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

$$S[f] = w * f$$

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

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

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

$$w' = aw + bv$$

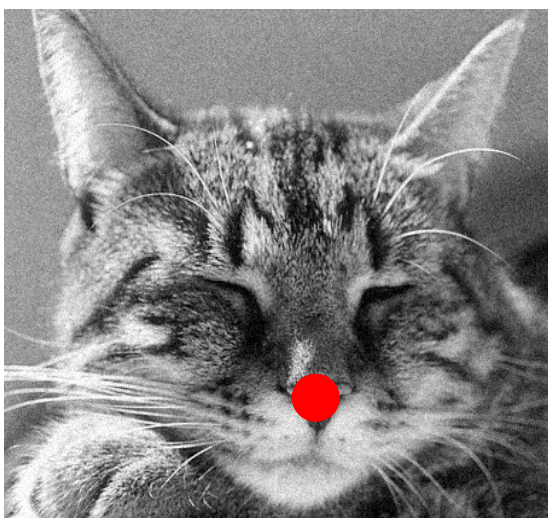
$$w' \otimes f = a(w \otimes f) + b(v \otimes f)$$

Shift equivariance

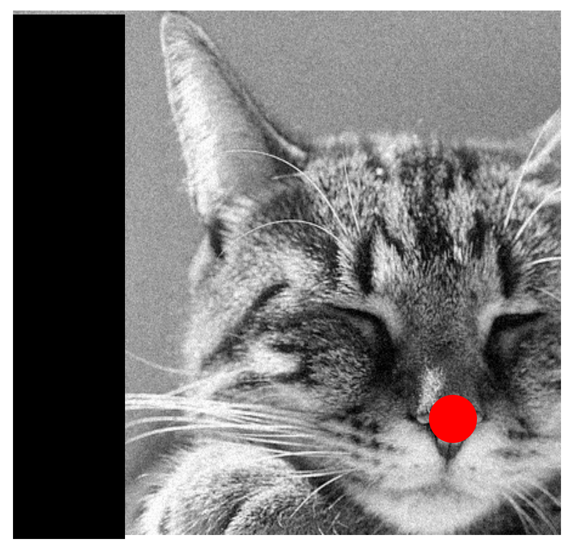
$$f'(m,n) = f(m - m_0, n - n_0)$$

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

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

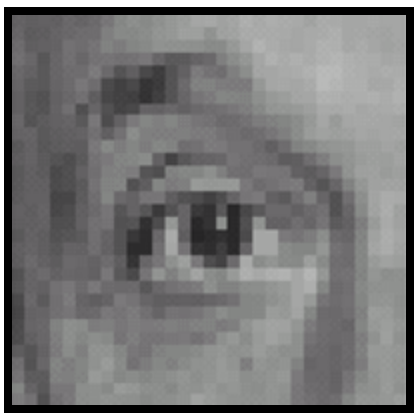


f



f'

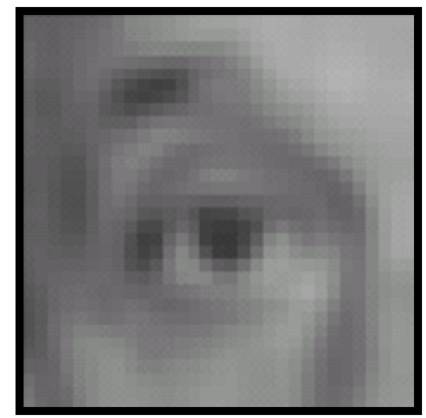
Filters: examples



Original (f)

$$* \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad =$$

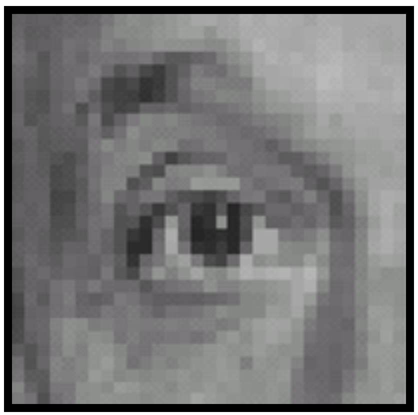
Kernel (k)



Blur (with a mean filter) (g)

Source: D. Lowe

Filters: examples



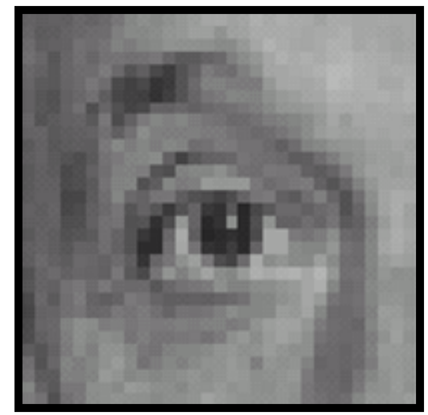
Original (f)

$*$

0	0	0
0	1	0
0	0	0

Kernel (k)

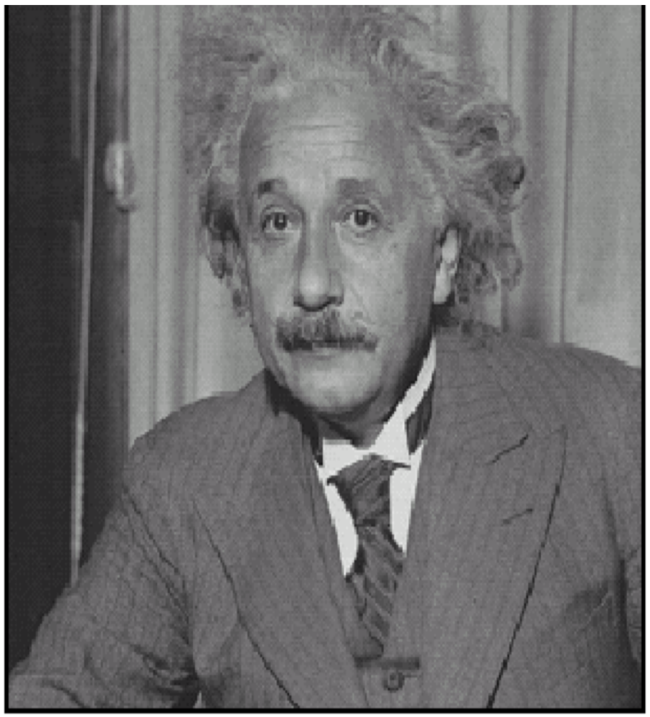
$=$



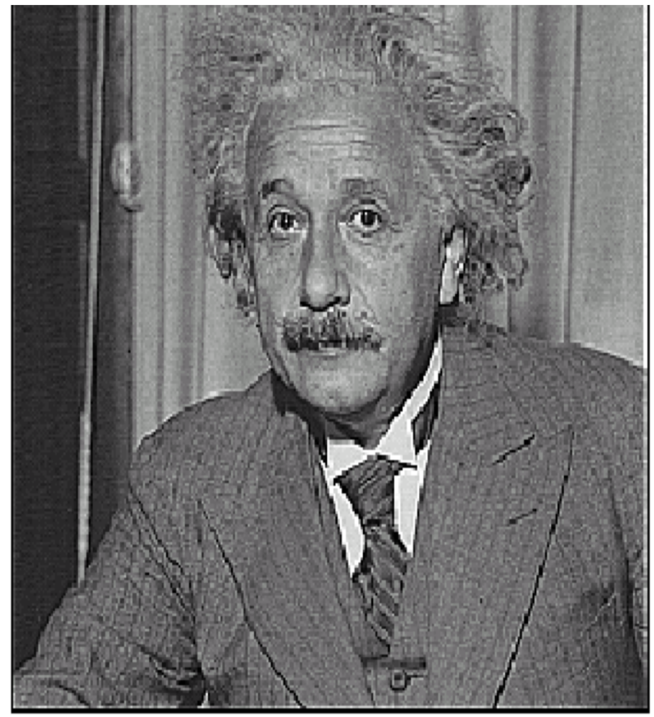
Identical image (g)

Source: D. Lowe

Sharpening



before

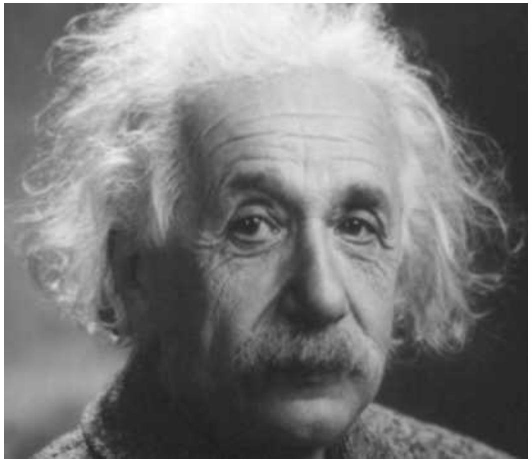


after

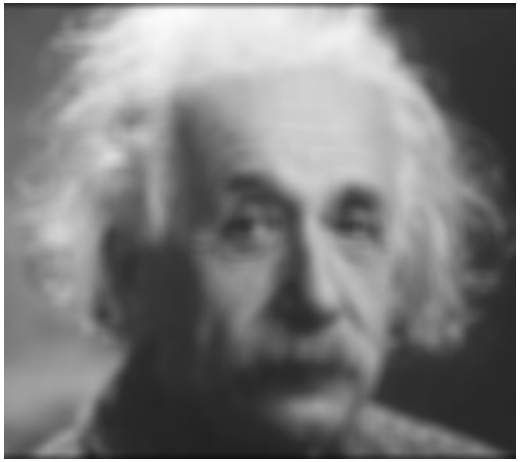
Source: D. Lowe

Sharpening

- What does blurring take away?

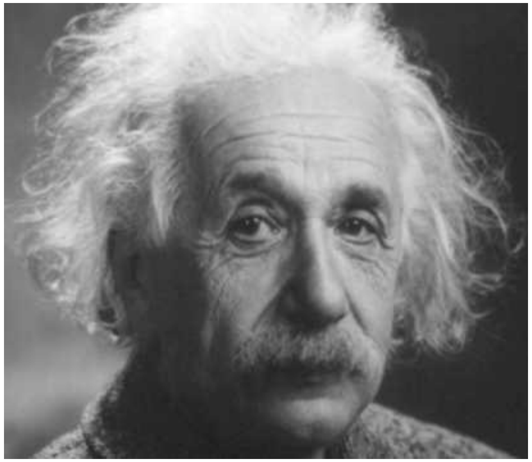


–



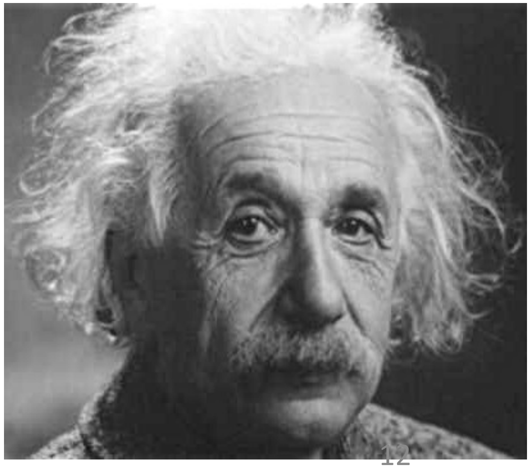
=

Let’s add it back:

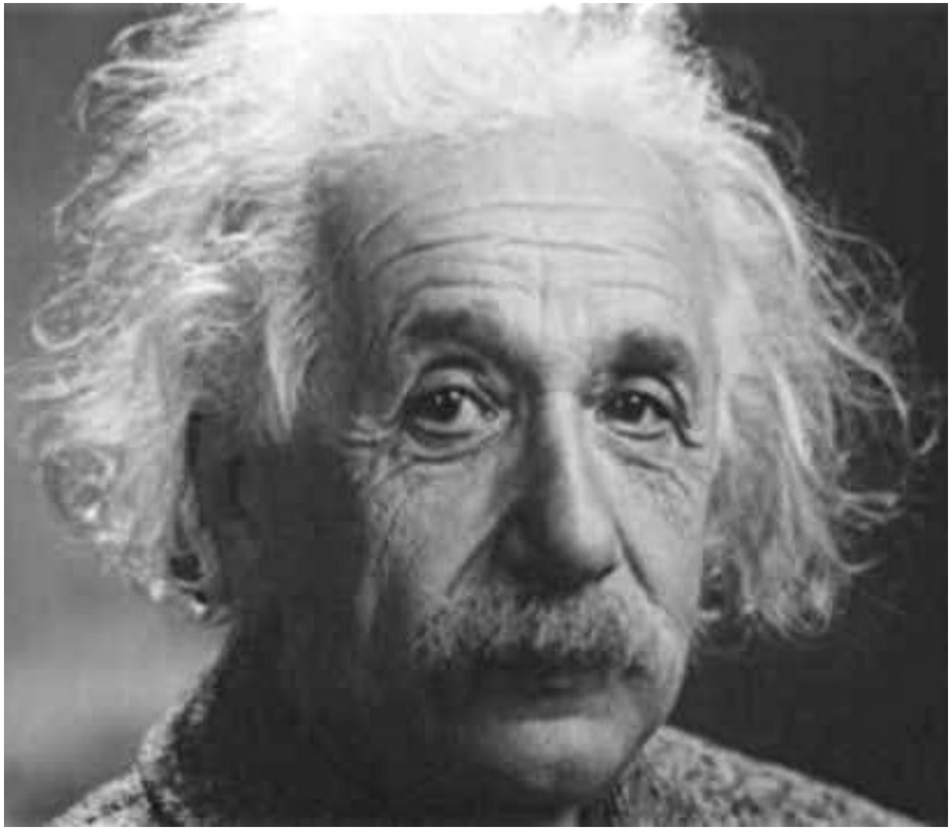


+ α

=

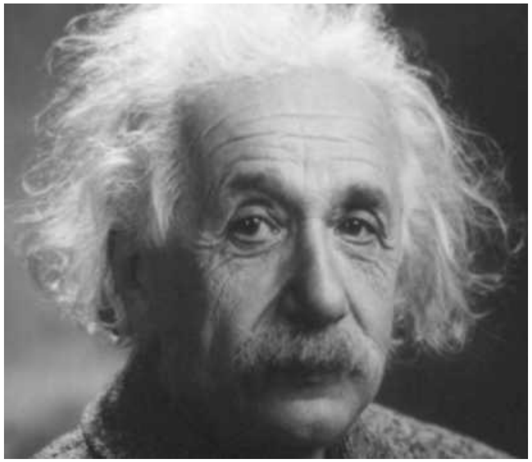


Sharpening

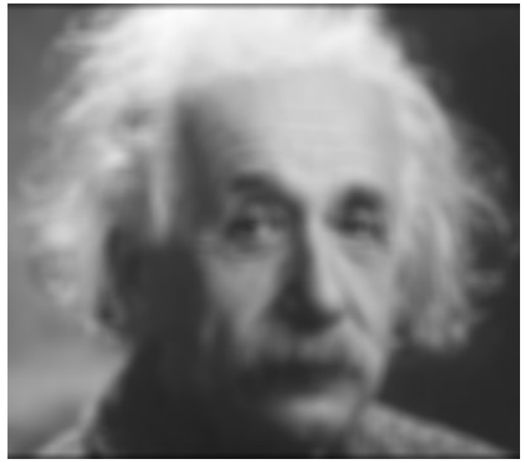


Sharpening

- What does blurring take away?

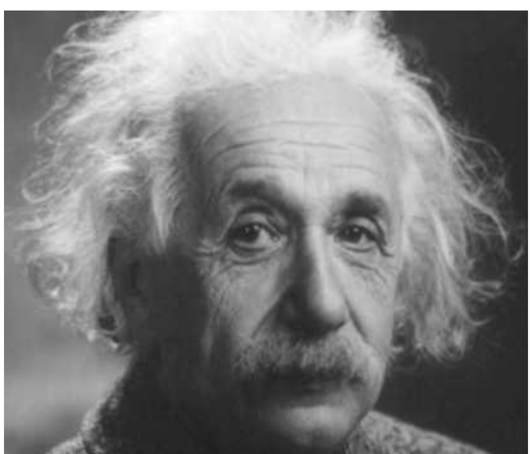


–



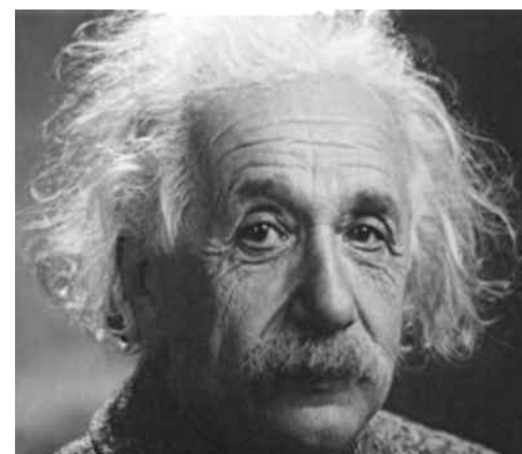
=

Let’s add it back:



+ α

=



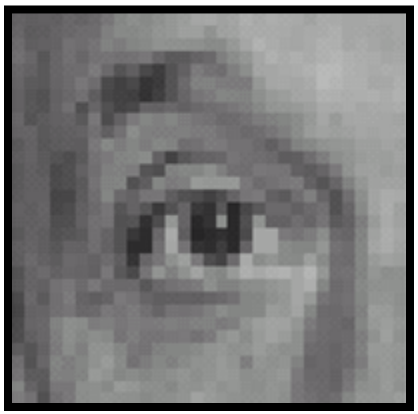
Sharpening

$$
\begin{aligned}
f_{sharp} &= f + \alpha(f - f_{blur}) \\
&= (1+\alpha)f - \alpha f_{blur} \\
&= (1+\alpha)(w * f) - \alpha(v * f)
\end{aligned}
$$

$$
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\qquad
\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
$$

$$
= ((1+\alpha)w - \alpha v) * f
$$

Sharpening filter



Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

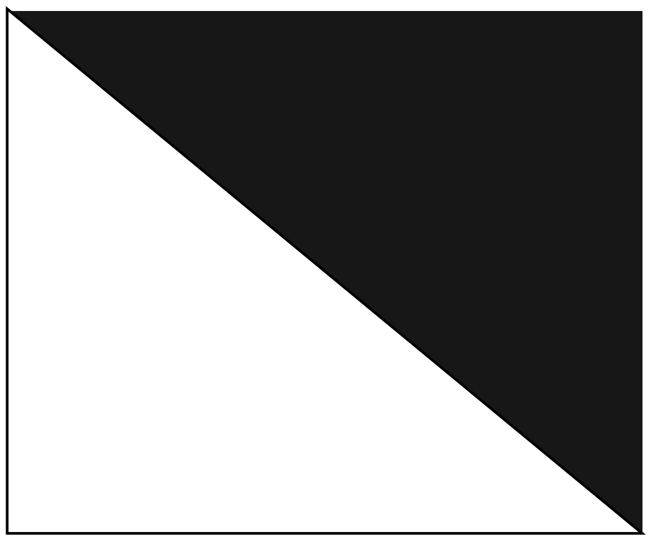
Sharpening filter
(accentuates edges)

Source: D. Lowe

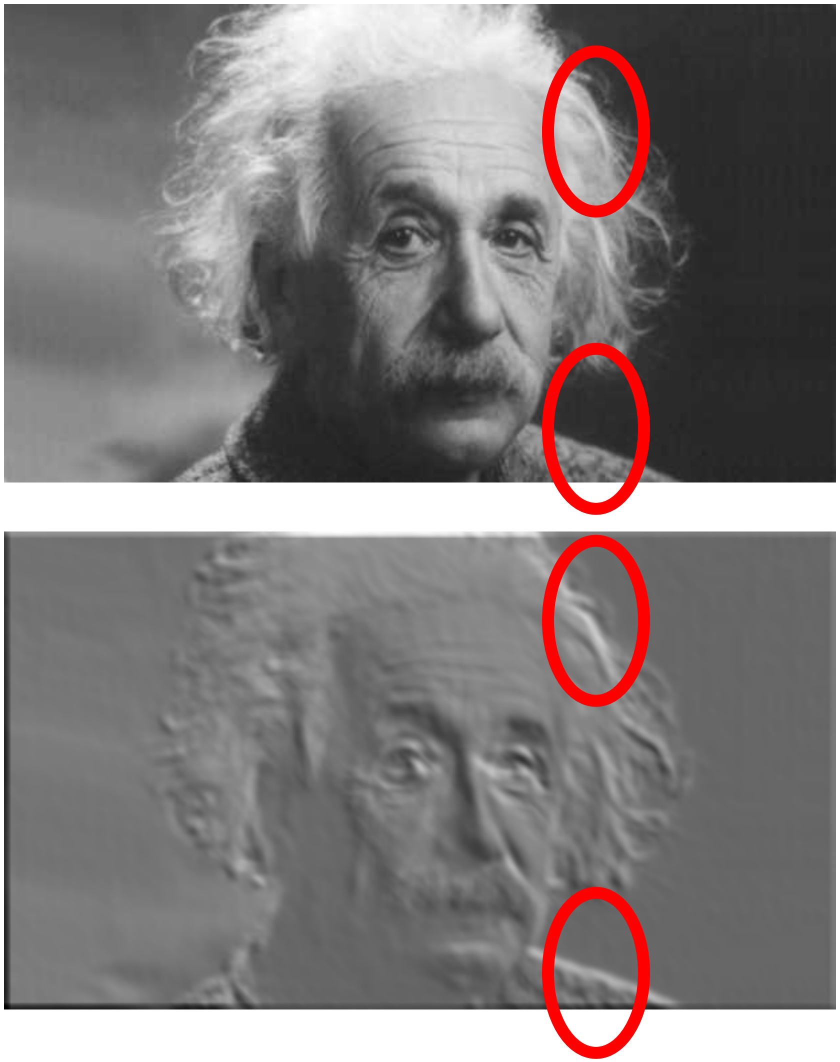
Another example

-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	-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	-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	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	-1

Another example



Another example



More properties of convolution

$$
\begin{aligned}
(w * f)(m,n) &= \sum_i \sum_j w(i,j) f(m-i, n-j) \\
&= \sum_i \sum_j w(m-i', n-j') f(i,j) \\
&= (f * w)(m,n)
\end{aligned}
$$

$$
\begin{aligned}
i' = m - i &\Rightarrow i = m - i' \\
j' = n - j &\Rightarrow j = n - j'
\end{aligned}
$$

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

1/25

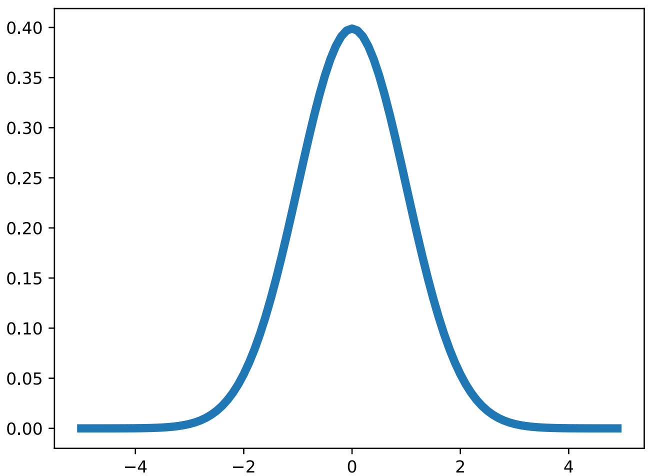
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

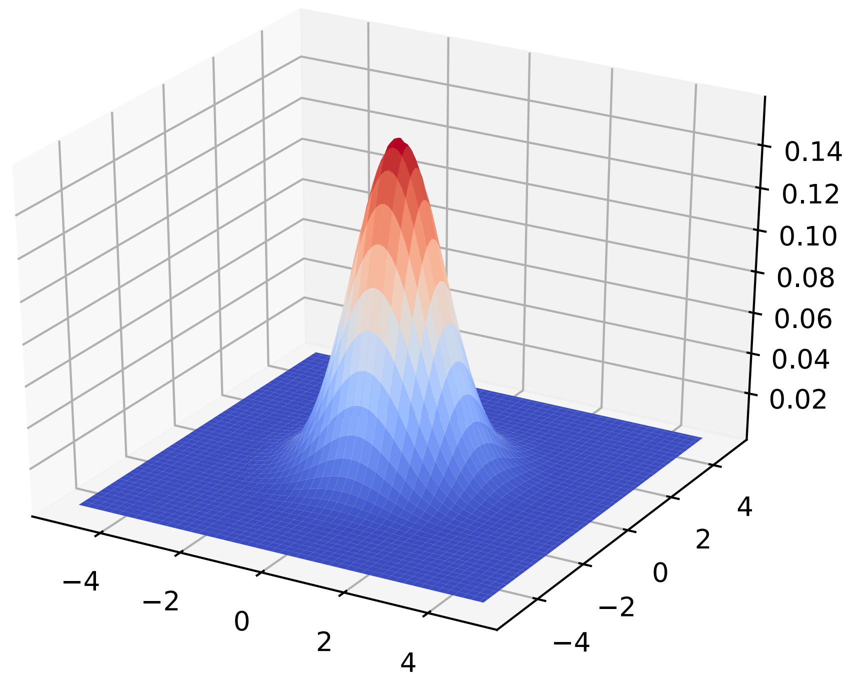
- But nearby pixels are more correlated than far-away pixels
- Weigh nearby pixels more

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

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

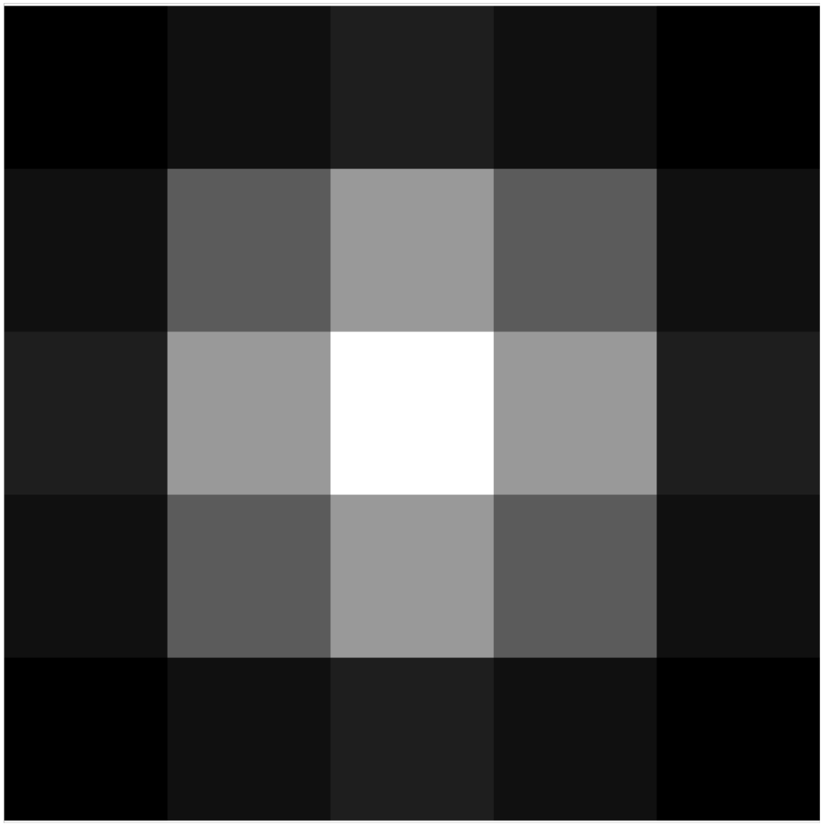
- Ignore factor in front, instead, normalize filter to sum to 1

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

5x5, σ=1

Gaussian filter

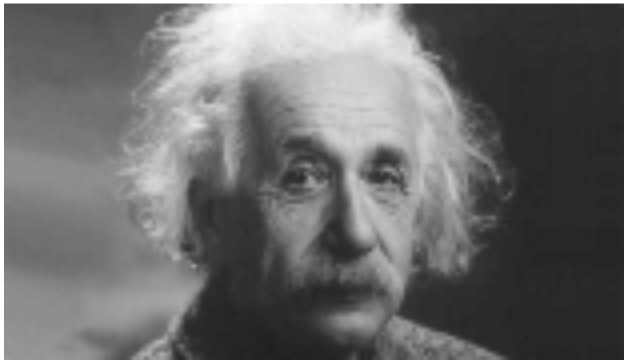
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

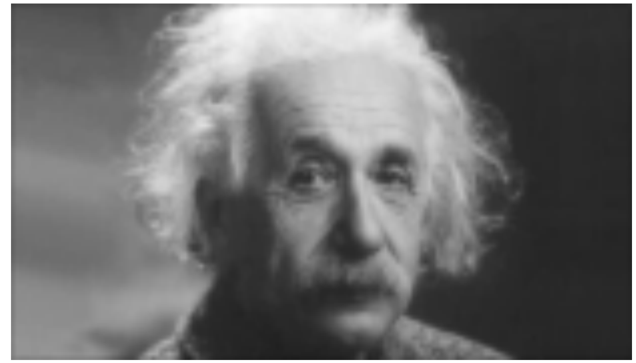


- Ignore factor in front, instead, normalize filter to sum to 1

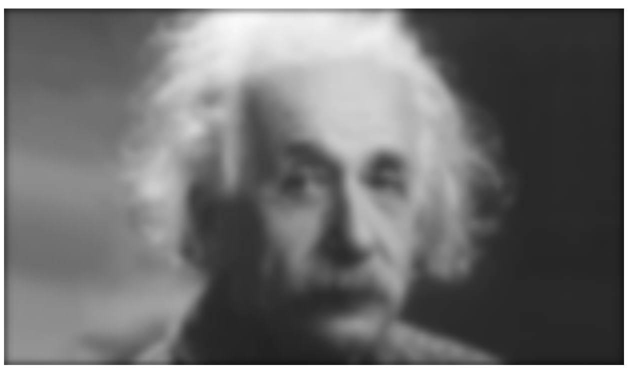
5x5, σ=1

Gaussian filter

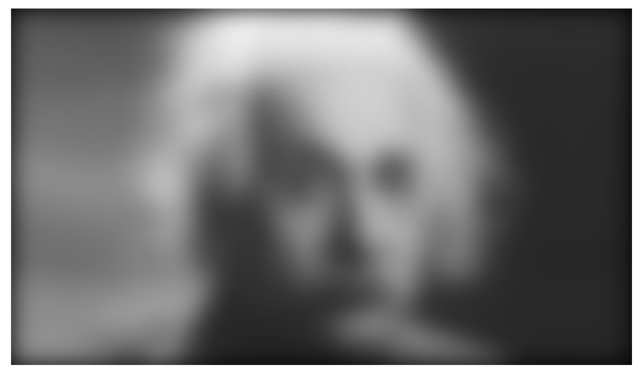




21x21, σ=0.5

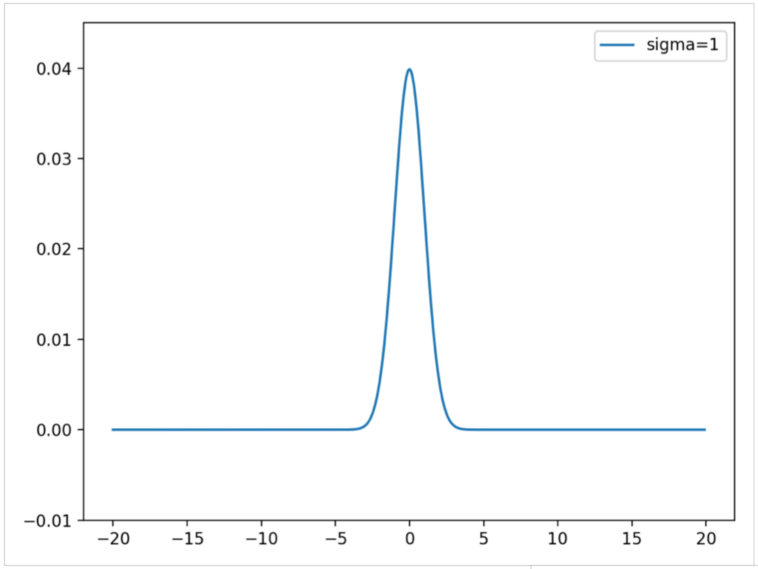


21x21, σ=1

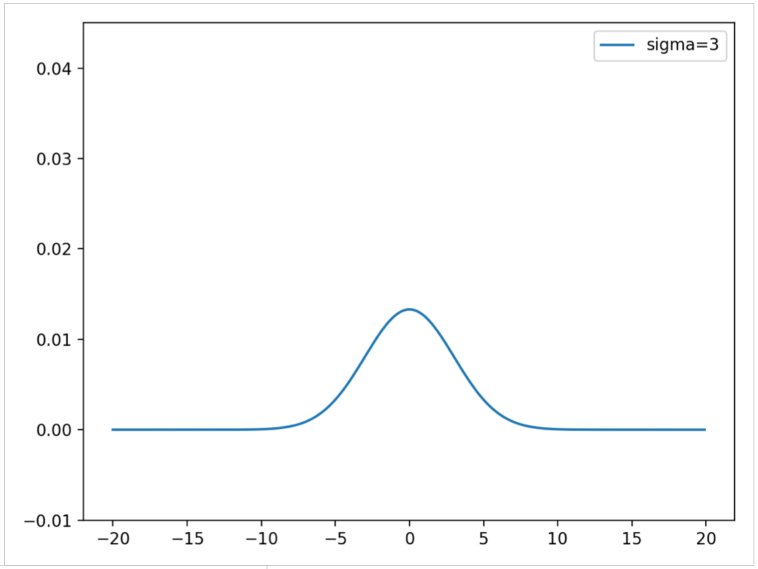


21x21, σ=3

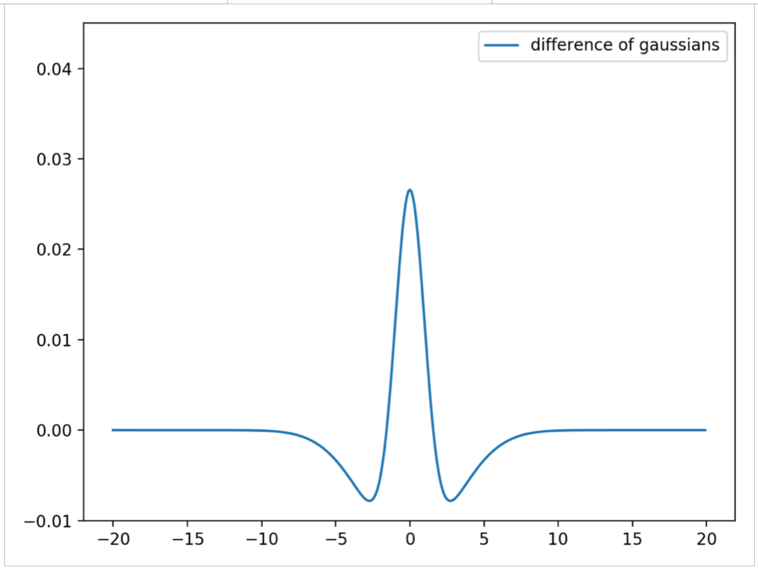
Difference of Gaussians (1D)



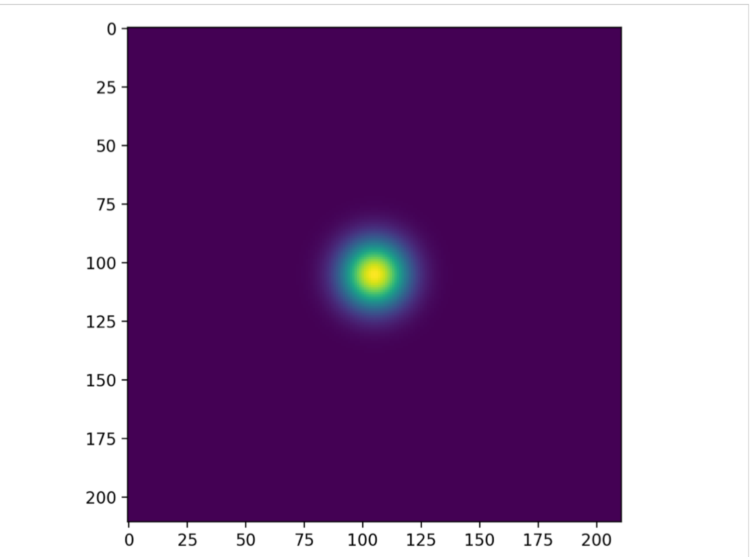
-



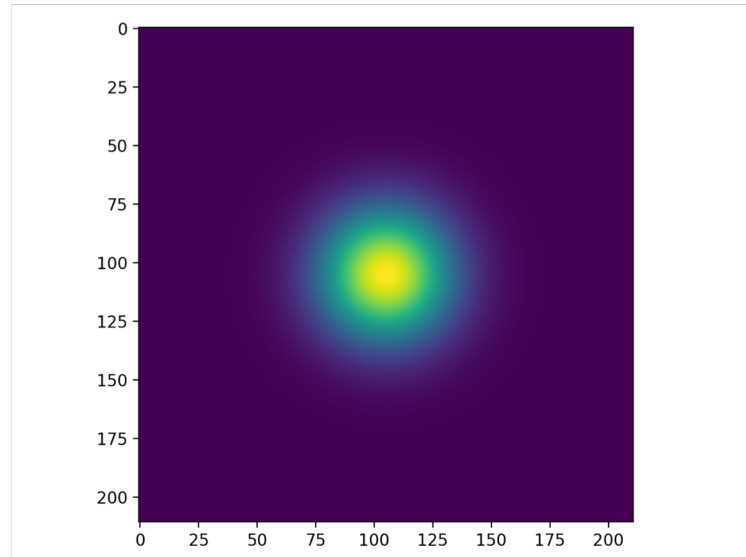
=



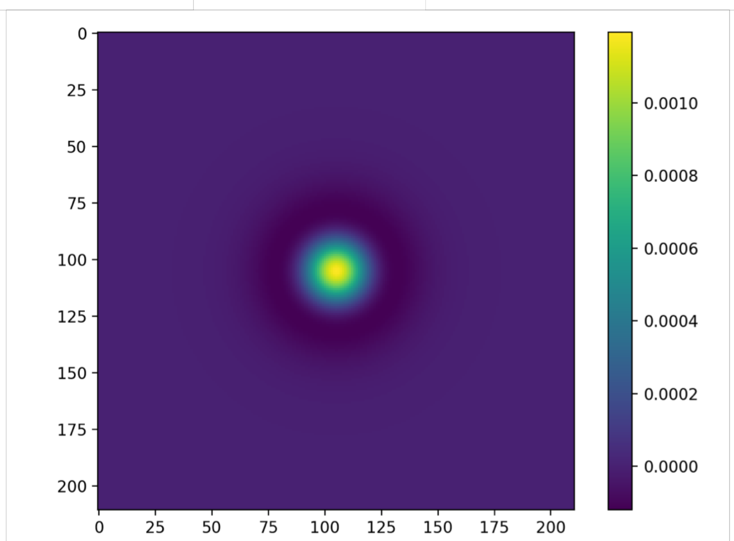
Difference of Gaussians (2D filter)



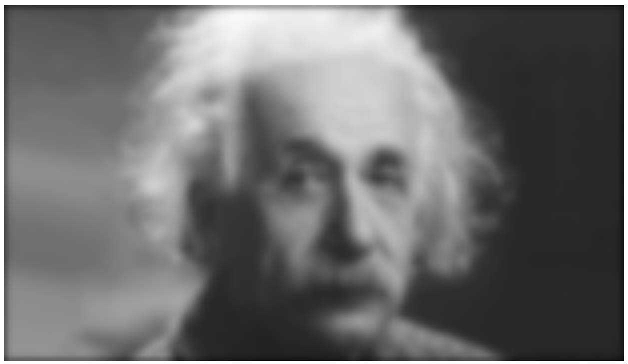
-



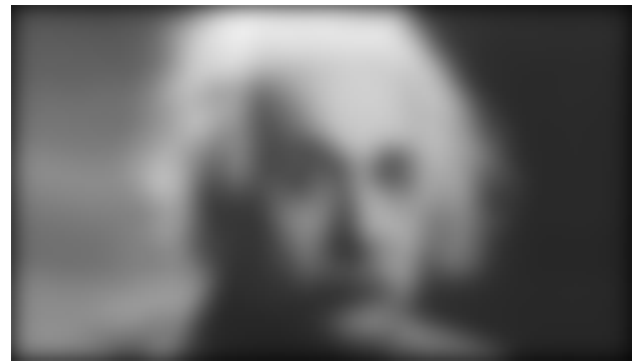
=



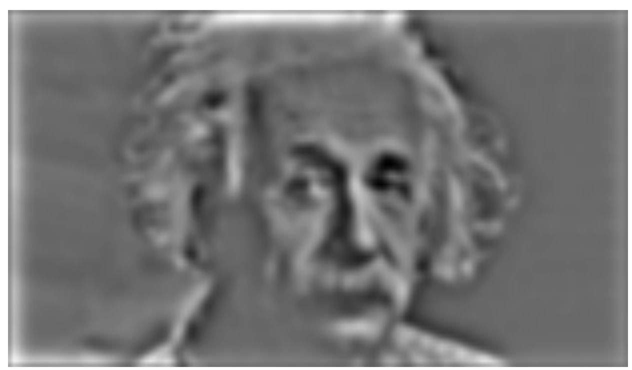
Difference of Gaussians



21x21, σ=1



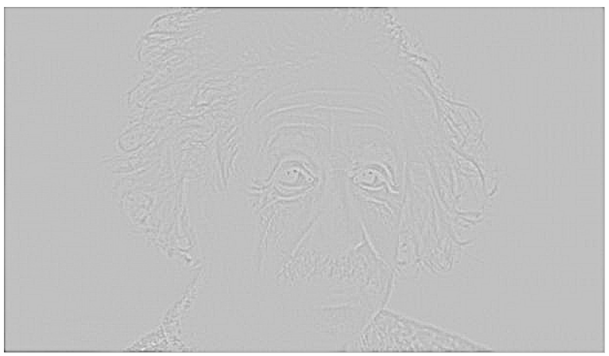
21x21, σ=3

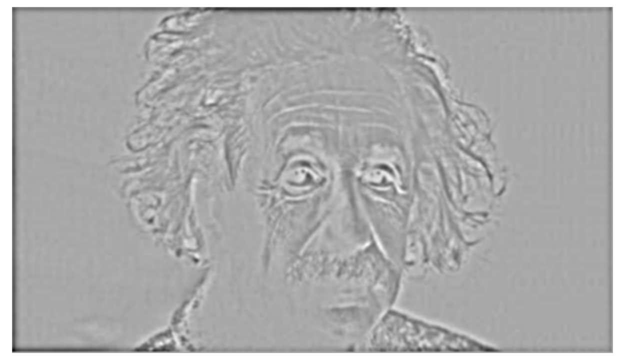


DoG filter

Difference of Gaussians

- Different standard deviations capture structures at different scales





Time complexity of convolution

- Image is w x h
- Filter is k x 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:
  - $O(whk^2)$

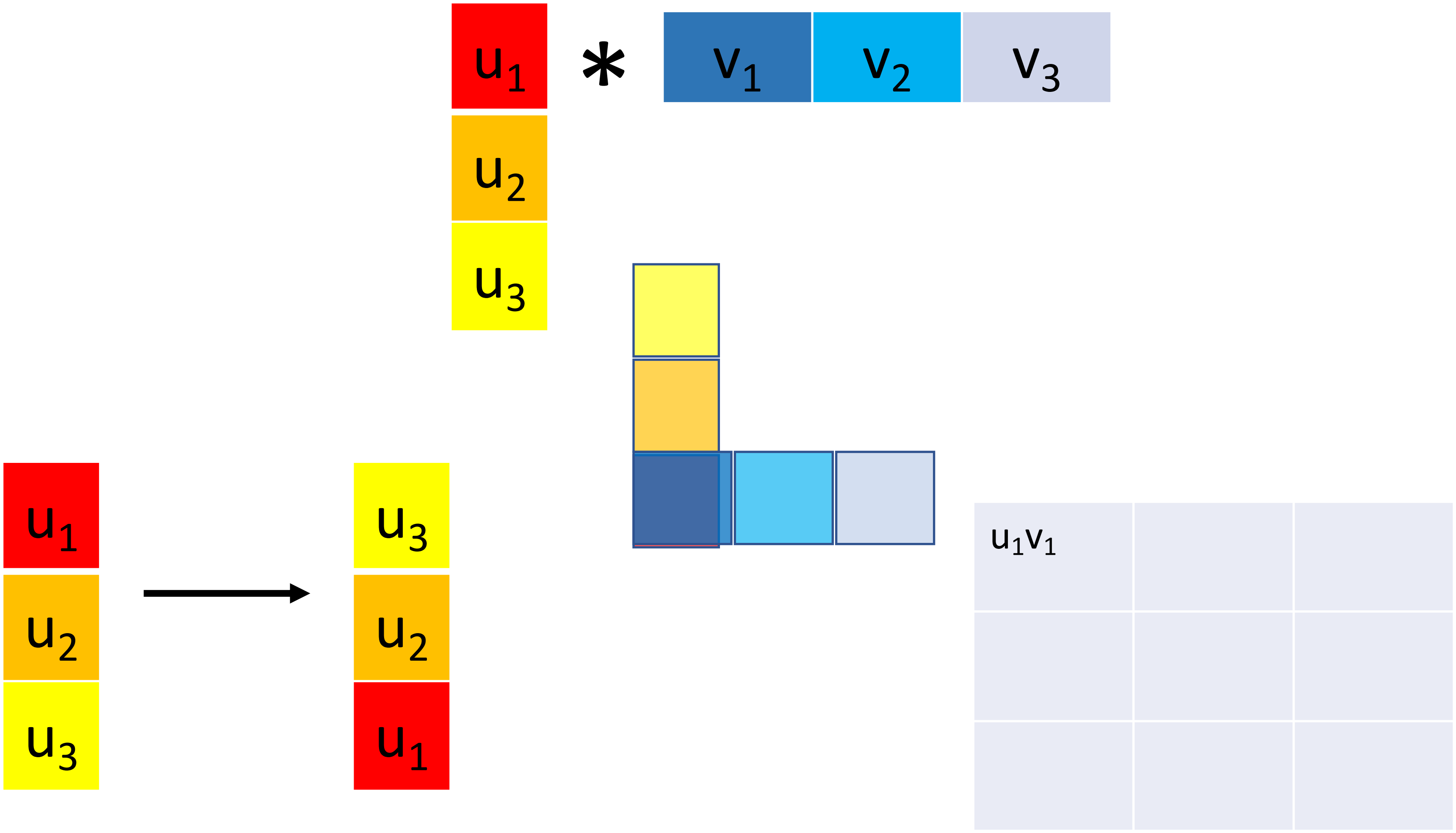
Optimization: separable filters

- basic alg. is $O(r^2)$: large filters get expensive fast!
- definition: $w(x,y)$ is *separable* if it can be written as:

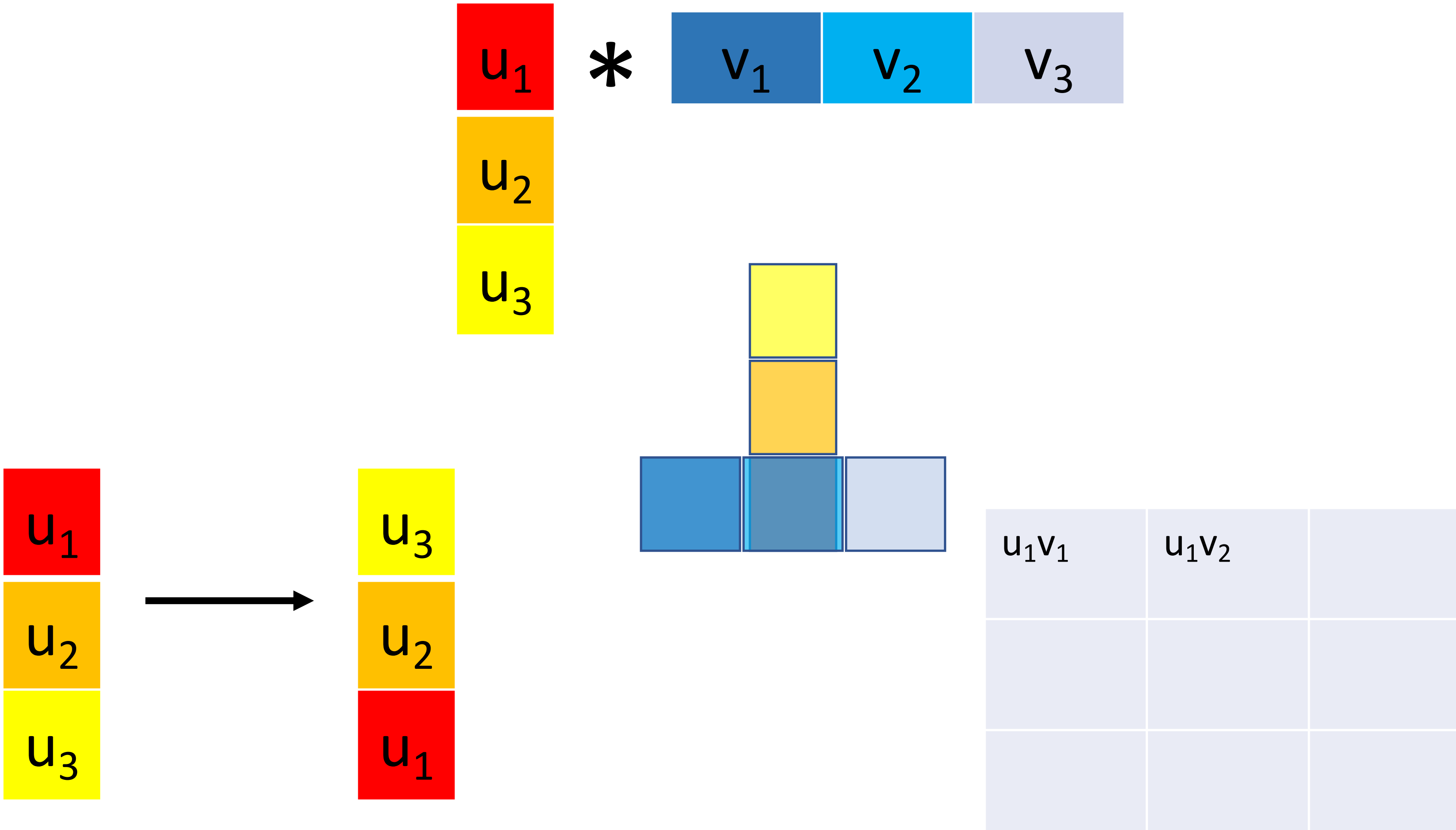
$$w(i,j) = u(i)v(j)$$

- Write u as a k x 1 filter, and v as a 1 x k filter
- Claim: $w = u * v$

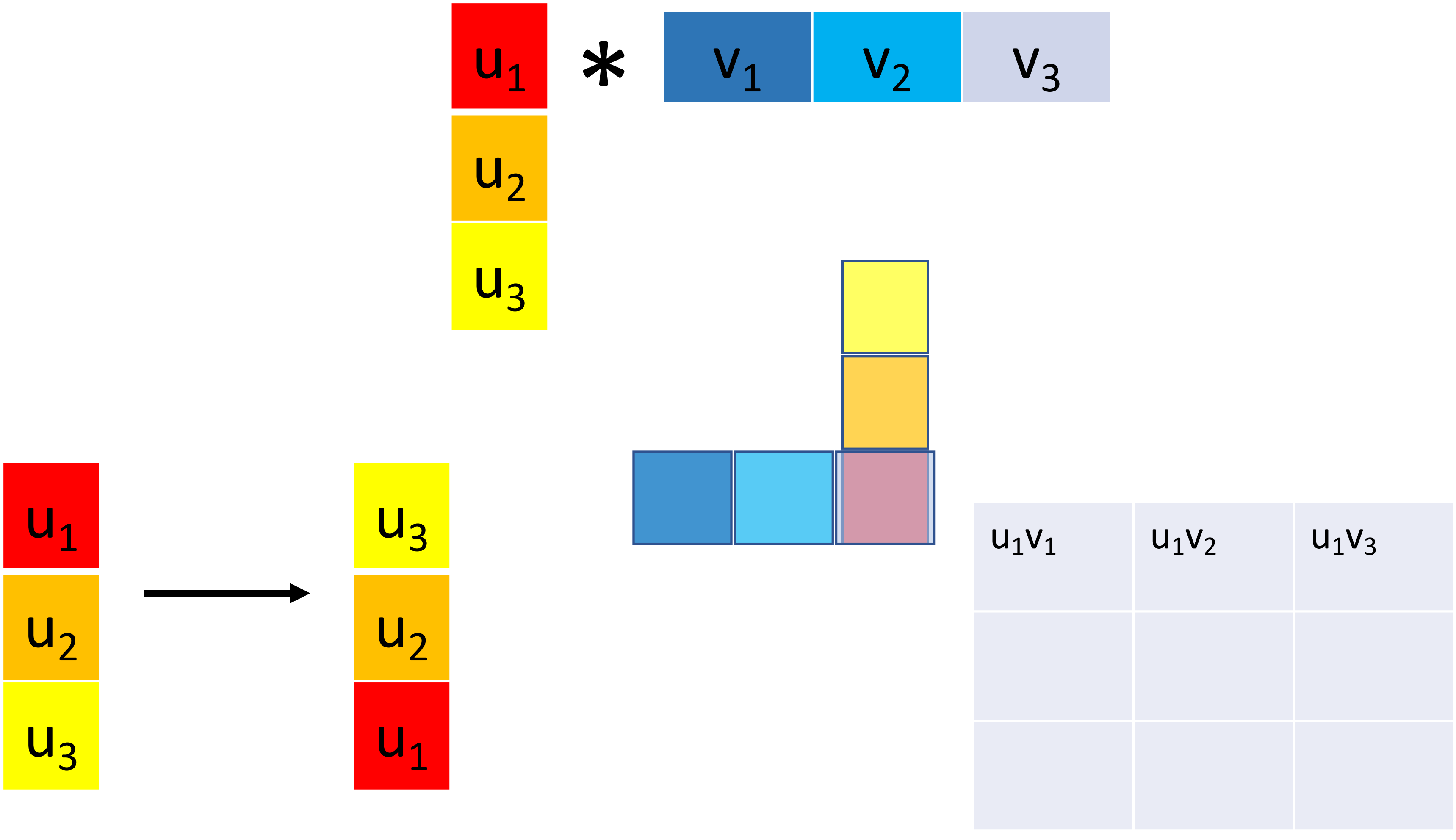
Separable filters



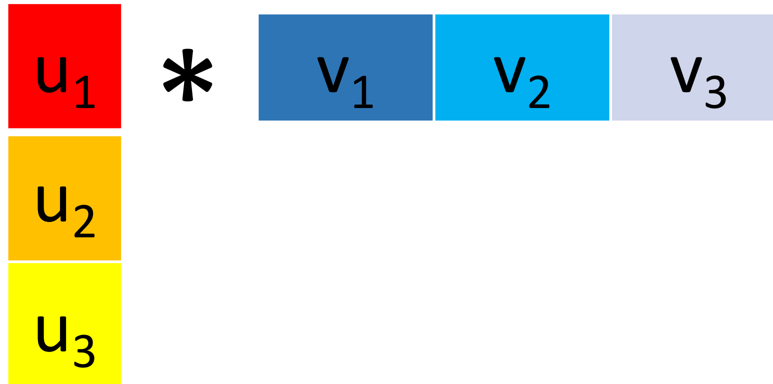
Separable filters

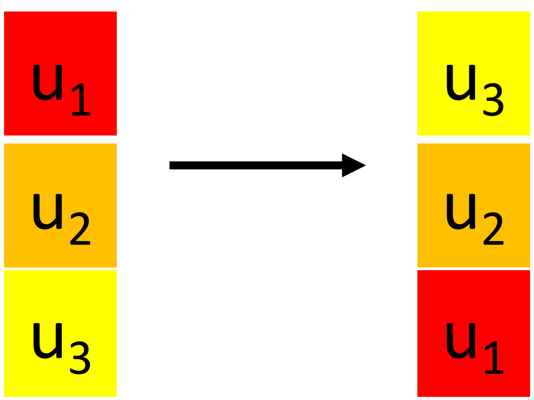


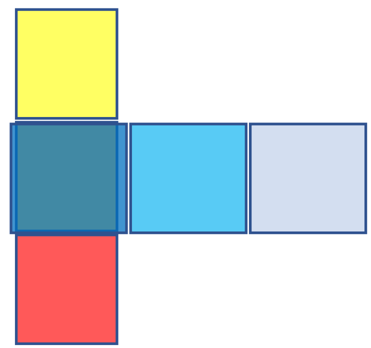
Separable filters



Separable filters

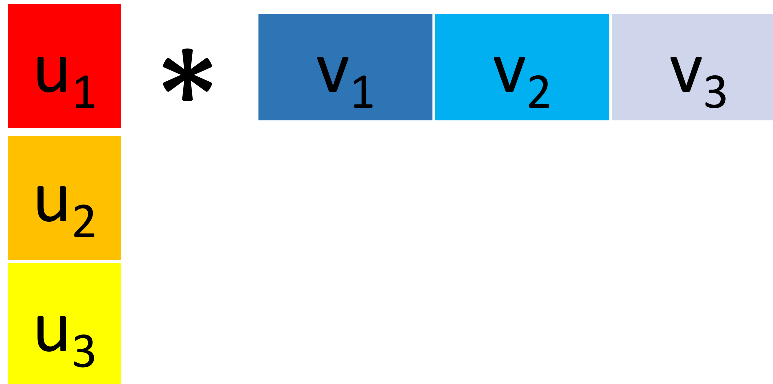


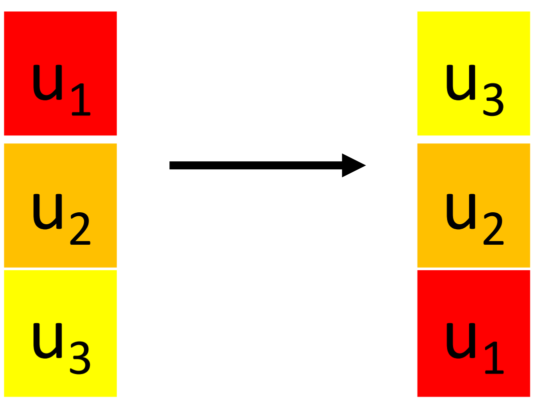


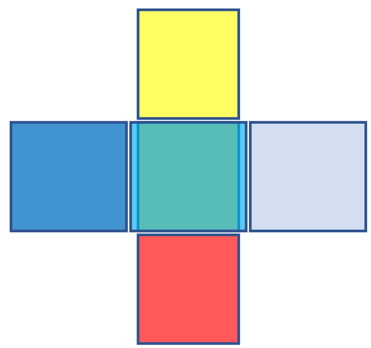


u_1v_1	u_1v_2	u_1v_3
u_2v_1		

Separable filters

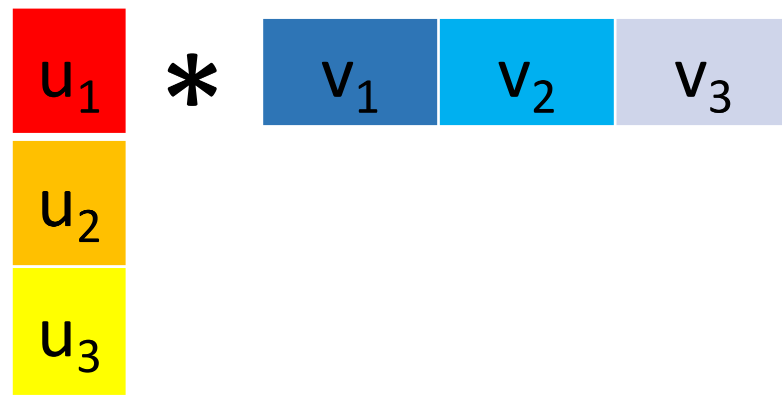


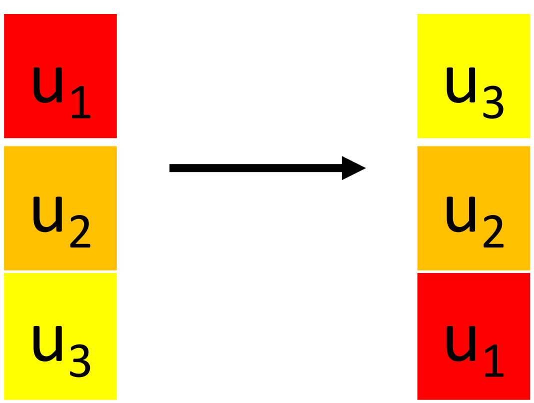


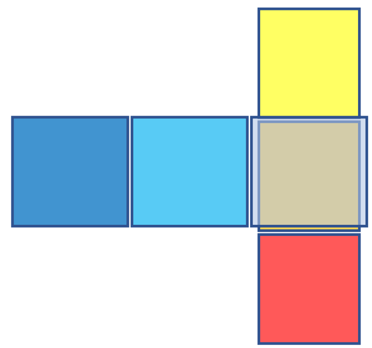


u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	

Separable filters

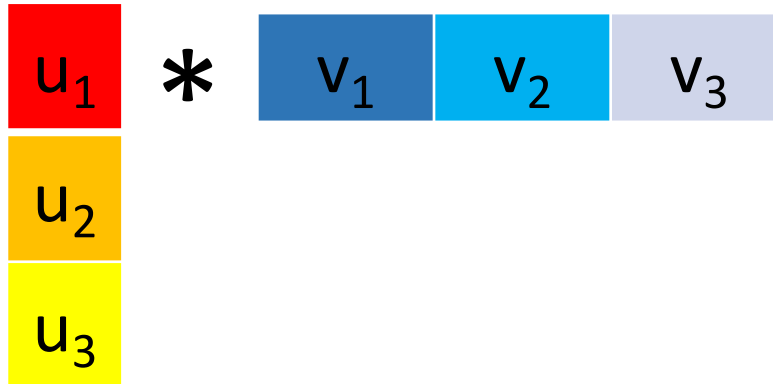


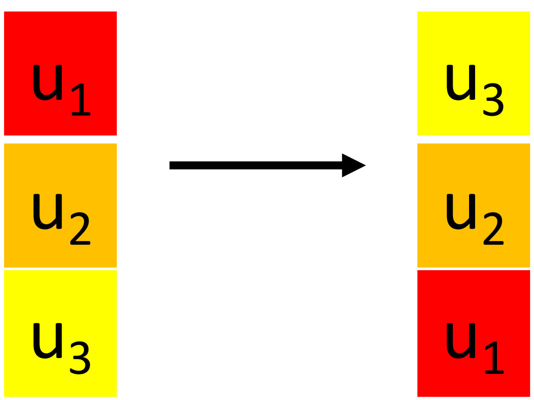


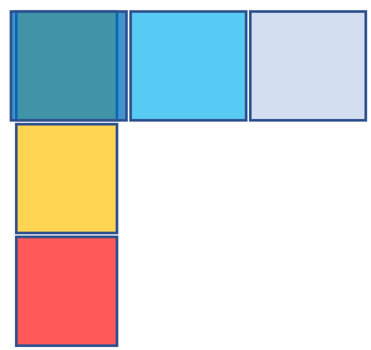


u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	u_2v_3

Separable filters

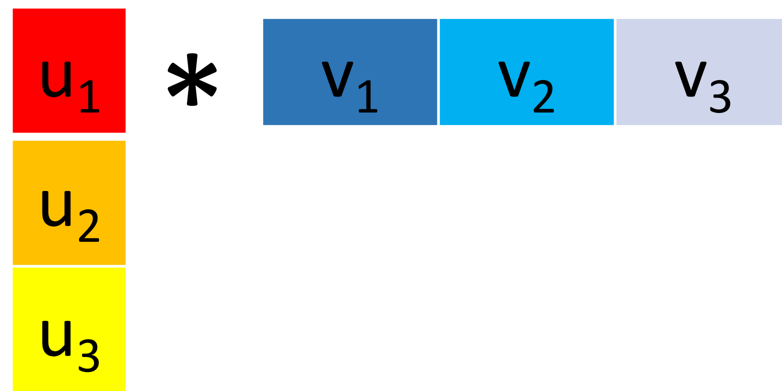


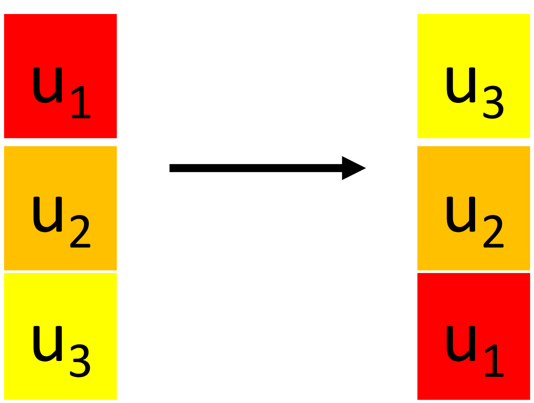


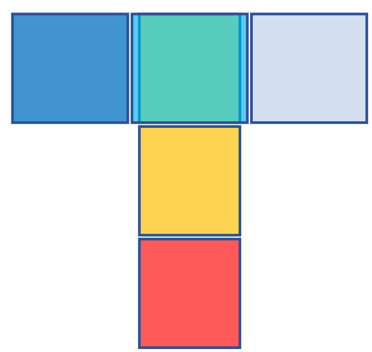


u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	u_2v_3
u_3v_1		

Separable filters

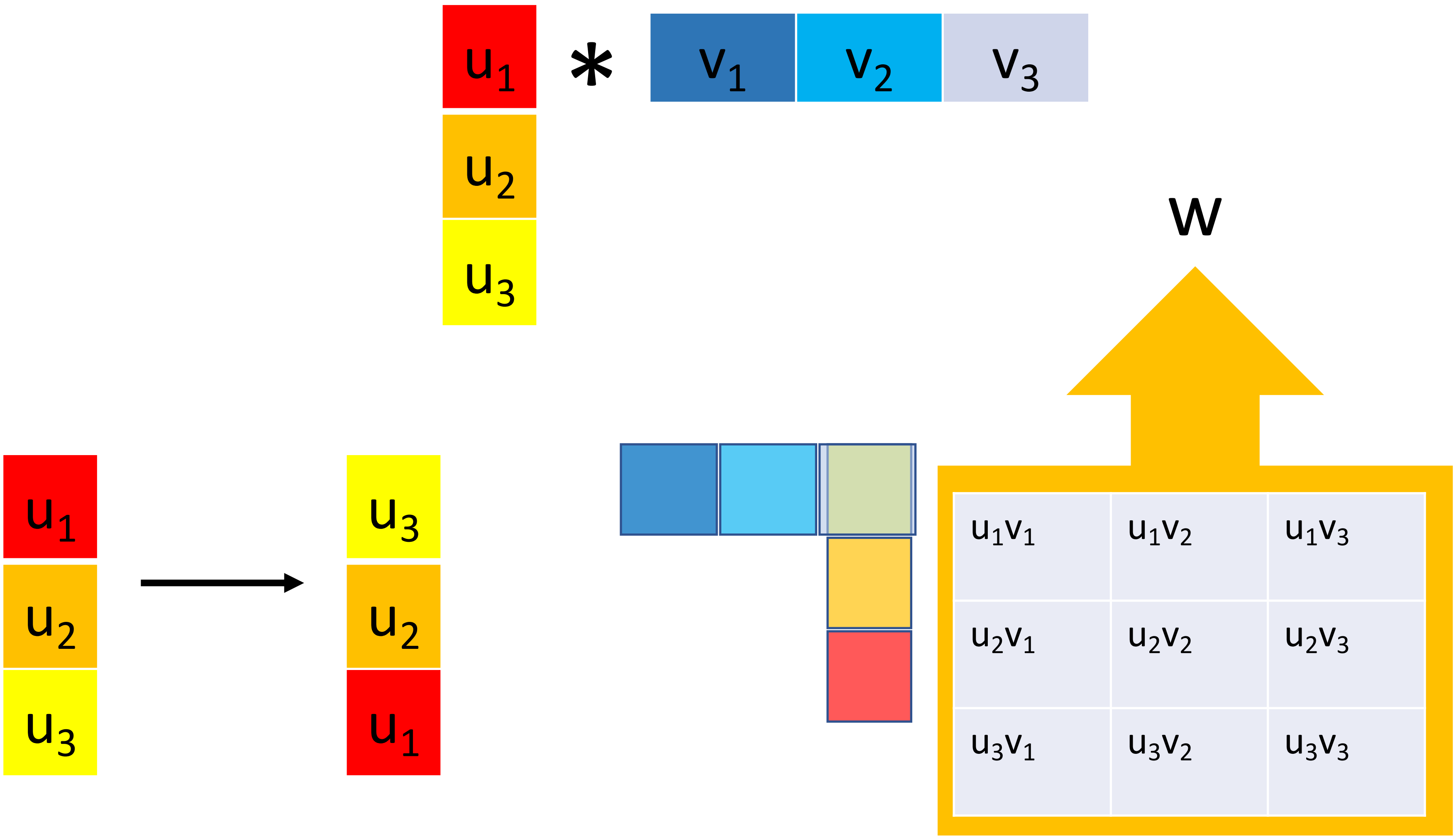






u_1v_1	u_1v_2	u_1v_3
u_2v_1	u_2v_2	u_2v_3
u_3v_1	u_3v_2	

Separable filters

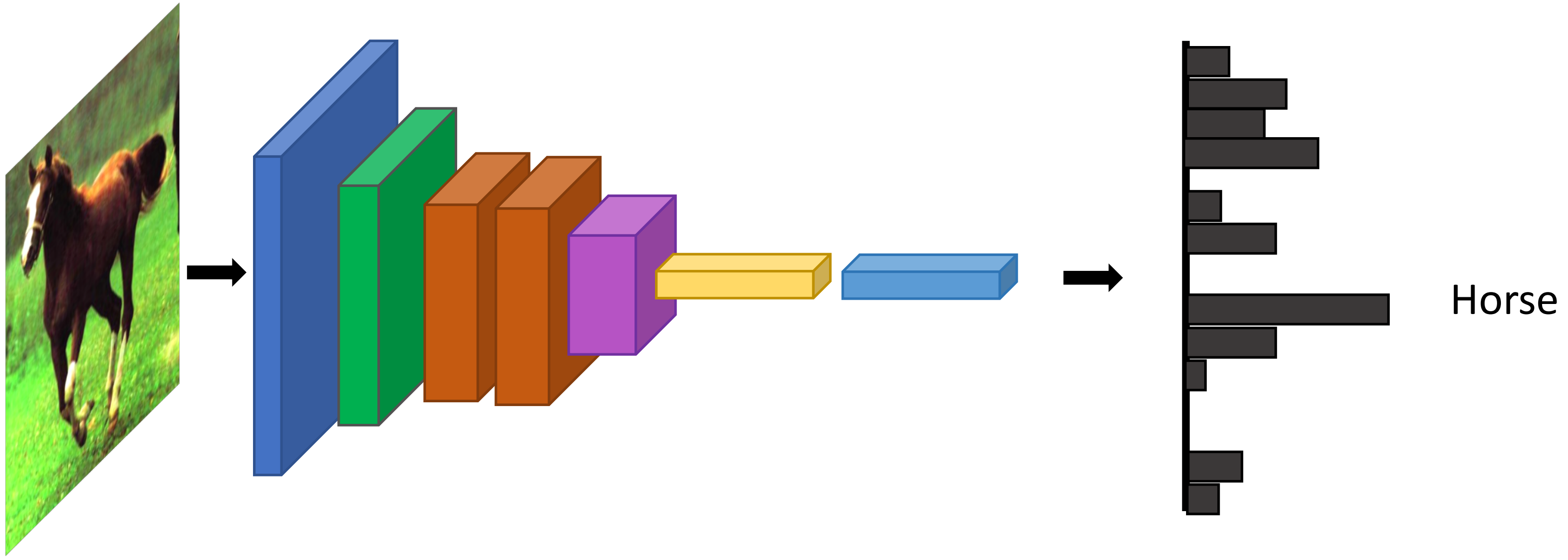


Separable filters

$$\begin{aligned} w * f &= (u * v) * f \\ &= u * (v * f) \end{aligned}$$

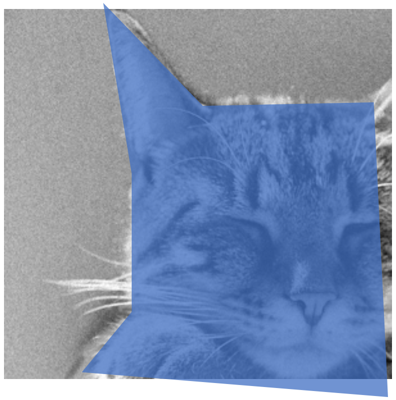
- Time complexity of original : $O(whk^2)$
- Time complexity of separable version : $O(whk)$

Convolution is everywhere



Why is convolution important?

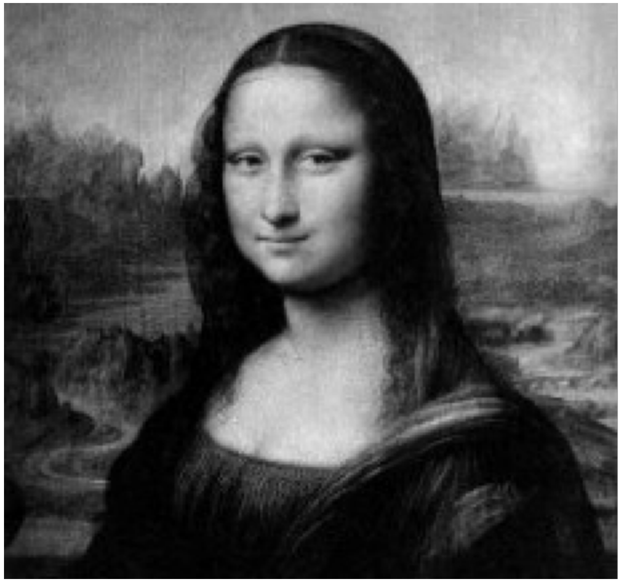
- Shift equivariance is a crucial property



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



$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

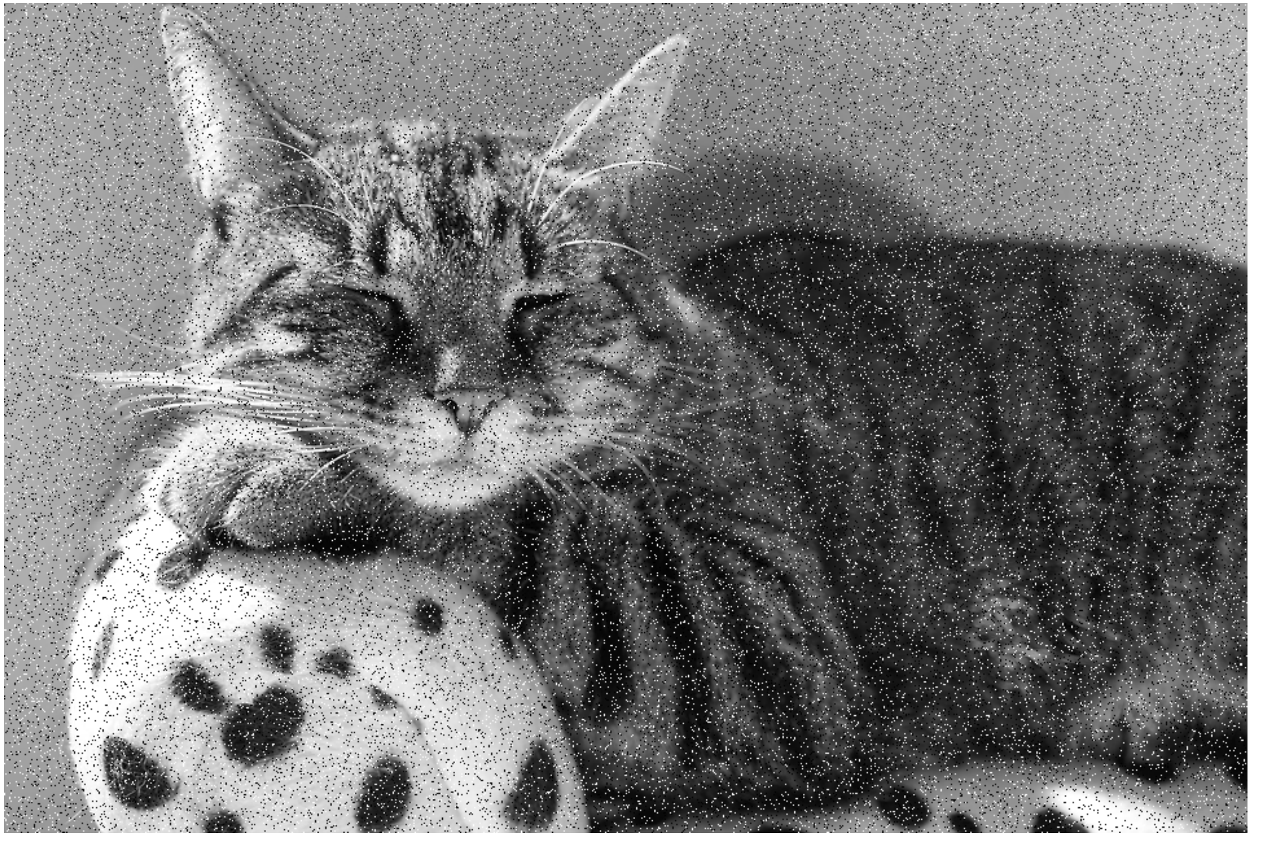
Non-linear filters: Rectification

- 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