

Convolution Kernels

Owen McClure

Spring 2026

1 Convolution Kernels

In image processing, convolution is used to apply various kernels or filters to digital images. A kernel is a small matrix of weights that is convolved with the image to produce a new image (referred to as a feature map). For a two-dimensional image g and a kernel h , the feature map is obtained by sliding the kernel across the image, multiplying overlapping entries, and summing the result at each location [1]. In digital image processing a convolution is often defined as:

$$f(x, y) = (g * h)(x, y) = \sum_m \sum_n g(x + m, y + n) h(m, n). \quad (1)$$

where f is the resulting feature map, g is the original image, h is the kernel, and (x, y) represents a specific pixel location. m and n represent summation indices over the kernel coordinates (vertical and horizontal offsets). So at the the output pixel location (x, y) , the term $g(x + m, y + n)h(m, n)$ means that we are taking the pixel at a position shifted by m and n from the current location (x, y) , multiplying it by the corresponding kernel weight at position (m, n) , and summing this product over all kernel positions (m, n) to get the new pixel value at (x, y) in the feature map.

From a strictly mathematical perspective, this definition is actually cross-correlation. But common practice in image processing is to call it convolution, even though the kernel is not flipped. This is because for convolutional kernels that are symmetric about the origin, convolution and cross-correlation operations will produce the same output.

In practical computer-vision settings, the kernel h is usually much smaller than the image g and can be designed to apply a number of different filtering operations such as applying smoothing or detecting features like edges, textures, or repeated patterns. If we want the feature map to have the same dimensions as the original image, we can use what is known

as padding, and is implemented in most convolution functions in image processing libraries. Padding adds extra invented pixels around the border of the image, and if we pad with zeros (referred to as zero-padding) these artificially added pixels will be rendered as black. If there is no padding, this is called valid convolution, and the resulting feature map will be smaller than the original image. If we choose to pad the image such that the resulting feature map has the same dimensions as the original image, this is called same convolution. It is common practice in image processing to use odd-sized kernels, such as 3×3 or 5×5 , so that the kernel has a well-defined center pixel. This allows for symmetric padding and helps preserve the spatial dimensions of the feature map [2]. Different padding choices can introduce artifacts at image edges, as we will see in later demonstrations.

A common term you will see in image processing is “2D convolution” which refers to applying a 2D kernel across both dimensions of the image simultaneously. The kernel is a two-dimensional matrix, and the convolution operation is performed in two dimensions, meaning that we are sliding the kernel across both the height and width of the image, our definition in equation 8 is a 2D convolution. There are also cases of separable convolution, where a 2D kernel can be expressed as the outer product of two 1D kernels, allowing for more efficient computation. In that case, we can perform two separate 1D convolutions instead of one 2D convolution. To imagine a 1D convolution along one axis, we simply remove one of our summation indices, leaving us with something similar to the equation seen in our example of sliding arrays from the convolution in probability section [1].

In both probability and image processing, convolution is acting as a local weighted averaging and accumulation process. In probability, we are averaging over the possible values of one random variable weighted by the distribution of another. In image processing, we are averaging over local pixel neighborhoods weighted by the kernel values. But, this is only one part of the connection between probability and convolution in image processing.

1.1 Noise + Images

Image processing is a cross-disciplinary field that is often thought of as a subarea of digital signal processing. In this paper, multiple different digital images are used for experimentation. To understand why convolution is the right operation in image processing, it is necessary to provide some information on how images are acquired and represented in a digital format.

One case of image acquisition is capturing an image with a vision system such as a camera. At the origin of image acquisition is the electromagnetic spectrum. Photography is a common

example of visible-light imaging.

To acquire a 2D digital image, a vision system often uses a matrix of sensors, such as a camera sensor, to capture light intensity at different locations in a scene. Each sensor in the matrix corresponds to a pixel in the resulting image, and the captured intensity is represented numerically. The resulting image can then be represented as a 2D array of pixel intensities, where each element corresponds to one sensor location.

To replicate human color perception, image acquisition mimics the three cone-cell sensitivities in the eye. This is typically implemented with a color filter array (CFA) on the sensor, so each sensing element captures only one of the three primary channels (red, green, or blue).

Finally, digitization involves sampling and quantization. Sampling maps a continuous scene onto a discrete pixel grid. Quantization maps the continuous range of measured intensities into a finite set of values. Often, images are quantized to 256 gray levels, represented using 8 bits per pixel. In that case, each pixel can take one of 256 intensity values, from 0 (black) to 255 (white). The result is a discrete representation of the scene that can be processed and analyzed with operations such as convolution. An important distinction in convolution operations is grayscale vs RGB images. Color channels add a third dimension, so 2D convolution still slides over height and width but there are a few different choices one must make for handling the channel axis. One way of handling color channels is per-channel (depthwise) convolution: applying the same (or different) 2D kernel to each color channel independently producing one output per channel. Each RGB channel is convolved separately and then stacked back into an RGB image. Examples of this method will be used in the following section. Another method popular in many classical image-processing operations, is to simply covert RGB to one channel and apply a single 2D kernel.

One additional term you are likely to encounter in digital imaging literature is Point Spread Function (PSF). In imaging, PSF describes how a point source of light spreads in an image due to the imaging system response. The PSF can be modeled as a convolutional kernel, and in many image processing texts the terms PSF, kernel, and filter are used interchangeably.

In digital photography, each step of acquisition introduces real randomness. Noise can appear during light capture by the sensor, during digitization, and during storage or transmission. For our purposes, image acquisition is not the main focus of this paper, but it is important to understand how images are acquired and represented in a digital format, as our connection between probability and convolution in image processing collides even more directly when we consider noise.

Noise is naturally a random phenomenon and can be modeled using probability distribu-

tions. For digital images, the noise at each pixel can be modeled as an independent random variable [?].

A common model in vision systems is $Observed\ Image = True\ Image + Noise$. This can be defined mathematically as:

$$y = x + b \tag{2}$$

where y is the observed image, x is the true image, and b is the noise. In this model, the noise b can be modeled as a random variable with a certain probability distribution. Another common model is: $Observed\ Image = Scene * Imaging\ System + Noise$. This describes how convolutions also appear in image formation, as most imaging systems can be modeled as linear systems that blur and mix light in a spatially structured way. Returning to the PSF, a point of light in a scene does not become a point of light on the sensor. Things like diffraction, defocus, and motion, cause it to spread out. Any scene can be decomposed into many point sources, each point produces a shifted copy of the PSF, and adding all those contributions together is convolution [?].

References

- [1] Mazet V. Basics of image processing. Universite de Strasbourg; 2025. Available from: <https://vincmazet.github.io/bip/>.
- [2] Bishop CM, Bishop H. Deep learning. Springer Nature; 2023.