Image Processing – 67829 – Coding Bootcamp 1

Version 1.0 - Last Update 16.12.2023

Welcome to the Image Processing Coding Bootcamp!

In this bootcamp, we aim to equip you with fundamental coding skills essential for image processing tasks. The challenges presented here are designed to be short and practical, serving as building blocks to enhance your proficiency with libraries such as numpy, PIL (Pillow), and scikit-image (skimage). You are expected to search online for answers to these problems, go over the documentation, explore, and see what tricks others use. Practice is the best way to learn these tools.

Note1: This bootcamp is not intended for formal submission, and it does not carry a grading component. Consider it a playground for you to experiment, learn, and strengthen your coding skills. The exercises are optional but highly recommended for those seeking a hands-on approach to mastering the foundational aspects of image processing.

Note2: You may need to install these tools on your local environment, simply google how to install them, it shouldn't take longer than a few seconds to find the answer.

Note3: Throughout this bootcamp you should strive to use as little loops as possible, try to perform all the operations using numpy in a vectorized fashion.

1 NumPy

NumPy is a fundamental library in the python ecosystem for numerical and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. The main philosophy behind numpy is centered on vector-oriented computing, enabling users to express complex operations on vectors and matrices without the need for explicit looping constructs. This not only simplifies code but also significantly enhances performance by leveraging optimized, low-level implementations. numpy forms the foundation for many other scientific computing libraries in python and serves as a cornerstone for tasks such as image processing, data analysis, and machine learning.

Complete the following tasks:

1. Create a numpy array of shape $[9, 1]$ and fill it with random values.
2. Turn it into a 3×3 matrix.
3. Print the shape and type of the matrix.
4. Get the first column of the matrix and multiply it by the third column of the matrix.
5. Create a new matrix of size $[9, 9]$ and fill it with the value from the previous section.
6. Create a new matrix of size $[9, 9]$ and fill it with random values.
7. Stack the two matrices along a new dimension to form a tensor of shape $[9, 9, 2]$.
8. Get the mean, sum, and standard deviation of the new matrices along the third dimension.
9. Get all the indices of the array for which the value is greater than the mean.
10. Replaces the values of items in these indices by 0.

Now, let's benchmark vectorized numpy operations and see that they are actually faster.

1. Create two random tensors of shape $512 \times 512 \times 3$
2. Multiply the two matrices, perform the multiplication once using numpy operations and once using a regular python loop.
3. Repeat this 1000 times (this can be in a loop)
4. Time each of the multiplication methods to see what is faster

2 Images

There are many python libraries for working with images. Some of the most popular ones are PIL, skimage, and OpenCV. While OpenCV is powerful, it is often recommended to use PIL or skimage for basic image manipulation tasks due to their simplicity and ease of integration with other scientific libraries.

One notable consideration when using OpenCV is its use of the BGR (Blue, Green, Red) color convention, as opposed to the more common RGB (Red, Green, Blue) convention. Users need to be aware of this difference when working with image data to avoid color-related issues. In addition to image processing libraries, there are many plotting libraries. One of the most famous plotting libraries is called matplotlib. **Note:** Although saving images via matplotlib is possible, you should not use it to save actual images but only for figures. When saving images via matplotlib the quality of the images are degraded, this is usually fine for plotting, but not when you need the real values of the images.

Complete the following tasks (Try to complete these tasks once with PIL and once with skimage):

1. Read an image from your local file system into a numpy array.
2. Print the dimensions and type of the image (i.e. the numpy array).
3. Display the RGB image using matplotlib.
4. Transform the image into a grayscale image.
5. Print the dimensions and type of the grayscale image (i.e. the numpy array).
6. Display the grayscale image using matplotlib (Note: make sure the displayed image looks grayscale!).
7. Save the grayscale image to the disk (Note: use the image processing library and not matplotlib, matplotlib will reduce the image quality, use it for plotting but avoid using it for saving actual images).
8. Using the original, RGB image, transform the range of values from [0, 255] with type uint8 to [0, 1] with type float32.
9. Subtract the mean value of the image from all the pixels of the image.
10. Normalize the pixel range back to [0, 1].
11. Transform the pixel range back to [0, 255] with type uint8.
12. In the same figure, display the original RGB image, the grayscale image, and the manipulated RGB image (Hint: Use matplotlib's subplots)

3 Videos

Just like with images, there are many python libraries for working with videos. mediapy is a good library for basic video reading and writing. Use it to complete the following tasks:

1. Read a video from your local file system into a numpy array.
2. Print the dimensions, type, and FPS (frames per second) of the video (i.e. the numpy array).
3. Transform the video into a grayscale video.
4. Save the grayscale video to the disk (Note: for this you can probably loop over the frames as it may be a bit harder to perform without any loops).
5. Save the grayscale video to the disk at half the original FPS.
6. Plot a figure showing a graph of the mean, sum, std, and max value per frame of the original video and the grayscale video.
7. Save the figure to the disk.