Image Convolution from Scratch

Manish Gour
2019H1030025G
04 April 2020

1. Problem Statement

Write a computer program (using any programming language you are familiar with) to performance convolution operation on images. The following may be necessary: • Call the function as myconv2D(args). The arguments (args) to the function are the following:

– An image (either grey scale or RGB image) – Number of filters (positive integer, m) – Set of 2D matrices as filters – Stride (positive integer, s) • Let's assume that each filter matrix is of dimension r×c, and an image is of size l ×b×d, where d is the depth of the image. Let's assume that total number of filter matrices passed to the function is n. • Your program should be able to check the consistency of the arguments passed:

– Number of filter is adequate i.e. m = n/d? Return: Error, if FALSE. – Dimension of the filter is correct. Using l,b,r,c,s, you can check whether the filter dimension results in valid convolution or not. Return: Error, if FALSE.

• If everything is alright, your program should be able to perform convolution and return an output volume. The depth of the output volume should be m.

2. Setup

Program is written in python containing below modules.

1. Loading the image:

Image has been loaded with skimage.data module.

2. Convolutional2d:

```
 # This function which takes an input (Tensor) and a kernel (Tensor)
    # and returns the convolution of them
    # Args:
    #   conv_input: a numpy array of size [input_height, input_width, i
nput # of channels].
    #   conv_kernel: a numpy array of size [kernel_height, kernel_width
, input # of channels,
    #         output # of channels] represents the kernel of the Convol
utional Layer's filter.
    #   strides: a tuple of (convolution vertical stride, convolution h
orizontal stride).
```

```
#   padding: type of the padding scheme: 'same' or 'valid'.
    # if stride is not valid then program, will add the padding
# Returns:
#   a numpy array (convolution output).
```

3. Kernel Setup

Two kernels for Edge detection and Blurring image are passed to function Convolutional2d().

```
Edge Detection [-1, -1, -1], [-1, 8, -1], [-1, -1, -1]
Blur Image [1, 1, 1], [1, 1, 1], [1, 1, 1]
```

4. Output Image

Output image matrix can be obtained if all the conditions are satisfied which can be visualized using plt.imshow(image, cmap) function.

3. Method:

CNN's make use of filters (also known as kernels), to detect what features, such as edges, are present throughout an image. A filter is just a matrix of values, called weights, that are trained to detect specific features. The filter moves over each part of the image to check if the feature it is meant to detect is present. To provide a value representing how confident it is that a specific feature is present, the filter carries out a convolution operation, which is an element-wise product and sum between two matrices.

```
for x in range(output_width):  # Loop over every pixel of the output
    for y in range(output_height):
        # element-wise multiplication of the kernel and the image
        output[y, x, ch] = (conv_kernel[..., ch] *
                            conv_input[y * strides[0]:y * strides[0] + kernel_h,
                            x * strides[1]:x * strides[1] + kernel_w, :]).sum()
```

When the feature is present in part of an image, the convolution operation between the filter and that part of the image results in a real number with a high value. If the feature is not present, the resulting value is low.

A filter can be slide over the input image at varying intervals, using a stride value. The stride value dictates by how much the filter should move at each step. The output dimensions of a strided convolution can be calculated using the following equation:

nout=(floor(nin-f)/s)+1

Where n_in denotes the dimension of the input image, f denotes the window size, and s denotes the stride.

In general, you have two main options for padding scheme which determine the output size, namely 'SAME' and 'VALID' padding schemes. In 'SAME' padding scheme, in which we have zero padding, the size of output will be

```
if padding == 'same':
    output_height = int(ceil(float(input_h) / float(strides[0])))
    output_width = int(ceil(float(input_w) / float(strides[1])))
```

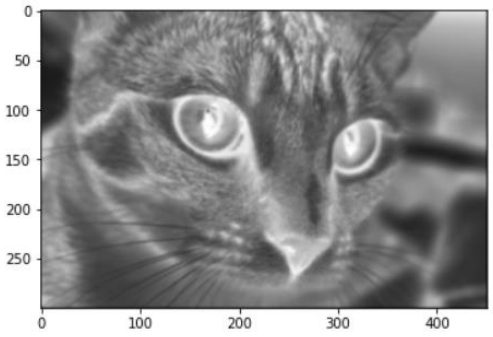
Similarly, in the 'VALID' padding scheme which we do not add any zero padding to the input, the size of the output would be

```
elif padding == 'valid':
    output_height = int(ceil(float(input_h - kernel_h + 1) / float(strides[0])))
    output_width = int(ceil(float(input_w - kernel_w + 1) / float(strides[1])))
```

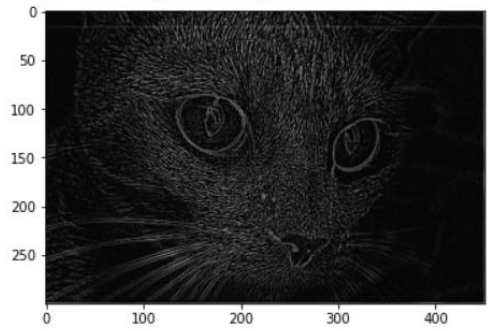
4. Result

Convoluted images using edge detection kernel and Blur image operations are shown below same shape as input image.

```
(300, 451)
<matplotlib.image.AxesImage at 0x7ff08221ca58>
```




```
(300, 451)
<matplotlib.image.AxesImage at 0x7ff08223cba8>
```



5. Conclusion:

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of `multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.