# Convolution

Convolution is a important concept in a lot of different fields aswell in the field of signal processing and analysis. Convolution is able to create an output out of any input signal and an impulse response.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | X |  |
|  |  |  |

In convolution a calculation is performed at a pixel which is set by the sum of the neigbouring pixels.  
This could be a + sign a 3x3 matrix or bigger depending on the situation.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | X |  |
|  |  |  |

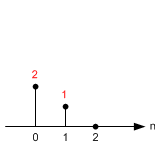
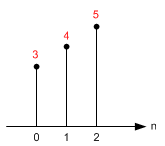
Example neighborhoods

The square or neighborhood is also know as the convolution kernel. And the size of this kernel will dictate the calculations that will have to happen.   
Then there is the impulse response which is what the system is going to run against the current values and will be better understandable as the filter.  
A certain setting of the impulse could be a low pass filter where in a different one could be a high pass filter.  
More on those filters later in this chapter.  
Convolution definition comes out of the mathematical domain.   
And its equation is as follows:  
Definition of 1D Convolution   
x[n] is an input signal  
h[n] is the impulse response  
y[n] is output  
\* is the notation for convolution.  
  
Important is that we multiply x[k] by the terms of time-shifted h[n] and add them up.

Key to understanding convolution is understanding the way the impulses are being handled.

## 1D convolution

For this example we are given the following:  
*x*[n] = { 3, 4, 5 }  🡨 is an input signal  
*h*[n] = { 2, 1 } 🡨 is the impulse response



Impulse Response: h[n]

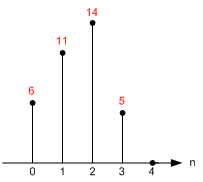
Input: x[n]

And with this information we are going to calculate the output step for step.  
For the calculation of a straight line we don’t have to think about the kernel just yet.

y[0] = x[k] · h[0-k] = x[0] · h[0] = 3·2 = 6  
y[1] = x[k] · h[1-k] = x[0] ·h[1-0] + x[1] ·h[1-1] = 3·1+4·2 = 11  
y[2] = x[k] · h[2-k] = x[0] ·h[2-0] + x[1] ·h[2-1]   
 + x[2] ·h[2-2] = 3·0+4·1+5·2 = 14  
y[3] = x[k] · h[3-k] = x[0] ·h[3-0] + x[1] ·h[3-1]   
 + x[2] ·h[3-2] + x[3] ·h[3-3] = 0+0+5x1+0 = 5

y[4] = x[k] · h[4-k] = x[0] ·h[4-0] + x[1] ·h[4-1]  
 + x[2] ·h[4-2] + x[3] ·h[4-3]  
 + x[4] ·h[4-4] = 0+0+0+0+0 = 0

As you might as well have spotted there is no use to continue after y[4] they will all be 0’s.



Output: y[n]

## 2D Convolution

Keep in mind that everything is based on column and row.   
Just to be clear that is horizontal and then vertical.

Notice that the origin of impulse response is always centered. (h[0,0] is located at the center sample of kernel, not the first element.)

Let's start calculate each sample of the output one by one.

First, flip the kernel, which is the shaded box, in both horizontal and vertical direction. Then, move it over the input array. If the kernel is centered (aligned) exactly at the sample that we are interested in, multiply the kernel data by the overlapped input data.

The accumulation (adding these 9 multiplications) is the last thing to do to find out the output value.

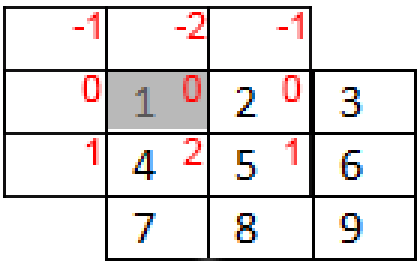
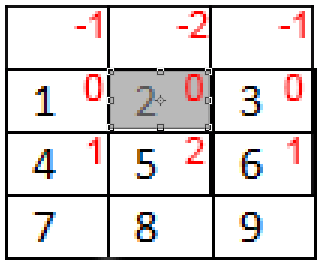
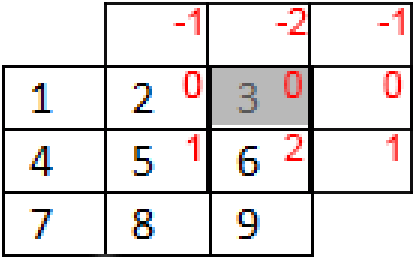
Note that the matrices are referenced here as [column, row], not [row, column]. M is horizontal (column) direction and N is vertical (row) direction.

For this example we are using the following:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Input x[n] Impulse Response: h[n]

Y[0,0] Y[0,1] y[0,2]

The impulse response is always centered around the value that we want to change.

This means that the value that your are calculating the value of h[0,0] is centered on that value.( This is one in the example).

However for the value x[0,0] is almost the first value top left. (in this example that is 1.)

For calculating y[0,0] we need the following.

Y[0,0] =   
 X[-1,-1] · h[1,1] + X[0,-1] · h[0,1] + X[1,-1] · h[-1,1]   
+ X[-1,0] · h[1,0] + X[0,0] · h[0,0] + X[1,0] · h[-1,0]  
+ X[-1,1] · h[1,-1] + X[0,1] · h[0,-1] + X[1,1] · h[-1,-1]

0x1 + 0x2 + 0x1   
+ 0x0 + 1x0 + 2X0   
+ 0x1 + 4x-2 + 5x-1 = -13

Y[1,0] =   
 X[0,-1] · h[1,1] + X[1,-1] · h[0,1] + X[2,-1] · h[-1,1]

+ X[0,0] · h[1,0] + X[1,0] · h[0,0] + X[2,0] · h[-1,0]

+ X[0,1] · h[1,-1] + X[1,1] · h[0,-1] + X[2,1] · h[-1,-1]

0x1 + 0x2 + 0x1  
+ 1x0 + 2x0 + 3x0

+ 4x-1 + 5x-2 + 6x-1 = -20

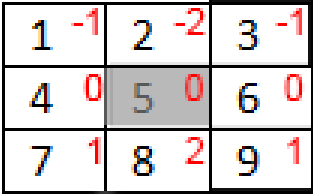
Y[2,0] =   
 X[1,-1] · h[1,1] + X[2,-1] · h[0,1] + X[3,-1] · h[-1,1]

+ X[1,0] · h[1,0] + X[2,0] · h[0,0] + X[3,0] · h[-1,0]

+ X[1,1] · h[1,-1] + X[2,1] · h[0,-1] + X[3,1] · h[-1,-1]

0x1 + 0x2 + 0x1  
+ 2x0 + 3x0 + 0x0

+ 5x-1 + 6x-2 + 0x-1 = -17



Y[1,1] = X[0,0] · h[1,1] + X[1,0] · h[0,1] + X[2,0] · h[-1,1]

+ X[0,1] · h[1,0] + X[1,1] · h[0,0] + X[2,1] · h[-1,0] +

X[0,2] · h[1,-1] + X[1,2] · h[0,-1] + X[2,2] · h[-1,-1]

1 + 4 + 3 + 0 + 0 + 0 + -7 + -16 + -9 = -24 =

|  |  |  |
| --- | --- | --- |
| -13 | -20 | -17 |
|  | -24 |  |
|  |  |  |

Output: y

-9 + -16 + -7 -32

8 24

Lowpass =  
1 1 1  
1 1 1   
1 1 1   
------  
 9

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Highpass  
-1 -1 -1  
-1 8 -1  
-1 -1 -1