Lab Assignment #3
Due Monday, October 17 by 11:59pm
(upload as a single PDF to CatCourses)

Title: Image resizing
In this lab, you will implement a function that resizes a grayscale image. It will perform either nearest neighbor or bilinear interpolation to generate the resized image. Resizing an image to a smaller size is called downsampling. Resizing an image to a larger size is called upsampling.

Read through this assignment in detail before starting.

The lab consists of four experiments:

1. Down then upsample an image using *nearest neighbor* interpolation. Compare the original image to the down then upsampled one by computing root mean square error (RMSE).
2. Down then upsample an image using *bilinear* interpolation. Compare the original image to the down then upsampled one using RMSE.
3. Up then downsample an image using *nearest neighbor* interpolation. Compare the original image to the up then downsampled one using RMSE.
4. Up then downsample an image using *bilinear* interpolation. Compare the original image to the up then downsampled one using RMSE.

See the Python script `test_myimresize.py` which performs these four experiments. You can use this script. Your task in this lab is to write the Python functions `myImageResize()`, `mybilinear()`, and `myRMSE()`.

Read through the Python script `test_myimresize.py` to see how it:

- Reads the image you will be resizing: `Lab_03_image.tif`. This is the original image.
- Creates a numpy matrix from the original image so we can access its pixels. NOTE THAT WE WE ARE CREATING A FLOAT32 ARRAY SINCE WE WILL BE DOING FLOATING POINT OPERATIONS IN THIS LAB.
- Creates a downsampled numpy matrix using nearest neighbor interpolation. (This requires calling the function `myImageResize()` which you will write.)
- Upsample this downsampled matrix using nearest neighbor interpolation.
- Creates an image from this matrix to display and save.
- Computes the RMSE between the original matrix and the downs then upnsampled one. This completes experiment 1.
- Repeats the above down then upsampling using bilinear interpolation. This is experiment 2.
- Performs experiment 3 which is up then downsampling using nearest neighbor interpolation.
- Performs experiment 4 which is up then downsampling using biliner interpolation.

Create the Python script `MyImageFunctions.py` and write the following functions:

1. Write the function `myimresize`() that takes as input

    - A numpy matrix representing a grayscale image.
    - The number of rows in the resized image.
    - The number of columns in the resized image.
    - A string with values 'nearest' or 'bilinear'.

    and outputs the resized numpy matrix image. This function should use either nearest neighbor or bilinear interpolation to determine the values in the output matrix. This function will need to:

- Determine the dimensions of the input numpy matrix.
- Create a numpy matrix for the resized image.
- For each pixel in output matrix, find corresponding location in input matrix. (See problem 5 from HW 1.)
- Use interpolation (nearest neighbor or bilinear) to determine the value at this location.

2. The nearest neighbor interpolation can be performed inside the `myimresize()` function. The bilinear interpolation should be performed by the function `mybilinear()` which you must write (inside the Python script `MyImageFunctions.py` and which should be called by your `myimresize()` function). This function takes as input:

    - Four pixels locations (these should be integer values)
    - The pixel values at those locations
    - The location of the interpolated pixel (this need not be integer valued)

    and provides, as output, the value of the interpolated pixel. You can implement either of the two approaches to bilinear interpolation that we covered in lecture.

3. Write the function `RMSE()` (inside the Python script `MyImageFunctions.py`) which takes as input:

    - Two numpy matrices representing grayscale images.

    and outputs:

    - The RMSE between the two matrices.

    For two matrices of size MxN, the RMSE can be computed as:

$$RMSE = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I1(m,n) - I2(m,n)\right)^2} \ .$$

RMSE is a numeric method for computing the difference between two matrices. It is commonly used to evaluate the effectiveness of image reconstruction algorithms by computing the pixelwise difference between the original and the reconstructed images

See the provided Python script `test_myimresize.py` for examples of how the functions `myImageResize()`, `mybilinear()`, and `myRMSE()` are called. Again, this assignment requires that you write these three functions.

Once you have written the three functions, you should be able to run the provided Python script `test_myimresize.py`. Here is the terminal output that I get when I run it:

```
Downsample/upsample with myimresize using nearest neighbor interpolation = 22.746414
Downsample/upsample with myimresize using bilinear interpolation = 16.839830

Upsample/downsample with myimresize using nearest neighbor interpolation = 0.000000
Upsample/downsample with myimresize using bilinear interpolation = 5.414557
```

These are the RMSE values.

Some notes:

- You might want to focus on getting your image resizing function (including the computation of the RMSE) to work using nearest neighbor interpolation before considering the more challenging bilinear interpolation case.

- Unlike in lab 02, make sure to create numpy matrices of type float32 from images. For example:
    - `orig_im_pixels = asarray(orig_im, dtype=np.float32)`

  (The only place I had to create a numpy matrix from an image is in the provided Python script `test_myimresize.py.)`
- Note that Python doesn't allow you to index into arrays using floating point numbers even if they don't have a decimal component. For example, if `x=1.0`, you will get an error if you write `inImage_pixels[x][y].` You need to typecast the float to an integer before using it to index: `x1 = int(x).`
- Provide comments for key lines of code.
- Provide headers for your functions. Headers are comments which appear right after the function declaration that summarize what the function does its calling syntax, the input and output parameters, and the function history. You must provide headers for the three functions you will write: `myImageResize()`, `mybilinear()`, and `myRMSE()`.

Questions:

1. Visually compare the two downsampled images, one using nearest neighbor interpolation and one using bilinear interpolation. How are they different and why based on what you know about the two interpolations?
2. Visually compare the two down then upsampled images, one using nearest neighbor interpolation and one using bilinear interpolation. How are they different? Which one looks better to you? Does this agree with the RMSE values?
3. Visually compare the two up then downsampled images, one using nearest neighbor interpolation and one using bilinear interpolation. How are they different? Which one looks better to you? Does this agree with the RMSE values?
4. If your image resizing is implemented correctly, you should get an RMSE value of zero between the original image and the up then downsampled one using nearest neighbor interpolation. Why is this the case?
5. What was the most difficult part of this assignment?

What to submit:

A lab report as a single PDF containing:

- Lab number and title, your name, your lab section, your TA's name, and the date.
- Abstract: In a few sentences, describe the purpose of this lab.
- Qualitative results: Include figures with captions of the following four resized versions of Lab_03_image.tif:
    - Downsampled to size (100, 175) using nearest neighbor interpolation.
    - Downsampled to size (100, 175) using bilinear interpolation.
    - Upsampled to size (500, 625) using nearest neighbor interpolation.
    - Downsampled to size (500, 625) using bilinear interpolation.
- Quantitative Results: A table showing the RMSE values between the original image and the down/upsampled and up/downsampled versions for both nearest neighbor and bilinear interpolation.
- Questions: Your answers to the five questions above.
- The Python script: `test_myimresize.py`. (This could be exactly the same as the one provided or one with some modifications.)
- The Python file `MyImageFunctions.py` which contains your three functions: `myImageResize()`, `mybilinear()`, and `myRMSE()`.

See Lab_03_sample_report.pdf as an example.

What your report will be graded on

- Whether you included an abstract.
- Whether your report includes the images you were asked to include and they have appropriate captions.
- Whether you included a table with the RMSE values.
- Whether you answered assignment questions.
- Whether your report includes the code you were asked to include.
- Whether your code is correct.
- Whether your code is commented and your functions have headers.