Convolution with Python

Dr. Samuel Smith
samuel.smith@bcu.ac.uk

October 18, 2023

Introduction

Part A: 1D and 2D Convolution with Numpy

TASK 1: Apply 1D Convolution Function

Utilize the following function code to perform convolution of a simple 1D input array of numbers using Python and verify its functionality.

```
import numpy as np

def simple_1d_convolution(input_array, conv_mask):
    """
    Perform 1D convolution on an input array using a convolution mask.

    Parameters:
        input_array (list or np.array): The input numbers array.
        conv_mask (list or np.array): The convolution mask.

    Returns:
        np.array: The convolution result.
    """
    return np.convolve(input_array, conv_mask, mode='full')
```

TASK 2: Implement 2D Convolution Function

Implement a function that performs 2D convolution on a 2D array of data. For this you will need the scipy library. to install scipy use pip:

```
pip install scipy
```

```
from scipy.signal import convolve2d
import numpy as np

def simple_2d_convolution(input_array, conv_mask):
    """
    Perform 2D convolution on an input array using a convolution mask.

    Parameters:
    input_array (np.array): The 2D input array.
    conv_mask (np.array): The convolution mask.

    Returns:
    np.array: The convolution result.
    """
    return convolve2d(input_array, conv_mask, mode='full')
```

TASK 3: Function Testing

Test your function with various input arrays and convolution masks. To display the results, use Python's print function. Verify the function by examining the output.

```
input_array = np.array([1, 2, 3, 4, 5, 6])
conv_mask = np.array([1, 1, 1, 1, 1])

convolution_result = simple_1d_convolution(input_array, conv_mask)
print("Convolution Result:", convolution_result)
```

```
input_array2 = np.array([[1,2,3,2], [2,1,1,2],
↪  [50,55,50],[55,50,50,55]])
conv_mask2 = np.array([[1,1,1], [0, 0, 0], [-1,-1,-1]])

convolution_result2 = simple_2d_convolution(input_array2, conv_mask2)
print("2D Convolution Result:", convolution_result2)
```

TASK 4: coding my own convolution function

Implement the following code to create another function called MyLoopConvolution which applies the cross correlation using for loops. Use np.flip() to flip the mask so it implements convolution.

```
def my_loop_convolution(my_input_array, my_conv_mask):
    """
    Perform 1D convolution on an input array using a convolution mask
    ↪  with loops.

    Parameters:
        my_input_array (list or np.array): The input numbers array.
        my_conv_mask (list or np.array): The convolution mask.

    Returns:
        np.array: The convolution result.
    """
    import numpy as np

    # Ensure the input arrays are numpy arrays
    my_input_array = np.array(my_input_array)
    my_conv_mask = np.array(my_conv_mask)


    # Initialize an empty array with the same length as the input array
    my_result = np.zeros(len(my_input_array) + len(my_conv_mask) - 1)

    # Loop through the data in the arrays
    for i in range(len(my_input_array)):
```

```
        for j in range(len(my_conv_mask)):
            # Ensure we do not go out of array bounds
            if i + j < len(my_result):
                my_result[i + j] += my_input_array[i] * my_conv_mask[j]

    # Display the results
    print("Convolution Result:", my_result)

    return my_result
```

OPTIONAL TASK The code above is a 1-D example, can you modify the code so it works for 2-D?

Part B: Image Processing with OpenCV

TASK 3: Averaging Window

Use OpenCV to apply a simple averaging window for blurring an image.

```
import cv2
import numpy as np

def apply_averaging_blur(image_path, kernel_size):
    """
    Apply an averaging blur using a specific kernel size.

    Parameters:
        image_path (str): Path to the image file.
        kernel_size (int): Size of the averaging kernel. Must be odd.
    """
    image = cv2.imread(image_path)
    blurred_image = cv2.blur(image, (kernel_size, kernel_size))

    cv2.imshow("Original", image)
    cv2.imshow("Averaging Blur", blurred_image)
    cv2.waitKey(0)
```

```
    cv2.destroyAllWindows()
```

Questions

TASK: I1

Test your function for image convolution using convolve2d(). For this apply different convolution masks. Look to the lecture for the masks to apply.

TASK: I2

Create a function which based on a user definened parameter creates a mask to be used within your 2D image convolution function from task 1. Save this function as 'MyMaskCreator" and allow it to create the masks from the lecture.

TASK: I3

Test your "MyMaskCreator" function from task I2 by calling it from within thefunction "simple_2d_convolution()" and allowing various masks to be applied.