

[Back](#)

Image Kernels

Explained Visually

Tweet

Beğen 762

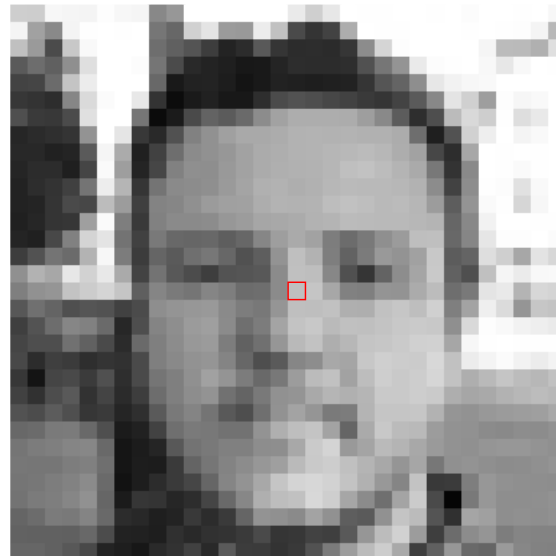
Paylaş

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.

```
208 205 247 245 244 253 247 245 138 151 255 255 255 255 255 255 234 207 231 255 254 254 255 255 255 254 255 253 255 254 255 247
244 181 137 244 254 255 254 255 118 103 229 225 155 153 238 193 74 52 85 173 255 254 254 255 255 255 254 255 254 253 244 154
162 154 75 200 240 255 255 255 110 96 54 81 35 44 59 53 44 45 43 54 140 213 253 255 255 255 255 254 187 156 176 223
90 109 26 143 233 255 255 255 117 75 41 39 21 24 28 38 45 44 44 48 51 118 148 254 252 254 255 245 231 245 255 254
87 89 107 195 235 255 255 104 35 34 35 29 20 25 34 32 30 33 34 53 55 100 142 221 242 247 249 255 255 255 255
55 51 45 134 215 251 255 232 51 12 26 33 34 24 46 73 82 75 71 89 55 53 87 90 136 225 205 155 253 248 249 255
79 55 56 75 234 255 255 118 11 27 74 99 51 106 140 162 173 173 173 155 137 92 46 78 187 217 208 254 222 233 255
35 43 47 52 147 255 225 56 41 81 129 145 180 189 189 172 175 179 179 179 177 177 172 110 31 82 209 235 255 244 249 255
40 40 33 38 90 245 171 32 85 110 139 145 151 162 171 174 175 175 182 184 187 183 173 182 71 45 187 255 254 255 254 255
37 44 44 31 69 250 155 38 70 139 142 142 153 162 171 175 177 175 182 181 194 185 180 170 120 51 137 255 254 250 254 255
34 45 51 64 116 237 151 53 116 139 142 142 154 164 176 175 174 177 183 186 185 185 183 175 140 86 141 254 252 225 249 255
24 36 52 74 71 155 156 63 131 134 144 155 180 181 173 175 175 175 189 192 190 185 187 182 156 93 145 250 254 214 247 255
32 35 52 54 159 250 126 57 129 135 135 140 151 156 185 185 171 175 180 187 188 185 185 183 150 102 136 242 255 255 254 254
36 32 72 129 212 225 115 85 121 104 102 104 94 103 134 155 170 182 125 126 121 142 155 190 191 104 134 230 253 253 250 251
81 82 118 107 175 247 124 80 101 90 111 119 105 81 54 147 191 175 136 56 133 153 147 181 200 82 100 232 207 187 227 215
144 178 187 231 210 232 170 87 115 85 76 82 83 85 88 139 182 190 135 83 53 59 141 189 201 97 79 192 245 235 248 249
127 145 149 195 204 213 197 95 133 122 117 133 125 105 110 139 191 187 189 129 145 145 147 171 185 110 121 225 223 180 215 212
87 112 100 79 85 82 85 75 142 145 151 153 135 125 120 149 191 190 193 175 174 193 195 190 208 127 183 239 219 149 195 195
83 83 109 134 129 106 39 78 132 142 155 159 139 111 134 164 195 200 186 192 191 195 202 202 200 142 217 253 249 242 233 234
89 78 78 113 97 74 42 108 127 140 152 155 125 97 112 150 155 194 174 183 196 196 202 205 209 186 247 254 255 254 254 254
72 44 83 59 48 52 49 74 137 137 146 149 132 125 78 90 134 141 185 185 199 207 204 203 216 193 236 244 251 242 238 243
35 20 89 73 59 50 46 74 117 137 144 181 140 134 105 120 156 187 193 182 199 208 201 205 214 194 174 155 197 185 183 193
85 49 77 59 50 85 43 81 109 137 141 147 113 100 121 145 145 189 181 176 181 201 201 205 202 174 186 189 175 183 155 154
52 76 92 79 54 56 37 47 90 121 132 116 59 78 111 148 183 149 122 134 180 197 197 195 175 149 152 155 157 159 185
104 107 122 123 125 79 27 33 86 111 132 120 114 114 147 175 190 196 183 121 170 200 187 185 185 148 145 135 137 141 140 145
117 124 127 133 135 105 21 28 37 85 115 121 125 126 141 142 185 202 212 152 184 186 180 185 154 146 144 147 151 151 147 144
118 118 118 125 125 111 21 29 28 95 100 118 131 140 151 155 186 201 205 192 193 185 149 188 118 144 147 143 143 141 144 145
117 118 126 130 130 106 18 29 44 56 70 102 133 147 186 197 212 219 210 195 177 152 133 195 97 59 136 191 149 143 142 141
115 123 126 134 145 102 37 54 52 38 45 89 105 125 175 189 193 216 206 188 139 111 164 203 74 5 121 151 143 142 143 146
101 105 123 121 123 105 44 40 31 25 37 44 55 101 147 144 135 183 145 94 90 145 195 187 94 45 185 180 142 144 142 145
95 97 97 96 104 76 34 23 30 48 41 49 51 55 74 33 55 86 83 89 150 188 209 156 82 105 140 149 125 132 131 131
102 102 97 85 73 35 30 22 42 30 85 41 90 80 59 51 57 82 123 157 187 205 189 82 96 151 105 101 154 125 130 129
```



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.

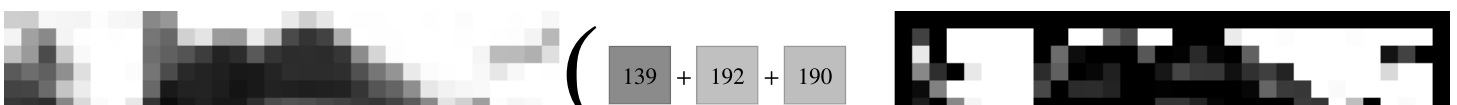


Image Kernels explained visually

$$\begin{array}{r}
 \times 0 \quad \times -1 \quad \times 0 \\
 + \begin{array}{|c|} \hline 139 \\ \hline \end{array} + \begin{array}{|c|} \hline 191 \\ \hline \end{array} + \begin{array}{|c|} \hline 197 \\ \hline \end{array} \\
 \times -1 \quad \times 5 \quad \times -1 \\
 + \begin{array}{|c|} \hline 149 \\ \hline \end{array} + \begin{array}{|c|} \hline 191 \\ \hline \end{array} + \begin{array}{|c|} \hline 190 \\ \hline \end{array} \\
 \times 0 \quad \times -1 \quad \times 0
 \end{array}
)$$

$$= 236$$

kernel:

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 where 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.

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



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.

28 Comments Explained Visually

29

Sort by Best



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 ^ | v • 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 ^ | v • Reply • Share ›

**tar van kriecken** → 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 :)

^ | v • Reply • Share ›

**renan jegouzo** → tar van kriecken • a year ago

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

^ | v • 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

^ | v • Reply • Share ›

**Christopher Silvia** • 3 years ago

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

3 ^ | v • 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 ^ | v • 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.

^ | v • Reply • Share ›

**Lee Meng** • 3 days ago

Wonderful explanation!

^ | v • Reply • Share ›

**Amadeo** • 5 months ago

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

^ | v • Reply • Share ›

**grubberr** • a year ago

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

^ | v • Reply • Share ›

**jagadkanihal** • a year ago

Awesome post, very intuitive

^ | v • Reply • Share ›



Orion Osborn • a year ago

How are the negative numbers translated to the greyscale?

^ | v • 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

^ | v • Reply • Share ›



Luis Javier Merino → Adam Hancock • 17 days ago

That is normally known as clamping, or saturating arithmetic.

^ | v • 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?

^ | v • Reply • Share ›



Pedro Tytgat • a year ago

Very nice page. Thank you!

^ | v • Reply • Share ›



Prajwel P.J • a year ago

Beautiful explanation!

^ | v • 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!!!

^ | v • Reply • Share ›



Harry Santoso • a year ago

so if we want sharpening image with convolution we must make it into grayscale right?

^ | v • 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..

^ | v • Reply • Share ›



CommanderWaffles • a year ago

Excellent demo :)

^ | v • 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.

^ | v • Reply • Share ›



Abhijit • 3 years ago

nice work man !!!

^ | v • 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.

^ | v • 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.

^ | v • Reply • Share ›

**Archi** • 2 years ago

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

^ | v • 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 $(4 \times -1) + 5 = 1$. If you put -2s there, you have to change 5 for 9. Effect is much stronger now, image is oversharpned. 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 ^ | v • Reply • Share ›

[✉ Subscribe](#) [➦ Add Disqus to your site](#)[Add DisqusAdd](#) [🔒 Privacy](#)