Chandigarh College of Engg.& Tech.

CCET

CS-701

DIGITAL IMAGE PROCESSING

ASSIGNMENT ON

WEB APP FOR IMAGE COMPRESSION USING DCT ALGORITHM

SUBMITTED TO:

Dr. Gulshan Goyal

CSE Department

SUBMITTED BY:

Vivsvaan Sharma

CO16362

CSE 7th Sem

Objective

This project aims to develop a GUI to demonstrate how the DCT can be used for image

compression, implementing a simple JPEG-based algorithm. The App has been developed using Python and the Django Web Framework.

Introduction

Image Compression – Image is stored or transmitted with having pixel value. It can be compressed by reducing the value its every pixel contains. Image compression is basically of two types:

1. Lossless compression – In this type of compression, after recovering image is exactly become same as that was before applying compression techniques and so, its quality didn't get reduced.

2. Lossy compression – In this type of compression, after recovering we can't get exactly as older data and that's why the quality of image gets significantly reduced. But this type of compression results in very high compression of image data and is very useful in transmitting image over network.

Discrete Cosine Transform is used in lossy image compression because it has very strong energy compaction, i.e., its large amount of information is stored in very low frequency component of a signal and rest other frequency having very small data which can be stored by using very less number of bits (usually, at most 2 or 3 bit).

To perform DCT Transformation on an image, first we have to fetch image file information (pixel value in term of integer having range 0 – 255) which we divides in block of 8 X 8 matrix and then we apply discrete cosine transform on that block of data.

The dct2 function computes the two-dimensional discrete cosine transform (DCT) of an image. The DCT has the property that, for a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, the DCT is often used in image compression applications. For example, the DCT is at the heart of the international standard lossy image compression algorithm known as JPEG. (The name comes from the working group that developed the standard: the Joint Photographic Experts Group).

The App works both with color and grayscale images. The former are converted in grayscale applying the standard RGB conversion formula. While the algorithm is designed to operate with .bmp images, our Python implementation also works properly providing .jpg images.

Input Arguments Meaning

As previously explained, the algorithm takes two integers as inputs:

- F determines how big are the blocks into which the image is split; the higher this value, the faster the algorithm but the lossier is the compression.
- d must have a value between 0 and 2F -1 and it determines how many frequencies will be cut out. the lower this value, the more aggressive the compression. It doesn't affect time performances.

Examples

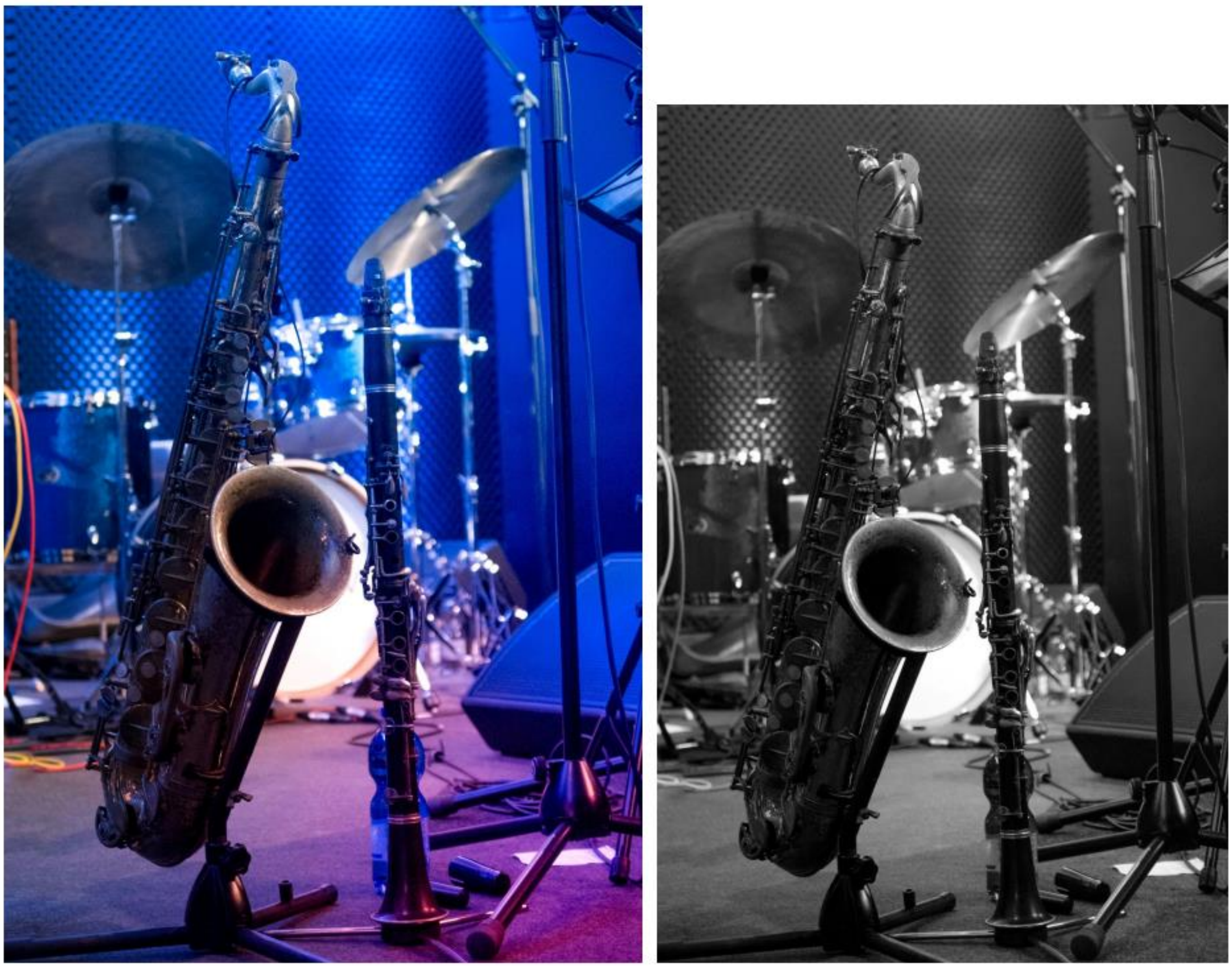


Figure 1 – f/2, 1/50 sec, ISO 1000, 1365x2048 pixels

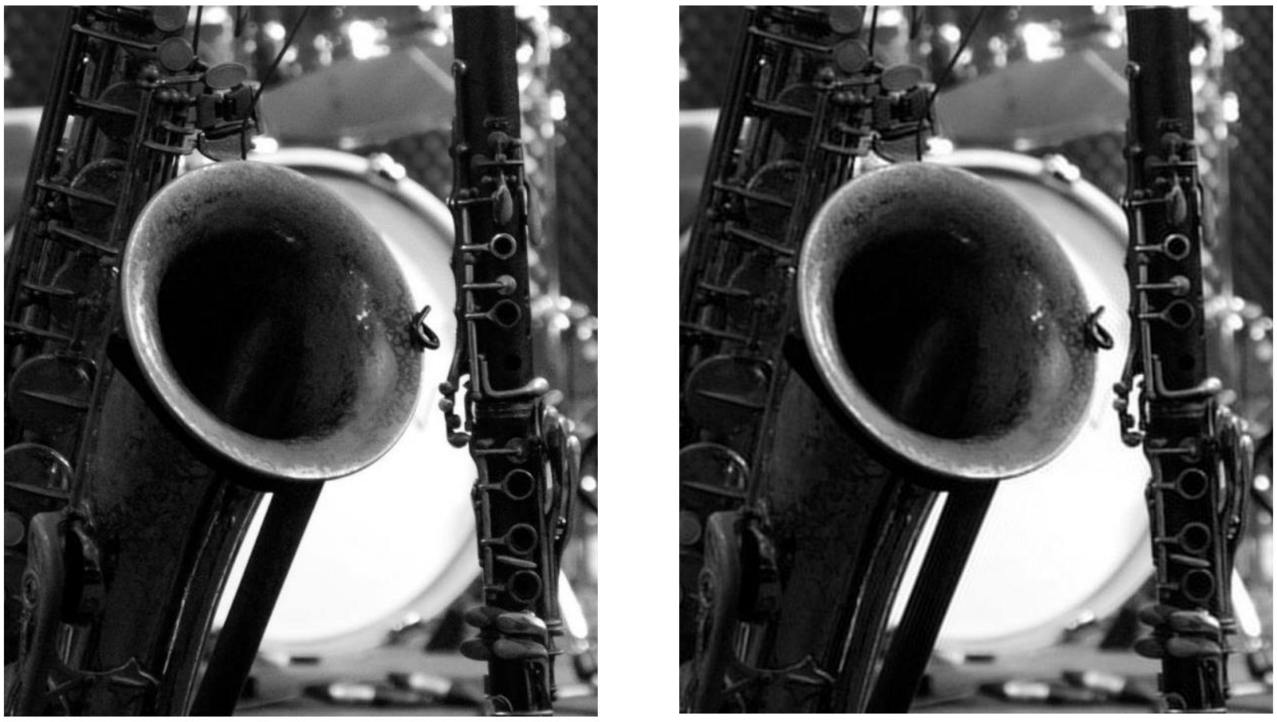


Figure 2 – Original grey-scaled (Left) and Compressed (Right) F = 100, d = 50

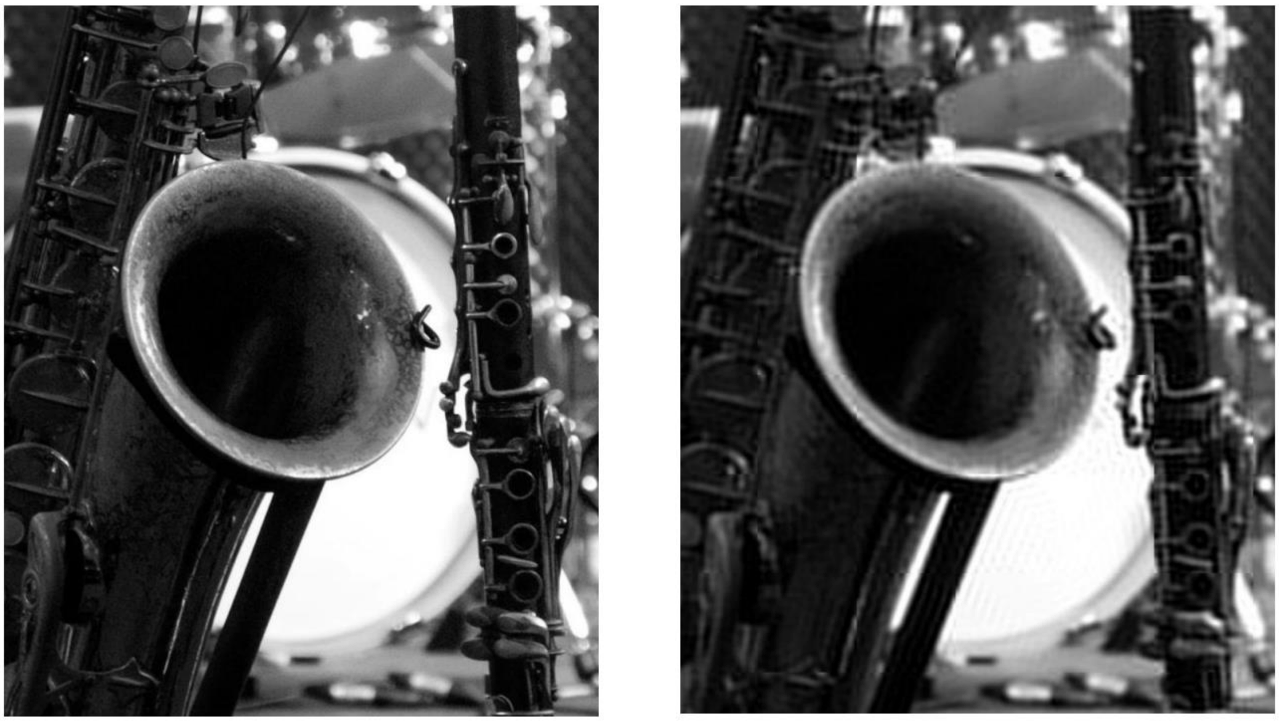


Figure 3 – Original grey-scaled (Left) and Compressed (Right) F = 100, d = 25

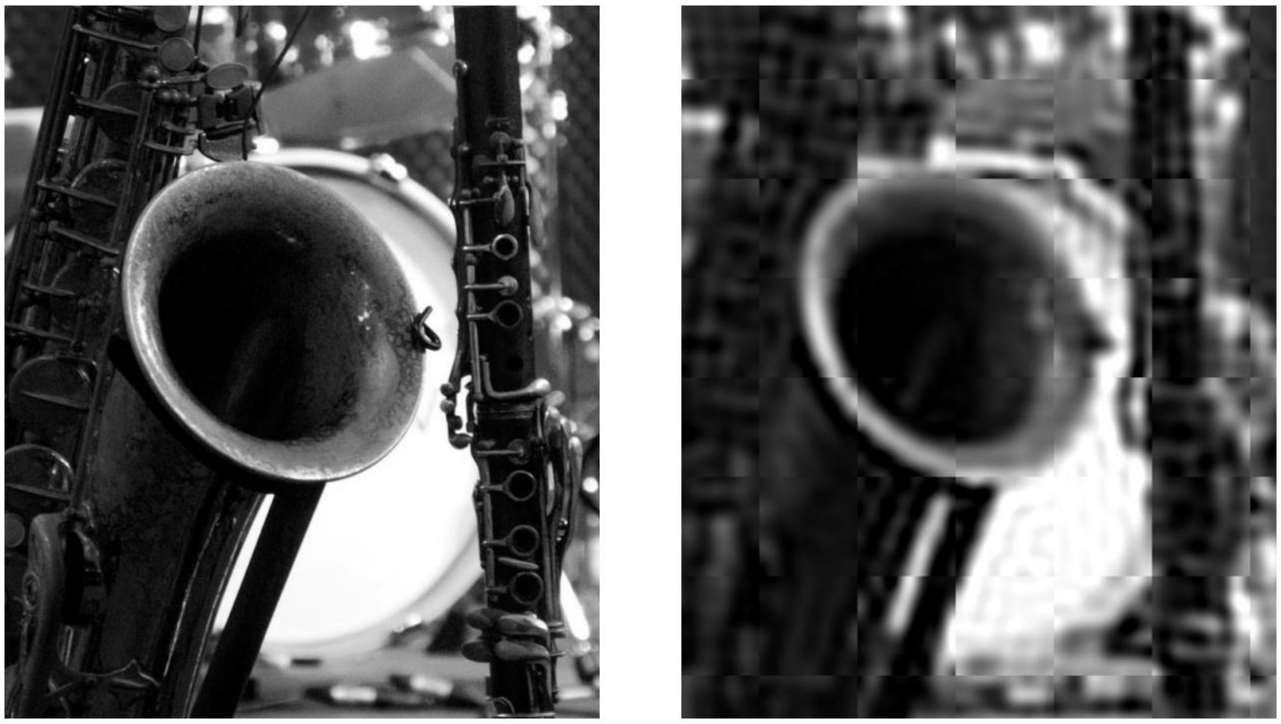


Figure 4 – Original grey-scaled (Left) and Compressed (Right) F = 100, d = 10

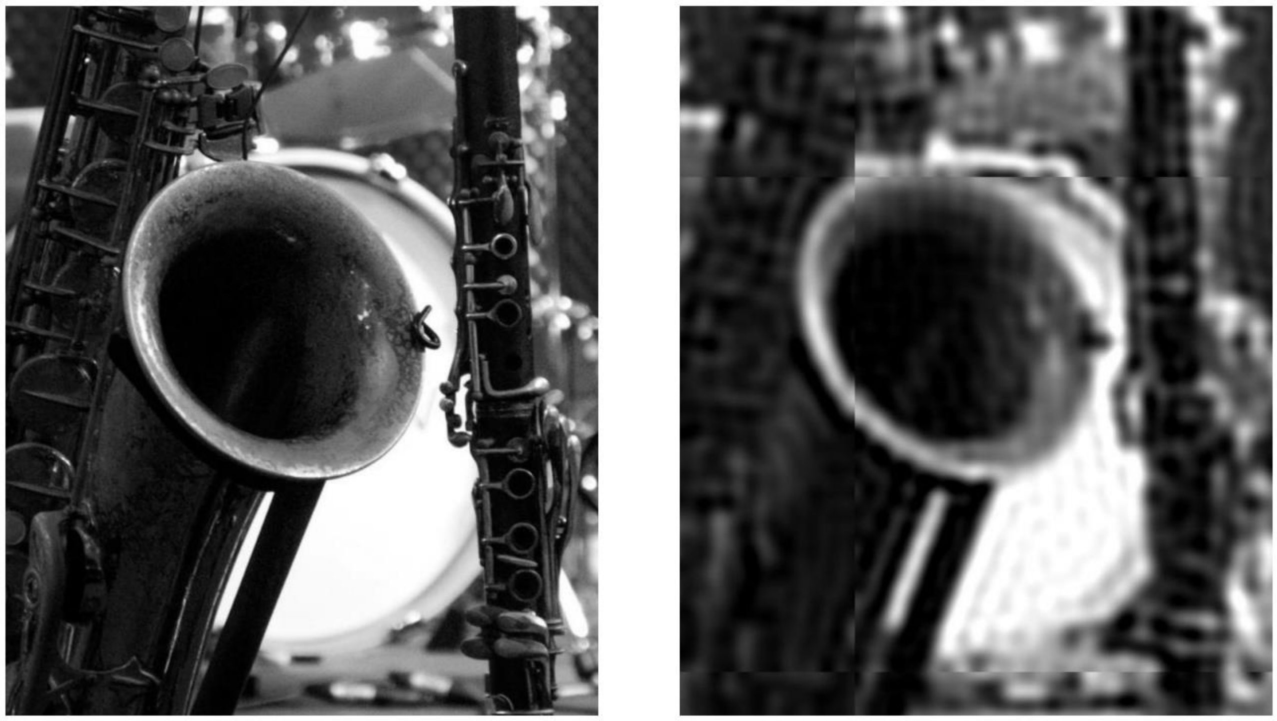


Figure 5 – Original grey-scaled (Left) and Compressed (Right) F = 500, d = 50

Algorithm – DCT Image Compression

```
INPUT –   Takes an image and two integers (F and d)
OUTPUT –  Rebuilds the image after performing following steps
Step 1 –  splits the image into FxF pixel blocks
Step 2 –  for each block
Step 3 –  applies the DCT2
Step 4 –  filters frequencies where row+col >= d
Step 5 –  applies the inverse DCT2
Step 6 –  normalizes obtained values
Step 7 –  rebuilds the image as output
```

Source Code

Utils.py

```
def compress_image(user, image, F, d):

    # paths
    imagename = randomString(10)
    image_folder = os.path.join('media', user, imagename)

    if not os.path.exists(image_folder):
        os.makedirs(image_folder)

    image_path = os.path.join(image_folder, '1-original.jpg')
    image_grey_path = os.path.join(image_folder, '2-grey.jpg')
    image_compress_path = os.path.join(
        image_folder, f'3-compress-F{F}_d{d}.jpg')

    # input image
    img = imageio.imread(image)
    if img.ndim == 3:  # colored images
        # bnimg = img[:, :, 0]
        bnimg = 0.2989 * img[:, :, 0] + 0.5870 * img[:, :, 1] + 0.1140 * 
img[:, :, 2] # RGB to grey
    else:  # grey images
        bnimg = img

    imageio.imwrite(image_path, img)  # original image
    imageio.imwrite(image_grey_path, bnimg)  # black & white image

    (rows, cols) = bnimg.shape

    # splitting bnimg in FxF blocks
    for i in range(0, int(rows / F)):
        for j in range(0, int(cols / F)):

            rowsLower = i * F
            colsLower = j * F
            rowsUpper = (i+1) * F
            colsUpper = (j+1) * F
            block = bnimg[rowsLower:rowsUpper, colsLower:colsUpper]

            # c = DCT2(f)
            c = dctn(block, type=2, norm='ortho')

            # filtering frequences to the right of d-diagonal
            (blockRows, blockCols) = block.shape
            for k in range(0, blockRows - 1):
                for l in range(0, blockCols - 1):
                    if(k + l >= d):
                        c[k, l] = 0
            # ff = IDCT2(c)
            ff = idctn(c, type=2, norm='ortho')

            # normalizing idct
            ff = np.round(ff)
            for index, value in np.ndenumerate(ff):
                if value < 0:
                    ff[index] = 0
                elif value > 255:
                    ff[index] = 255

            bnimg[rowsLower:rowsUpper, colsLower:colsUpper] = ff

    imageio.imwrite(image_compress_path, bnimg)  # compress image
    # return image_path, image_compress_path

def randomString(stringLength=10):
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(stringLength))
```

views.py

```
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import render, redirect
from django.urls import reverse
from django.http import JsonResponse

import os
import shutil
import webui.utils as utils

# Create your views here.
```

```
def index(request, image=None, image_compress=None):
    return render(request, 'index.html')


def compress(request):
    try:
        # Trying to get POST data
        image = request.FILES['image']
        F = int(request.POST['F'])
        d = int(request.POST['d'])
        user = request.POST['user']
    except (KeyError):
        # Redisplay the form in case of error
        return render(request, 'index.html', {
            'image': image,
            'F': F,
            'd': d,
            'error_message': "You missed something.",
        })
    else:
        utils.compress_image(user, image, F, d)
        return redirect('index')


def get_user_images(request):
    user = request.GET['user']
    userpath = os.path.join('media', user)
    image_paths = []

    if os.path.exists(userpath):
        for directory in os.listdir(userpath):
            for root, dirs, files in os.walk(os.path.join(userpath, direc
tory)):
                original_image_path = utils.get_first_matching_image(root
, files, 'original')
```

```
                grey_image_path = utils.get_first_matching_image(root, fi
les, 'grey')
                compress_image_path = utils.get_first_matching_image(root
, files, 'compress')

                image_paths.append(utils.ImageDTO(directory, original_ima
ge_path, grey_image_path, compress_image_path).__dict__)

    return JsonResponse({'images': image_paths})



def delete_image(request, user, image):
    imagefolder = os.path.join('media', user, image)
    if os.path.exists(imagefolder):
        shutil.rmtree(imagefolder)

    return redirect('index')

def delete_all_for_user(request, user):
    imagefolder = os.path.join('media', user)
    if os.path.exists(imagefolder):
        shutil.rmtree(imagefolder)

    return redirect('index')
```

urls.py

```
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('compress', views.compress, name='compress'),
    path('get_user_images', views.get_user_images, name='get_user_images'
),
    path('delete_image/<slug:user>/<slug:image>', views.delete_image, nam
e='delete_image'),
    path('delete_all_for_user/<slug:user>', views.delete_all_for_user, na
me='delete_all_for_user'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

base.html

```
<!DOCTYPE html>

<html>

<head>
    <title>DCT Image Compression</title>

    <!-- CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boots
trap/4.1.3/css/bootstrap.min.css"
        integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossor
igin="anonymous">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.
8.2/css/all.css"
        integrity="sha384-
oS3vJWv+0UjzBfQzYUhtDYW+Pj2yciDJxpsK1OYPAYjqT085Qq/1cq5FLXAZQ7Ay" crossor
igin="anonymous">

    <!-- Scripts -->
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/
umd/popper.min.js"
        integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossor
igin="anonymous">
    </script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bo
otstrap.min.js"
```

```
        integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossor
igin="anonymous">
    </script>
</head>


<body>

    <!-- Menu -->
    <nav class="navbar navbar-expand-md navbar-dark bg-dark">
        <a class="navbar-brand" href="/">DCT Image compression</a>
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarCollapse"
            aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Home</a>
                </li>
            </ul>
        </div>
    </nav>

    <!-- Content -->
    <div class="container" style="margin-top: 30px;">
        {% block content %}{% endblock %}

        <footer style="margin-bottom: 30px;">
            <hr>
            Marco Ferri (807130) - Nassim Habbash (808292) / <a href="htt
ps://github.com/mferri17/dct-image-
compression" target="_blank">View source code</a>
        </footer>
    </div>

</body>

</html>
```

index.html

```
{% extends "base.html" %}
{% load staticfiles %}

{% block content %}

<h2>Upload an image to compress</h2>
<hr>

{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif
 %}

<!-- --------------------- UPLOAD FORM --------------------- -->

<form id="upload" class="row" action="{% url 'compress' %}" method="post"
 enctype="multipart/form-data">
    {% csrf_token %}
    <div class="col-md-6">
        <input type="hidden" name="user" />
        <div class="form-group row">
            <div class="col">
                <label for="image">Choose image</label>
                <input type="file" class="form-control form-control-
file" accept="image/jpg, image/jpeg, .bmp"
                    name="image" id="image" required style="padding-
bottom: 36px;" />
            </div>
        </div>
        <div class="form-group row">
            <div class="col">
                <input type="number" class="form-
control" name="F" placeholder="F = blocks size" required />
            </div>
            <div class="col">
                <input type="number" class="form-
control" name="d" placeholder="d = frequency treshold" required />
            </div>
        </div>
        <input class="btn btn-primary" type="submit" value="Submit" />
    </div>
    <div class="col-md-6">
```

```
        <img id="loading" src="https://loading.io/spinners/cutiefox/lg.cu
tie-fox-spinner.gif" width="180" style="display:none;" />
    </div>
    <br />
    <hr>
</form>

<!-- --------------------- HISTORY TABLE ---------------------- -->

<style>
    #user-images td {
        text-align: center;
    }

    #user-images img {
        max-width: 300px;
        max-height: 280px;
    }
</style>

<div class="col-md-4" style="margin:30px 0 150px 0;">
    <h3>
        Your images
        <a id="delete-all" class="btn btn-sm btn-
danger" role="button" href="#"
            onclick="return confirm('Are you sure?');">Delete all</a>
    </h3>
    <table id="user-images" class="table table-bordered table-striped">
        <thead>
            <tr>
                <th>Original</th>
                <th style="white-space:nowrap;">Greyscale</th>
                <th>Compress</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
        </tbody>
    </table>
    <i>Click an image to see it bigger</i>
</div>

<script>
```

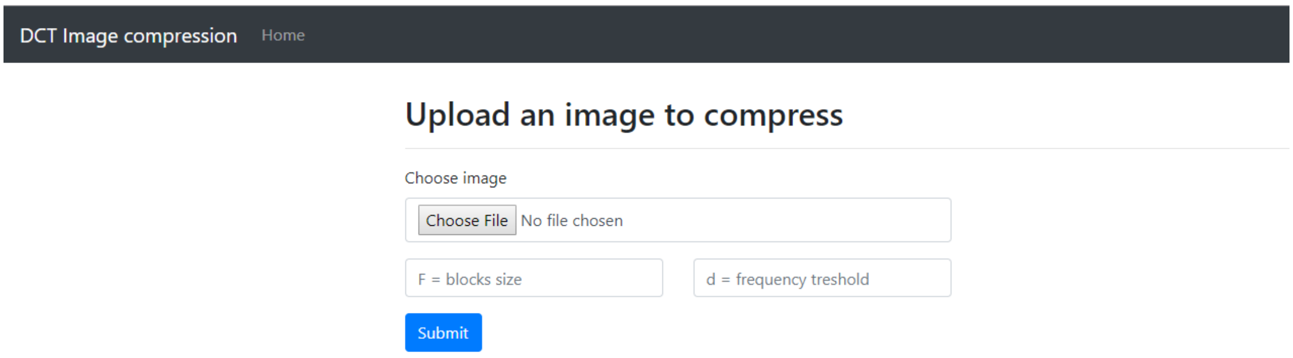
```
    $(document).ready(function () {
        var user = setUser();
        getUserImages(user);
        $('#delete-all').attr('href', '/delete_all_for_user/' + user);

        $('form#upload').submit(function (event) {
            //event.preventDefault();
            var F = $(event.target).find('[name=F]').val();
            var d = $(event.target).find('[name=d]').val();
            if (d < 0 || d > 2 * F - 2) {
                alert('Frequency treshold (d) must be between 0 and 2F-
2');
                return false;
            }
            else{
                $('#loading').show();
                $('[type=submit]').attr('disabled', true);
            }
        });
    });

    function setUser() {
        var user = window.localStorage.getItem('user');
        if (!user) {
            user = randomString();
            window.localStorage.setItem('user', user);
        }
        $('[name=user]').val(user);
        return user;
    }

    function getUserImages(user) {
        $.ajax({
                method: "GET",
                url: "/get_user_images",
                data: {
                    user: user,
                }
            })
            .done(function (response) {
                $(response.images).each(function (index, element) {
                    var original = '<td><a href="' + element.original + '
" target="_blank"><img src="' +
                              element.original + '"><a/></td>';
                         var grey = '<td><a href="' + element.grey + '" target
="_blank"><img src="' + element
                              .grey + '"><a/></td>';
                         var compress = '<td><a href="' + element.compress + '
" target="_blank"><img src="' +
                              element.compress + '"><a/></td>';
                         var remove = '<td><a href="/delete_image/' + user + '
/' + element.name +
                              '" class="fa fa-trash text-danger"></a></td>';

                         var tr = $('<tr>').append(original).append(grey).appe
nd(compress).append(remove);
                         $('#user-images tbody').append(tr);
                    });
               });
     }

     function randomString() {
          // https://stackoverflow.com/questions/1349404/generate-random-
string-characters-in-javascript
          return Math.random().toString(36).substring(7);
     }
</script>

{% endblock %}
```

User Interface

Upload an image by choosing the image file.

Set the value of blocks (i.e. F) and frequency threshold (i.e. d)

Click Submit Button



Upload an image to compress

Choose image

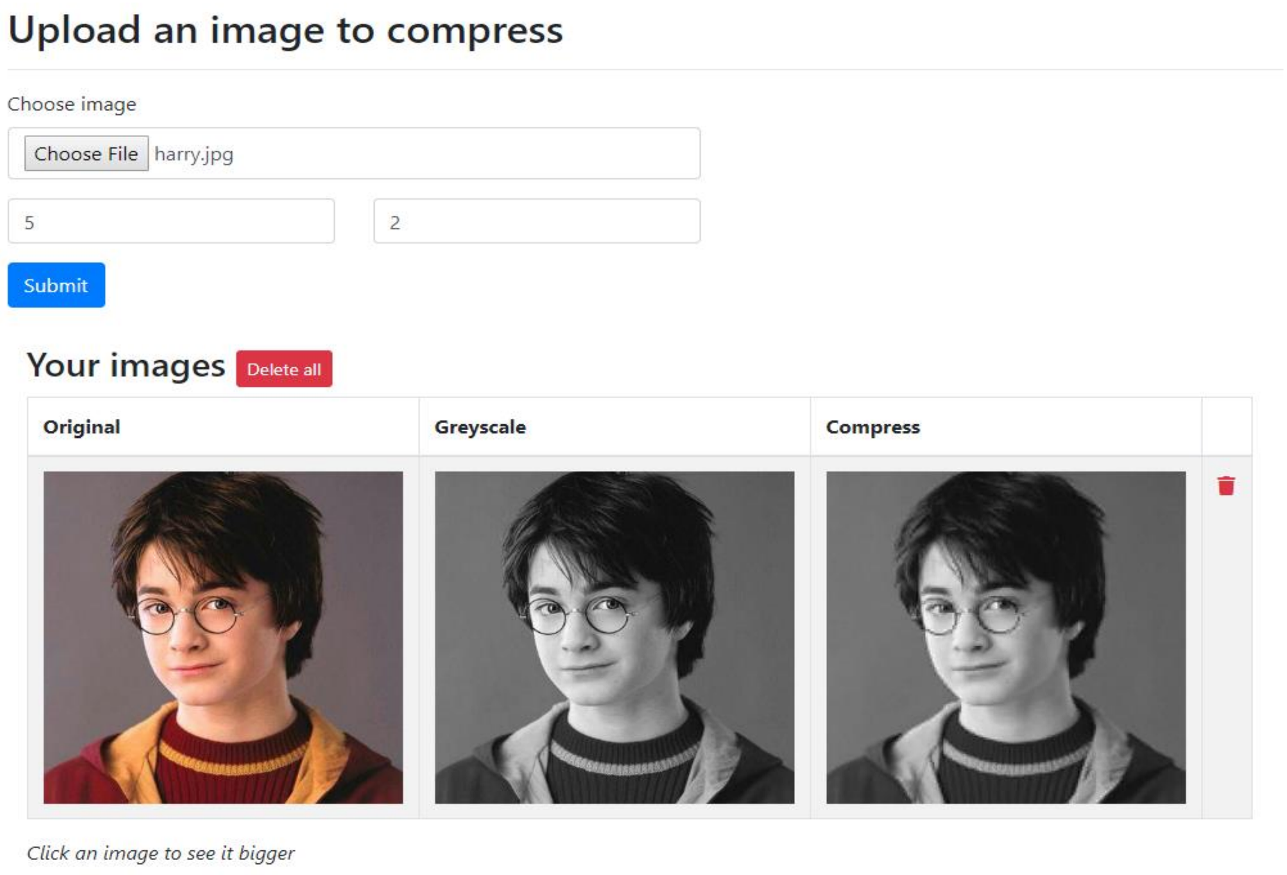
No file chosen

F = blocks size

d = frequency treshold

Figure 1 – Uploading the image

After the compression, the original image, grey-scaled image and the compressed image are displayed



Upload an image to compress

Choose image

harry.jpg

5

2

Your images

Original	Greyscale	Compress	

Click an image to see it bigger

Figure 2 – Original, grey-scaled and compressed image