statement

April 27, 2021

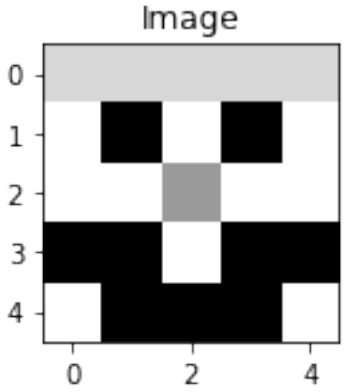
1 Convolutional filter

The task is to make a programm applying a convolutional filter to a given image. Here we work with a grayscale images in png format. They are stored as a matrix of integer numbers ranging from 0 to 255. Each cell of a matrix corrsponds to a pixel and the number is brightness of the pixel with 0 being a black pixel (minimum brightness) and 255 being a white pixel (maximum brightness). You can see some examples below:

[7]:
```
show_array_and_image("sample_image_1.png")
show_array_and_image("sample_image_2.png")
```

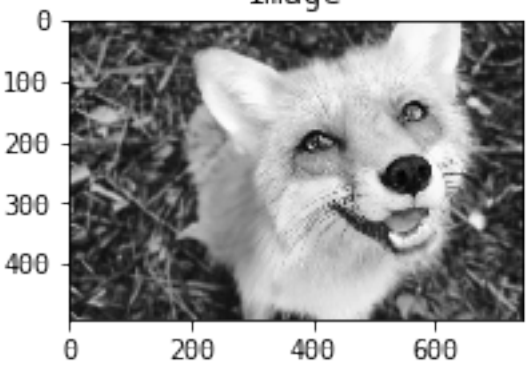
Array representation

```
[[215 215 215 215 215]
 [255   0 255   0 255]
 [255 255 154 255 255]
 [  0   0 255   0   0]
 [255   0   0   0 255]]
```



Array representation

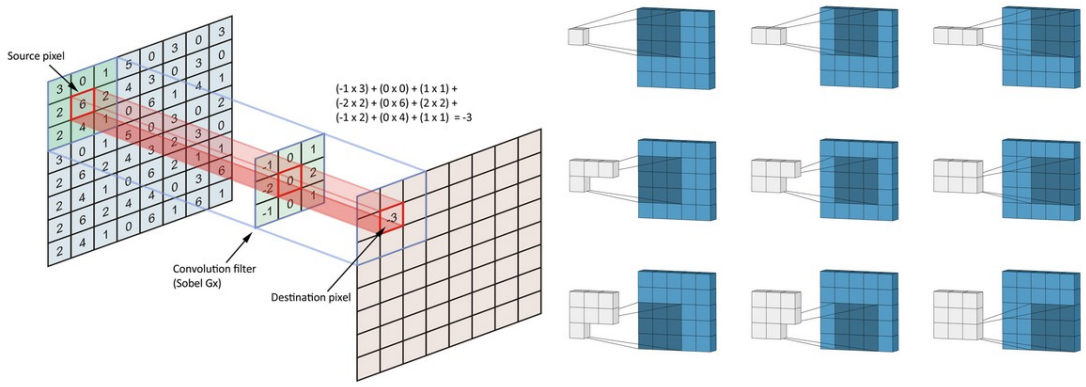
```
[[174 177 176 ... 129 140 147]
 [177 176 173 ... 125 137 145]
 [173 169 164 ... 122 133 143]
                ...
 [ 90  71  62 ...  30  31  34]
 [116  95  82 ...  30  30  33]
 [140 125 113 ...  30  30  31]]
```



A convolutional filter is defined by a kernel – usually a small matrix. The algorythm of applying the filter is the following:

1) Choose the part of the picture at the upper left angle of the image. The size of this part must be the same as the kernel size. This chosen part is called “perception field”
2) Perform elementwise multiplication of perceprtion field’s pixels values with the kernel
3) Sum up the results and store them in the new output pixel
4) Move the perception field one pixel right. If you reached the right border of the image, move it to the left border and one pixel down.

The images below visualise the algorythm:



You can find more about convolution filters here:

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

https://en.wikipedia.org/wiki/Kernel_(image_processing)

http://setosa.io/ev/image-kernels/

Also note, that after applying the filter some output values can be less than 0 or greater than 255. To fix this:

1) If there are pixels with the value less than zero, make their value equal to zero
2) After the step 1, if there are pixels with value greater than 255, let them be 255

2 Restrictions

In your code you should work with numpy arrays. Use cv2 (or other image-processing librares) only for reading or writng image to the file.

In you code you should use no more than two explicit “for” loops. Numpy arrays support broadcasting – a technique based dealing with arrays of different shape. Broadcasting operations, being

written in C language, are much faster than Python "for" cycles and thus usful for scientific calculations. You can find more about the broadcasting here:

https://machinelearningmastery.com/broadcasting-with-numpy-arrays/

3 Input data

In the folder "input" you can find a set of subfolders named as numbers: "1", "2", "3" … Each subfolder contains:

"image.png" – the grayscale image in png format "conv_kernel.txt" – convolution kernel. This file contains a kernel array. Each row is stored in a separate line, with elements being space separated

For reading the image you can use function imread from cv2 library: image = cv2.imread(path, 0) Here "0" parameter indicates that the image is taken in the grayscale mode

4 Output

The output should be a non-compressed grayscale png image "output.png", being the result of applying convolutional filter from "conv_kernel.txt" to image from "image.png". Save this image into the corresponding subfolder, containing input files.

If the kernel size is bigger than the image, raise an exception.

5 A kind of modification

An image can be stored not only as a matrix $M \times N$, but also as a vector. The simplest way to do it is to split the image into column vectors c_i and to stack them one on top of the other:

$$[c_1, c_2, ..., c_N] \rightarrow \begin{pmatrix} c_1 \\ c_2 \\ ... \\ c_N \end{pmatrix}$$

Where $\dim(c_i) = M$, and the height of the whole vector equals $M \times N$

As soon as application of a convolutional filter is a linear operation, it can be rewritten as matrix multiplication:

$$w = Av$$

Where * v is vector representation of input image * A stands for new convolutional operator, corresponding to convoutional filter K * w is vector-shaped resulting image

6 Task

The task is to implement a function, which for given convolutional kernel K makes corresponding matrix A, and apply this new filter to all input images. You may also need *to_vector* and $from\_vector$ functions, which convert images from one representation to another.

6.1 Please, keep in mind the following equation:

Supposing * X is the input image and v is its vector form * K is the convolutional filter, A is corresponding matrix

Then

$$from\_vector(Av) == K \otimes X$$

After writing code you should run it on all input images and make sure, that these results do not differ from non-modifed convolution

7 Final note

After showing me the algorithm you will be asked to add an additional simple feature to your code. The feature should be implemented in classroom, with the possibility of using the internet but without asking other people for help. The key thing you should be aware of is the usage of numpy library, including broadcasting.

If you need to clarify anything in this problem, please contact me: anton-naymov@yandex.ru +7(985)565-00-70 – cell phone, WhatsApp, Telegram

Or Andrey Ivanov: @pinkhedgehog – telegram