Filtering
Last time: Convolution and cross-correlation

• Cross correlation

\[ S[f] = w \otimes f \]
\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j) \]

• Convolution

\[ S[f] = w \ast f \]
\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m - i, n - j) \]
Last time: Boundary conditions

• Setup
  • m x m image
  • k x k kernel

• Output size?
Last time: Boundary conditions in practice

- “Full convolution”: compute if any part of kernel intersects with image
  - requires padding
  - Output size = m+k-1
- “Same convolution”: compute if center of kernel is in image
  - requires padding
  - output size = m
- “Valid convolution”: compute only if all of kernel is in image
  - no padding
  - output size = m-k+1
“Full” convolution / cross-correlation

\[(w * f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m - i, n - j)\]

• What if \(m - i < 0\)?
• What if \(m - i > \) image size
• Assume \(f\) is defined for \([-\infty, \infty]\) in both directions, just 0 everywhere else
• Same for \(w\)

\[(w * f)(m, n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} w(i, j) f(m - i, n - j)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f' = af + bg\]

\[w \otimes f' = a(w \otimes f) + b(w \otimes g)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[w' = aw + bv\]

\[w' \otimes f = a(w \otimes f) + b(v \otimes f)\]
Shift equivariance

\[ f'(m, n) = f(m - m_0, n - n_0) \]

\[ (w \otimes f')(m, n) = (w \otimes f)(m - m_0, n - n_0) \]

- Shift, then convolve = convolve, then shift
- Output of convolution does not depend on where the pixel is (modulo boundary conditions)
Filters: examples

Original (f) \ast \frac{1}{9} [\begin{array}{lll} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}] = Blur (with a mean filter) (g)

Source: D. Lowe
Filters: examples

Original (f) \* Kernel (k) = Identical image (g)

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Sharpening

• What does blurring take away?

Let’s add it back:

original + $\alpha$ = sharpened image
Sharpening
Sharpening

• What does blurring take away?

Let’s add it back:

\[ \text{original detail} + \alpha = \]
Sharpening

\[ f_{\text{sharp}} = f + \alpha (f - f_{\text{blur}}) \]
\[ = (1 + \alpha) f - \alpha f_{\text{blur}} \]
\[ = (1 + \alpha) (w * f) - \alpha (v * f) \]
\[ = ((1 + \alpha) w - \alpha v) * f \]
Sharpening filter

Original

\[
\begin{pmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{pmatrix} \ast \left( \begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix} - \frac{1}{9} \right) = \text{Sharpening filter (accentuates edges)}
\]
Another example

<table>
<thead>
<tr>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Another example
Another example
More properties of convolution

$$(w * f)(m, n) = \sum_{i} \sum_{j} w(i, j) f(m - i, n - j)$$

$$= \sum_{i} \sum_{j} w(m - i', n - j') f(i, j)$$

$$= (f * w)(m, n)$$

$i' = m - i \Rightarrow i = m - i'$

$j' = n - j \Rightarrow j = n - j'$
More properties of convolution

• Convolution is linear
• Convolution is shift-invariant
• Convolution is commutative \((w*f = f*w)\)
• Convolution is associative \((v*(w*f) = (v*w)*f)\)
• Every linear shift-invariant operation is a convolution
More convolution filters

• Mean filter

$$\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}$$

1/25

• But nearby pixels are more correlated than far-away pixels

• Weigh nearby pixels more
Gaussian filter

\[ G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \]

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]
Gaussian filter

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

- Ignore factor in front, instead, normalize filter to sum to 1

\[
\begin{array}{cccccc}
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\
0.022 & 0.098 & 0.162 & 0.098 & 0.022 \\
0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
\end{array}
\]

5x5, \sigma=1
Gaussian filter

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

- Ignore factor in front, instead, normalize filter to sum to 1

5x5, \( \sigma = 1 \)
Gaussian filter

21x21, $\sigma=0.5$

21x21, $\sigma=1$

21x21, $\sigma=3$
Difference of Gaussians (1D)
Difference of Gaussians (2D filter)
Difference of Gaussians

21x21, $\sigma=1$

21x21, $\sigma=3$

DoG filter
Difference of Gaussians

• Different standard deviations capture structures at different scales
Time complexity of convolution

• Image is $w \times h$
• Filter is $k \times k$
• Every entry takes $O(k^2)$ operations
• Number of output entries:
  • $(w+k-1)(h+k-1)$ for full
  • $wh$ for same
• Total time complexity:
  • $O(whk^2)$
Optimization: separable filters

- basic alg. is $O(r^2)$: large filters get expensive fast!
- definition: $w(x,y)$ is *separable* if it can be written as:
  \[ w(i, j) = u(i)v(j) \]
- Write $u$ as a $k \times 1$ filter, and $v$ as a $1 \times k$ filter
- Claim: $w = u \ast v$
Separable filters
Separable filters

\[ u_1 \ast \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \]

\[ \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \]

\[ \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \]

\[ \begin{bmatrix} u_1v_1 & u_1v_2 \\ \vdots & \vdots \\ \vdots & \vdots \end{bmatrix} \]
Separable filters

\[
\begin{align*}
  & u_1 \\
  & u_2 \\
  & u_3 \\
\end{align*}
\quad \ast \quad
\begin{align*}
  & v_1 \\
  & v_2 \\
  & v_3 \\
\end{align*}
\]

\[
\begin{array}{ccc}
  & u_1v_1 & u_1v_2 & u_1v_3 \\
  & u_3 & u_2 & u_1 \\
\end{array}
\]
Separable filters
Separable filters

\[ u_1 \ast v_1 \quad v_2 \quad v_3 \]

\[ u_1 \quad u_2 \quad u_3 \]

\[ u_1 \quad u_2 \quad u_3 \]

\[ u_1 v_1 \quad u_2 v_1 \quad u_1 v_3 \]

\[ u_1 v_2 \quad u_2 v_2 \]
Separable filters

\[ u_1 * v_1 \quad v_2 \quad v_3 \]

\[
\begin{array}{ccc}
    u_1 & u_2 & u_3 \\
    u_1 & u_2 & u_3 \\
    u_1 & u_2 & u_3 \\
\end{array}
\]

\[
\begin{array}{ccc}
    v_1 & v_2 & v_3 \\
    v_1 & v_2 & v_3 \\
    v_1 & v_2 & v_3 \\
\end{array}
\]

\[
\begin{array}{ccc}
    u_1v_1 & u_1v_2 & u_1v_3 \\
    u_2v_1 & u_2v_2 & u_2v_3 \\
    \quad & \quad & \quad \\
\end{array}
\]

38
Separable filters

\[
\begin{array}{c}
\begin{array}{ccc}
  & v_1 & v_2 \\
 u_1 & v_3 \\
 u_2 & \\
 u_3 &
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{ccc}
 u_1 & u_3 & \\
 u_2 & u_2 & \\
 u_3 & u_1 &
\end{array}
\end{array}
\]

\[
\begin{array}{c|c|c}
 u_1v_1 & u_1v_2 & u_1v_3 \\
 u_2v_1 & u_2v_2 & u_2v_3 \\
 u_3v_1 & \\
\end{array}
\]
Separable filters

\[ u_1 \ast v_1 \]

\[ u_2 \ast v_2 \]

\[ u_3 \ast v_3 \]

\[ u_1 \]

\[ u_2 \]

\[ u_3 \]

\[ v_1 \]

\[ v_2 \]

\[ v_3 \]

\[ u_1 \]

\[ u_2 \]

\[ u_3 \]

\[ u_1 \]

\[ u_2 \]

\[ u_3 \]

\[
\begin{array}{ccc}
  u_1 v_1 & u_1 v_2 & u_1 v_3 \\
  u_2 v_1 & u_2 v_2 & u_2 v_3 \\
  u_3 v_1 & u_3 v_2 & \\
\end{array}
\]
Separable filters

\[ u_1 \ast v_1 \ast v_2 \ast v_3 \]

\[
\begin{array}{ccc}
  u_1 & v_1 & v_2 & v_3 \\
  u_2 & u_1 & v_2 & v_3 \\
  u_3 & u_2 & u_1 & v_3 \\
  u_3 & u_2 & u_1 & v_3 \\
\end{array}
\]

\[ W \]
Separable filters

\[ w * f = (u * v) * f \]
\[ = u * (v * f) \]

- Time complexity of original : \( O(whk^2) \)
- Time complexity of separable version : \( O(whk) \)
Convolution is everywhere
Why is convolution important?

- Shift equivariance is a crucial property
Why is convolution important?

• We *like* linearity
  • Linear functions behave predictably when input changes
  • Lots of theory just easier with linear functions

• *All linear shift-equivariant systems can be expressed as a convolution*
Non-linear filters: Thresholding

\[ g(m, n) = \begin{cases} 
255, & f(m, n) > A \\
0, & \text{otherwise} 
\end{cases} \]
Non-linear filters: Rectification

• $g(m,n) = \max(f(m,n), 0)$

• Crucial component of modern convolutional networks
Non-linear filters

• Sometimes mean filtering does not work
Non-linear filters

• Sometimes mean filtering does not work
Non-linear filters

• Mean is sensitive to outliers
• Median filter: Replace pixel by median of neighbors
Non-linear filters
Takeaway

• Two general recipes:
  • convolution
  • cross-correlation

• Properties
  • Shift-equivariant: a sensible thing to require
  • Linearity: convenient

• Can be used for smoothing, sharpening

• Also main component of CNNs