Code for Convolution Operation on Images from Scratch without using libraries

Purva Taranekar

April 2020

1. Problem Statement

Given a set of 2D matrices as filters (filters), image (image), stride (s) and number of filters(m) as an inputs, write a function **myconv2D(image, m, filters, s)** which prints set of convoluted matrices for each filter input.

Parameter description is as follows: -

filters – it is a set of 2D matrices, each of which will be used as filters

image – it is a 2D or 3D matrix based on type of image (grayscale – 2D, RGB – 3D),

m – number of filters, that is the length of filters matrix

s – stride is the step taken while sliding

Output Description :

Output – it is a 2D matrix which is the convolution matrix of image

1. Background of Implementation

A lot of libraries exist that can find convolution of a matrix. Sometimes there is need to customise this implementation or apply improvements based on our need.

The following problems opens our thoughts towards understanding the complexity involved behind implementation of convolution algorithm.

Basic libraries like matplotlib, numpy have been used.

1. How it works

Input image is given in the form of image, which is then converted into numpy array for further computations. Stride is the slide step that the algorithm will follow while sliding across the image.

The output is expected to be same as filters size. For each filter, we need to find corresponding convoluted image of 2D. The same filter has been used on each channel, i.e., R,G,B and results of each are added to make one single 2D output.

Output image dimensions have been calculated using formula

Output\_length = (image\_length - filter\_length + 1) //s

Output \_Breadth = (breadth – filter\_breadth + 1) //s

1. Complexity Analysis

The algorithm has been implemented using simple for loops.

for each filter {

for each depth {

for each row {

for each column {

calculate each cell value

}

}

}

}

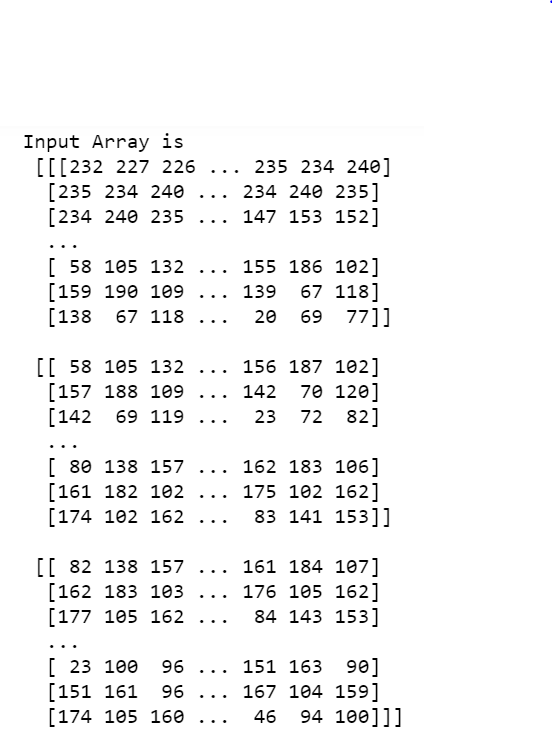
Time Complexity Analysis

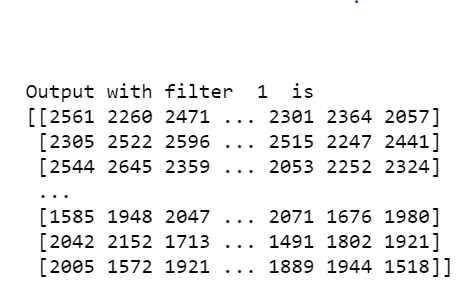
If image = l x b x d and filter = r x c, Time Complexity = m \* (d\*b\*l\*r\*c)

Space Complexity Analysis

output array of size r \* c, image array of size l \* b \* d

1. Results





Where Filter = [ [1,0,1],

[0,1,0],

[1,0,1] ]