Back

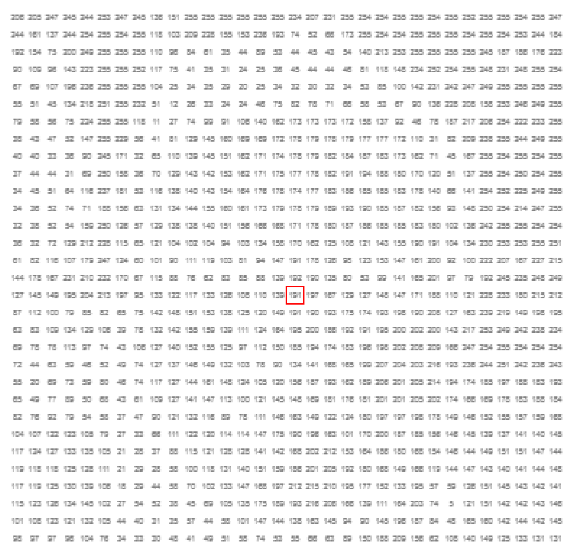
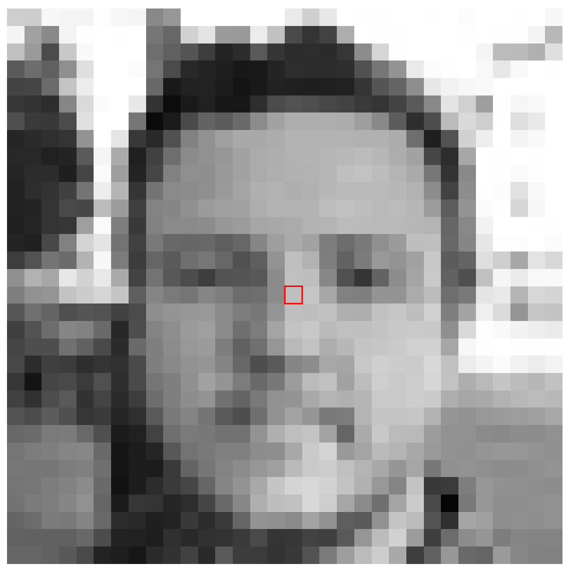
Image Kernels

Explained Visually

By Victor Powell

An image kernel is a small matrix used to apply effects like the ones you might find in Photoshop or Gimp, such as blurring, sharpening, outlining or embossing. They're also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as "convolution" (see: convolutional neural networks.)

To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.

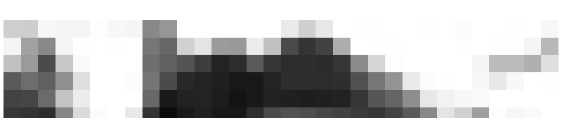




Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

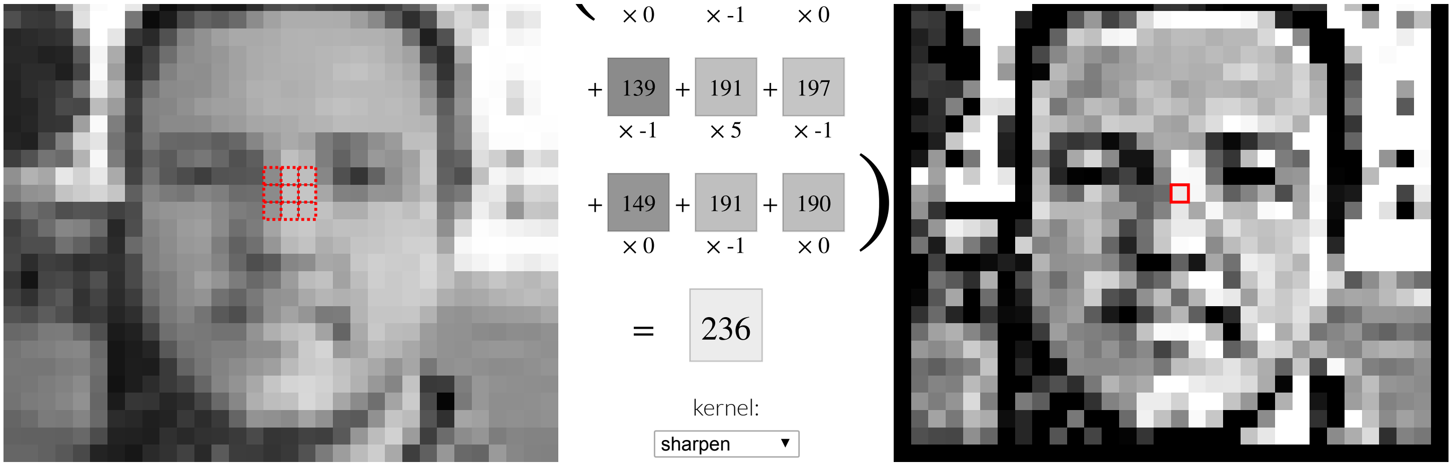
sharpen ▾

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



(139 + 192 + 190



input image output image

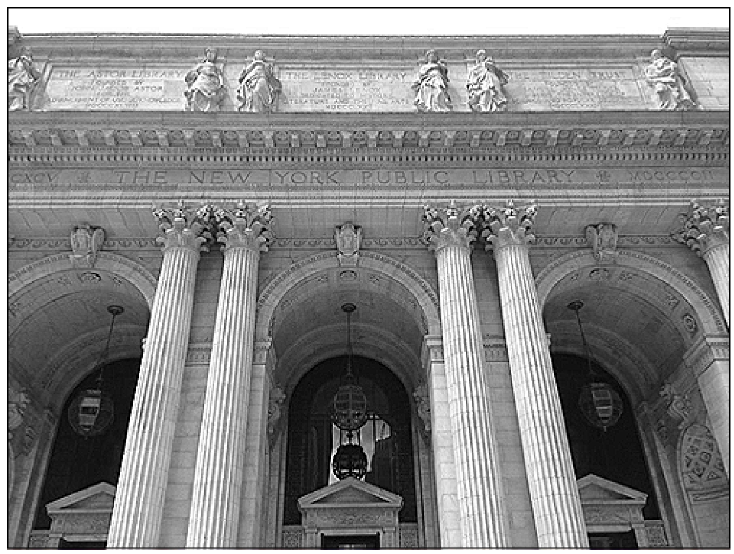
One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

Here's a playground were you can select different kernel matrices and see how they effect the original image or build your own kernel. You can also upload your own image or use live video if your browser supports it.

Dosya Seç Dosya seçilmedi Live video

0	-1	0
-1	5	-1
0	-1	0

sharpen



The **sharpen** kernel emphasizes differences in adjacent pixel values. This makes the image look more vivid.

For more, have a look at Gimp's excellent documentation on using Image kernel's. You can also apply your own custom filters in Photoshop by going to Filter -> Other -> Custom...

For more explanations, visit the Explained Visually project homepage.

Or subscribe to our mailing list.

Email address Subscribe

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Keeguon • 3 years ago

Just as a side note, though image kernels are a good way to produce a blurring effect on pictures it's not really the most efficient way to do it on large kernels. If you want to achieve more complex blurs efficiently it's more efficient to use a Fourier transform.

4 • Reply • Share ›

Zz Tux → Keeguon • a year ago

I prefer using firwin. Using one or more FIR filter works better than ifft.
After all a kernel is just a filter :D.

26 • Reply • Share ›

tar van krieken → Keeguon • 2 years ago

well, simply running the same kernel over the output picture multiple times gives the desired effect aswell and is slightly more efficient than large kernels :)

Reply • Share ›

renan jegouzo → tar van krieken • a year ago

for the blur it's faster to do it with 2 passes, one horizontally, one vertically

Reply • Share ›

Adam Hancock → renan jegouzo • 6 months ago

And it's faster still to use FFT to perform the convolution of the blur kernel in the frequency domain

Reply • Share ›

Christopher Silvia • 3 years ago

So how do Convolutional Neural Nets use these Kernels to detect features?

3 • Reply • Share ›

pwais → Christopher Silvia • 3 years ago

CNNs will learn a weighted combination of these kernels in order to match parts of objects (e.g. part of the wheel on a car). This is a nice visualization! Other helpful resources:
* See DeepViz: https://github.com/bruckner...
* See Layers >=2 plotted here: http://arxiv.org/pdf/1311.2...
* Another tool one can use to play with convolutional kernels is ShaderToy (https://www.shadertoy.com/)

5 • Reply • Share ›

Rafael Espericueta → Christopher Silvia • a year ago

Filters can be designed to find edges, which are places in an image with maximum information, as an edge implies a spatial change. Areas of an image with the same color contain very little information. So filters are used in image processing (even by our own retinas!) to extract information from images.

Reply • Share ›

Lee Meng • 3 days ago

Wonderful explanation!

Reply • Share ›

Amadeo • 5 months ago

Deep learning aside, how do you derive kernels like blur, sobel, etc?

Reply • Share ›

grubberr • a year ago

wow how simple it can be. Thanks a lot for explanation !!!

Reply • Share ›

jagadkanihal • a year ago

Awesome post, very intuative

Reply • Share ›

Orion Osborn • a year ago
How are the negative numbers translated to the greyscale?
Reply • Share ›

Adam Hancock → Orion Osborn • 6 months ago
by scaling the result or anything less than 0 is 0 and anything greater than 255 is 255
Reply • Share ›

Luis Javier Merino → Adam Hancock • 17 days ago
That is normally known as clamping, or saturating arithmethic.
Reply • Share ›

Edgar Cheung • a year ago
I am just learning about image kernel recently, and I would like to ask whether the kernels like the sobel kernels has been flipped before convolution? Or the kernels shown above have already been flipped?
Reply • Share ›

Pedro Tytgat • a year ago
Very nice page. Thank you!
Reply • Share ›

Prajwel P.J • a year ago
Beautiful explanation!
Reply • Share ›

TechnoWings Project • a year ago
Thank you for a wonderful article! You have taken lot of effort to explain convolution. Thank you so much!!!
Reply • Share ›

Harry Santoso • a year ago
so if we want sharpening image with convolution we must make it into grayscale right?
Reply • Share ›

Juraj → Harry Santoso • 10 months ago
No, the effect is best visible in grayscale. You can sharpen color images with this method as well..
Reply • Share ›

CommanderWaffles • a year ago
Excellent demo :)
Reply • Share ›

Let Me • 2 years ago
Good explanation of kernel effect. Could not thank you enough. I was really confuse as to how in Opengl is doing this by visualizing.
Reply • Share ›

Abhijit • 3 years ago
nice work man !!!
Reply • Share ›

Muggin • a year ago
Hey! First of all, thanks for the great post! I would just like suggest one adjustment. I believe this part "In this context the process is referred to more generally as "convolution"" is not entirely correct. In ConvNets a kernel is still called a kernel, or a filter, and convolution is the process of applying the kernel to the processed image.
Reply • Share ›

Anton Bacaj → Muggin • a year ago
The author said that the process is referred to as "convolution", I'm pretty sure you misunderstood what the author wrote. He did not say that kernels are called convolutions, but the act of applying the kernel in a CNN is referred to as the "convolution" part of the "C" in "CNN". Your suggestion needs to be re-evaluated.
Reply • Share ›

Archi • 2 years ago

Is Kernel matrix decided?? Or we can Choose manually kernel matrix?

• Reply • Share ›

Juraj → Archi • 10 months ago

Both yes and no. Depends what you want to do. With sharpening, sum in matrix should be 1. So in the example above you have (4 x -1) + 5 = 1. If you put -2s there, you have to change 5 for 9. Effect is much stronger now, image is overshrpened. Change again, and make negative numers -0.3, so central one should be 2.2 now. Sharpen is much weaker. You can change intensity, but some rules must be kept.

1 • Reply • Share ›