Image Processing and Imaging
Correlation and Convolution

Dominik Söllinger
Fachbereich Computerwissenschaften
Universität Salzburg

Winter term 2022/23

PARIS
LODRON
UNIVERSITY
SALZBURG

Correlation and **Convolution** are essential building blocks image processing pipeline. Unfortunately these two operations often get confused. Hence we will take a closer look on the differences and their properties.

Definitions (for continuous 1D signals):

Let $x(t)$ and $y(t)$ be two continuous 1-D signals. Then the correlation / convolution between the signals can be calculated as follows:

Correlation

$$(x \otimes y)(t) = \int_{-\infty}^{\infty} x(t+\tau) \cdot y(\tau)\, d\tau$$

Convolution

$$(x * y)(t) = \int_{-\infty}^{\infty} x(t-\tau) \cdot y(\tau)\, d\tau$$

Note the different signs in the formulas!

Similar to the continuous case, we can also define the correlation and convolution for discrete 1D signals.

Definitions (for discrete 1D signals):

Let $x[n]$ and $y[n]$ be two discrete 1-D signals.

Correlation

$$(x \otimes y)[n] = \sum_{k=-\infty}^{\infty} x[n+k] \cdot y[k]$$

Convolution

$$(x * y)[n] = \sum_{k=-\infty}^{\infty} x[n-k] \cdot y[k]$$

To better understand the difference between correlation and convolution, we study the difference based on a simple example.

Example:

Let the two discrete signals $x[n]$ and $y[n]$ be defined as follows:

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]

Goal:

Compute the correlated signal Corr[n] — What's the value of each n?

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
Corr[n]										

Example: Correlation (2)

For Corr[n=1]:

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
Corr[n]					**1**					

For Corr[n=2]:

n	...	-3	-2	-1	0	1	2	3	4	...	
x[n]	[...	2	-1	1	2	2	3	1	4	...]	
y[n]		[...	0	0	-1	0	1	0	0	0	...]
Corr[n]					**1**	**1**					

For Corr[n=3]:

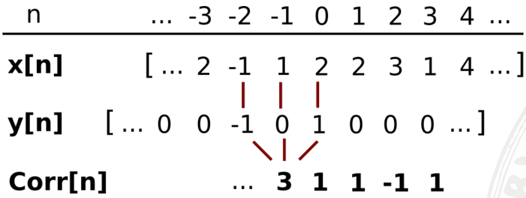
n	… -3 -2 -1 0 1 2 3 4 …
x[n]	[… 2 -1 1 2 2 3 1 4 …]
y[n]	[… 0 0 -1 0 1 0 0 0 …]
Corr[n]	**1 1 -1**

For Corr[n=4]:

n	… -3 -2 -1 0 1 2 3 4 …
x[n]	[… 2 -1 1 2 2 3 1 4 …]
y[n]	[… 0 0 -1 0 1 0 0 0 …]
Corr[n]	**1 1 -1 1**

We should not forget about negative *ns*!

For Corr[n=-1]:



Finally, we end up with something like this:

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
Corr[n]	**[**	**...**	**-1**	**3**	**1**	**1**	**-1**	**1**	**...**	**]**

To reproduce the result in Numpy simply run:

```
x = np.asarray([2,-1,1,2,2,3,1,4])
y = np.asarray([0,0,-1,0,1,0,0,0])
corr = np.correlate(x,y, mode='same')
```

Example: Convolution (1)

That was easy — But what about the convolution?

Goal:

Compute the convolved signal Conv[n] — What's the value of each n?

Example: Convolution (2)

For Conv[n=0]:

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
Conv[n]					**-1**					

For Conv[n=1]:

n	... -3 -2 -1 0 1 2 3 4 ...
x[n]	[... 2 -1 1 2 2 3 1 4 ...]
y[n]	[... 0 0 -1 0 1 0 0 0 ...]
Conv[n]	**-1 -1**

For Conv[n=2]:

n	… -3 -2 -1 0 1 2 3 4 …
x[n]	[… 2 -1 1 2 2 3 1 4 …]
y[n]	[… 0 0 -1 0 1 0 0 0 …]
Conv[n]	**-1 -1 1**

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
Conv[n]	**[**	**...**	**1**	**-3**	**-1**	**-1**	**1**	**-1**		**...]**

To reproduce the result in Numpy simply run:

```
x = np.asarray([2,-1,1,2,2,3,1,4])
y = np.asarray([0,0,-1,0,1,0,0,0])
conv = np.convolve(x,y, mode='same')
```

Convolving two signals seems somehow difficult to do "by hand".

But wait! What if we flip/mirror y[n] and substitute the convolution by a correlation?

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
y[-n]	[...	0	0	1	0	-1	0	0	0	...]

Correlation vs. Convolution (2)

Convolving two signals seems somehow difficult to do "by hand".

But wait! What if we flip/reverse y[n] and substitute the convolution by a correlation?

n	...	-3	-2	-1	0	1	2	3	4	...
x[n]	[...	2	-1	1	2	2	3	1	4	...]
y[n]	[...	0	0	-1	0	1	0	0	0	...]
y[-n]	[...	0	0	1	0	-1	0	0	0	...]
Corr*[n]	**[...**		**1**	**-3**	**-1**	**-1**	**1**	**-1**		**...]**
Conv[n]	**[...**		**1**	**-3**	**-1**	**-1**	**1**	**-1**		**...]**

The result is the same!

Proof:

Let $x(t)$ and $y(t)$ be two continuous signals. $y^*(t)$ is the reversed version of $y(t)$

More precisely, $y^*(t) = y(-t)$.

$$
\begin{aligned}
x(t) \otimes y^*(t) &= \int_{-\infty}^{\infty} x(t+\tau) \cdot y^*(\tau)\, d\tau \\
&= \int_{-\infty}^{\infty} x(t+\tau) \cdot y(-\tau)\, d\tau \\
&= \int_{-\infty}^{\infty} x(u) \cdot y(-(u-t))\, du \\
&= \int_{-\infty}^{\infty} y(t-u) \cdot x(u)\, du \\
&= y(t) * x(t)
\end{aligned}
$$

$$
\left|\;\begin{aligned} u &= t + \tau \\ \tau &= u - t \\ d\tau/du &= 1 \end{aligned}\right.
$$

What remains to be shown is that ...

$$y(t) * x(t) = x(t) * y(t)$$

Proof continued:

$$\begin{aligned} y(t) * x(t) &= \int_{-\infty}^{\infty} y(t-u) \cdot x(u)\, du \\ &= -\int_{-\infty}^{\infty} y(v) \cdot x(t-v)\, dv \\ &= \int_{-\infty}^{\infty} x(t-v) \cdot y(v)\, dv \\ &= x(v) * y(v) \end{aligned} \qquad \left| \begin{array}{l} v = t-u \\ u = t-v \\ du/dv = -1 \end{array} \right.$$

Important takeaways (1)

- A correlation can be easily replaced with a convolution and vice versa
  - $x(t) * y(t) = x(t) \otimes y(-t)$
  - $x(t) \otimes y(t) = x(t) * y(-t)$
  - **But** $x(t) \otimes y(t) \neq x(-t) * y(t)$
- The convolution is commutative
  - $x(t) * y(t) = y(t) * x(t)$

- A correlation can be easily replaced with a convolution and vice versa
  - $x(t) * y(t) = x(t) \otimes y(-t)$
  - $x(t) \otimes y(t) = x(t) * y(-t)$
  - **But** $x(t) \otimes y(t) \neq x(-t) * y(t)$
- The convolution is commutative
  - $x(t) * y(t) = y(t) * x(t)$

But what about the correlation?

Important takeaways (3)

- A correlation can be easily replaced with a convolution and vice versa
  - $x(t) * y(t) = x(t) \otimes y(-t)$
  - $x(t) \otimes y(t) = x(t) * y(-t)$
  - **But** $x(t) \otimes y(t) \neq x(-t) * y(t)$
- The convolution is commutative
  - $x(t) * y(t) = y(t) * x(t)$

But what about the correlation?

- No, the correlation is NOT commutative!

  See:

$$[0, 1, 2, 3] \otimes [1, 1] = [1, 3, 5]$$
$$[1, 1] \otimes [0, 1, 2, 3] = [5, 3, 1]$$

Summary of the most important properties

The convolution is ...

- commutative
- associative
- distributive

The correlation is ...

- NOT commutative
- NOT associative
- distributive

In other words, the convolutions are much nicer in terms of their properties!

Remark

When you search for "correlation" on the web, you will soon come across the terms "cross-correlation" and "auto-correlation".

Cross-correlation: Used to express that two different signals are correlated with each other (like we did in our example).

Auto-correlation: Used to express that a signal is correlated with itself (i.e., with a delayed copy).

Since we work in the field of image processing, we simply stick with the term "correlation" for the sake of simplicity.

So far, we have only look at at the correlation/convolution in the context of signal processing. Some additional steps need to be taken to make both operations work for 2D images.

$$(x * y)[n] = \sum_{k=-\infty}^{\infty} x[n-k] \cdot y[k]$$

— **What do we have to change?**

Correlation and Convolution for 2D images (1)

Definitions (for 2D images):

Let $I[n, m]$ be an image and H a kernel (matrix) of size $W \times H$.

Correlation

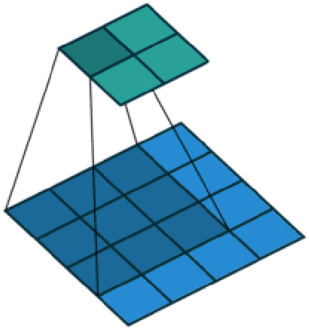
$$(I \otimes H)[n, m] = \sum_{k=0}^{W-1} \sum_{l=0}^{H-1} I[n + k, m + l] \cdot H[k, l]$$

Convolution

$$(I * H)[n, m] = \sum_{k=0}^{W-1} \sum_{l=0}^{H-1} I[n - k, m - l] \cdot H[k, l]$$

All properties still hold for 2-D correlations/convolutions!

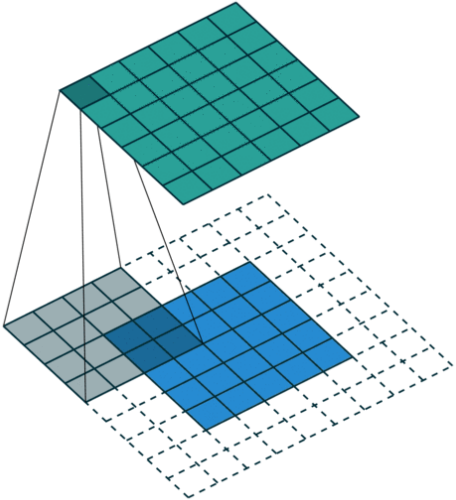
Correlation and Convolution for 2D images (2)



Convolution/Correlation changes the output size

— **How can we fix this?**

Border padding



Types of padding (1)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	2	0	0
0	0	3	4	5	0	0
0	0	6	7	8	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figure: Constant padding with 0s

Types of padding (2)

8	7	6	7	8	7	6
5	4	3	4	5	4	3
2	1	0	1	2	1	0
5	4	3	4	5	4	3
8	7	6	7	8	7	6
5	4	3	4	5	4	3
2	1	0	1	2	1	0

Figure: Reflective padding

Types of padding (3)

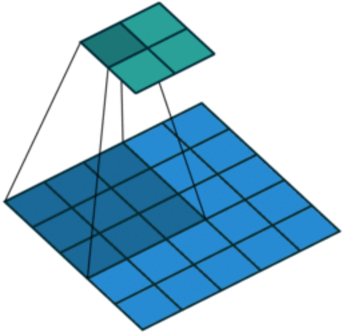
4	3	3	4	5	5	4
1	0	0	1	2	2	1
1	0	0	1	2	2	1
4	3	3	4	5	5	4
7	6	6	7	8	8	7
7	6	6	7	8	8	7
4	3	3	4	5	5	4

Figure: Symmetric padding

Types of padding (4)

4	5	3	4	5	3	4
7	8	6	7	8	6	7
1	2	0	1	2	0	1
4	5	3	4	5	3	4
7	8	6	7	8	6	7
1	2	0	1	2	0	1
4	5	3	4	5	3	4

Figure: Circular (a.k.a. wrap) padding



Output size calculation (1)

How do we calculate the output size (o) given the ...

padding (p), stride (s), kernel size (k) and image size (i) ?

How do we calculate the output size (o) given the ...

padding (p), stride (s), kernel size (k) and image size (i) ?

$$o = \lfloor \frac{i+2p-k}{s} \rfloor + 1$$

Words of warning

With the raise of deep learning (and Convolutional Neural Networks), convolutions have received a lot of attention. As a result you will find many article on the web explaining convolution but not correlation. Unfortunately it seems that many people are not aware of the differences!

- Take the resources explaining the convolution with a grain of salt.
- Even research papers might be misleading (see)
- Note that in PyTorch (deep learning framework) *torch.nn.Conv2d* performs a correlation. Only *torch.nn.functional.conv2d* performs a convolution.

Where is the eye located in the image?

Matching using Normalized Cross Correlation (NCC):

$$NCC[n,m] = \frac{\sum_{k,l} T[k,l] \cdot I[m+k, n+l]}{\sqrt{\sum_{k,l} T[k,l]^2} \sqrt{\sum_{k,l} I[m+k, n+l]^2}}$$

Matching using Normalized Cross Correlation (NCC):

$$NCC[n,m] = \frac{\sum_{k,l} T[k,l] \cdot I[m+k,n+l]}{\sqrt{\sum_{k,l} T[k,l]^2}\sqrt{\sum_{k,l} I[m+k,n+l]^2}}$$

What is the problem with this approach?

Matching using Normalized Cross Correlation (NCC):

$$NCC[n,m] = \frac{\sum_{k,l} T[k,l] \cdot I[m+k,n+l]}{\sqrt{\sum_{k,l} T[k,l]^2}\sqrt{\sum_{k,l} I[m+k,n+l]^2}}$$

What is the problem with this approach?

No invariance to intensity and contrast changes.

Improved matching using Zero Normalized Cross Correlation (ZNCC):

$$NCC[n, m] = \frac{\sum_{k,l}(T[k,l]-\bar{T})(I[m+k,n+l]-\bar{I}_{n,m})}{\sqrt{\sum_{k,l}(T[k,l]-\bar{T})^2}\sqrt{\sum_{k,l}(I[m+k,n+l]-\bar{I}_{n,m})^2}}$$

$\bar{I}_{n,m}$ ... Mean of the image patch (not the global image mean)

$\bar{T}$ ... Mean of the template