Coefficients are uniformly quantized in corresponds with the Quantizing Factor Q (hardcoded in Appendix Q.) which is the step size for input DCT Coefficient. It is further gets divided by factor K, which can be any value from 1 depending upon what is desired output quality of image. The result is then Rounded off, the purpose of this process is to get rid of unwanted or not visibly needed pixel removal.

- **Decoding:**

1. **Dequantizer**

   This block has two inputs one is Compressed image, and another is Multiply by Factor K. First the Compressed image is multiplied by uniform Q quantizer and K same values used in Encoding blocks and is then given to Dequantizer block which will decompress the image.

2. **IDCT – Inverse Discrete Cosine Transform**

   This will reverse the function done at DCT the encoded values get decoded back to reconstruct the given image.

3. **8*8 blocks**

   This block will collect the all 8*8 image samples from the IDCT block and form the Full image and We get the Reconstructed Image as a Result.

In Decoding blocks, it will reverse all part of encoding and we will get the Image with less quality, this is because of the **Rounding off** at the Quantizer's output.

Requirements Specification

- **CODEC Specifications:**

  - Block processing with 8x8 size blocks.
  - Forward and Inverse DCT
  - Use of quantization value Q.
  - **Brighten** the Image.
  - **Darken** the image.
  - Display **Discarded pixels** after compression.

- **GUI Specifications**:

  - Select the desired image grayscale or color
  - Select image via browsing
  - Value of K coefficient is input from user through GUI.
  - Display Original image and compare its visual quality against the compressed one.
  - Show the current value of K coefficient.
  - GUI should have Pop-up menu to select other three functionalities which is Brightness, Darkness and Discarded pixels
  - Display message when hovering over any button, so they would know its functionality.

Output: To put it in nutshell below are the functionalities user can view-

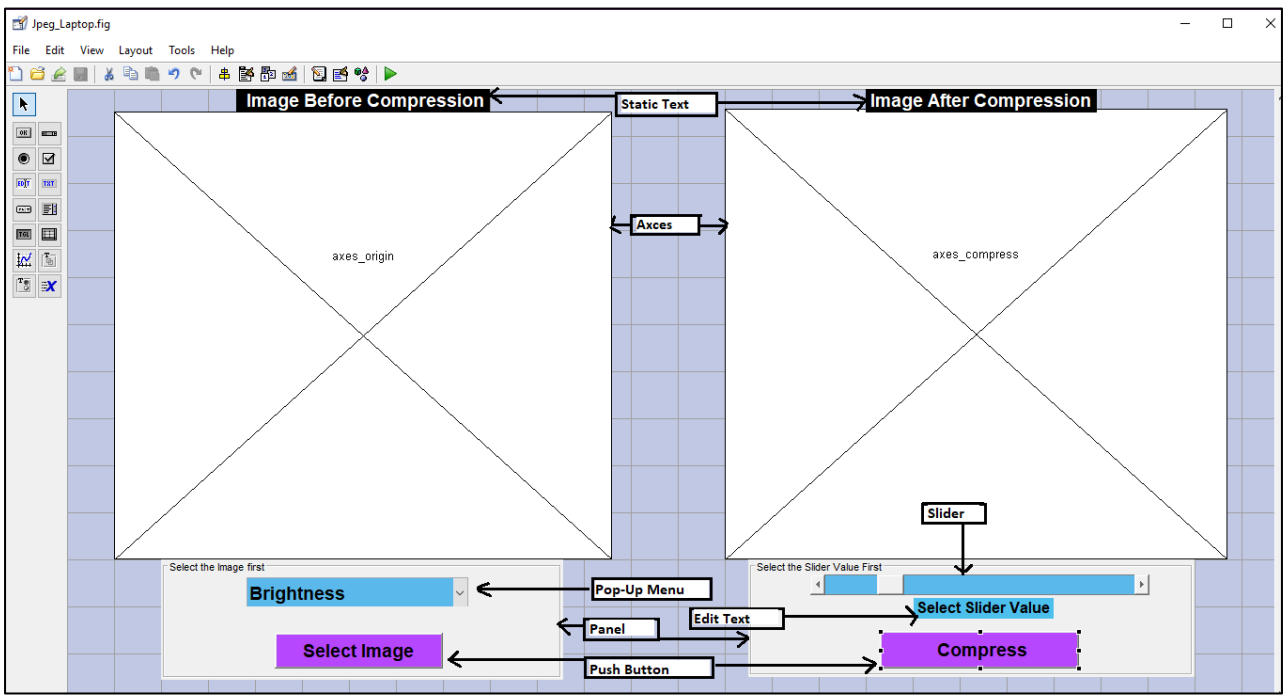
1. Original image, its Compressed image.
2. Brightened original Image, its compressed image.
3. Darker original Image, its compressed image.
4. Discarded Pixels from Each of above three factors (Original,Brighter,Darker).
5. Displays current value of K.

Design & Implementation

- **Coding**

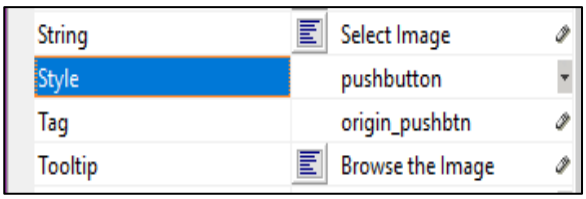
  - Input from user via browsing on origin_pushbtn function uigetfile ,saved it in Handles to make its scope global
  - Checked whether the image is color/grayscale using if-else condition on number of planes - Grayscale has 1 and Color image has 3 planes red, green, blue then used concatenation to add all planes in result.
  - Function for Reconstructed_image= idct2(((dct2(original Image))/(K*Q))*(K*Q))
  - Used blockproc function to process the original image by applying DCT/IDCT above function to each distinct 8*8 block and concatenating the results into the variable matrix.
  - edit_slider textbox will set the value from slider using str2num, on sliders slide it will get values by num2str
  - Initialized slider_k = 1 once and after user moves the control it will get the current values from GUI and pass it on for Multiplying/dividing Q.
  - Used Switch case for selecting:
    - Brightness takes the current values of image and adds the hardcoded values to pixels.
    - Darkness takes the current values of image and subtracts the hardcoded values to pixels
    - Discarded Pixels will subtract the Original image from Compressed image and shows output only with removed pixels

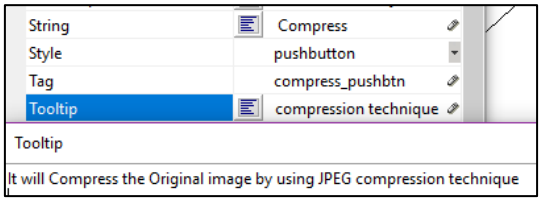
- **GUI:** Dragged and dropped below component



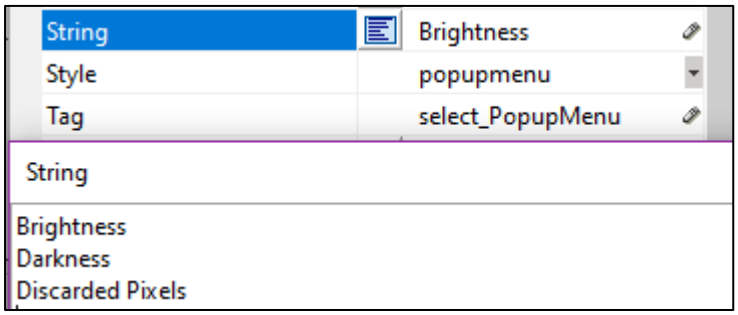
Changed **Properties**: BackgroundColor, ForegroundColor,
-String –to Display on screen,
-Tag –to identify component needed for function call,
-Tooltip –hovering will give comment on window.

- Push Button: Once clicked on this it will call the function 1st is used to open new window for browsing and selecting that image and showing it on axes_origin, and compress_pushbtn will call reconstructed image function and shows the compressed output on axes_compress

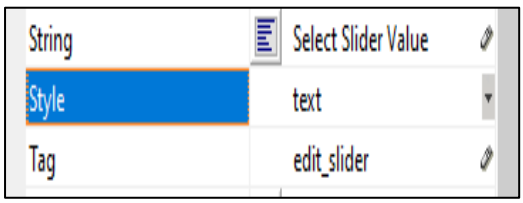


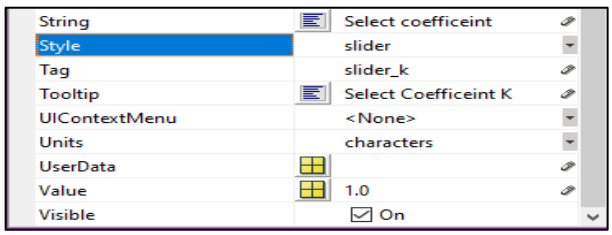


- Pop-up Menu has three options as String:

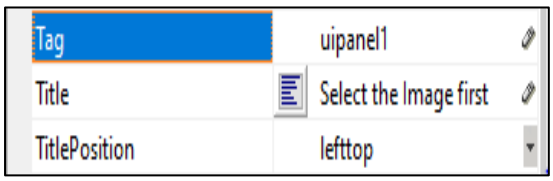


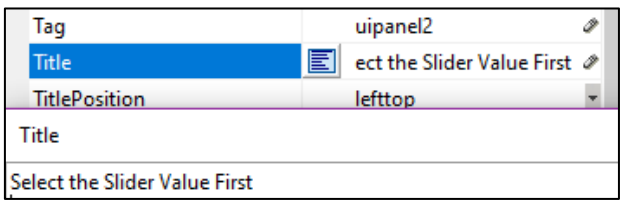
- Text and Slider



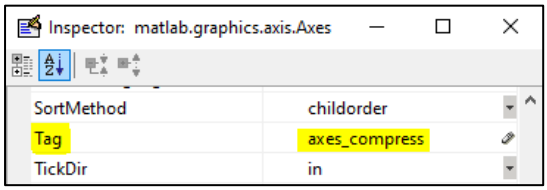


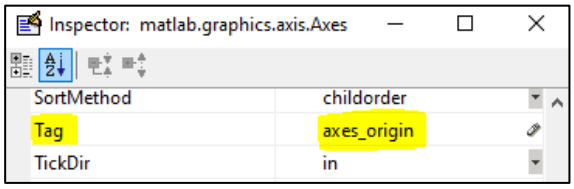
- Panel



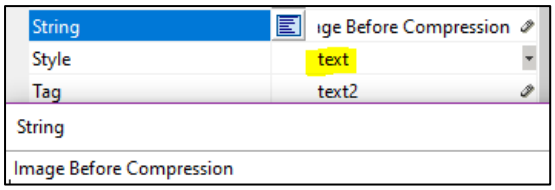


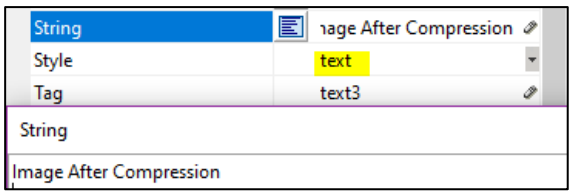
- Axis: visibility unchecked so no output till user selects the image.





- Text

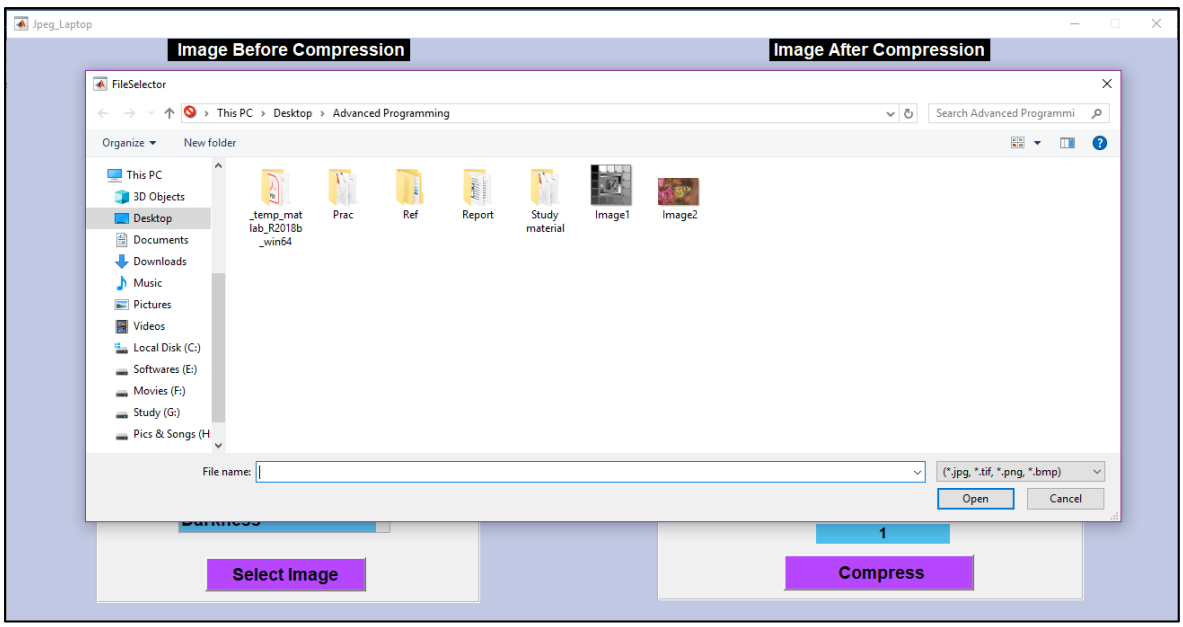




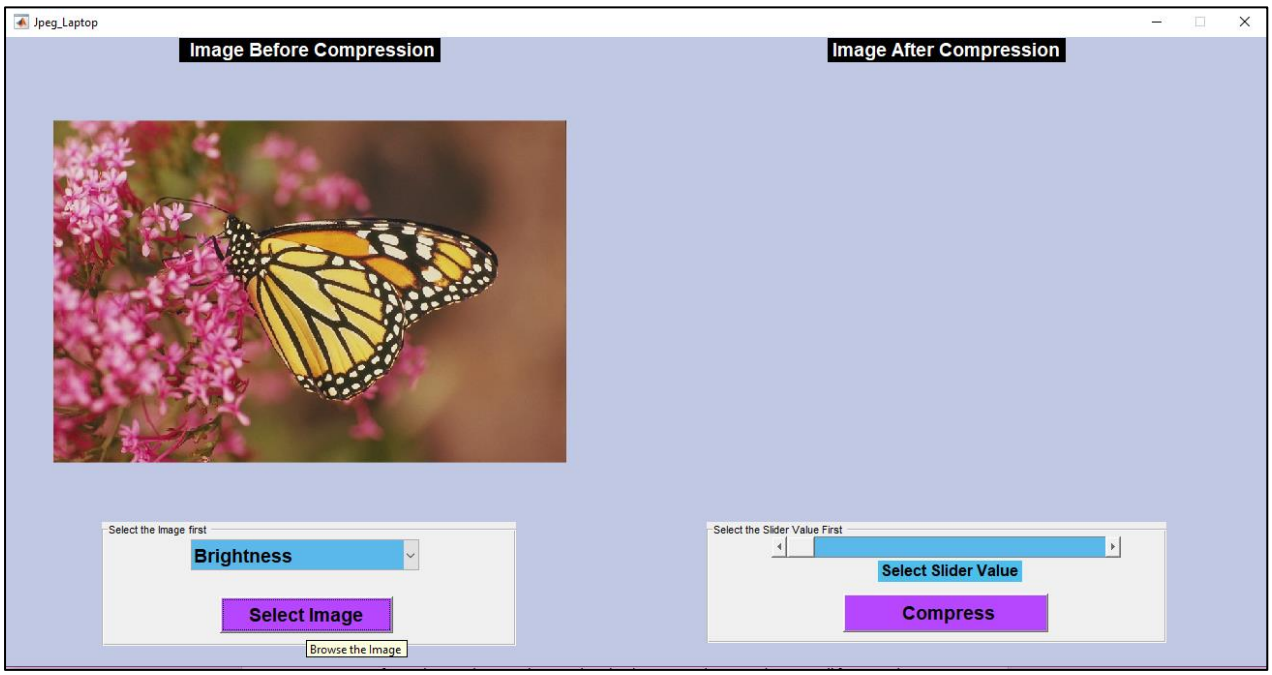
Testing & Evaluation

1. **Run** Program. shows below pop-up

2. Click on **Select Image,** when hover over Push Button it shows Tooltip as '**Browse Image**' and opens browsing window so user can select any image.

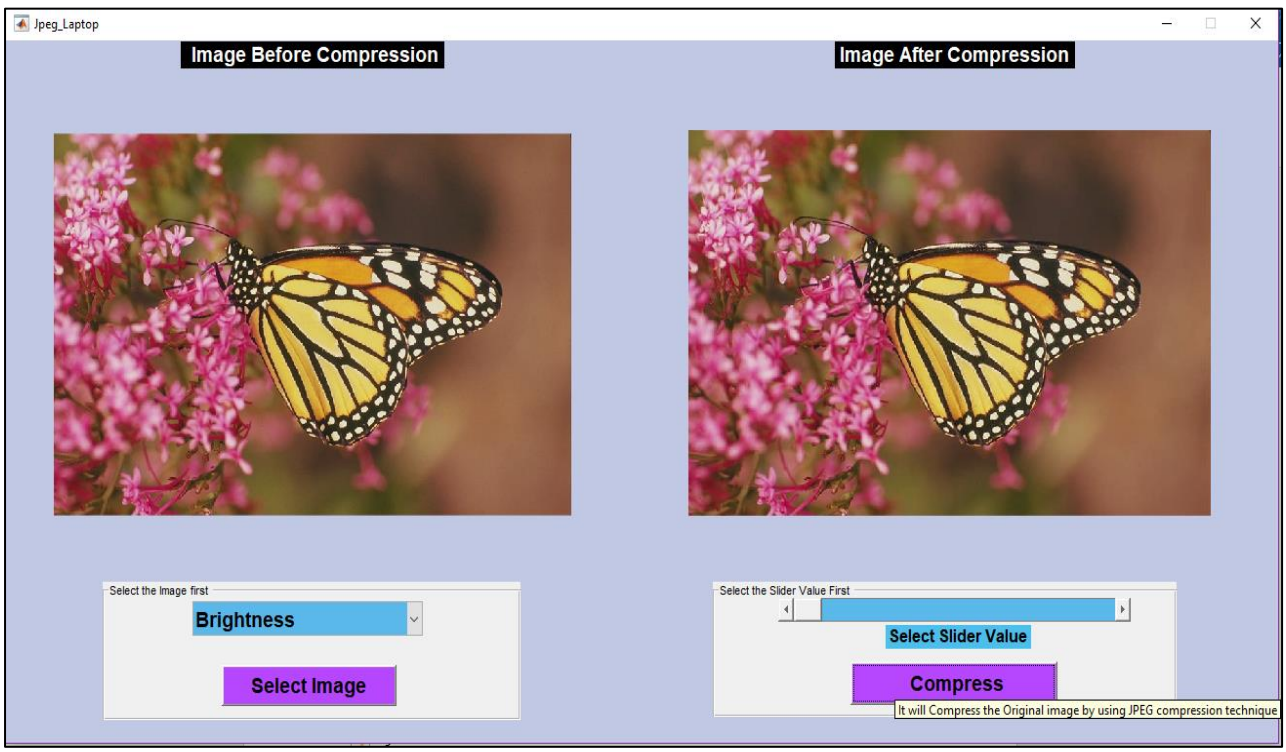


- **Color Image**

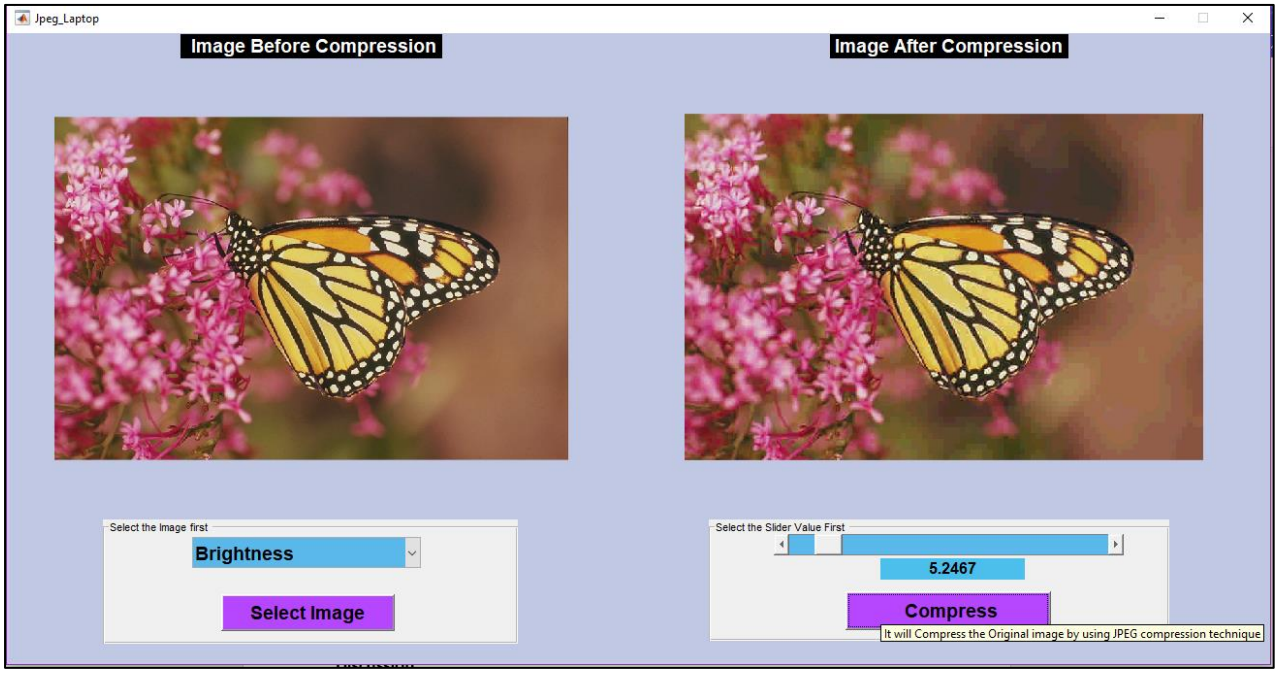


- **Original Image**

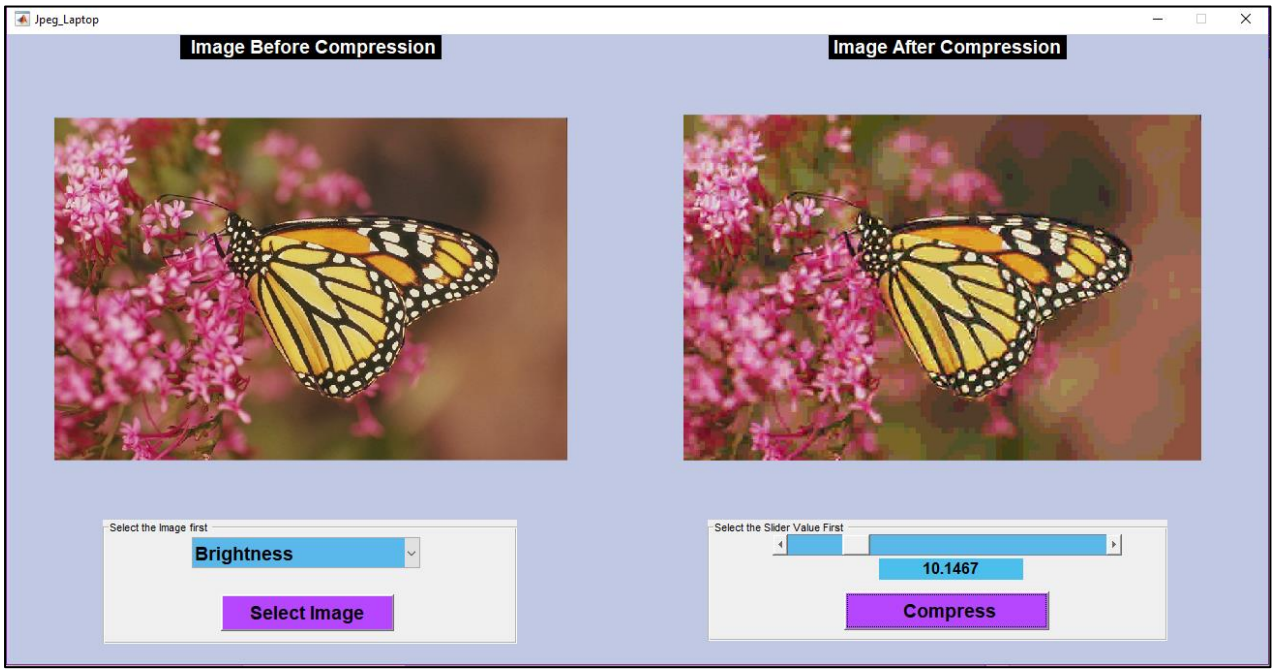
1. Click on **Compress**, it takes **K=1** initially as it is set as default value and from result we can't detect the compression



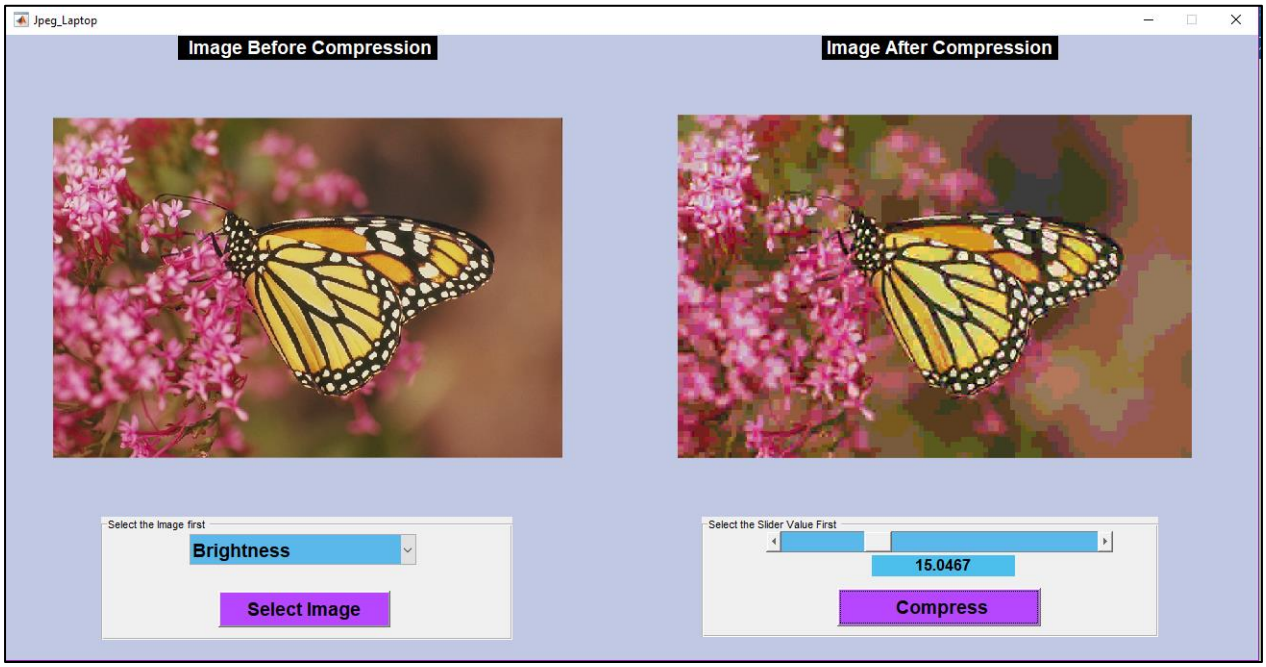
2. Click on Slider bar to select **K= 5.24** which is visible in text bar below slider and image shows little compression.



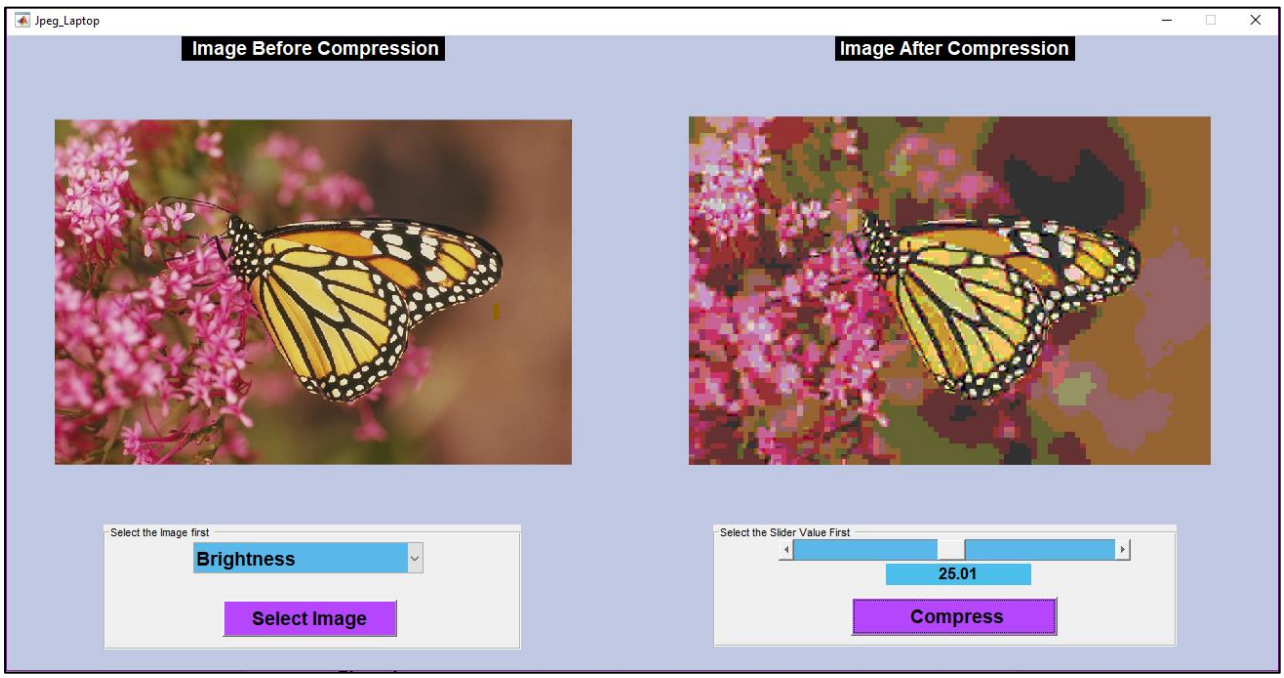
3. Slider value around, **K= 10.14** here the compress is clearly visible as the image starting to distort.



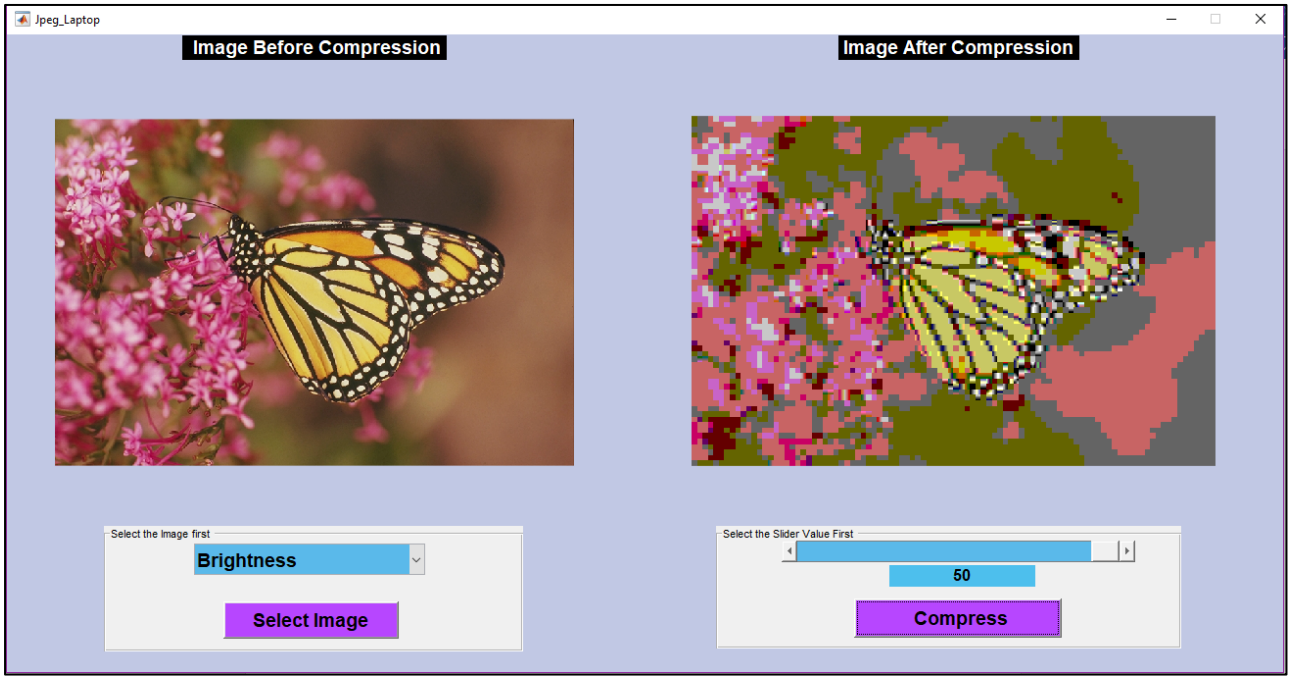
4. Slider **K= 15.04**



5. Compress at **K= 25.01**

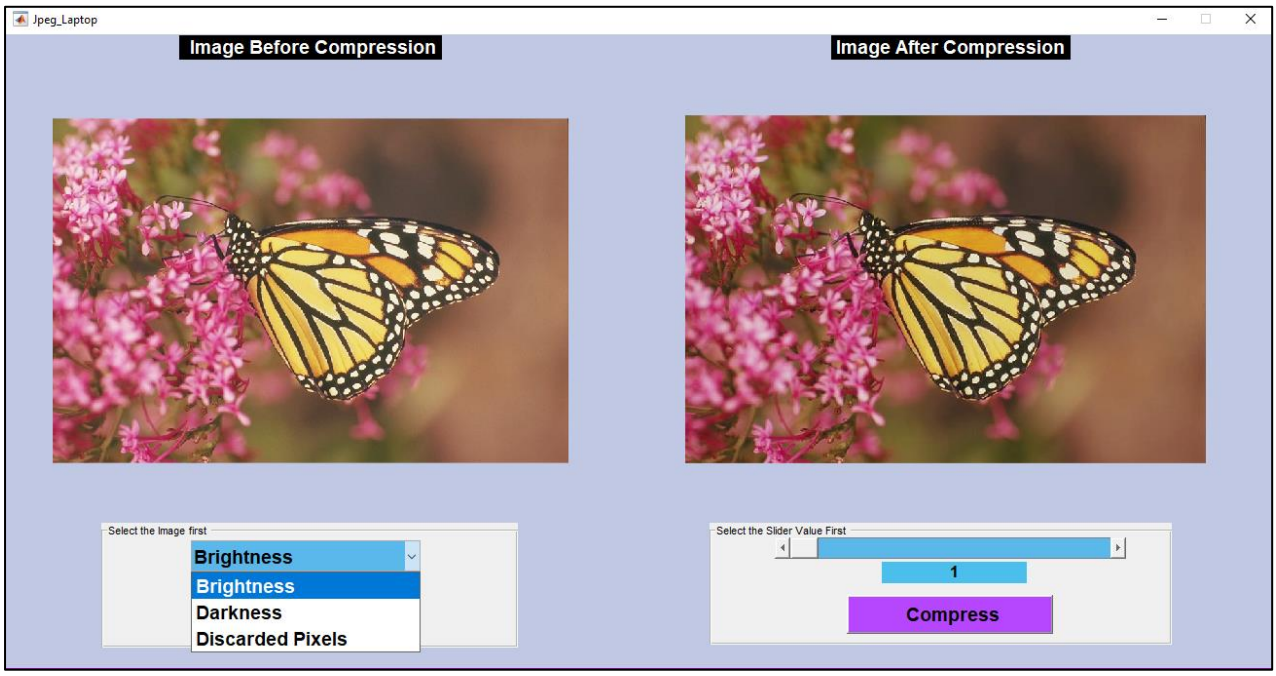


6. Compress at **K= 50,** this changes the color of given image as the multiplying coefficient is very high and its rounding off values with more difference.

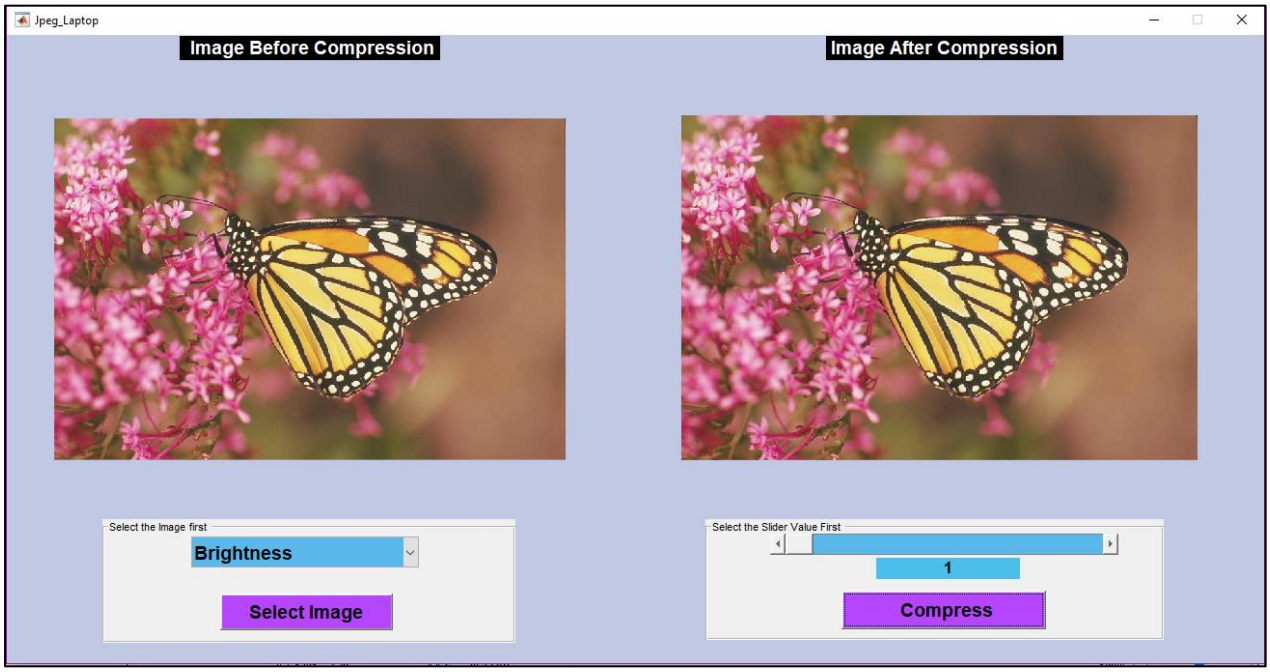


- **Pop up Menu Brightness:**

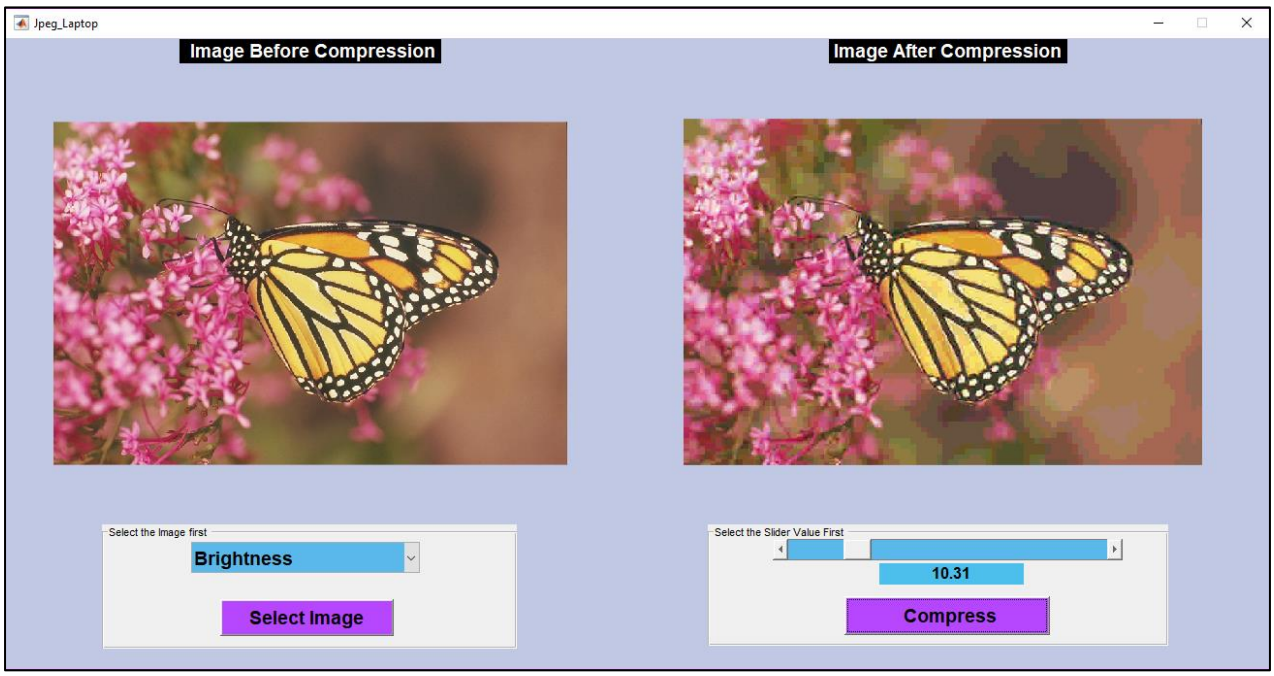
1. Select the **Brightness** from Pop-up Menu, the difference in Image can be seen as the second window shows the compression of original image with factor 1.



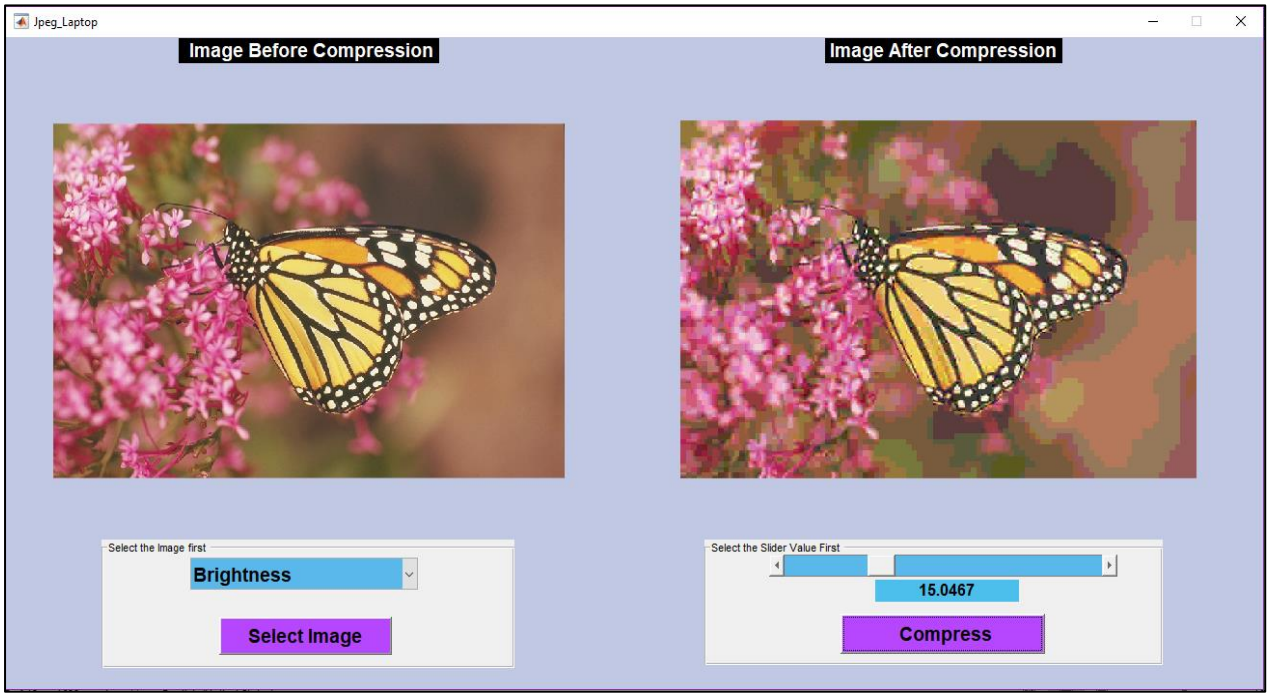
2. Compress with **k= 1**



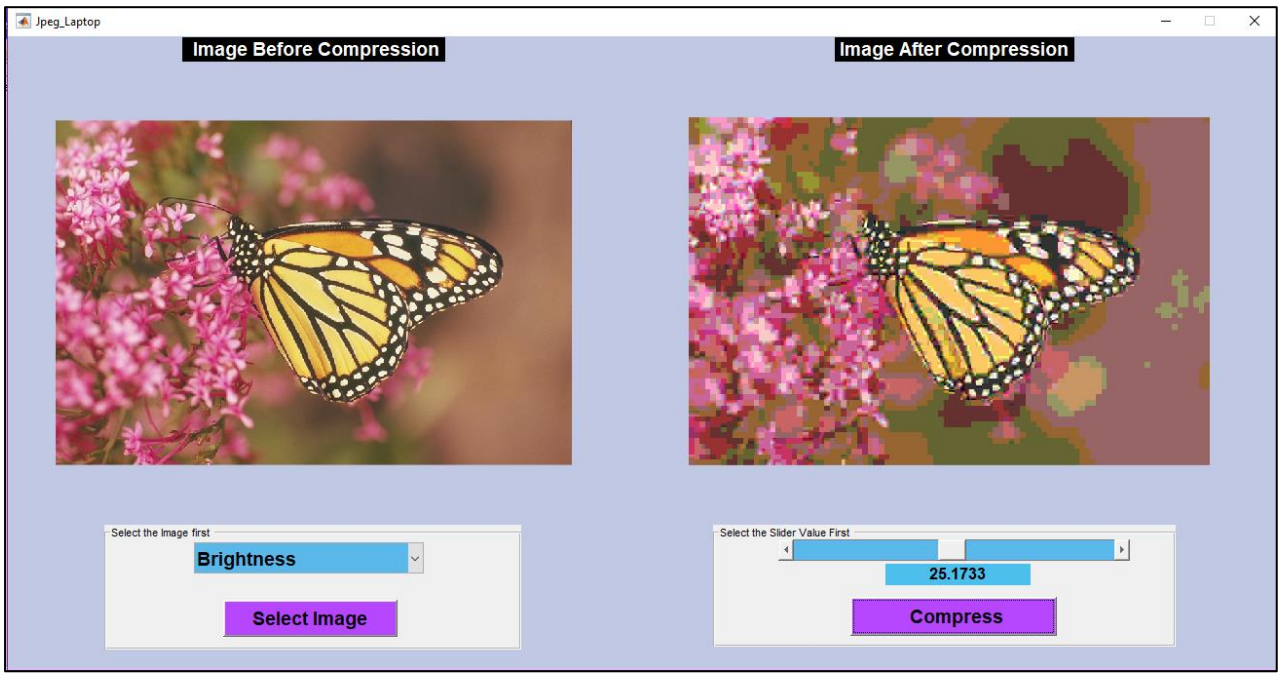
3. **K= 10.31**



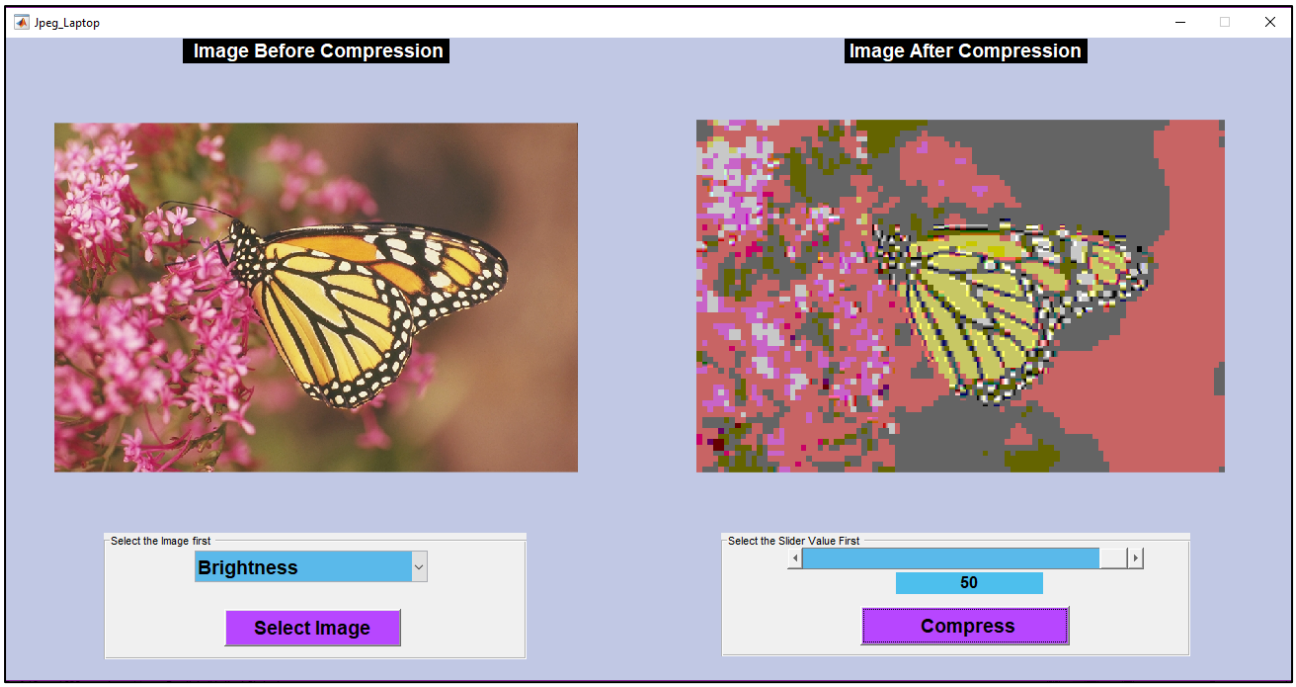
4. **K= 15.04**



5. **K= 25.17**

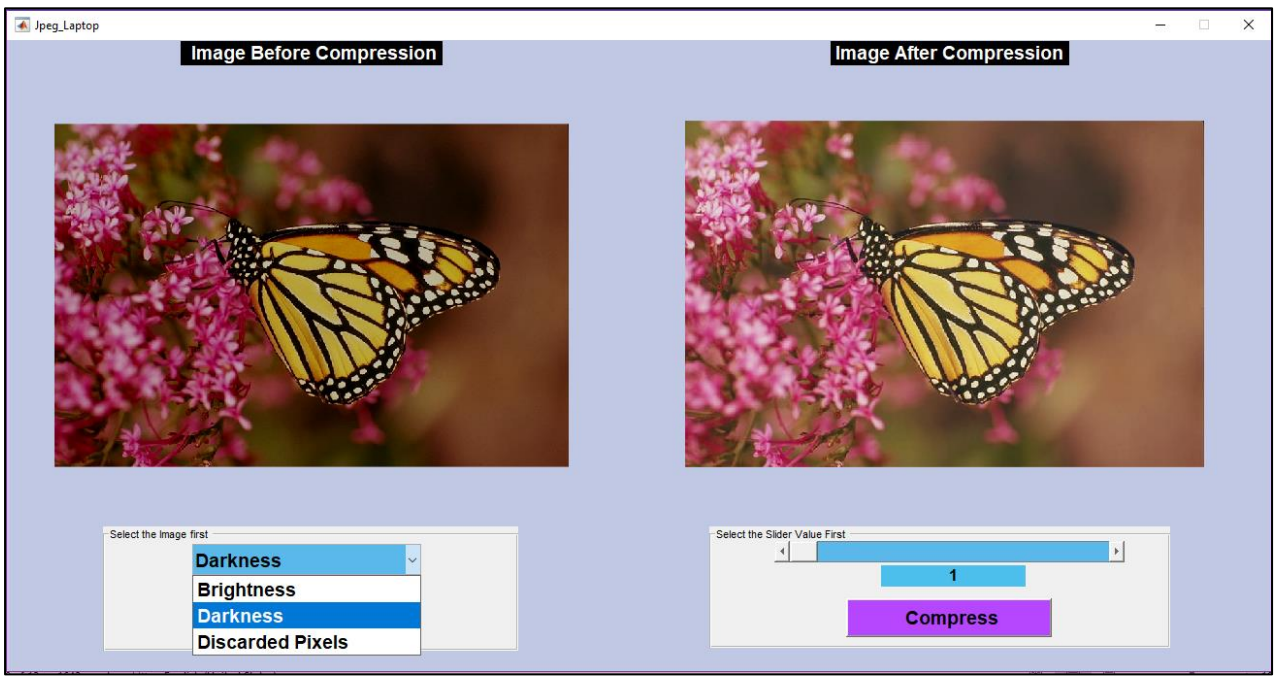


6. **K= 50**

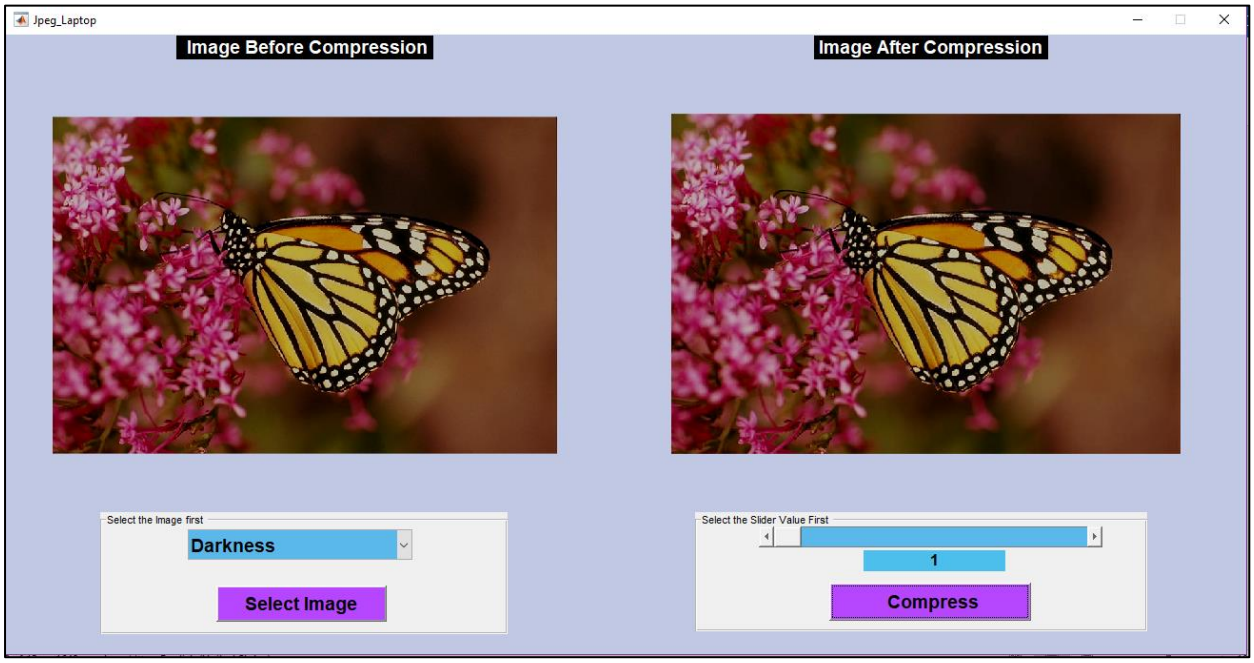


- **Pop up Menu Darkness**

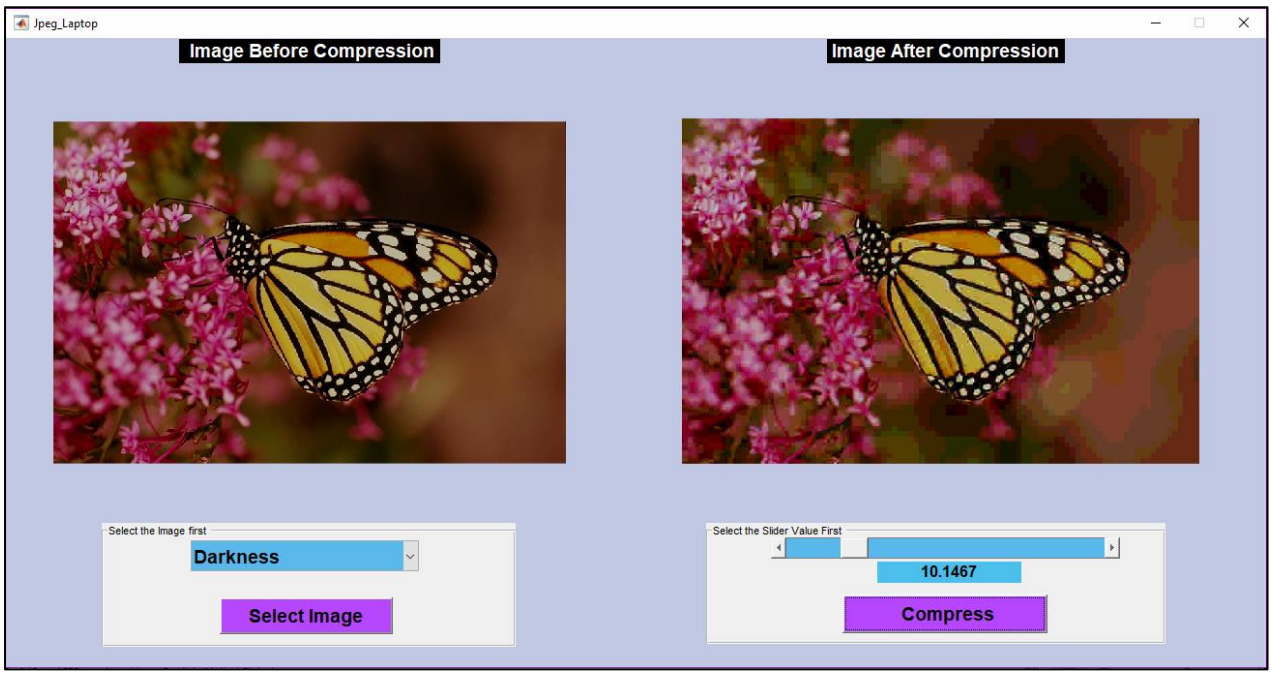
1. Select the **Darkness** from the Pop-up Menu, the difference in Image can be seen as the second window shows the compression of original image with factor 1.



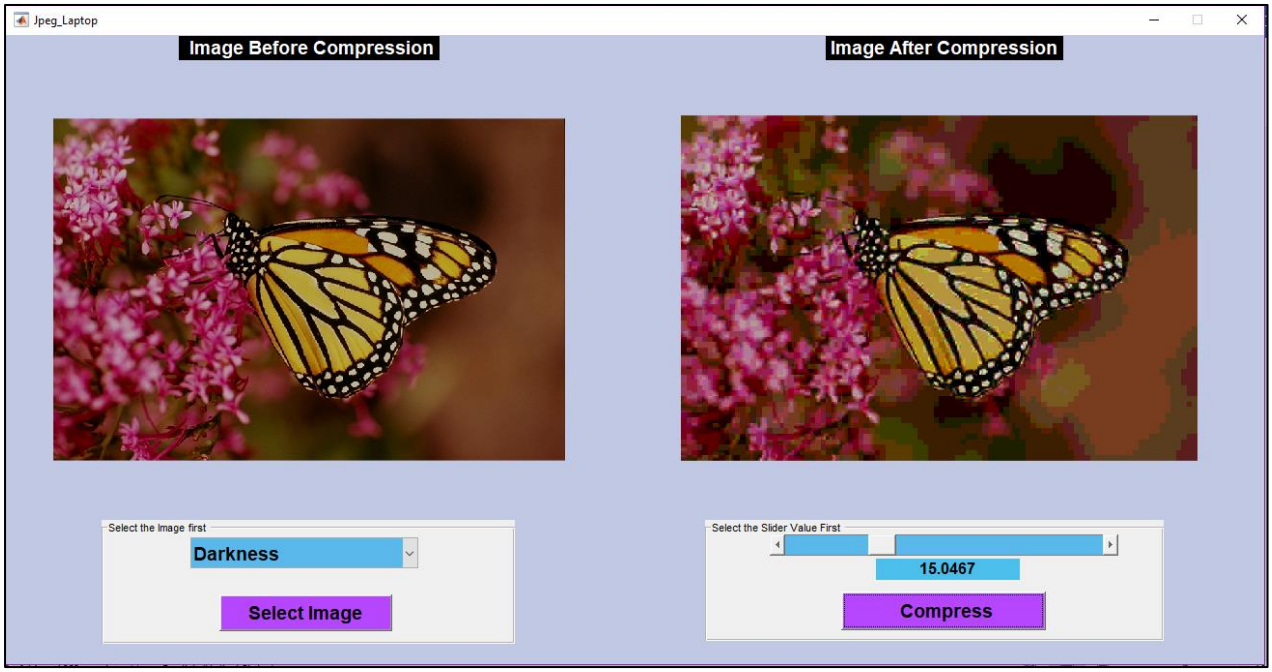
2. **K= 1**


Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Darkness
Select Image
Select the Slider Value First
1
Compress

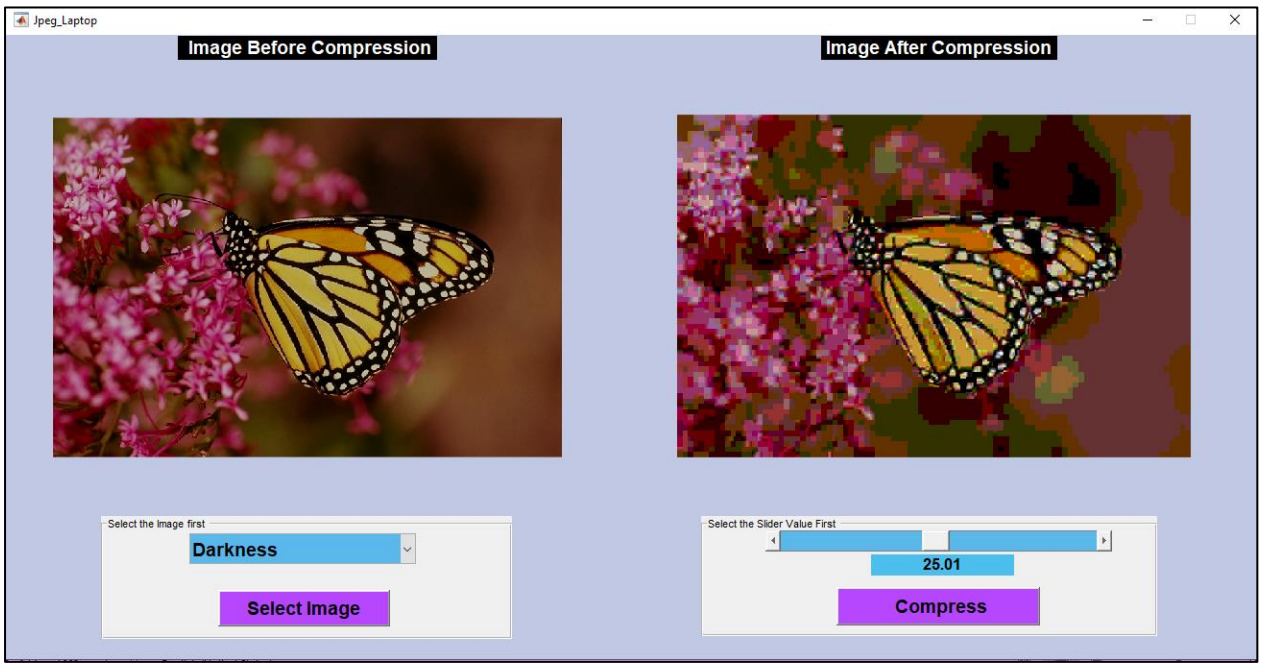
3. **K= 10.14**


Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Darkness
Select Image
Select the Slider Value First
10.1467
Compress

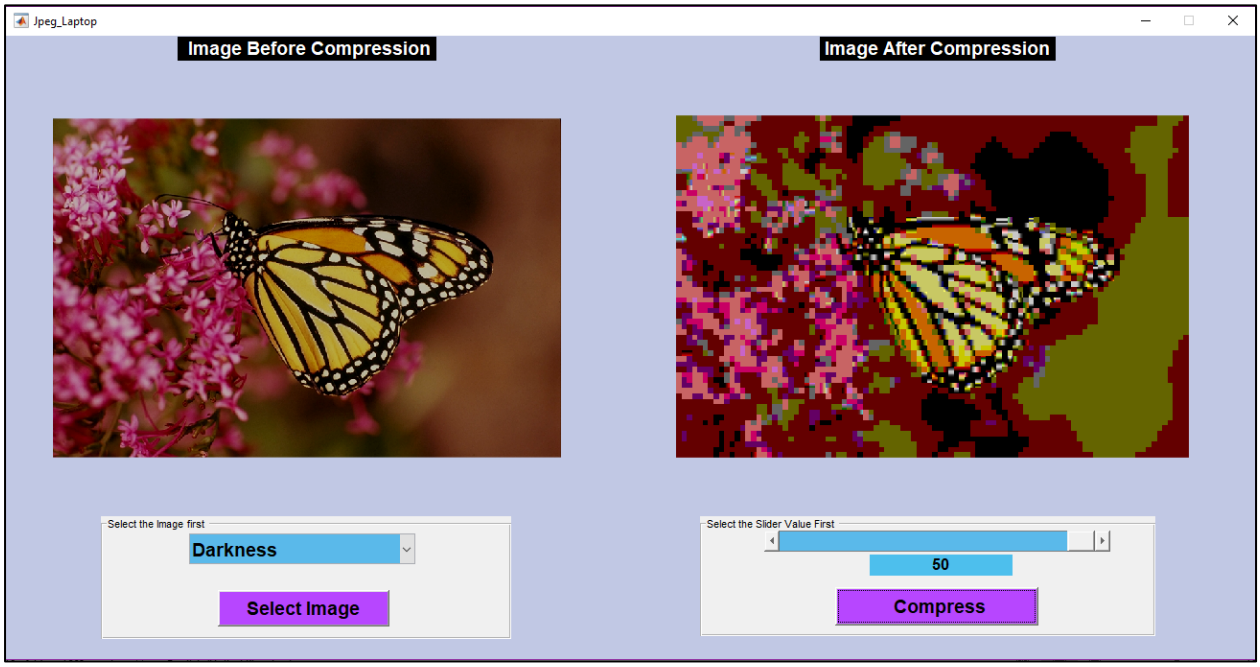
4. **K= 15.04**



5. **K= 25.01**

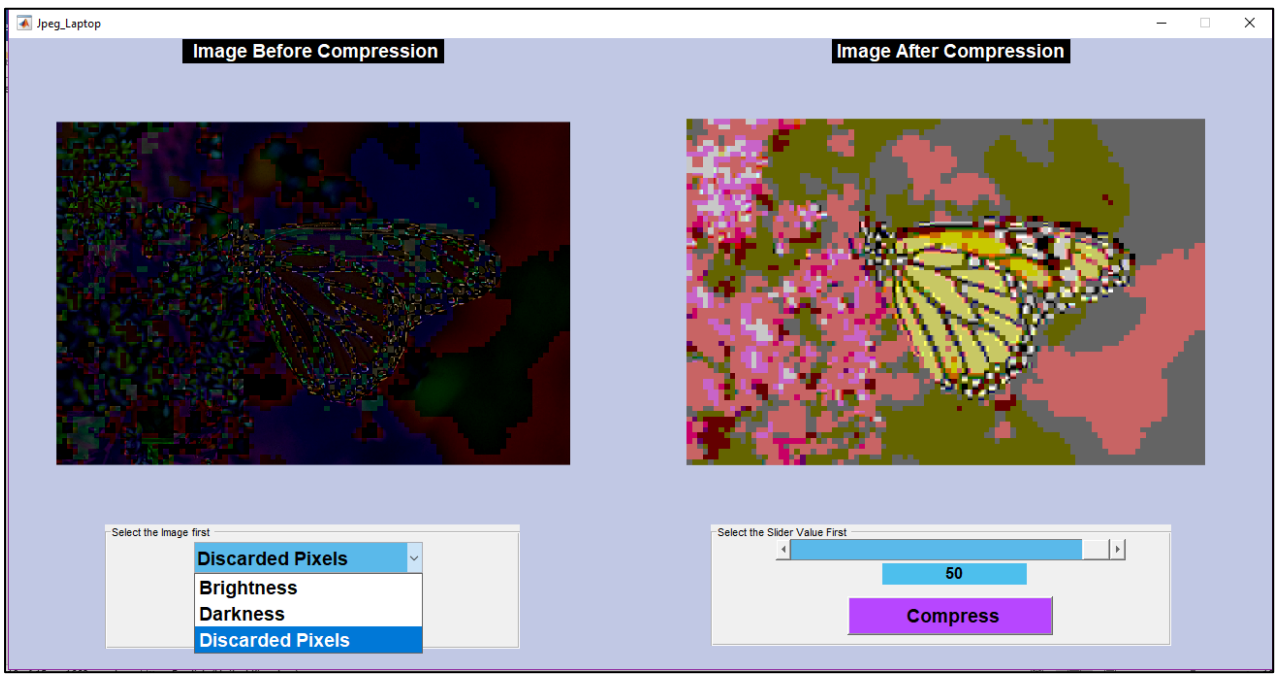


6. **K= 50,** most distorted image

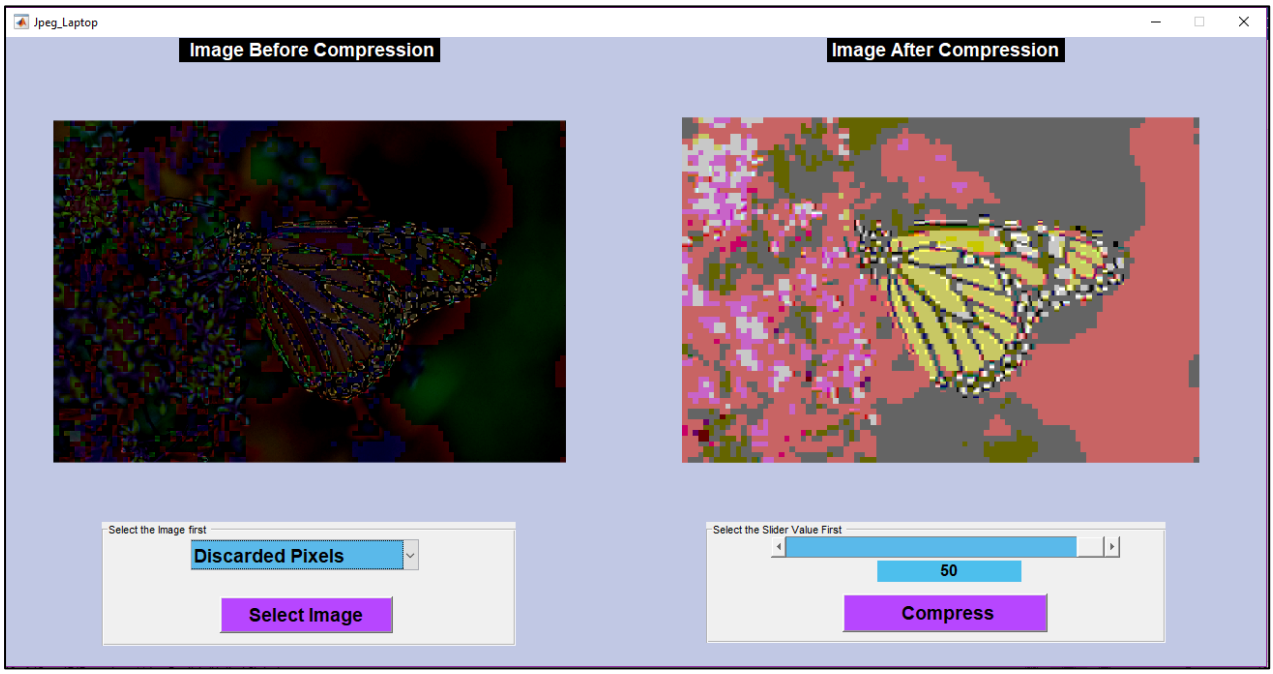


- **Discarded Pixels:**

1. Select the **Original** Image and then click on Compress format at **k= 50** (so the discarded pixels can be more visible), once both images are stored, Select **Discarded Pixels** from Pop-up menu



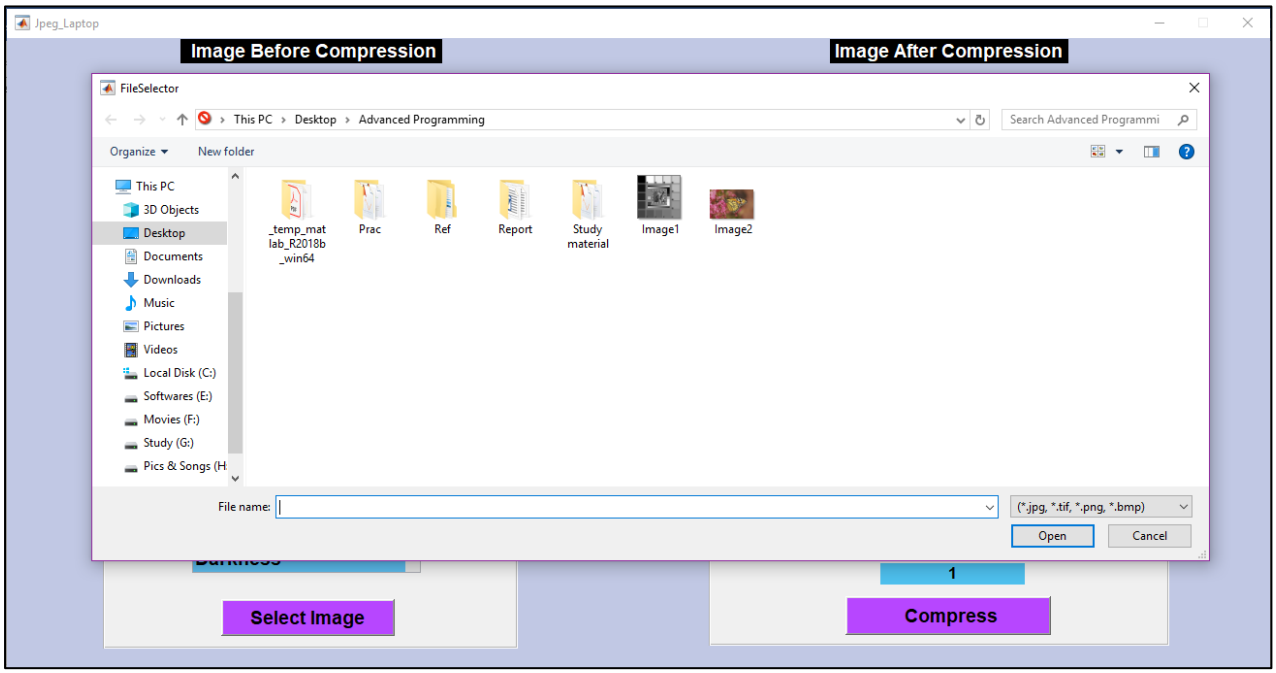
2. Select the image click on **Brightness**, click on Compress format at **k= 50**, once both images are stored, Select **Discarded Pixels**.



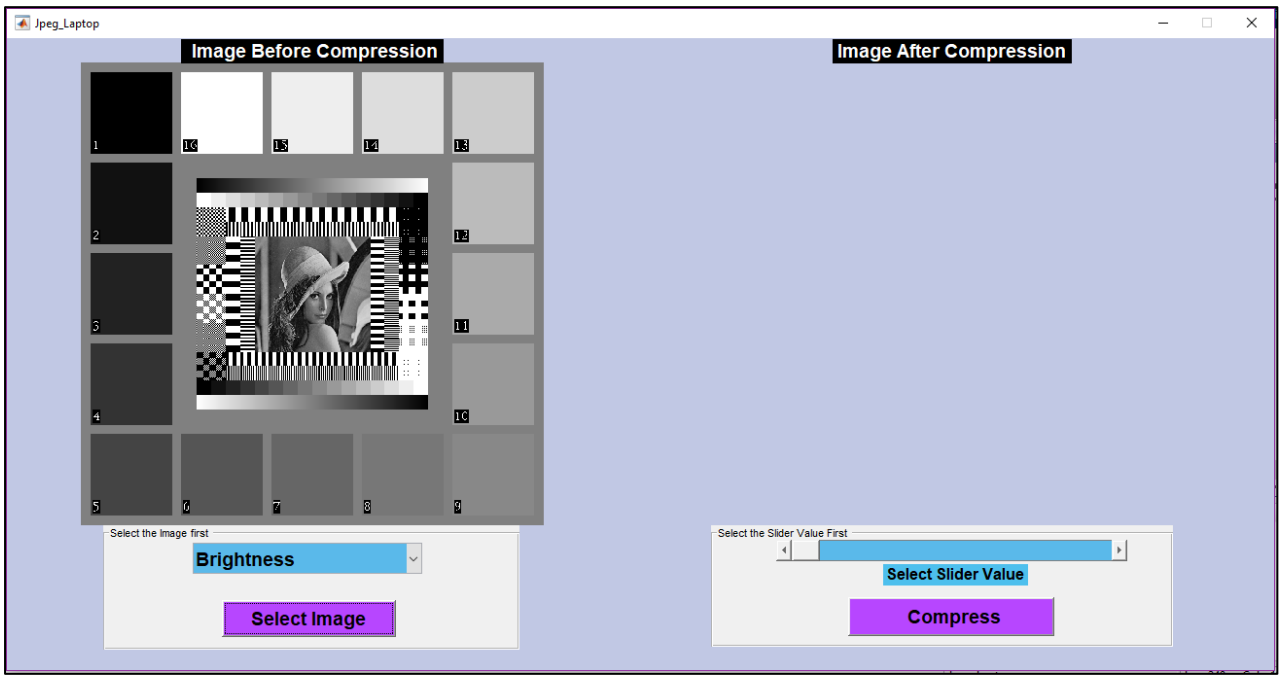
3. Select the image click on **Darkness**, click on Compress format at **k= 50**, once both images are stored, Select **Discarded Pixels**.

- **Grayscale Image**

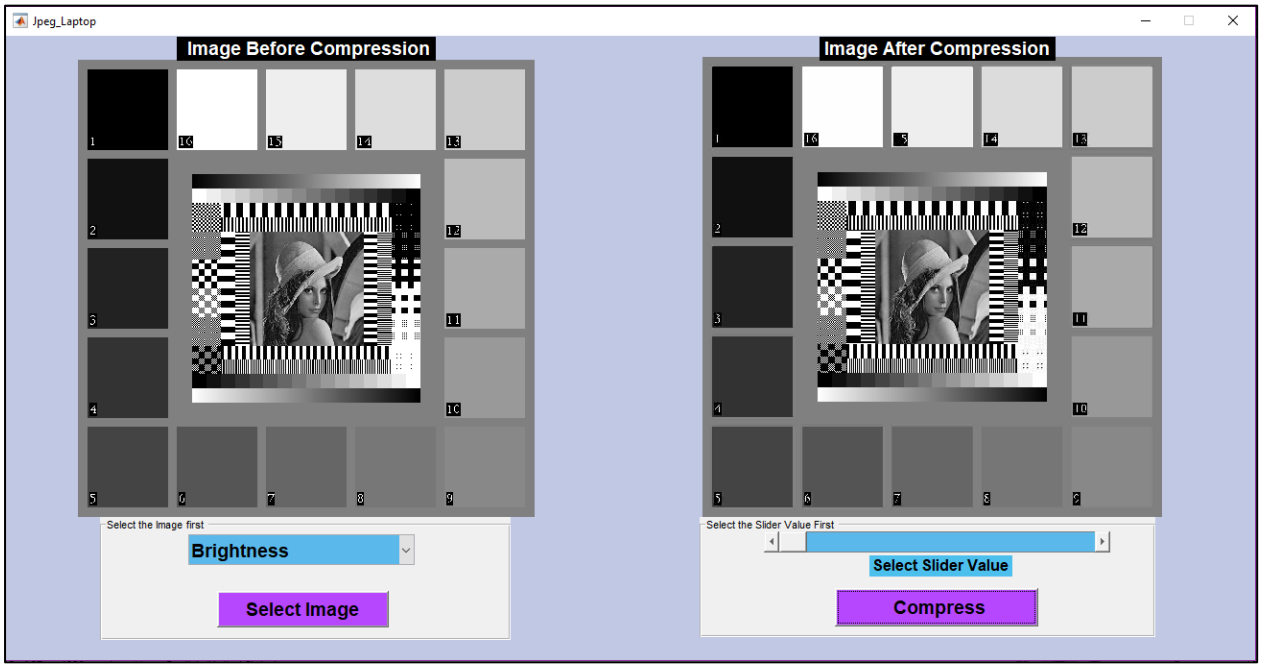
1. Run Program, Click on **Select Image**



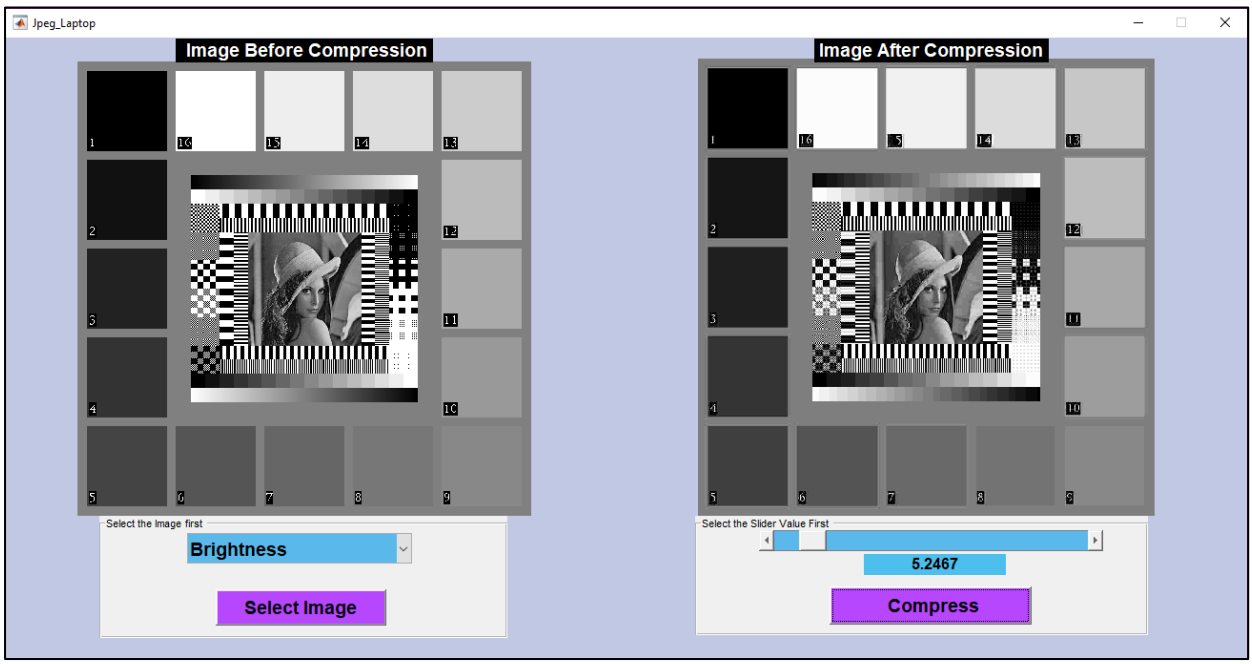
2. Selected Grayscale-Image



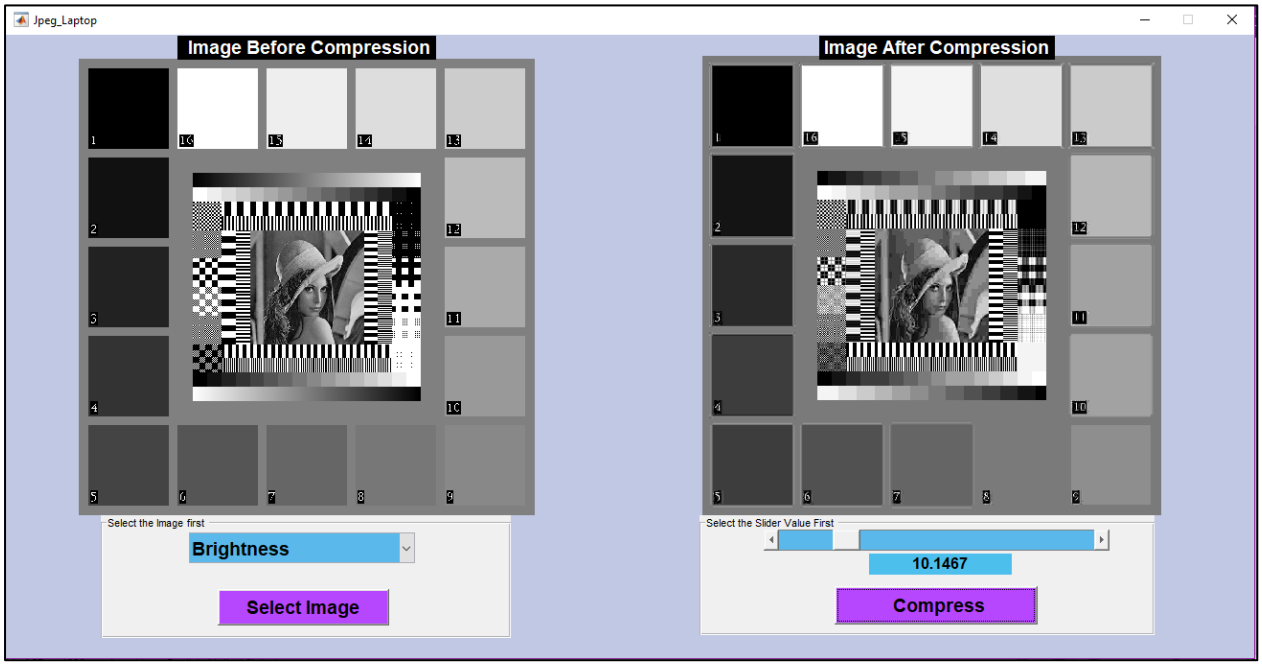
3. Click on **Compress**, it takes **K=1** default value



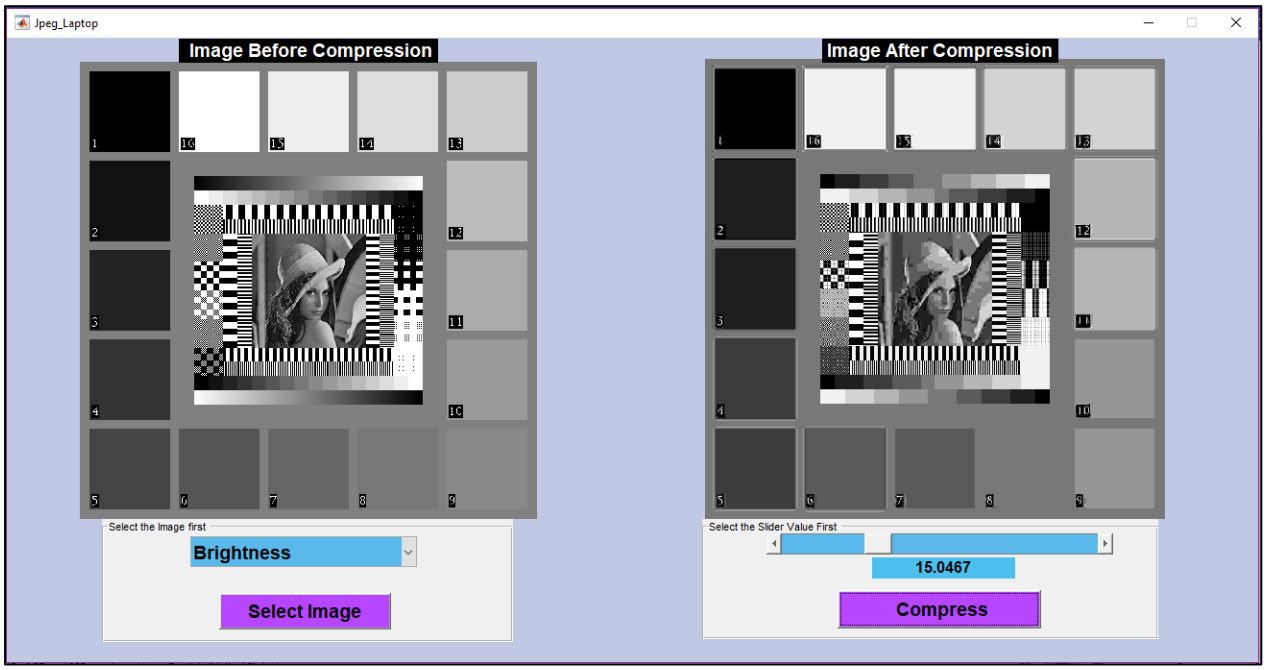
4. **K= 5.24**



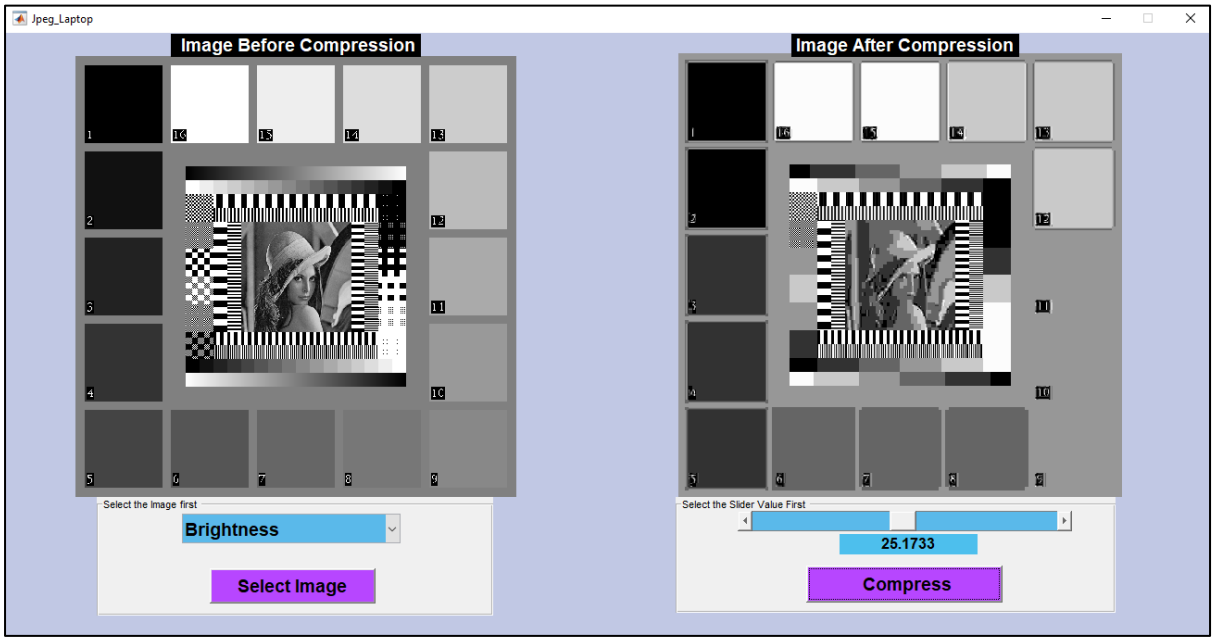
5. **K= 10.14**



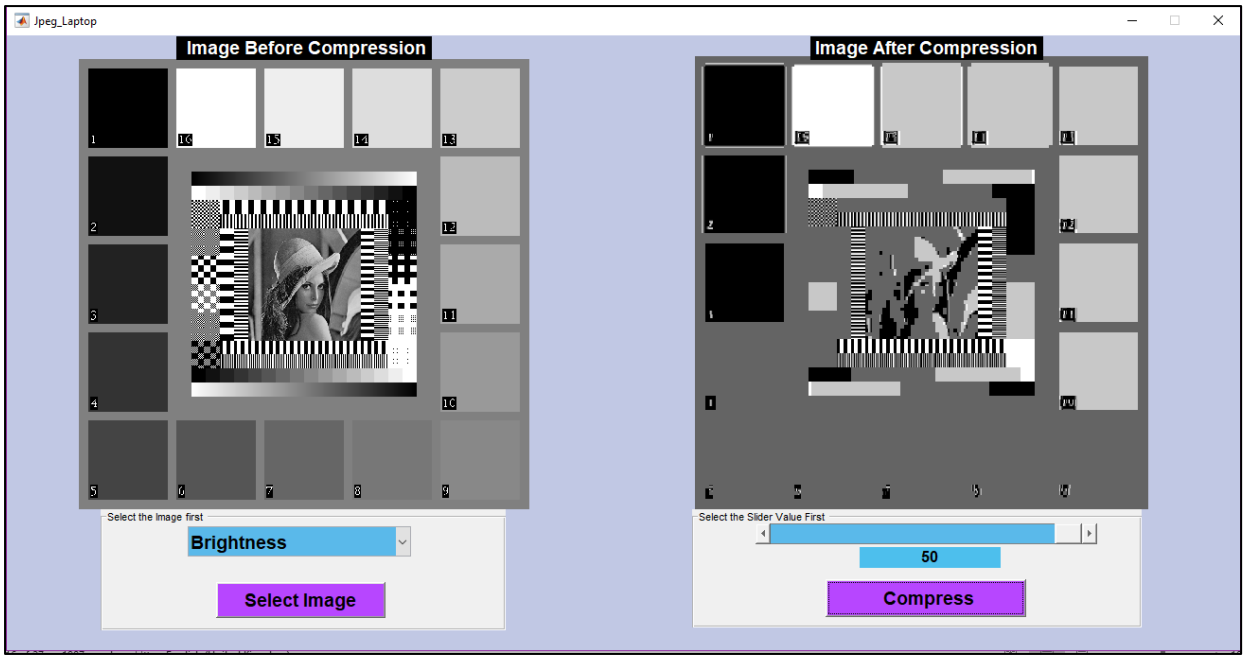
6. **K = 15.04**



7. **K= 25.17**

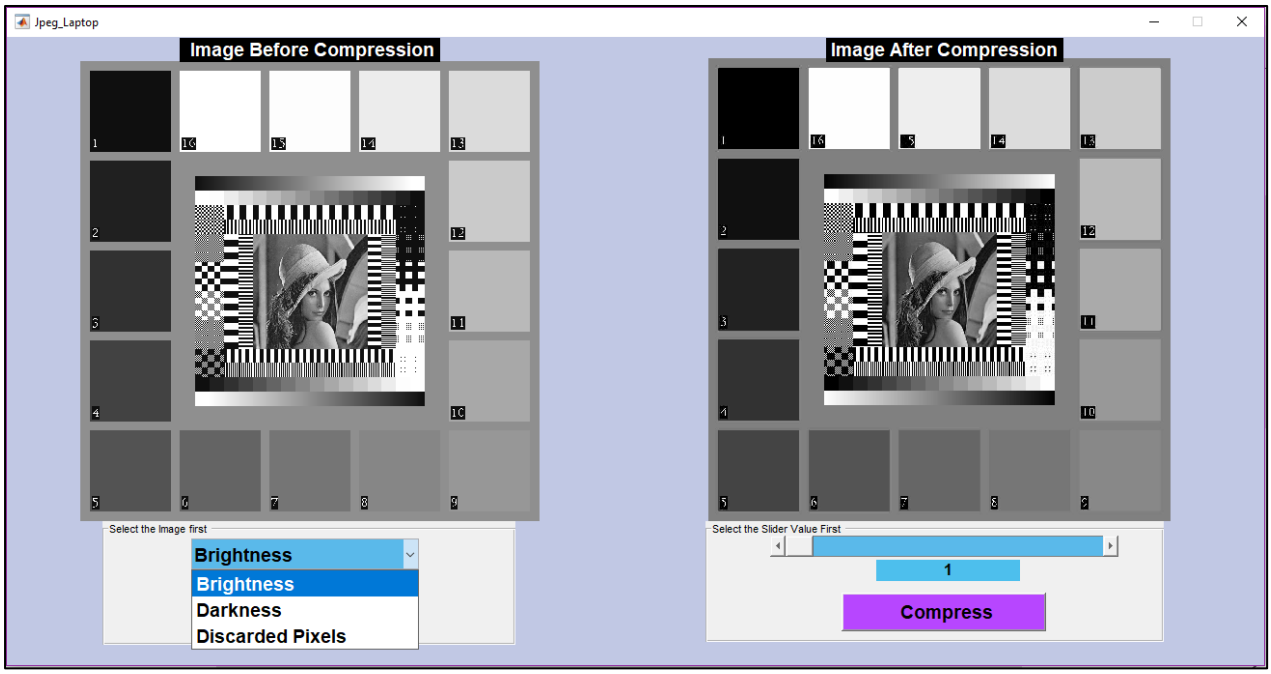


8. Compress at **K= 50**

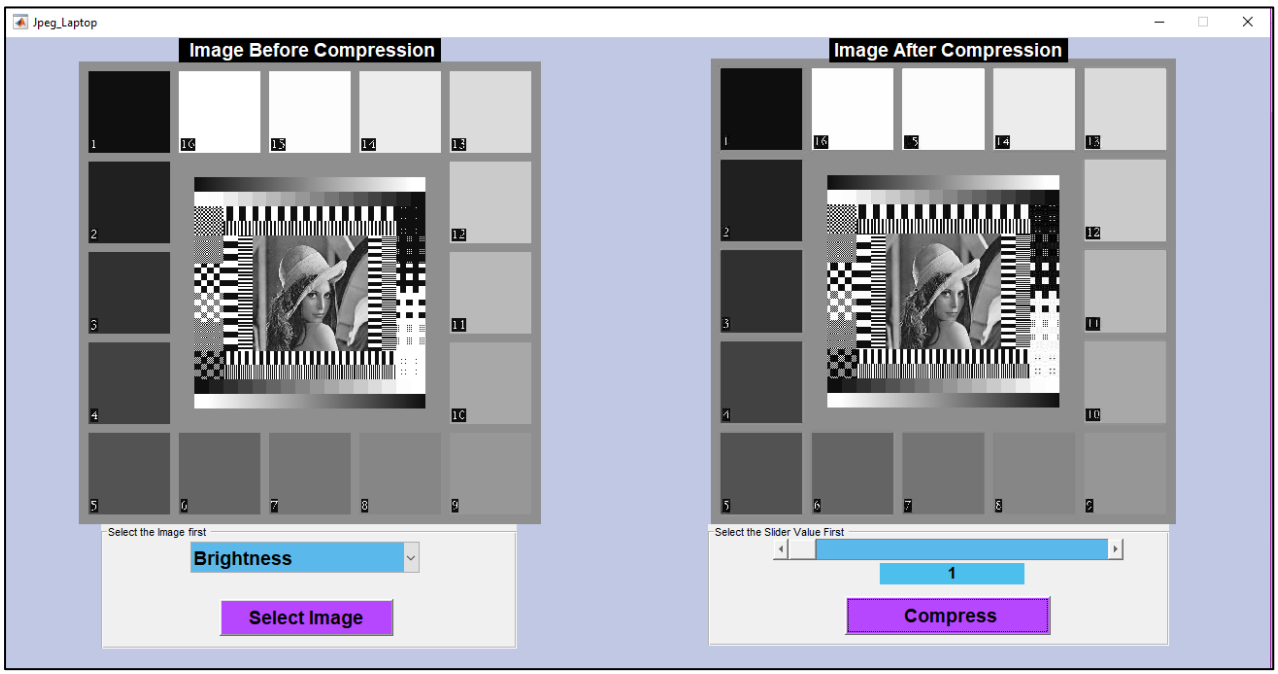


- **Pop up Menu Brightness:**

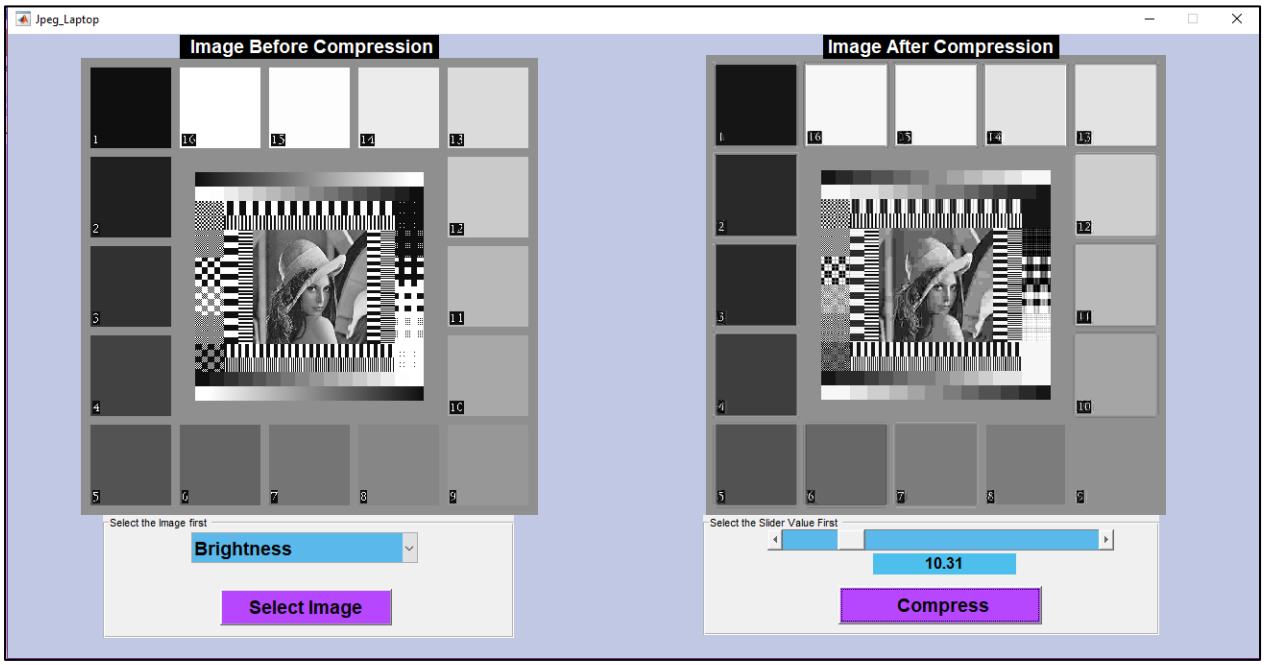
1. Select the **Brightness** from Pop-up Menu, the difference in Image can be seen as the second window shows the compression of original image with factor 1.



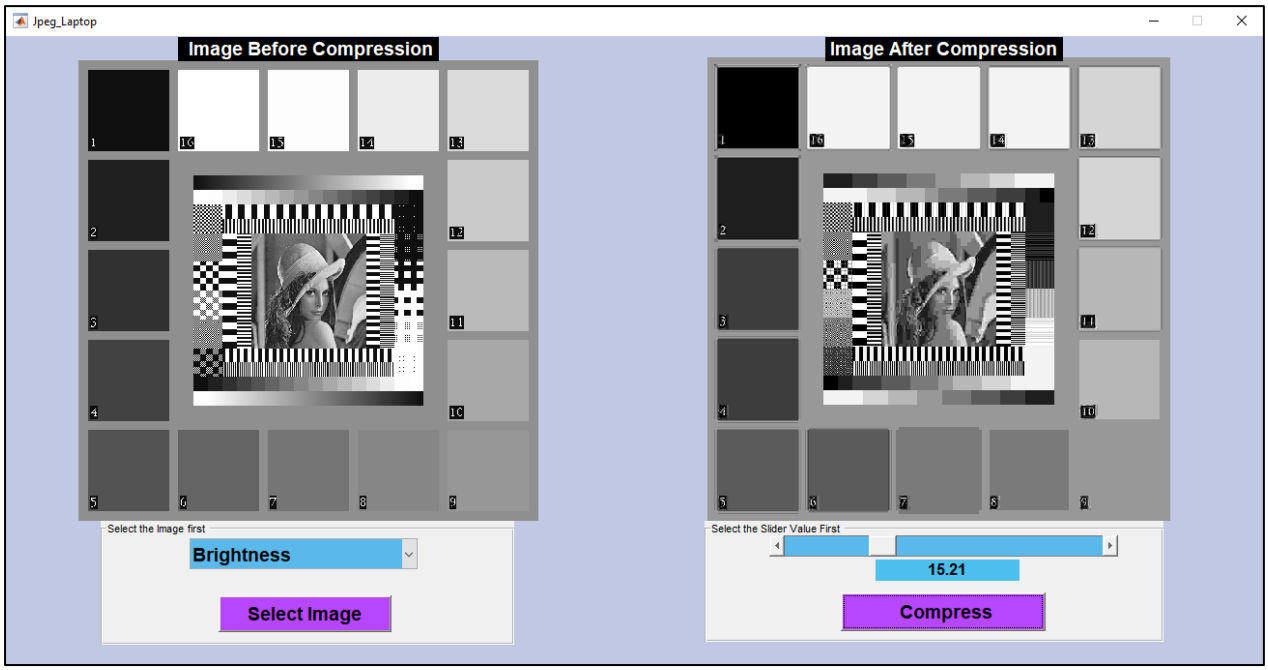
2. Compressing brighter image with **K= 1**



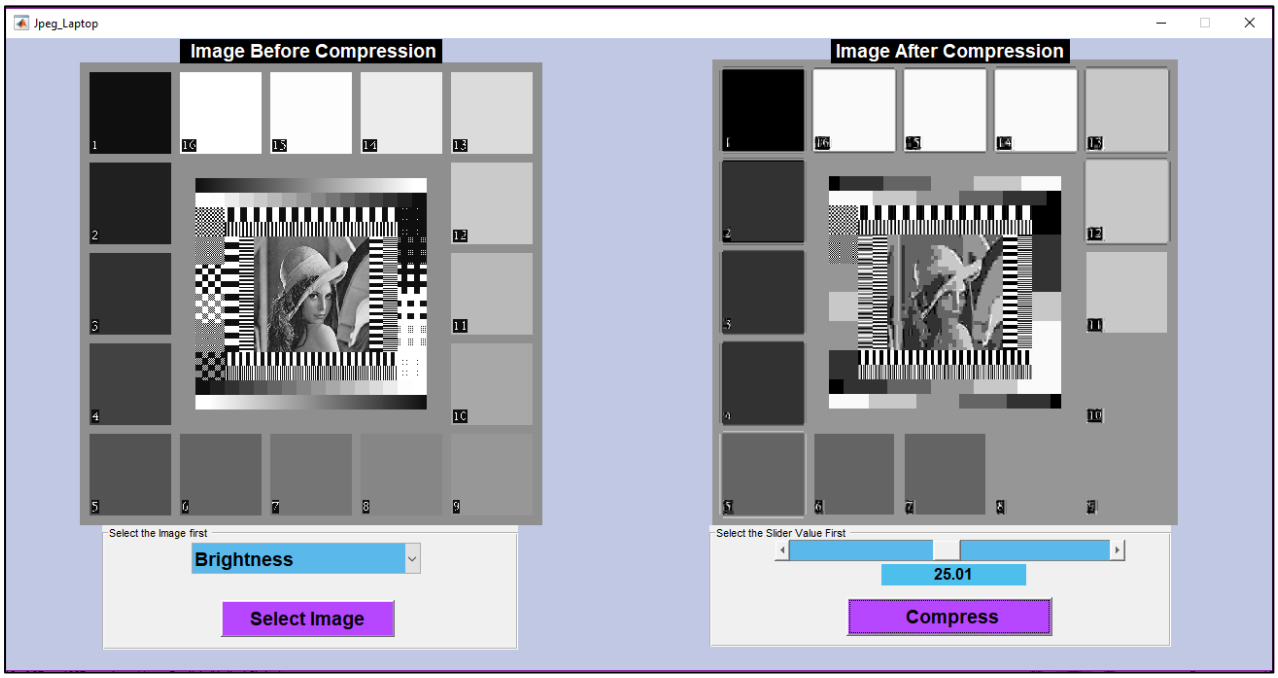
3. **K= 10.31**



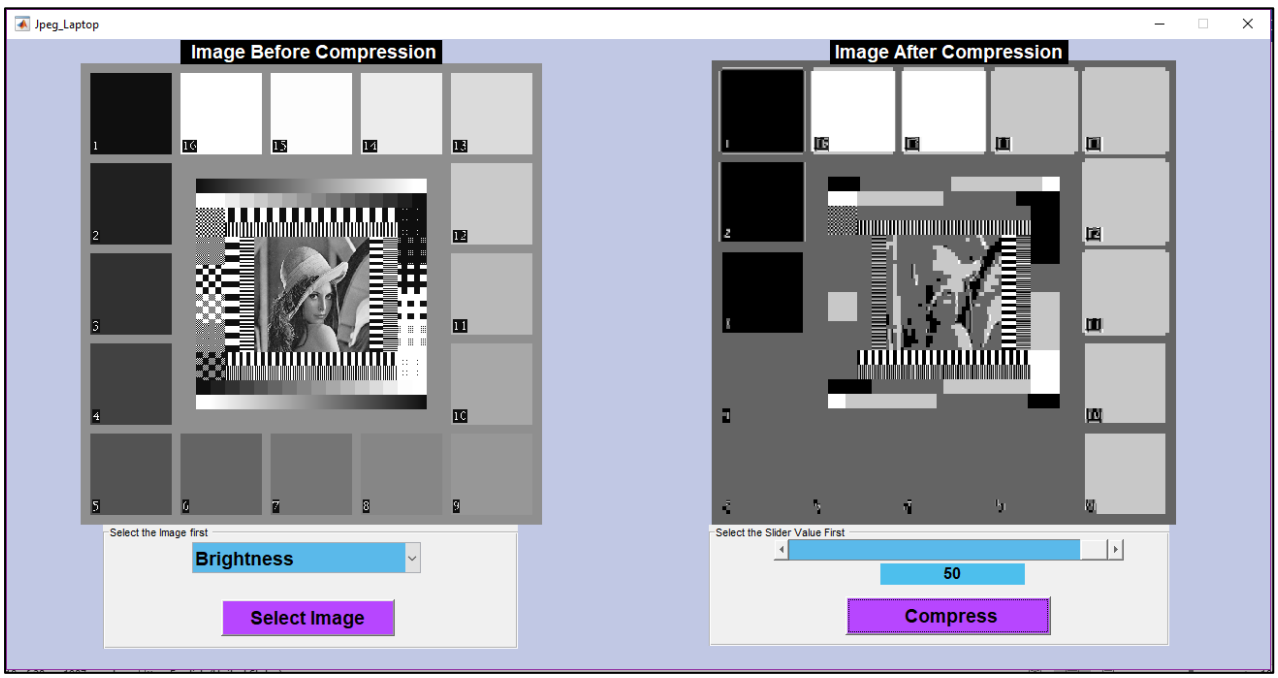
4. **K= 15.21**



5. **K= 25.01**

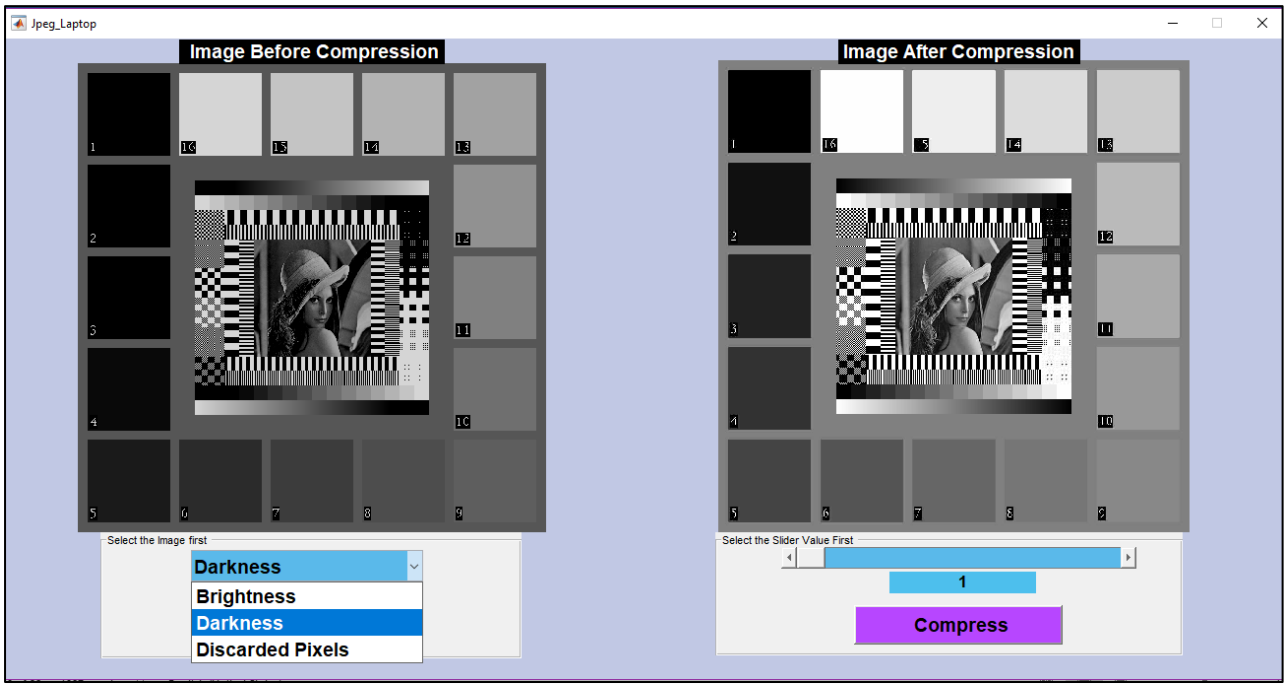

Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Brightness
Select Image
Select the Slider Value First
25.01
Compress

6. **K= 50**

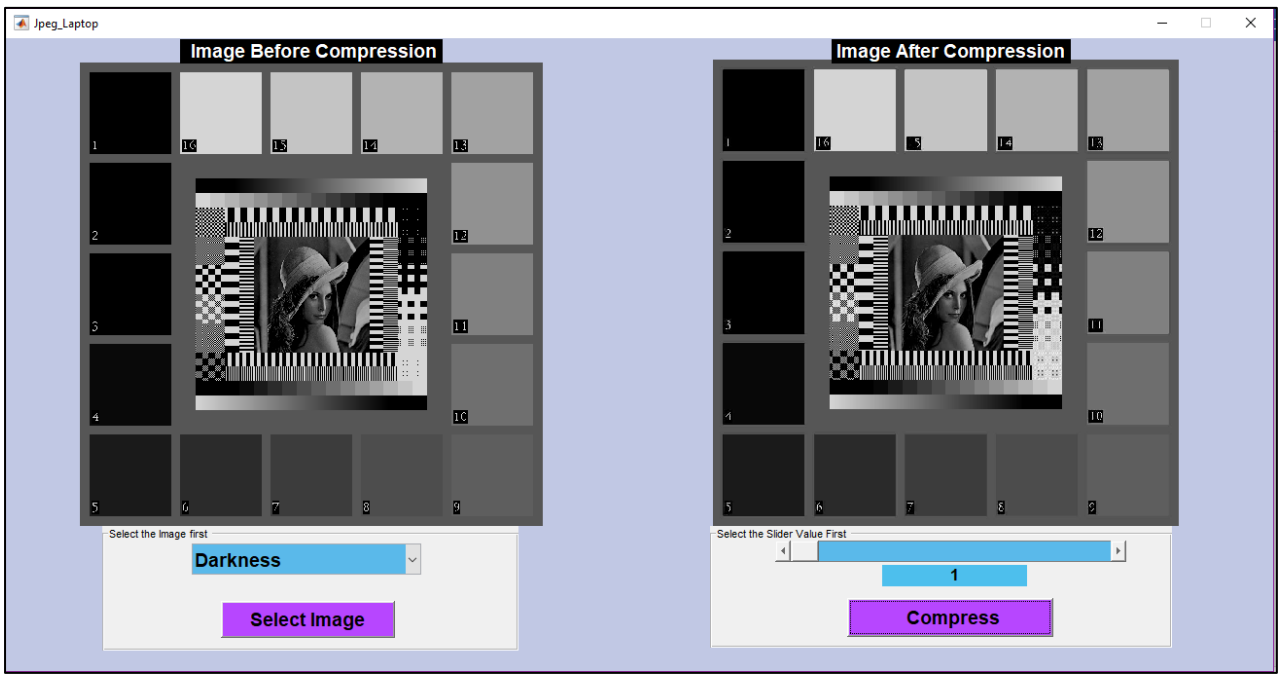

Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Brightness
Select Image
Select the Slider Value First
50
Compress

- **Pop up Menu Darkness**

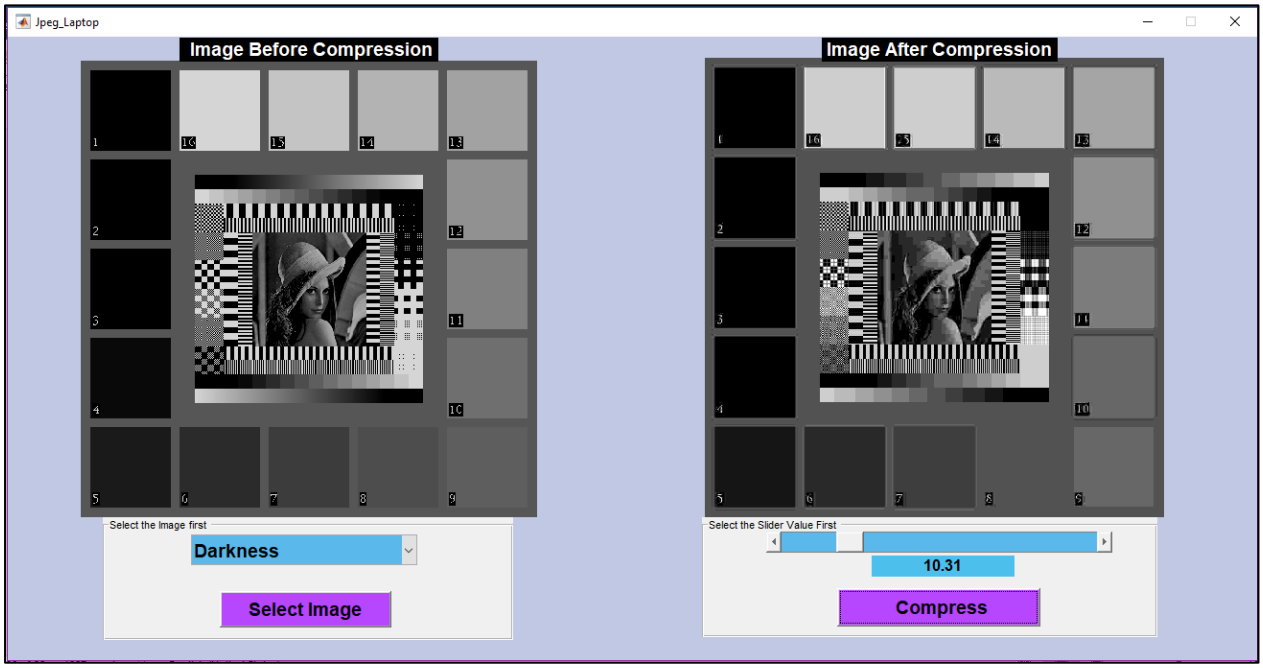
1. Select the **Darkness** from the Pop-up Menu, the difference in Image can be seen as the second window shows the compression of original image with factor 1.



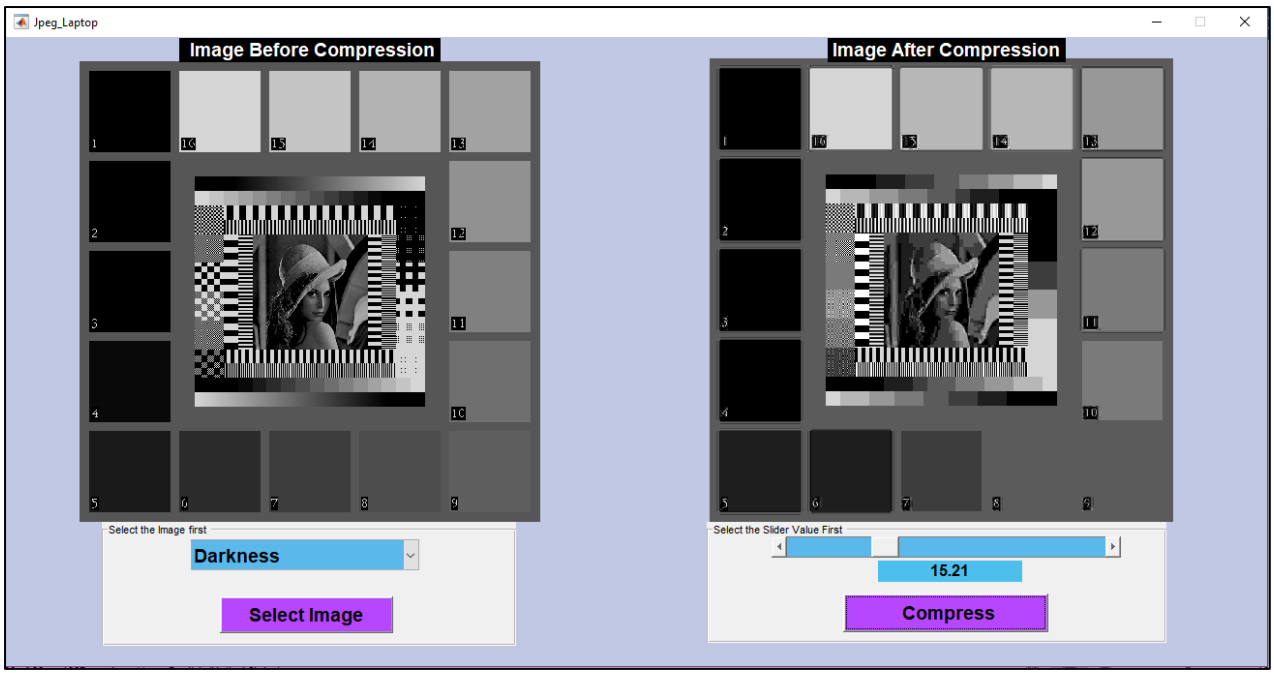
2. Darkness image is compressed **K= 1**



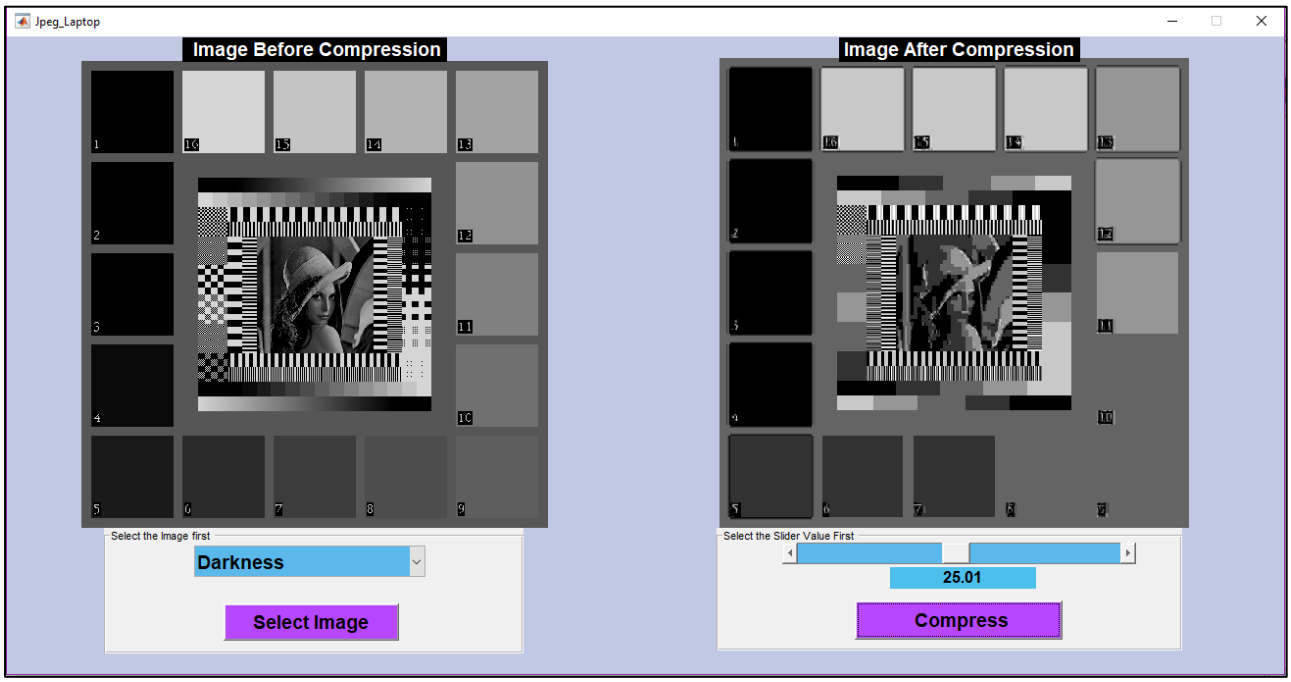
3. **K = 10.31**



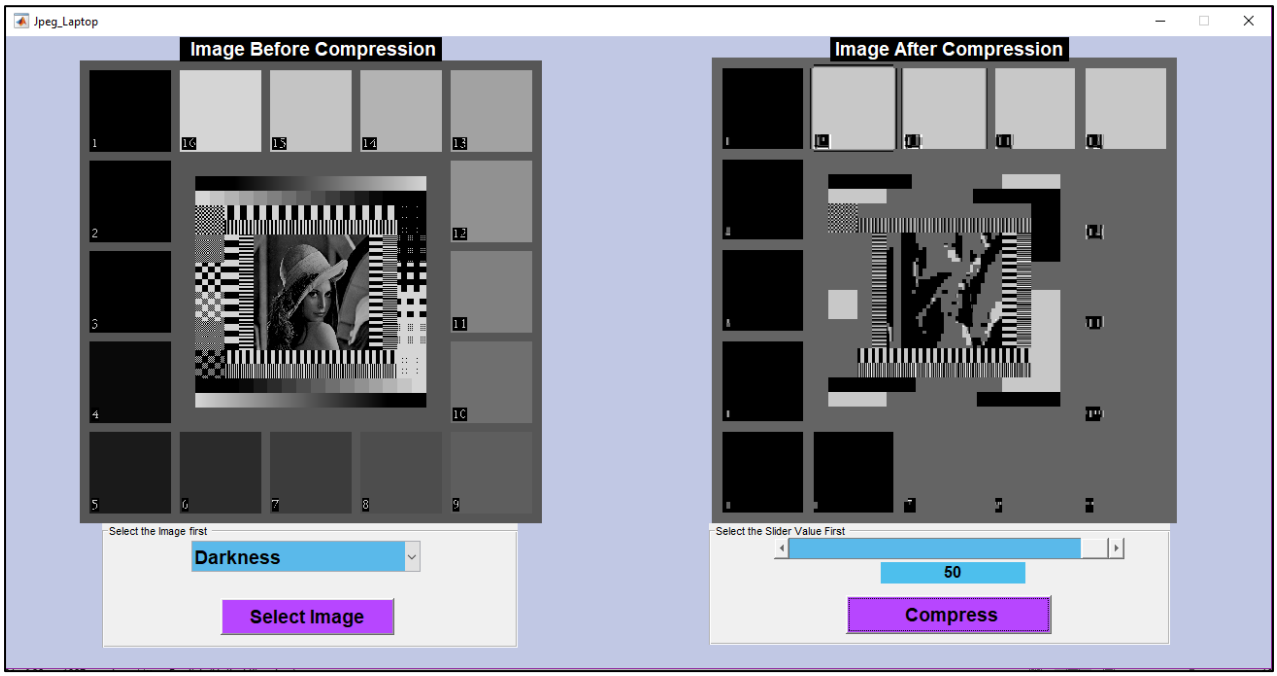
4. **K = 15.21**



5. **K = 25.01**

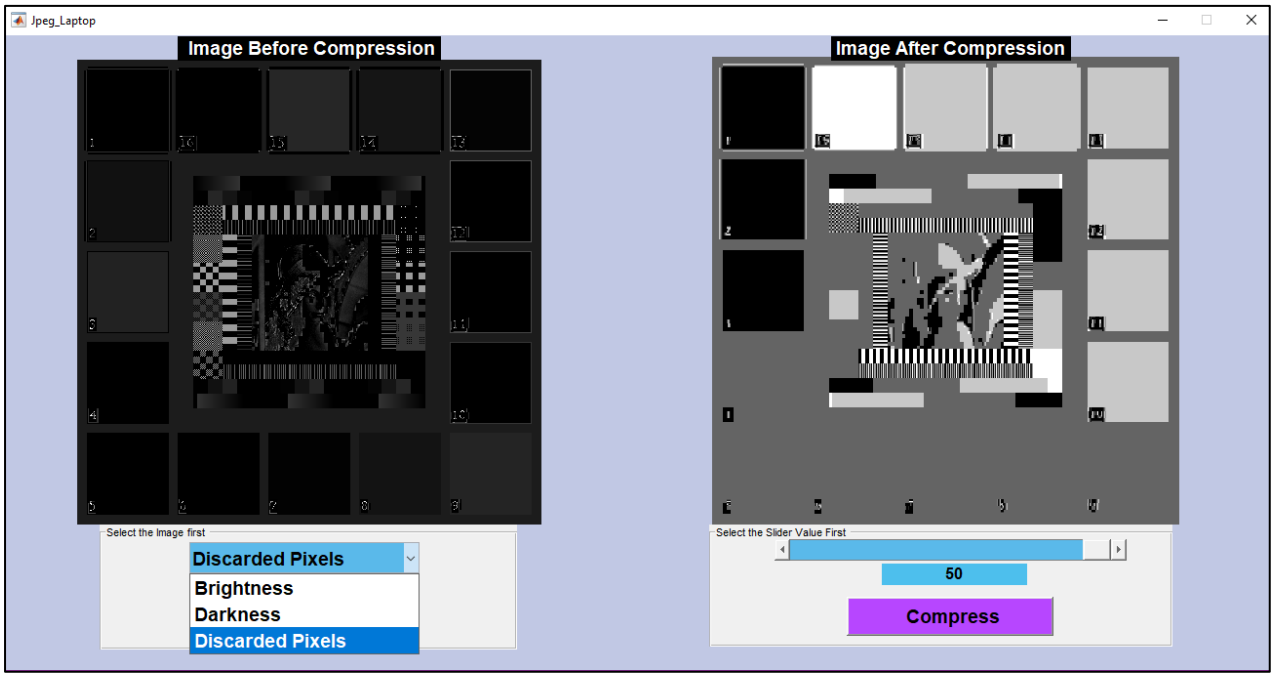

Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Darkness
Select Image
Select the Slider Value First
25.01
Compress

6. **K= 50**

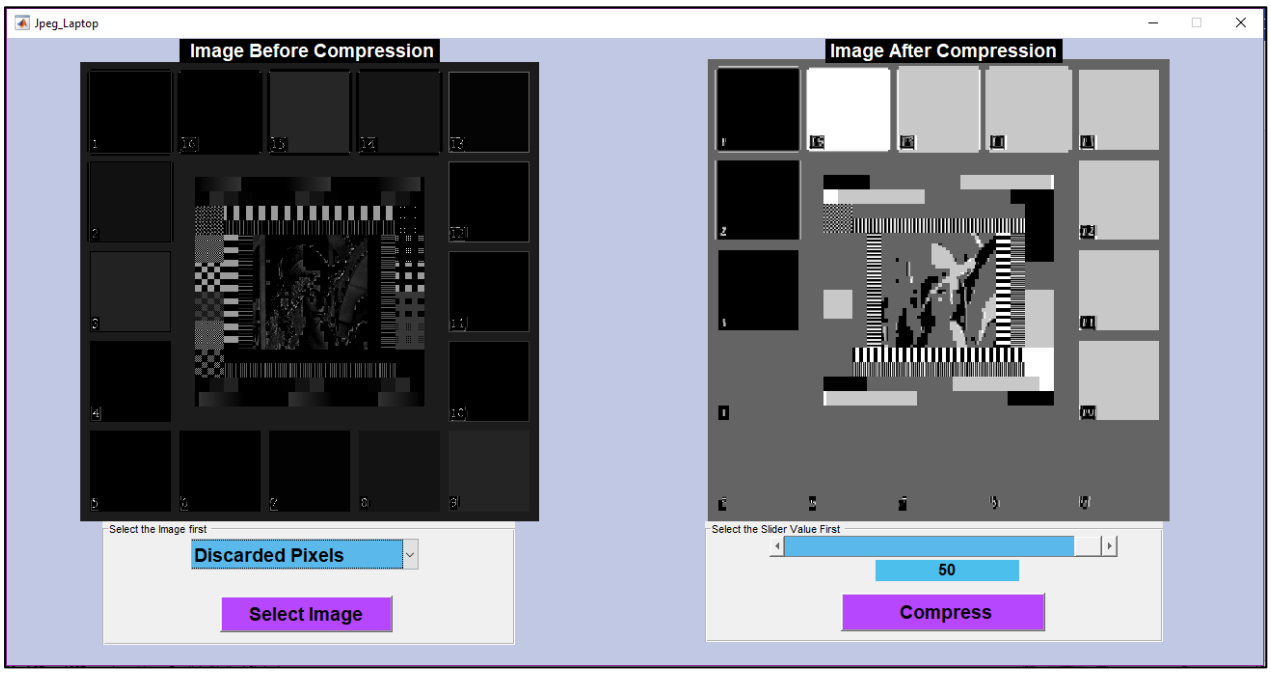

Jpeg_Laptop
Image Before Compression
Image After Compression
Select the Image first
Darkness
Select Image
Select the Slider Value First
50
Compress

- **Discarded Pixels:**

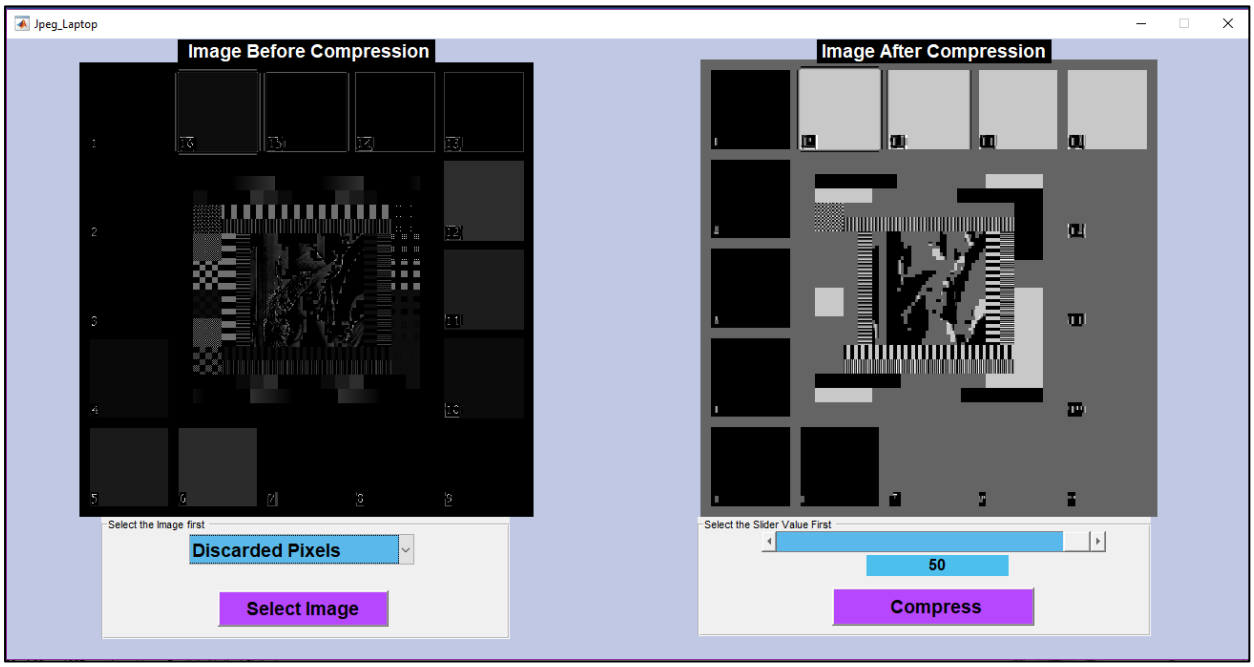
1. Select the **Original** Image and then click on Compress format at k= 50, once both images are stored, Select **Discarded Pixels** from Pop-up menu



2. Select the image click on **Brightness**, click on Compress format at k= 50, once both images are stored, Select **Discarded Pixels**.



3. Select the image click on **Darkness**, click on Compress format at k= 50, once both images are stored, Select **Discarded Pixels**.



Conclusion and Further work:

To conclude from above testing it is observed that as K increases the image distorts more and its pixel artifacts removal is more visible. But when the multiplying factor K is kept at 1 there is not much difference to human eye. Also, Brightness, Darkness Pixel has higher discarded pixels than same k with original image compression.

Tried incorporating Zoom and Crop function to view specific detail of image and its compress, I will work on this as a future scope.

Appendix

- **Q =** [16 11 10 16 24 40 51 61;
  12 12 14 19 26 58 60 55;
  14 13 16 24 40 57 69 56;
  14 17 22 29 51 87 80 62;
  18 22 37 56 68 109 103 77;
  24 35 55 64 81 104 113 92;
  49 64 78 87 103 121 120 101;
  72 92 95 98 112 100 103 99];

- **Code**

```
function varargout = Jpeg_Laptop(varargin)
% JPEG_LAPTOP MATLAB code for Jpeg_Laptop.fig
% This program Gives the Compressed output of given image using JPEG Lossy
% compression Technique.
%
%  JPEG_LAPTOP, by itself, creates a new JPEG_LAPTOP or raises the existing
%  singleton*.
%
%  H = JPEG_LAPTOP returns the handle to a new JPEG_LAPTOP or the handle to
%      the existing singleton*.
%
%   JPEG_LAPTOP('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in JPEG_LAPTOP.M with the given input
arguments.
%
%   JPEG_LAPTOP('Property','Value',...) creates a new JPEG_LAPTOP or raises
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before Jpeg_Laptop_OpeningFcn gets called.  An
%   unrecognized property name or invalid value makes property application
%   stop.  All inputs are passed to Jpeg_Laptop_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Jpeg_Laptop

% Last Modified by GUIDE v2.5 13-Oct-2018 15:22:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Jpeg_Laptop_OpeningFcn, ...
                   'gui_OutputFcn',  @Jpeg_Laptop_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Jpeg_Laptop is made visible.
function Jpeg_Laptop_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Jpeg_Laptop (see VARARGIN)


% Choose default command line output for Jpeg_Laptop
handles.output = hObject;

% Choose default command line output for myslider
set(handles.slider_k, 'Value', 1);                    %Initilaizing K=1
handles.slider_k = get(handles.slider_k,'Value');

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Jpeg_Laptop wait for user response (see UIRESUME)
% uiwait(handles.figure1);




% --- Outputs from this function are returned to the command line.
function varargout = Jpeg_Laptop_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider_k_Callback(hObject, eventdata, handles)
% hObject    handle to slider_k (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  slider_value = get(hObject, 'Value' ); % save the current slider value
  handles.slider_k = slider_value;       % make its scope global


  set(handles.edit_slider,'string',num2str(slider_value));% display in text
  guidata(hObject,handles)

% Hints: get(hObject,'Value') returns position of slider
%   get(hObject,'Min') and get(hObject,'Max') to determine range of slider
```

```
% --- Executes on button press in compress_pushbtn.
function compress_pushbtn_Callback(hObject, eventdata, handles)
% hObject    handle to compress_pushbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 origin = handles.image;                    %open Selected image
 slider_k = handles.slider_k ;              %Save slider value

if size(origin,3)==3 % Process data if its color image

  out_block = @(out) jpegmain(out.data , slider_k);
  % calls jpegmain function for every  8*8
  r = blockproc(origin(: , : , 1) ,[8 8], out_block);
  g = blockproc(origin(: , : , 2) ,[8 8], out_block);
  b = blockproc(origin(: , : , 3) ,[8 8], out_block);
  output = cat(3, r , g , b);
  axes(handles.axes_compress);
  imshow (output);


else % Process data if its grayscale image

  out_block = @(out) jpegmain(out.data , slider_k);
   % calls jpegmain function for every  8*8
  output = blockproc(origin ,[8,8], out_block);
  axes(handles.axes_compress);
  imshow (output);


end

handles.output = output;  %make the output value scope gobal
guidata(hObject,handles)

 function dctfun = jpegmain(out_blk, k)

 quant =[16  11  10  16  24  40  51  61; %Predefined Qunatization value
         12  12  14  19  26  58  60  55;
         14  13  16  24  40  57  69  56;
         14  17  22  29  51  87  80  62;
         18  22  37  56  68  109 103 77;
         24  35  55  64  81  104 113 92;
         49  64  78  87  103 121 120 101;
         72  92  95  98  112 100 103 99 ];

%DCT/IDCT lossy compression
 dctfun = uint8(idct2((round((dct2(out_blk))./(k.*quant))) .*(k.*quant)));



% --- Executes on selection change in select_PopupMenu.
function select_PopupMenu_Callback(hObject, eventdata, handles)
% hObject    handle to select_PopupMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = cellstr(get(hObject,'String')) returns select_PopupMenu
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
select_PopupMenu

% Determine the selected data set.
str = get(hObject, 'String');
val = get(hObject,'Value');

% Set current data to the selected data set.
switch str{val}

case 'Brightness' % User selects Brightness.
   img1 = handles.image;
   val = 15 * get(hObject,'Value') - 0.5 ; %get the current pixels then
+and* by predifined values
   img2 = img1 + val;
   axes(handles.axes_origin);
   imshow(img2);
   handles.image = img2;

case 'Darkness' % User selects Canny Image.
   img1 = handles.image;
   val = 15/get(hObject,'Value') -50 ; %get the current pixels then /and-
predifined values
   img2 = img1 + val;
   axes(handles.axes_origin);
   imshow(img2);
   handles.image = img2;

 case 'Discarded Pixels'
   original = handles.image;
   compress = handles.output;
   error = original- compress;  % Diffrence will show the removed pixels
   axes(handles.axes_origin);
   imshow(error);
   handles.image = error;

end
% Save the handles structure.
guidata(hObject,handles)



% --- Executes during object creation, after setting all properties.
function select_PopupMenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to select_PopupMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in origin_pushbtn.
function origin_pushbtn_Callback(hObject, eventdata, handles)
% hObject    handle to origin_pushbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[FileName , PathName] =
uigetfile('*.jpg;*.tif;*.png;*.bmp','FileSelector');
image = strcat( PathName ,FileName);  % Browse
axes(handles.axes_origin);            % Display it on 1st axes
imshow(image);

handles.image = imread(image);        % making its scope global
guidata(hObject,handles)              % saves the handles structure


function edit_slider_Callback(hObject, eventdata, handles)
% hObject    handle to edit_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit = get(hObject, 'string');                    %Takes the value of text
set(handles.slider_value,'string',str2num(edit));  %Displays the value of K
guidata(hObject,handles)

% Hints: get(hObject,'String') returns contents of edit_slider as text
%        str2double(get(hObject,'String')) returns contents of edit_slider
as a double


% --- Executes during object creation, after setting all properties.
function edit_slider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```