Neural Networks and Deep Learning: Convolution

Nicolas Thome

Conservatoire National des Arts et Métiers (Cnam)
Département Informatique

Convolution in 1D (Signal)

- **Discrete 1D convolution** with **Finite Impulse Response (FIR) filter h**, size d (odd)
- Input signal $f(i)$, $i \in \{1; N\}$
- Output signal $f'(i)$, $i \in \{1; N\}$
- Convolution: operator $T : f \to f' = T[f] = f \star h$

$$f'(i) = (f \star h)(i) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n)h(n)$$

Convolution in 2D (Images)

- **Discrete 2D convolution** with **FIR filter h** (size d odd), $T : f \to f' = T[f] = f \star h$:

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j)h(n,m)$$

$$h = \begin{pmatrix} w_9 & w_8 & w_7 \\ w_6 & w_5 & w_4 \\ w_3 & w_2 & w_1 \end{pmatrix}$$

- **Ex with a 3×3 filter:**

$$g = \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix}$$

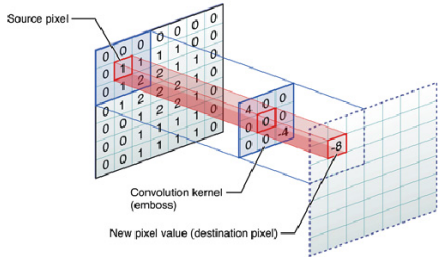
$$\begin{aligned} f'(i,j) &= w_1 f(i-1,j-1) + w_2 f(i-1,j) + w_3 f(i-1,j+1) \\ &+ w_4 f(i,j-1) + w_5 f(i,j) + w_6 f(i,j+1) \\ &+ w_7 f(i+1,j-1) + w_8 f(i+1,j) + w_9 f(i+1,j+1) \end{aligned}$$

- **Convolution processing:**
  1. Apply central symmetry to the filter: $h(n,m) \Rightarrow h(-n,-m) = g(n,m)$
  2. $\forall (i,j)$, compute weighted sum between image value around $f(i,j)$ and filter coeffs $g(n,m)$

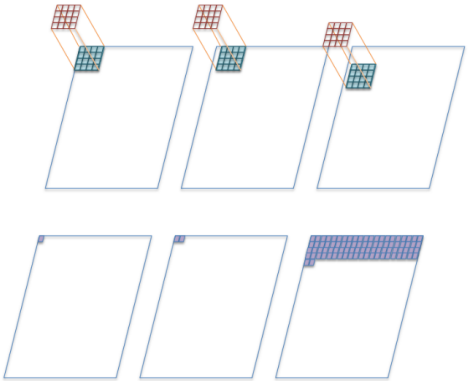
2D Convolution *vs* Cross-Correlation

- 2D Convolution: $f'(i,j) = (f \star h)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j)h(n,m)$
- Cross-Correlation: $f'(i,j) = (f \otimes h)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i+n, m+j)h(n,m)$
  - Cross-Correlation ~ Convolution without symmetrizing mask!

$$h = \begin{pmatrix} -4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \Rightarrow g = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -4 \end{pmatrix}$$



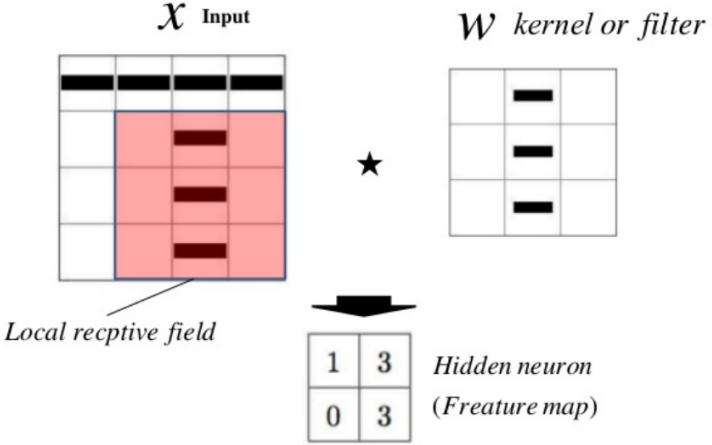
2D Convolution / Cross-Correlation: Interpretation



- Cross-Correlation: at each (i, j) location: **dot product** between **image region** and **filter h**
  - Large output $f'(i, j)$
    $\Rightarrow$ **filter h and image region around** $f(i, j)$ **aligned**
- **Input:** 2d image $\Rightarrow$ output: 2d map

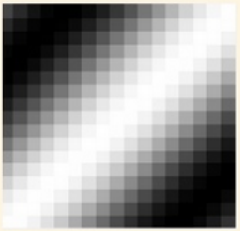
2D Convolution / Cross-Correlation: Example

- Cross-Correlation: output maps ⇒ location in input image similar (*i.e.* aligned, dot product) to input image content

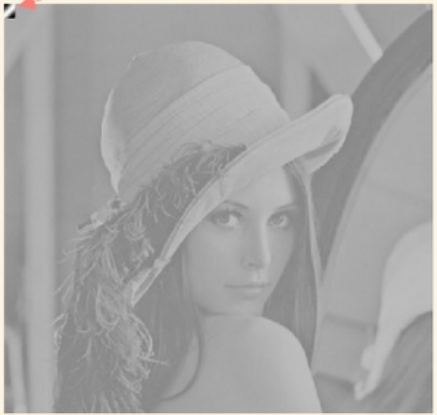


2D Convolution / Cross-Correlation: Real Image Example

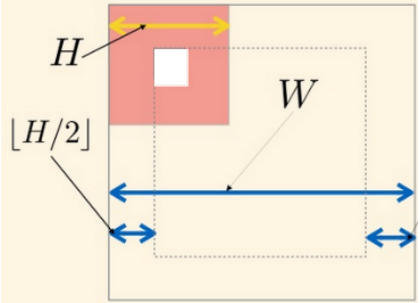
The filter size is • • •



like this.



Padding



- Filter of size H: problem with the borders $\lfloor \frac{H}{2} \rfloor$
- **Solution 1**: reduce processing to computable area ⇒ decreased output size
- **Solution 2**: padding, *i.e.* fill missing info with (arbitrary) values
  - Zero-padding, recopy, mirror, *etc*

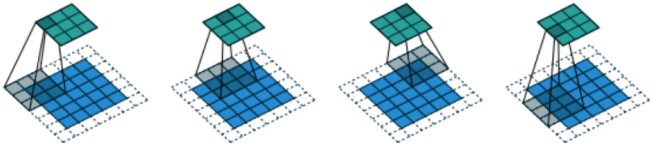
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	2	5	6	3	6	7	3	0	0	0
0	0	2	3	4	6	7	5	1	8	4	0	0	0
0	0	8	7	6	5	7	6	3	3	4	0	0	0
0	0	2	3	5	6	7	8	2	7	3	0	0	0
0	0	4	5	3	2	1	6	8	7	2	0	0	0
0	0	1	4	5	3	2	6	7	8	1	0	0	0
0	0	2	3	4	5	6	8	9	2	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Strided Convolution

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j)h(n,m)$$

- Standard convolution: stride 1 $\Rightarrow$ compute $f'(i,j)$ for $(i,j) \in \{1;N\} \times \{1;M\}$
- Strided convolution: compute $f'(i,j)$ for $i \in \{1, 1+s, 1+2s, ..., N\}$ (*resp.* $j \in \{1, 1+s, 1+2s, ..., M\}$)
- Ex: $s = 2$, $N = M = 5$, $d = 3$
  $\Rightarrow$ **reduced map size** (3×3)



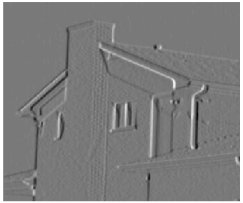
Convolution: Example for Gradient Computation

$I_x \approx I \star M_x$

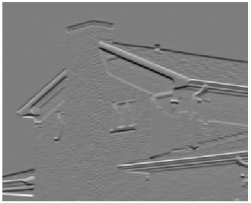
- Gradient vector:
  $\vec{G}(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix}^T = \begin{pmatrix} Ix & Iy \end{pmatrix}^T$
- I_x and I_y approximated by linear convolution:
  $I_x \approx I \star M_x$, $I_y \approx I \star M_y$

$$M_x = \frac{1}{4} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \frac{1}{4} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

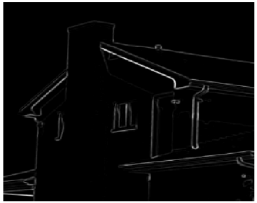
Convolution with Multiple Filters: Edge Detection



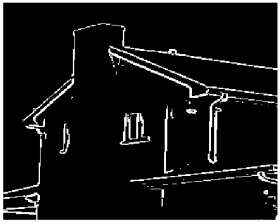
$Ix \sim$ filter 1



$Iy \sim$ filter 2



$I_e = Ix^2 + Iy^2$



$I_{e,t}$

$I_{e,t}$: thresholding $I_e \Rightarrow$ **edge detector!**

Edge detection with Convolutional Nets?

Convolution: Linear Filtering

- Convolution can be viewed as multiplication by a matrix
- 1D case: univariate Toeplitz matrix:

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}$$

- 2D case: doubly block circulant matrix

$$\begin{bmatrix} c_0 & c_{n-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{bmatrix}$$

Convolution: Conclusion

- Convolution for 1D signals, 2D images
  - Local Processing
  - Extracts feature maps: activations $\Leftrightarrow$ correlation wrt filter
  - Linear operation $\sim$ matrix multiplication
- **Non-linearity, nice properties of convolution?**
  $\Rightarrow$ **following!**