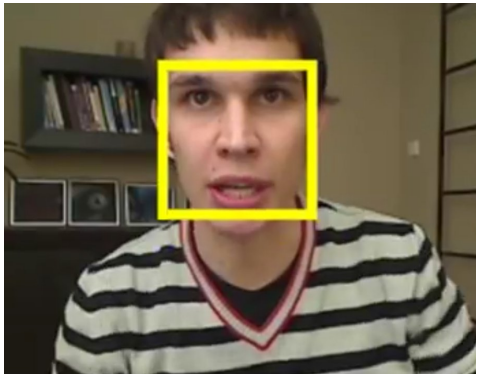
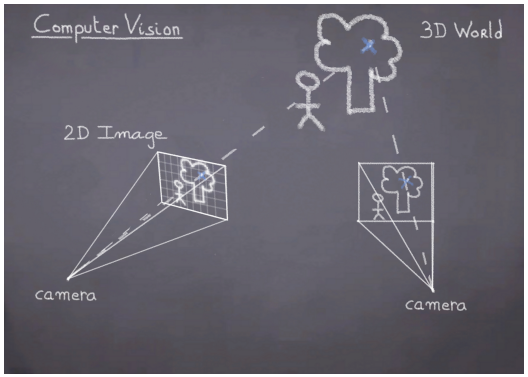
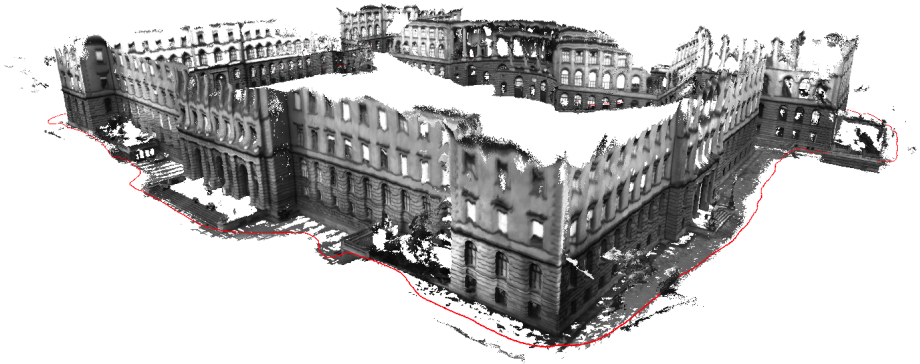
Computer vision and image processing introduction

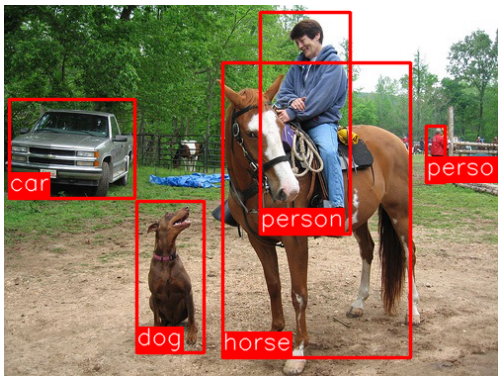
Ronan Sicre
Credits to Yannis Avrithis https://sif-dlv.github.io/

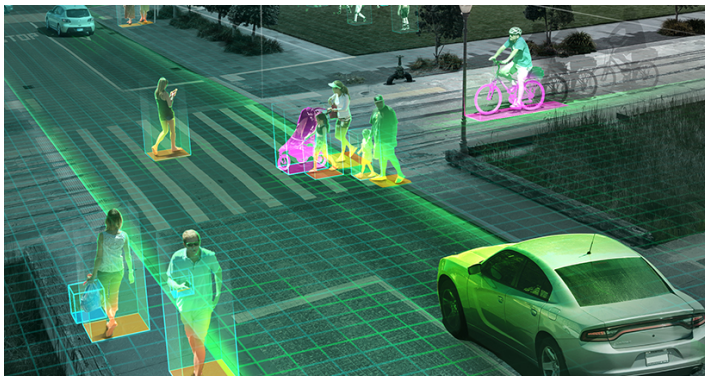
computer vision in images



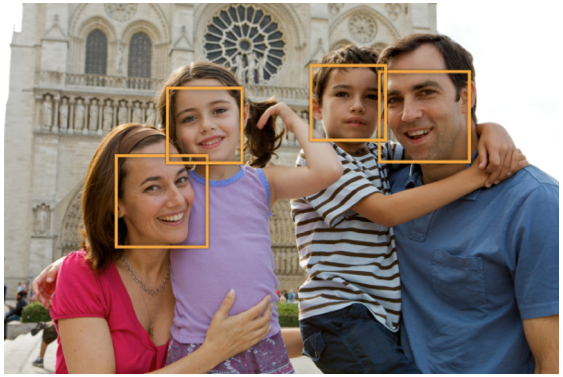


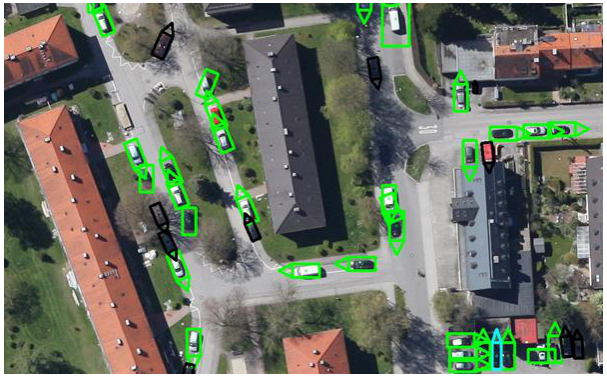




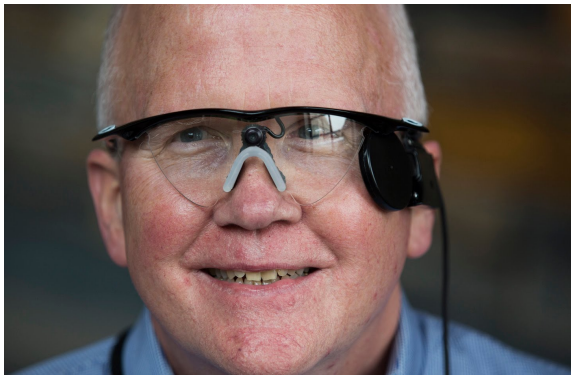
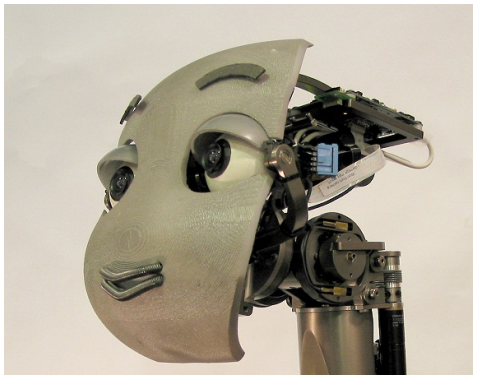


computer vision in images

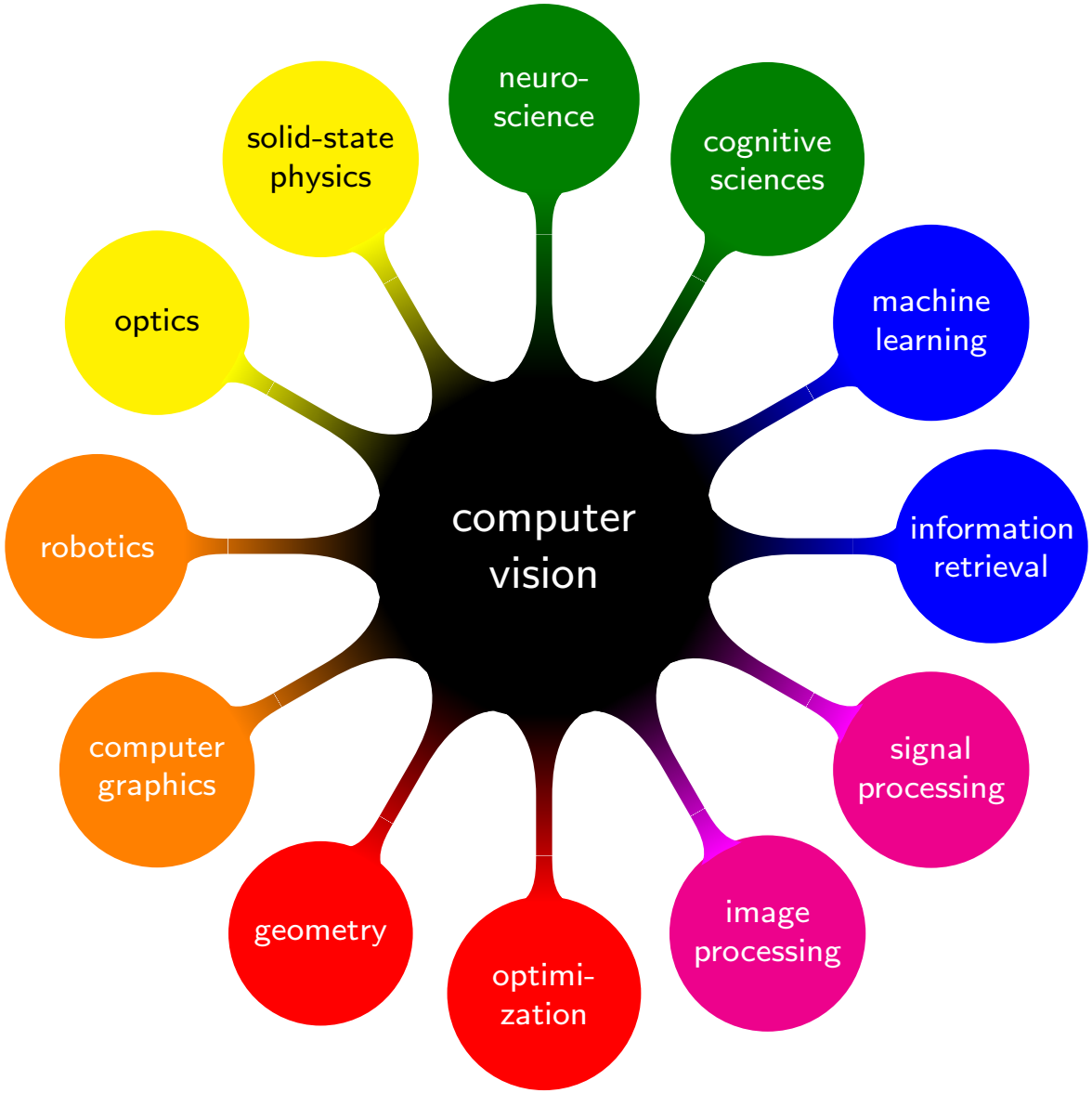




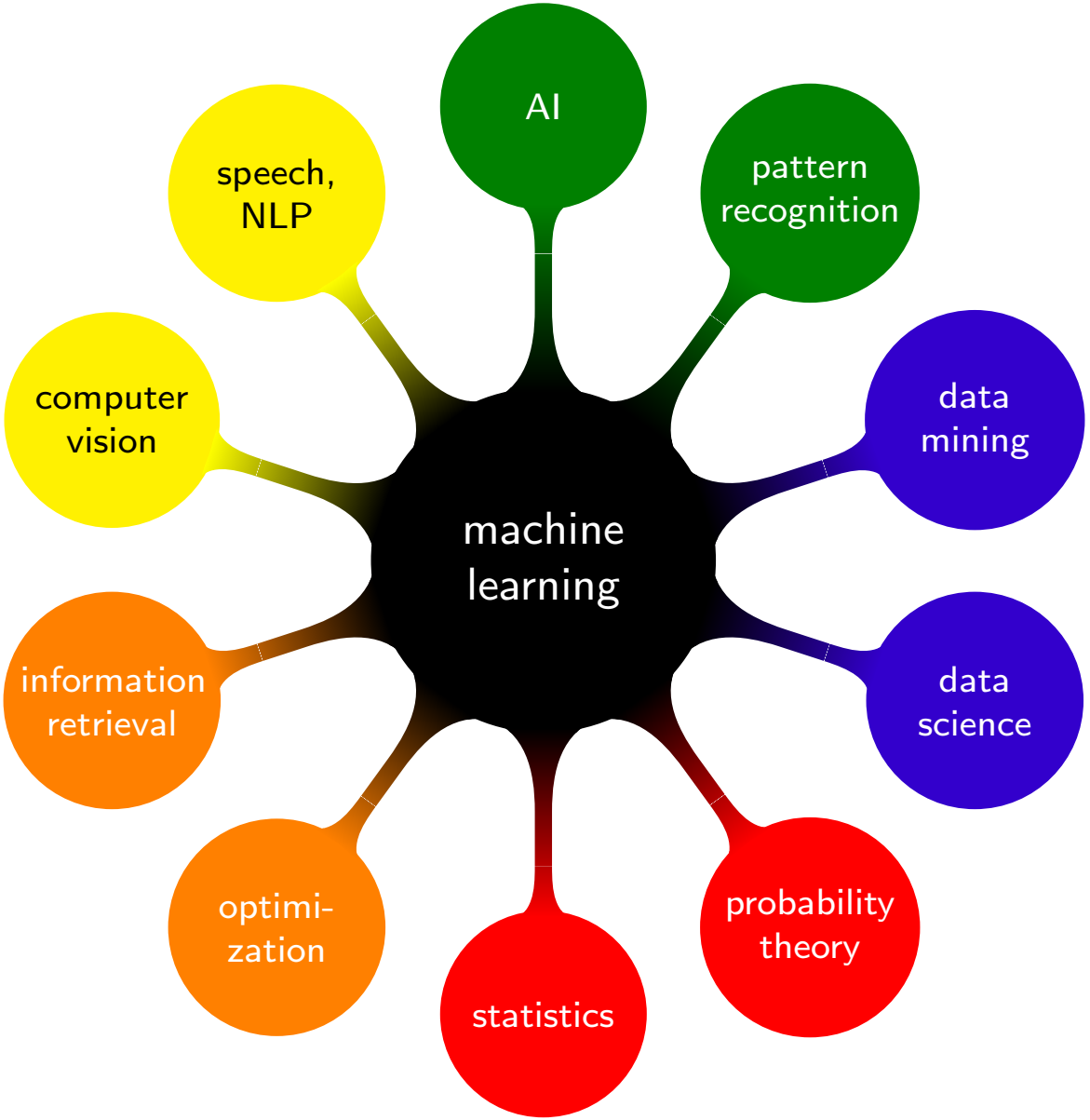


computer vision—related fields

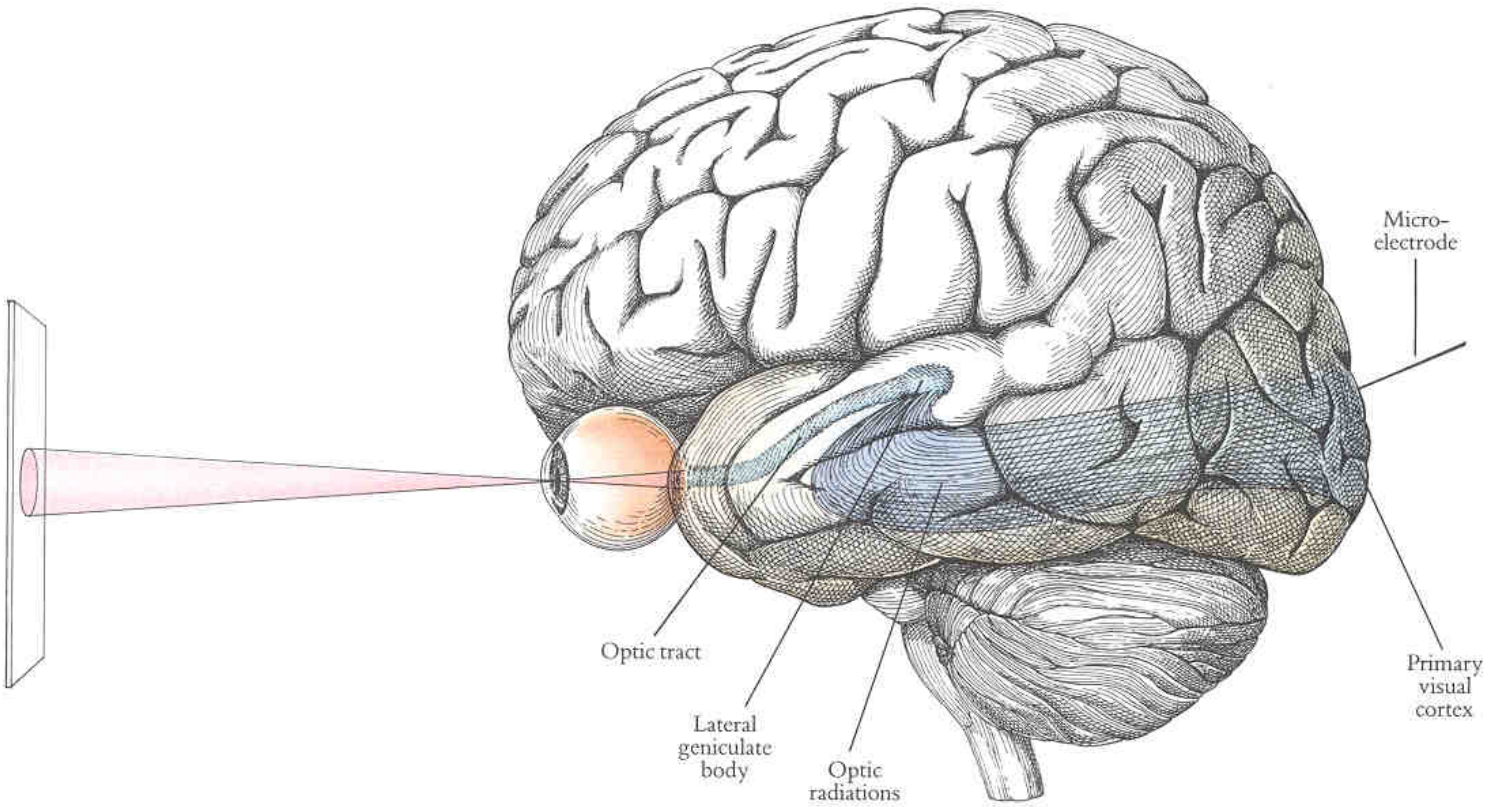


machine learning—related fields



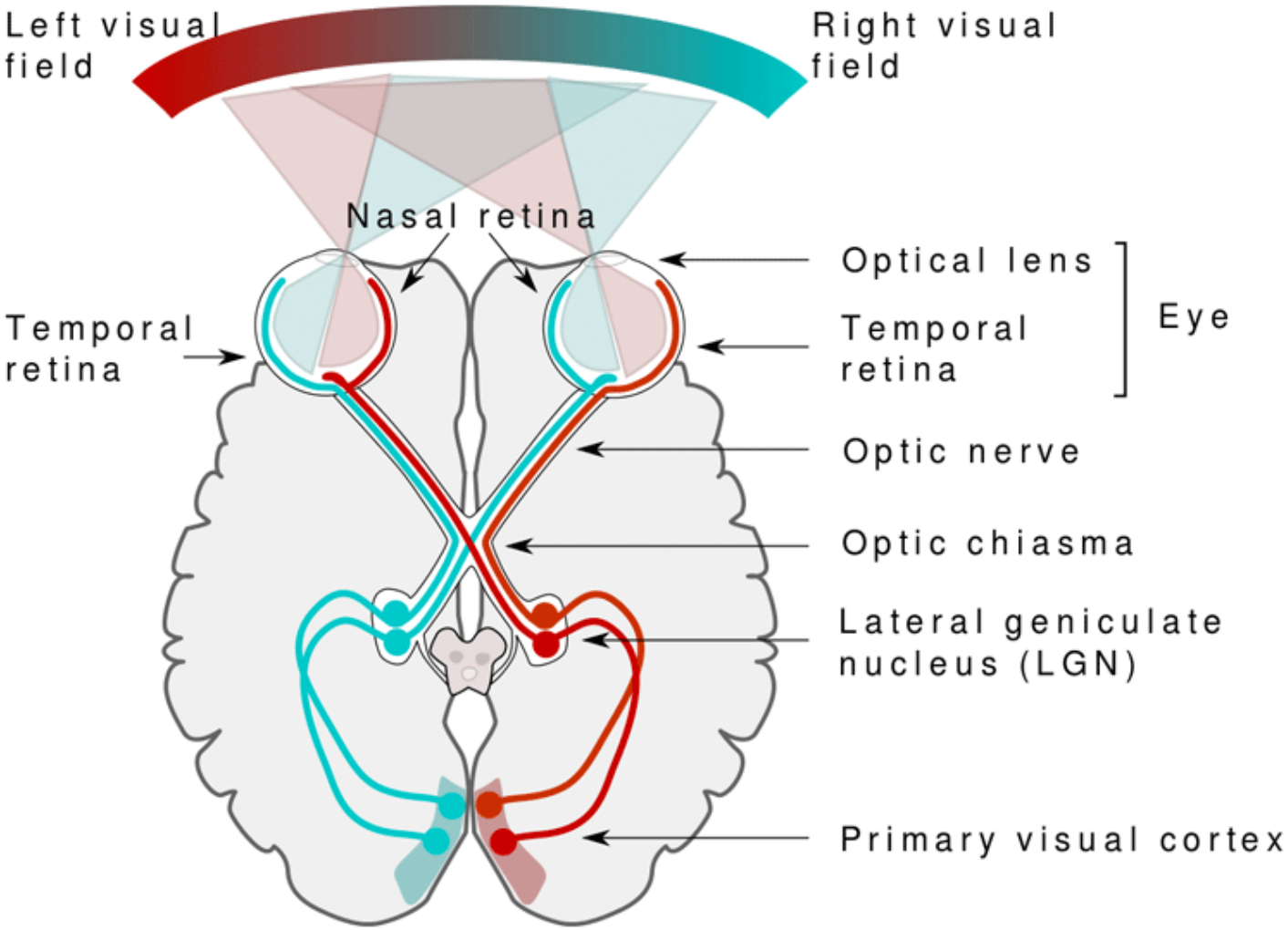
psychology and neuroscience background

visual system of mammals



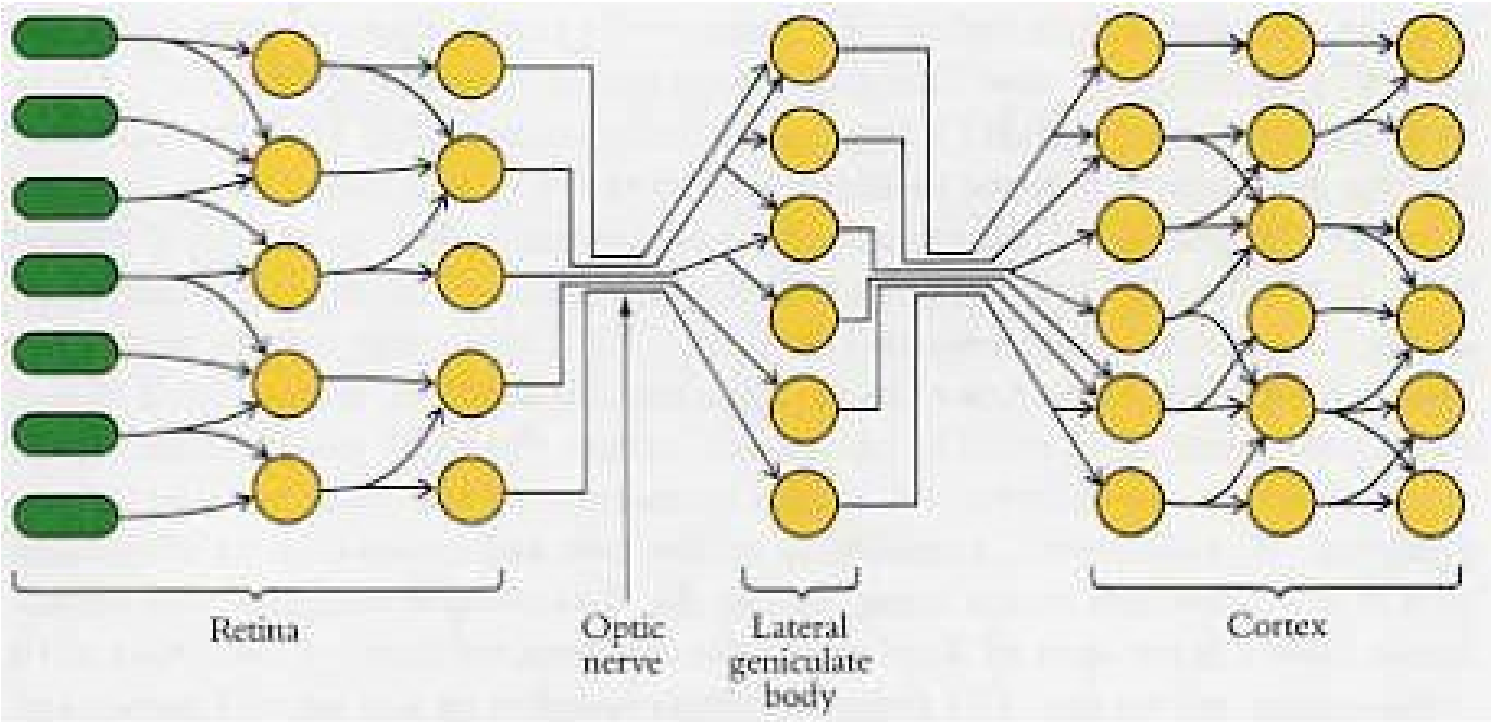
Hubel 1995. Eye, Brain, and Vision.

visual pathway



Hubel 1995. Eye, Brain, and Vision.

topographic representation



- as you move along the retina, the corresponding points in the cortex trace a continuous path
- each column represents a two-dimensional array of cells

Hubel 1995. Eye, Brain, and Vision.

computer vision background

the summer vision project

[Papert 1966]

"The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition"."

Papert 1966. The Summer Vision Project.

general goals

FIGURE-GROUND

> “divide a picture into regions such as likely objects, likely background areas and chaos”

REGION DESCRIPTION

> “analysis of shape and surface properties”

OBJECT IDENTIFICATION

> “name objects by matching them with a vocabulary of known objects”

Papert 1966. The Summer Vision Project.

edge detection

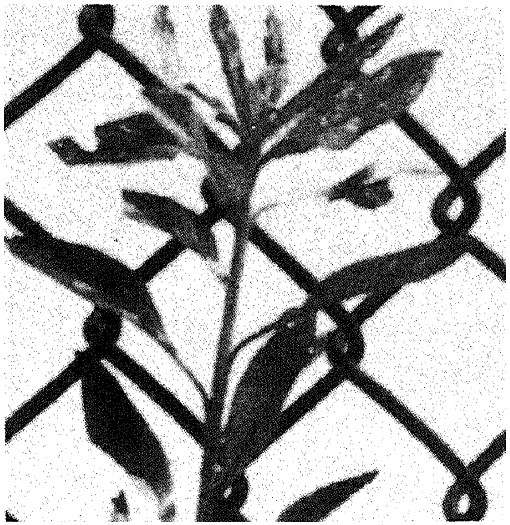
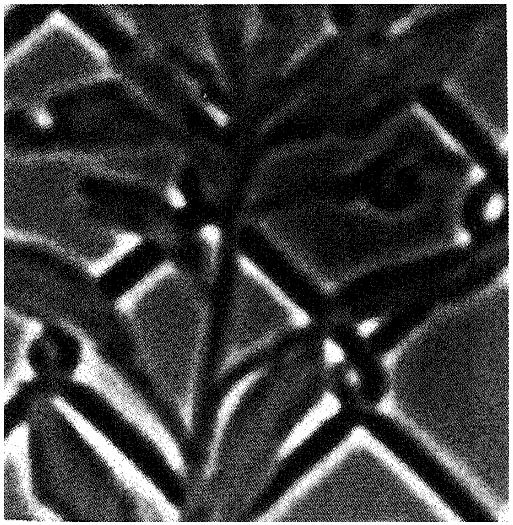
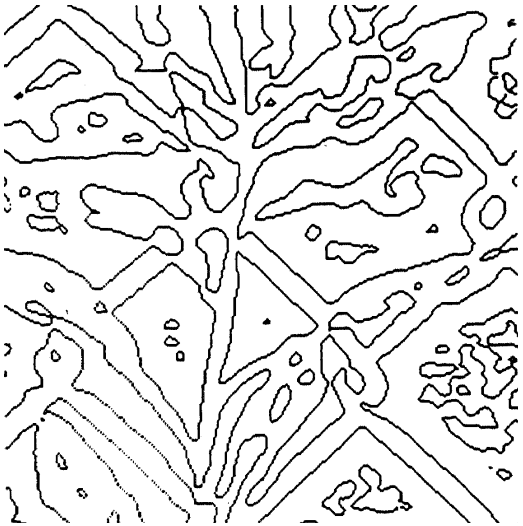


image I



$I * \nabla^2 G$

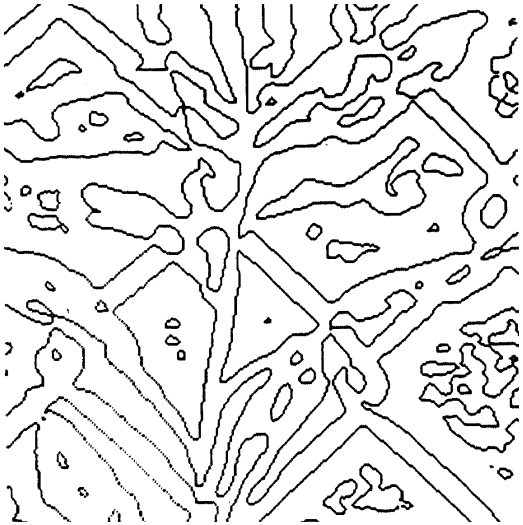
sign



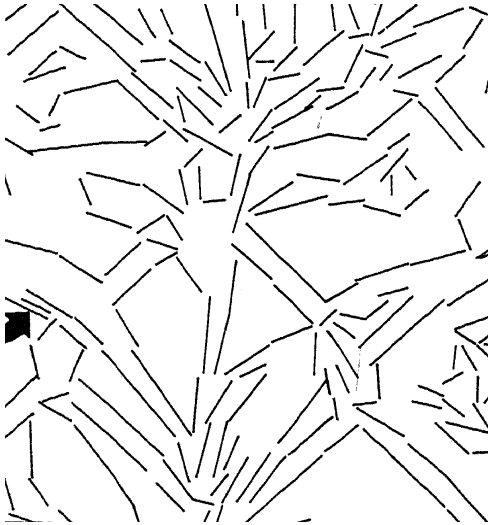
zero crossings

Marr 1982. Vision.

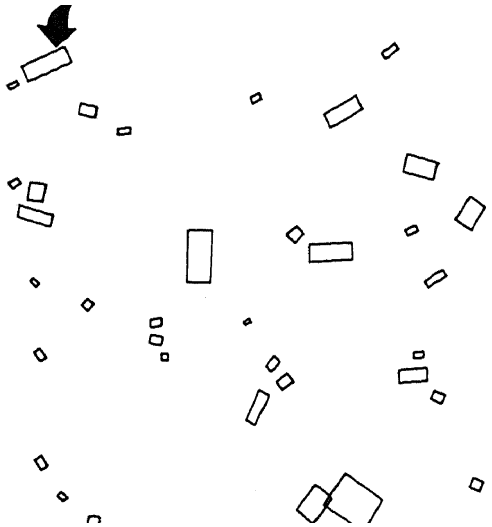
raw primal sketch



zero crossings



edge segments

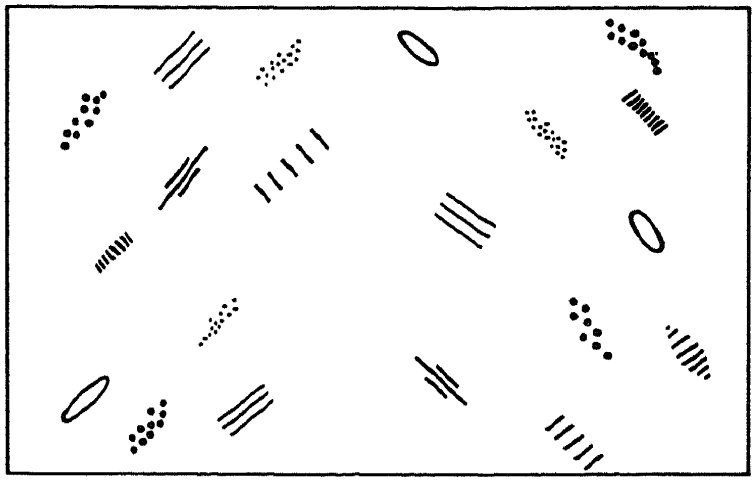


blobs

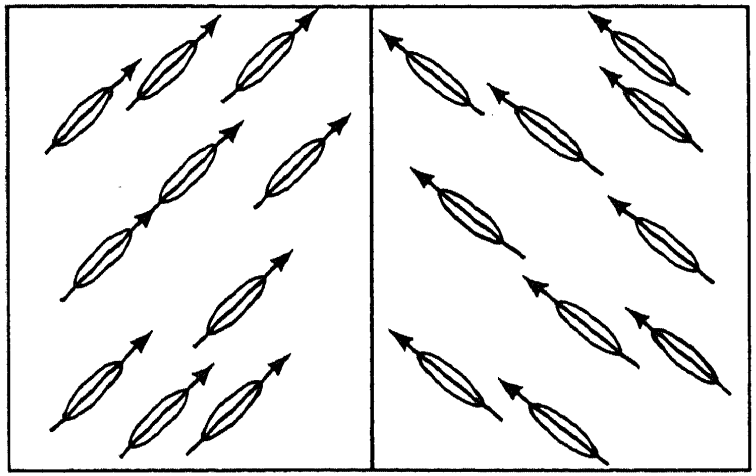
bars

Marr 1982. Vision.

full primal sketch



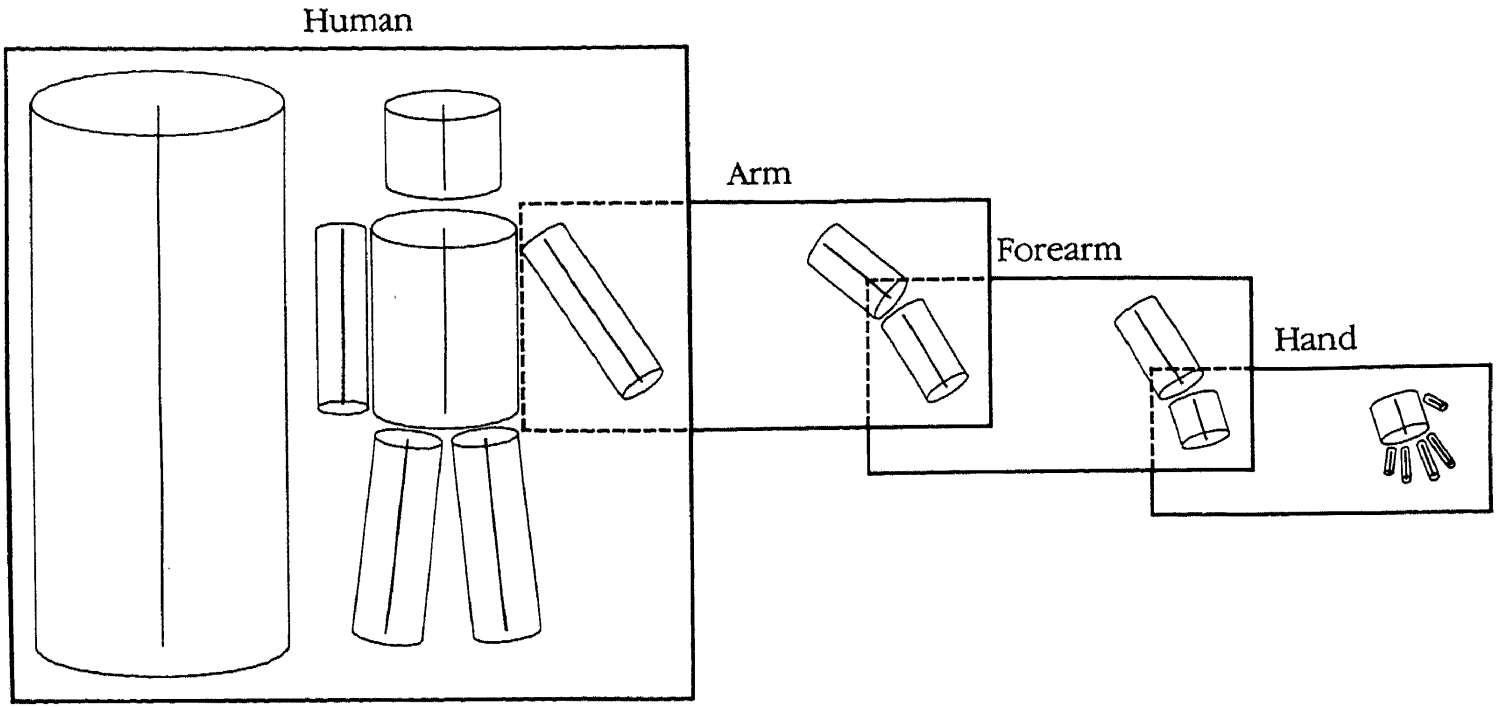
image



hierarchical grouping of tokens

Marr 1982. Vision.

3d model representation

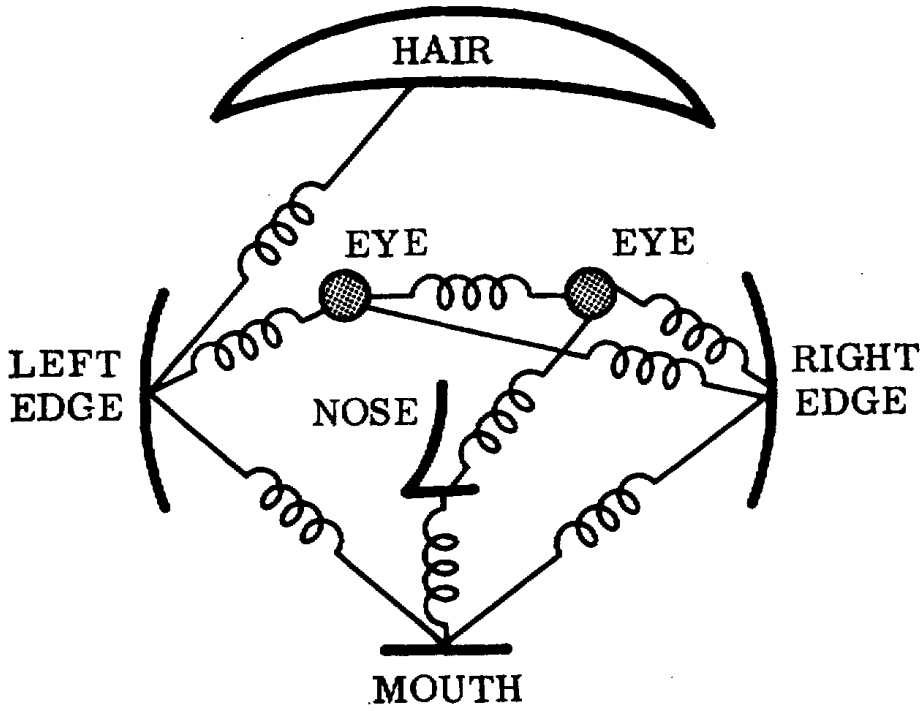


- hierarchical 3d model description
- parts of limited complexity, specified in local coordinate systems
- flexible, allowing for relative part transformation

Marr 1982. Vision.

pictorial structures

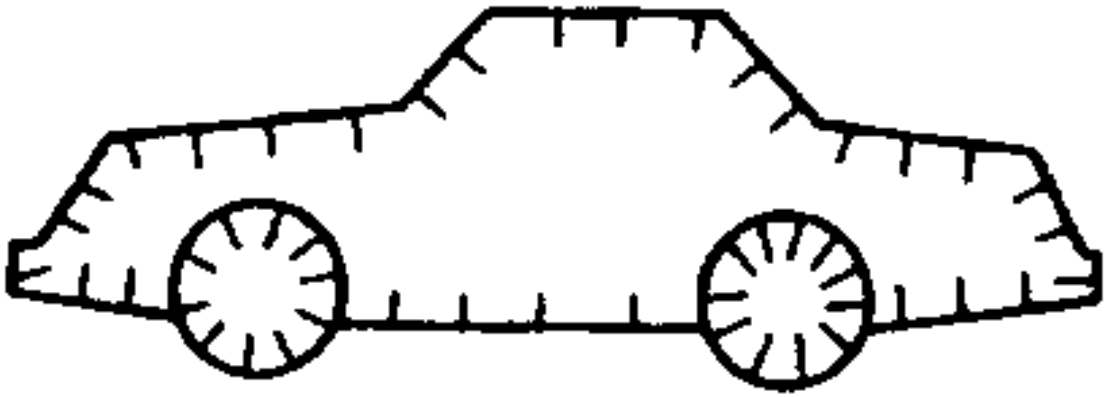
[Fischler and Elschlager 1973]



- manually specified object description
- parts-based model: part attributes and pairwise spatial relations
- efficient dynamic programming implementation

generalized Hough transform

[Ballard 1981]

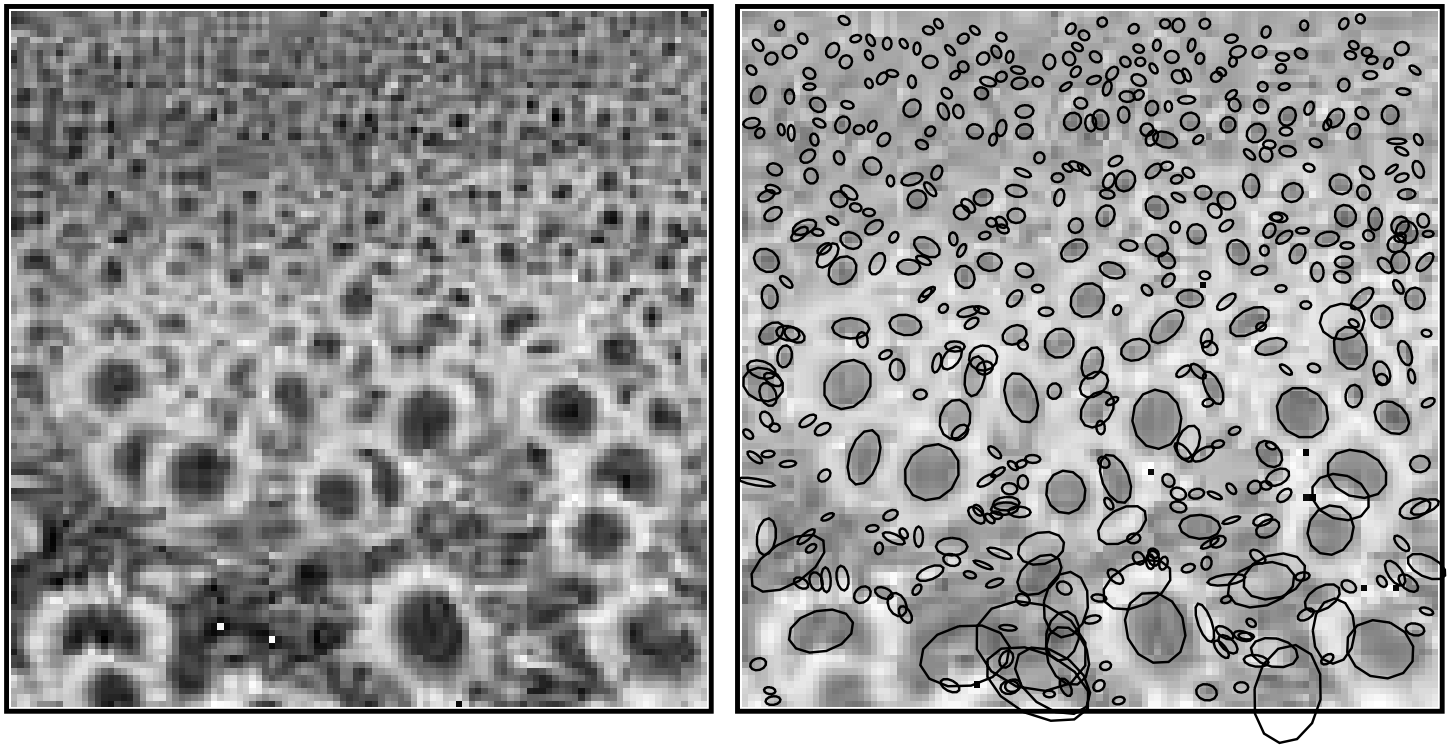


- Hough transform detects analytic curves in parameter space
- generalized version detects arbitrary non-analytic curves
- detection based on a voting process

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

scale selection

[Lindeberg 1993]



- scale-space and scale-normalized derivatives
- automatic scale selection at local maxima over scale
- applies to blobs, junctions, corners, edges or ridges

Lindeberg. SCIA 1993. On Scale Selection for Differential Operators.

scale-invariant feature transform (SIFT)

[Lowe 1999]

- scale selection by difference of Gaussians (DoG)
- orientation assignment, local descriptor
- Hough transform on affine space

Lowe. ICCV 1999. Object recognition from local scale-invariant features.

real-time face detection

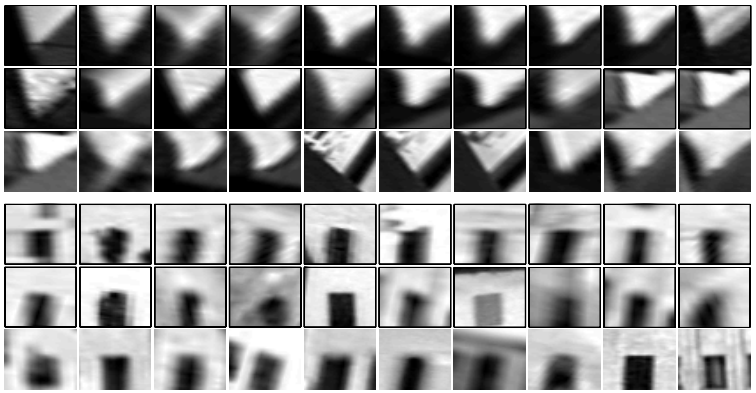
[Viola and Jones 2001]

- simple rectangle features in constant time on integral images
- learning weak classifiers by boosting
- classifier cascade provides a focus-of-attention mechanism

Viola and Jones. CVPR 2001. Rapid Object Detection Using a Boosted Cascade of Simple Features.

bag of words

[Sivic and Zisserman 2003]



visual vocabulary

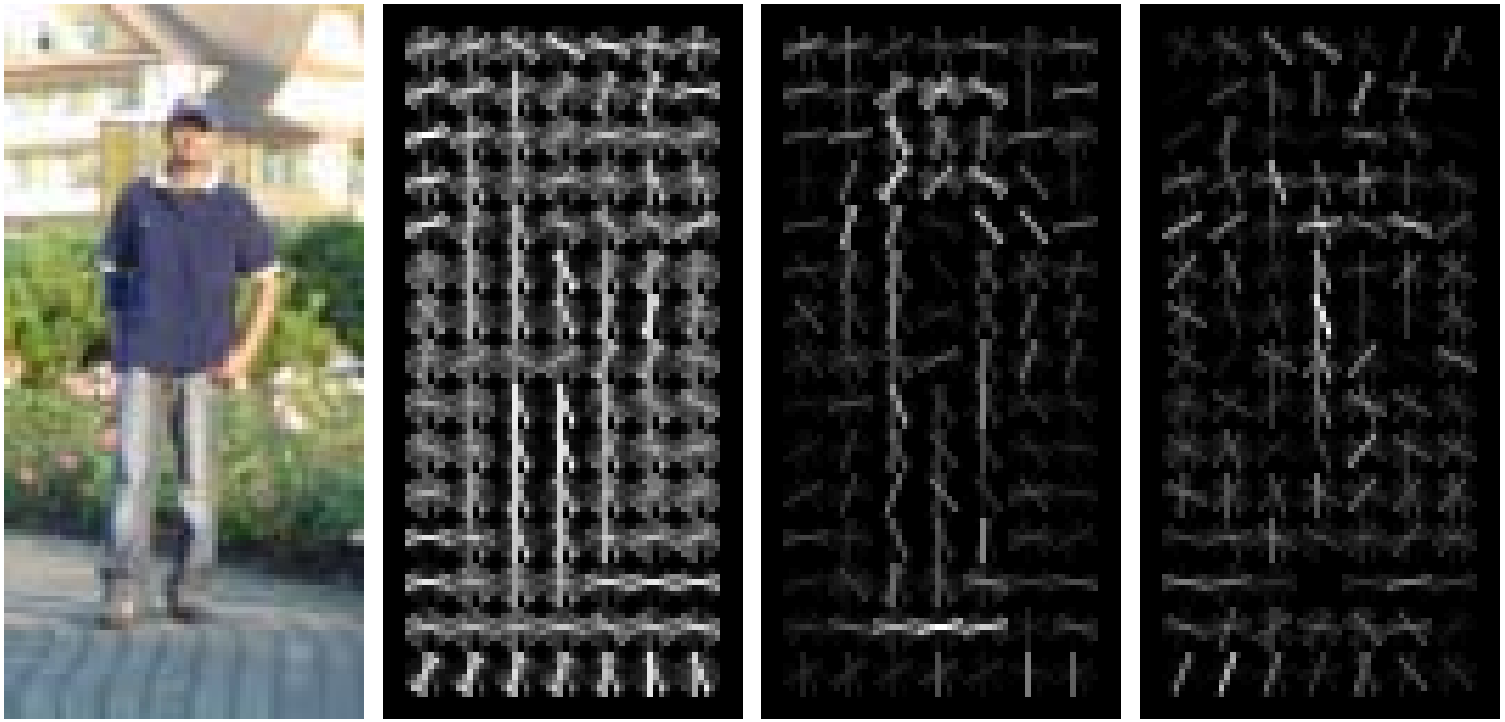
video retrieval

- "visual words" defined as clusters of SIFT descriptors
- images described by visual word histograms
- text retrieval methods applied to video retrieval

Sivic and Zisserman. ICCV 2003. Video Google: A Text Retrieval Approach to Object Matching in videos.

histogram of oriented gradients (HOG)

[Dalal and Triggs 2005]

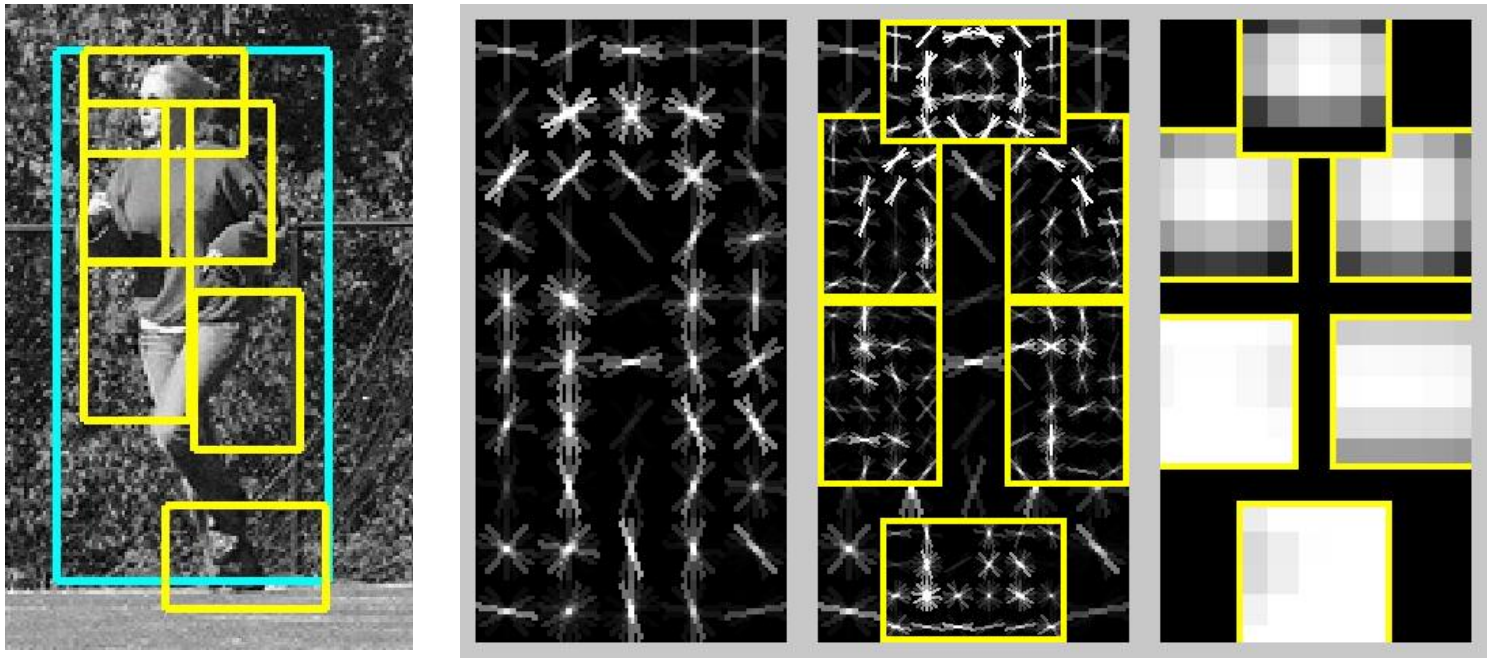


- dense, SIFT-like descriptors
- SVM classifier
- sliding window detection at all positions and scales

Dalal and Triggs. CVPR 2005. Histograms of Oriented Gradients for Human Detection.

deformable part model (DPM)

[Felzenszwalb et al. 2008]



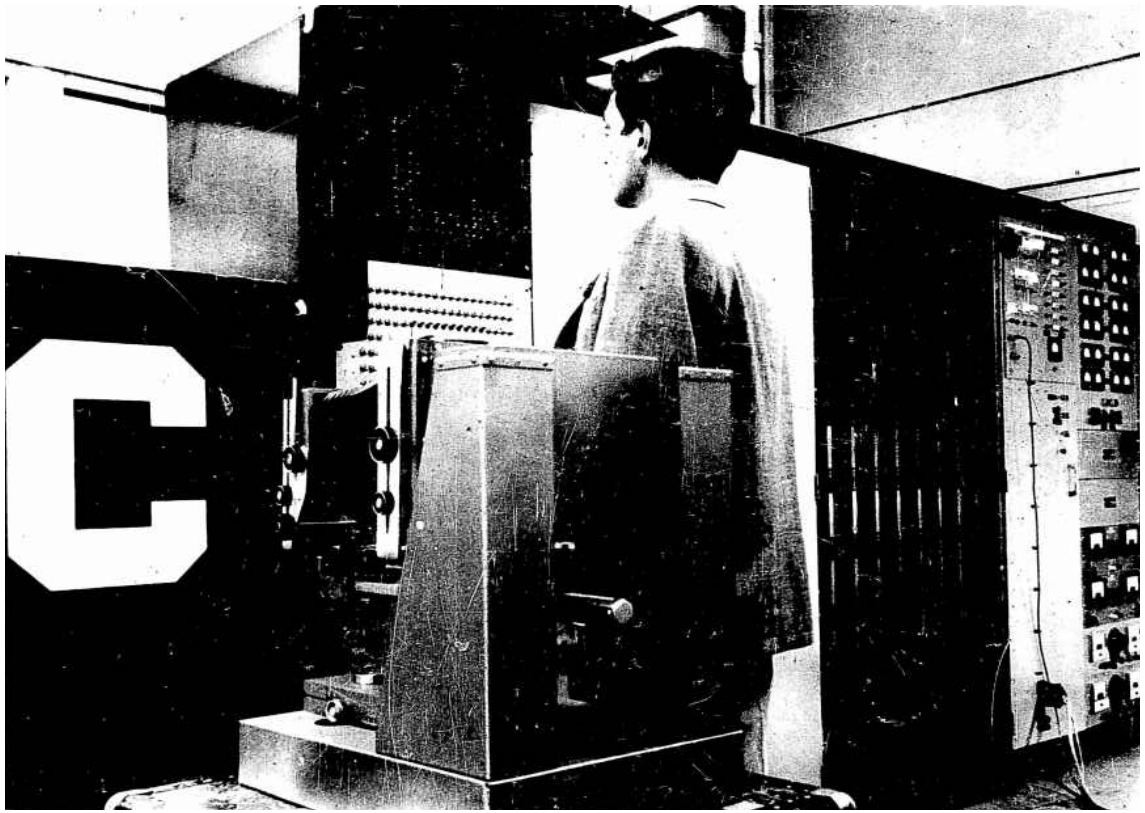
- appearance represented by HOG
- spatial configuration inspired by "pictorial structures"
- part locations treated as latent variables

Felzenszwalb, Mcallester and Ramanan. CVPR 2008. A Discriminatively Trained, Multiscale, Deformable Part Model.

machine learning background

perceptron

[Rosenblatt 1962]



- Mark-I perceptron
- analog circuit implementation; parameters as potentiometers

back-propagation

[Rumelhart et al. 1986]

The backward pass starts by computing $\partial E/\partial y$ for each of the output units. Differentiating equation (3) for a particular case, c, and suppressing the index c gives

$$\partial E/\partial y_j = y_j - d_j \tag{4}$$

We can then apply the chain rule to compute $\partial E/\partial x_j$

$$\partial E/\partial x_j = \partial E/\partial y_j \cdot \mathrm{d}y_j/\mathrm{d}x_j$$

Differentiating equation (2) to get the value of $\mathrm{d}y_j/\mathrm{d}x_j$ and substituting gives

$$\partial E/\partial x_j = \partial E/\partial y_j \cdot y_j(1-y_j) \tag{5}$$

This means that we know how a change in the total input x to an output unit will affect the error. But this total input is just a linear function of the states of the lower level units and it is also a linear function of the weights on the connections, so it is easy to compute how the error will be affected by changing these states and weights. For a weight w_{ji}, from i to j the derivative is

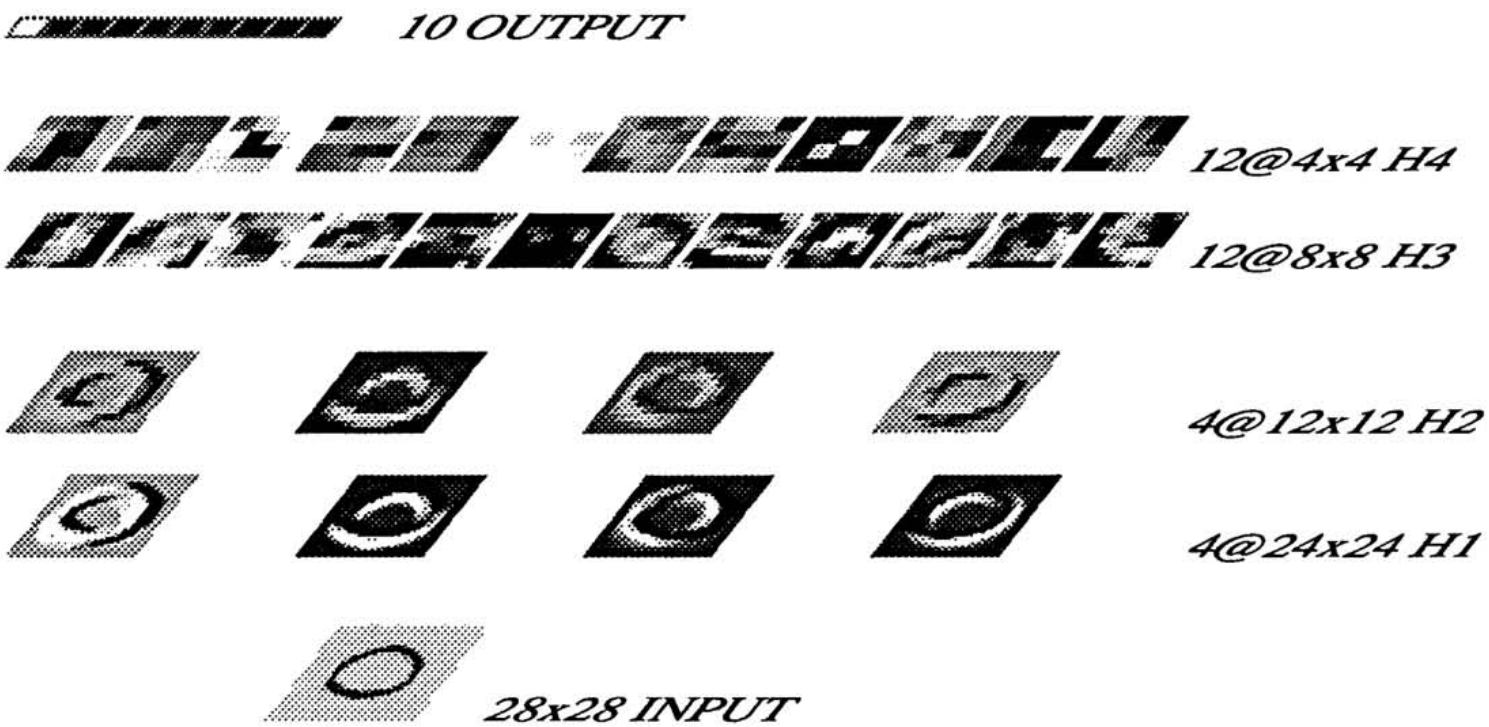
$$\begin{aligned}\partial E/\partial w_{ji} &= \partial E/\partial x_j \cdot \partial x_j/\partial w_{ji} \\ &= \partial E/\partial x_j \cdot y_i \end{aligned} \tag{6}$$

- introduce back-propagation in multi-layer networks with sigmoid nonlinearities and sum of squares loss function
- advocate batch gradient descent for supervised learning
- discuss online gradient descent, momentum and random initialization

Rumelhart, Hinton and Williams. N 1986. Learning Representations By Back-Propagating Errors.

convolutional networks

[LeCun et al. 1990]

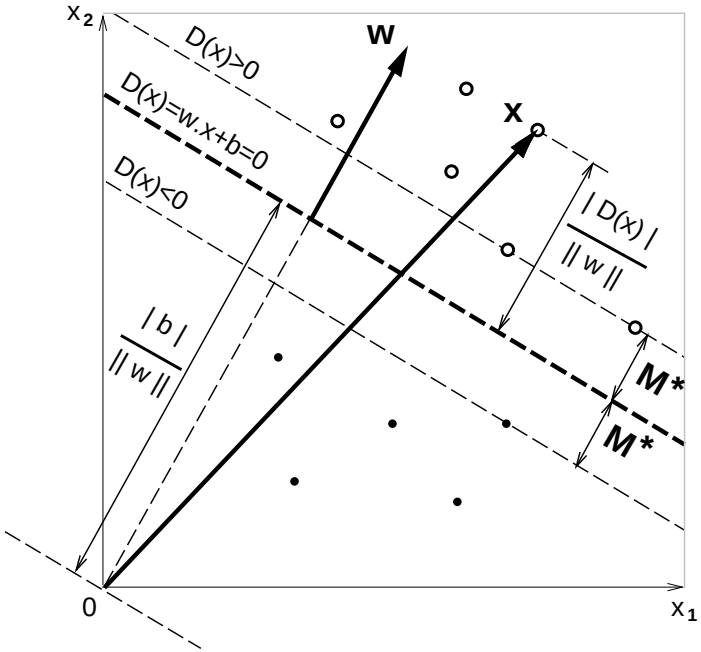


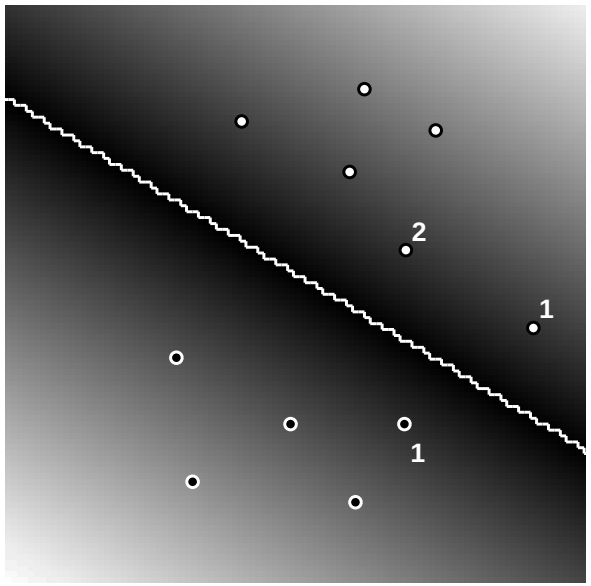
- train a convolutional network by back-propagation
- advocate end-to-end feature learning for image classification

LeCun, Boser, Denker *et al* . NIPS 1990. Handwritten Digit Recognition with a Back-Propagation Network.

support vector machines

[Boser et al. 1992]



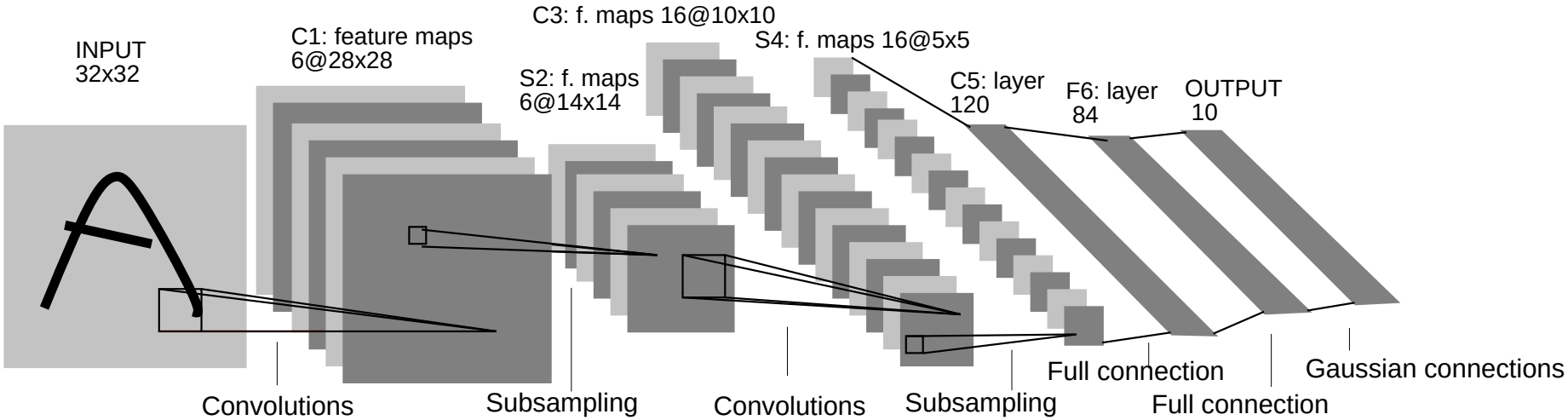


- linear classifier, made nonlinear via kernel trick
- convex optimization
- back to raw inputs; hand-crafted kernel functions
- shift focus from neural networks to kernel methods

Boser, Guyon and Vapnik. COLT 1992. A Training Algorithm for Optimal Margin Classifiers.

LeNet-5

[LeCun et al. 1998]



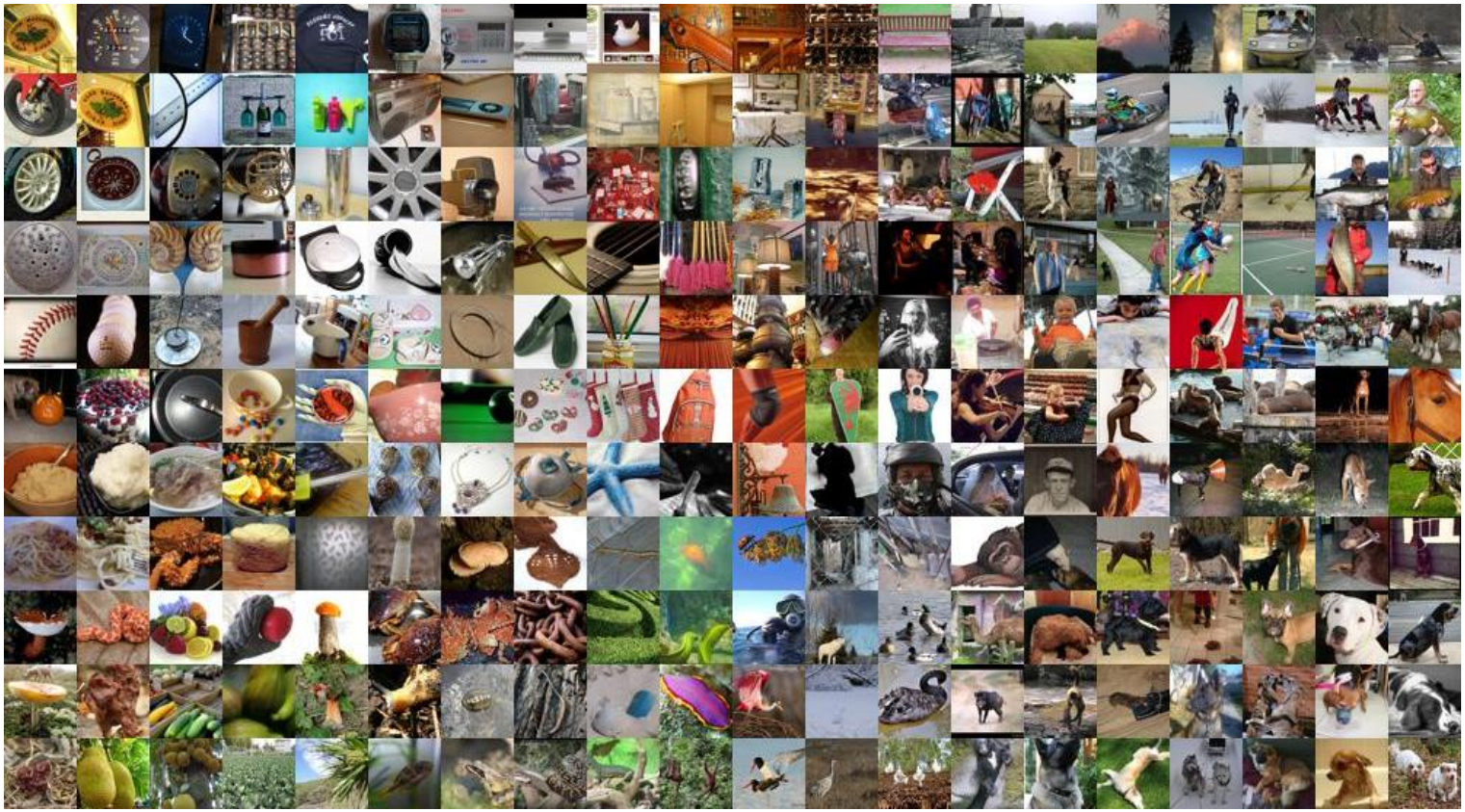
- sub-sampling gradually introduces translation, scale and distortion invariance
- non-linearity included in sub-sampling layers as feature maps are increasing in dimension

Lecun, Bottou, Bengio, Haffner. IEEE Proc. 1998. Gradient-Based Learning Applied to Document Recognition.

modern deep learning

ImageNet

[Russakovsky et al. 2014]

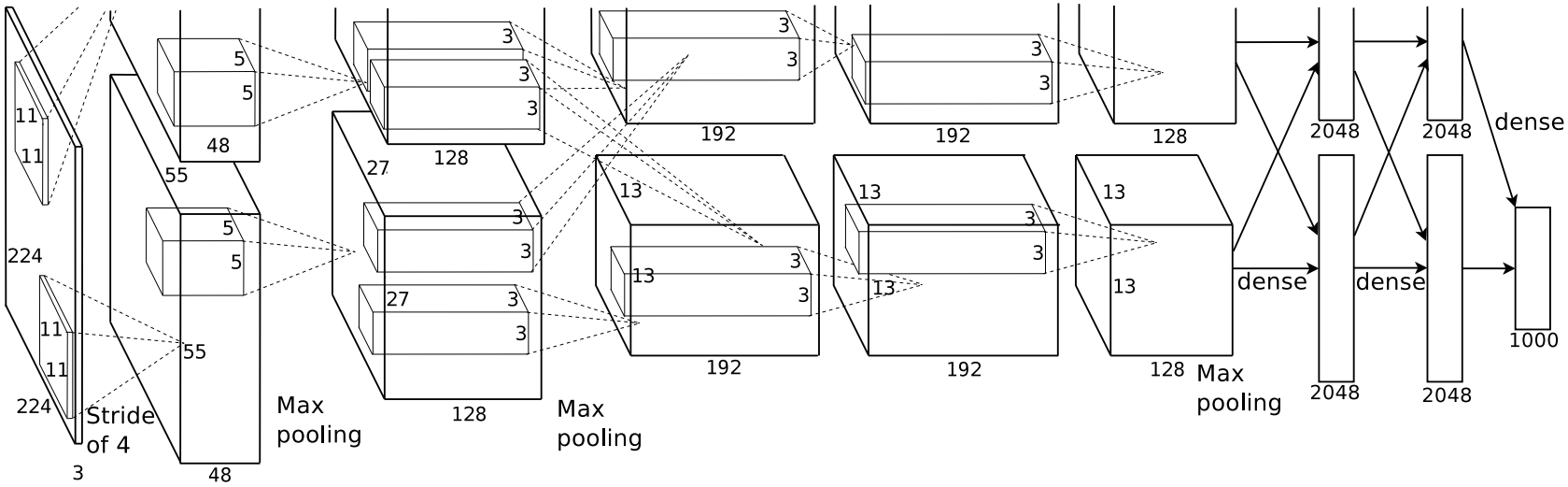


- 22k classes, 15M samples
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1000 classes, 1.2M training images, 50k validation images, 150k test images

Russakovsky, Deng, Su, Krause, *et al.* 2014. Imagenet Large Scale Visual Recognition Challenge.

AlexNet

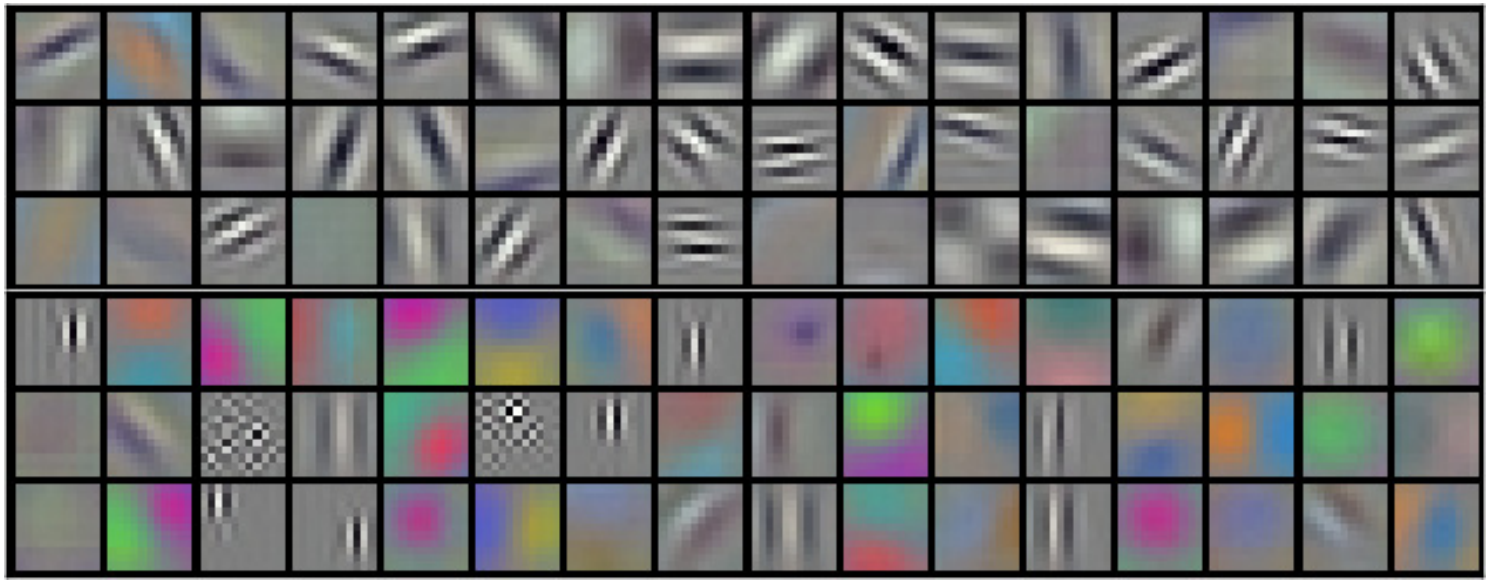
[Krizhevsky et al. 2012]



- implementation on two GPUs; connectivity between the two subnetworks is limited
- ReLU, data augmentation, local response normalization, dropout
- outperformed all previous models on ILSVRC by 10%

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

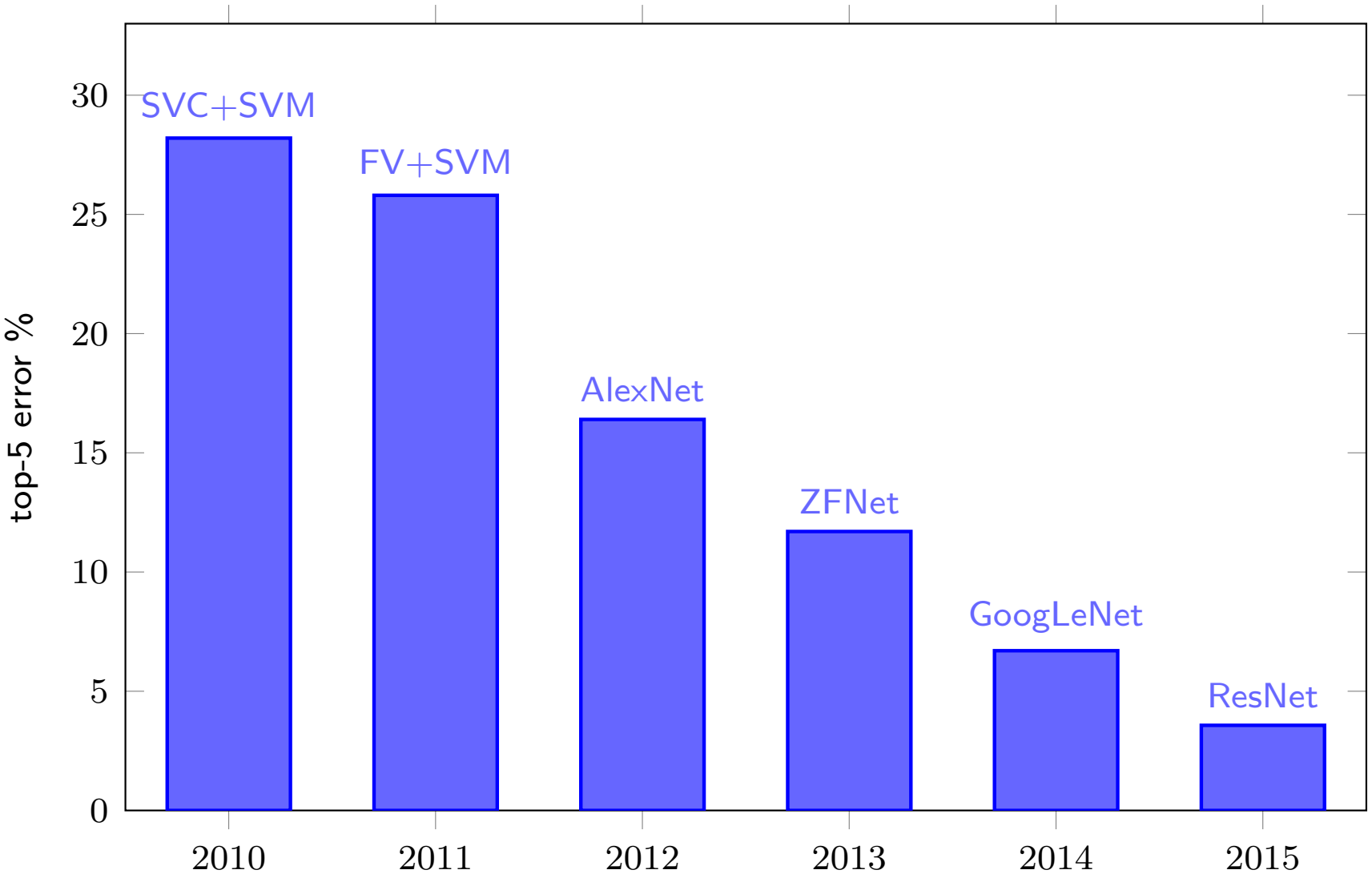
learned layer 1 kernels



- 96 kernels of size $11 \times 11 \times 3$
- top: 48 GPU 1 kernels; bottom: 48 GPU 2 kernels

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

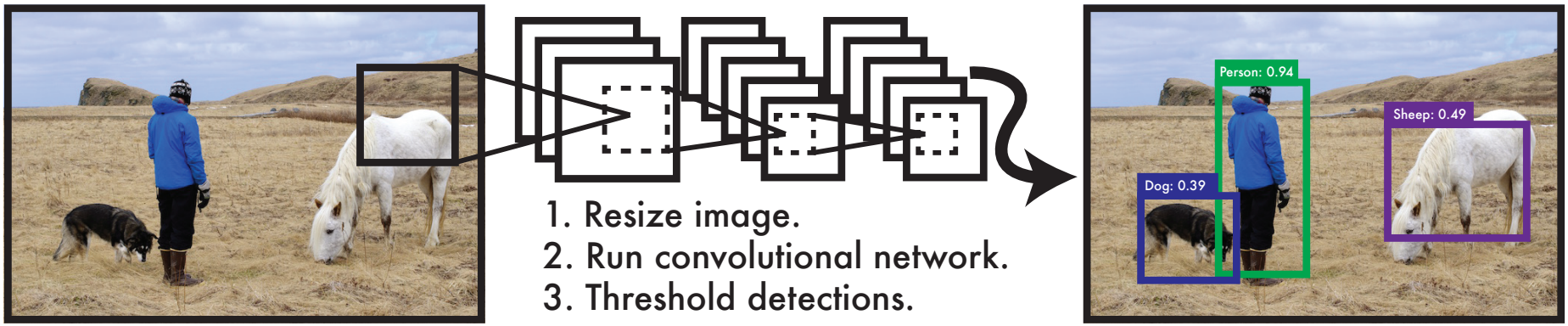
ImageNet classification performance



Russakovsky, Deng, Su, Krause, *et al*. 2014. Imagenet Large Scale Visual Recognition Challenge.

object detection

[Redmon et al. 2016]

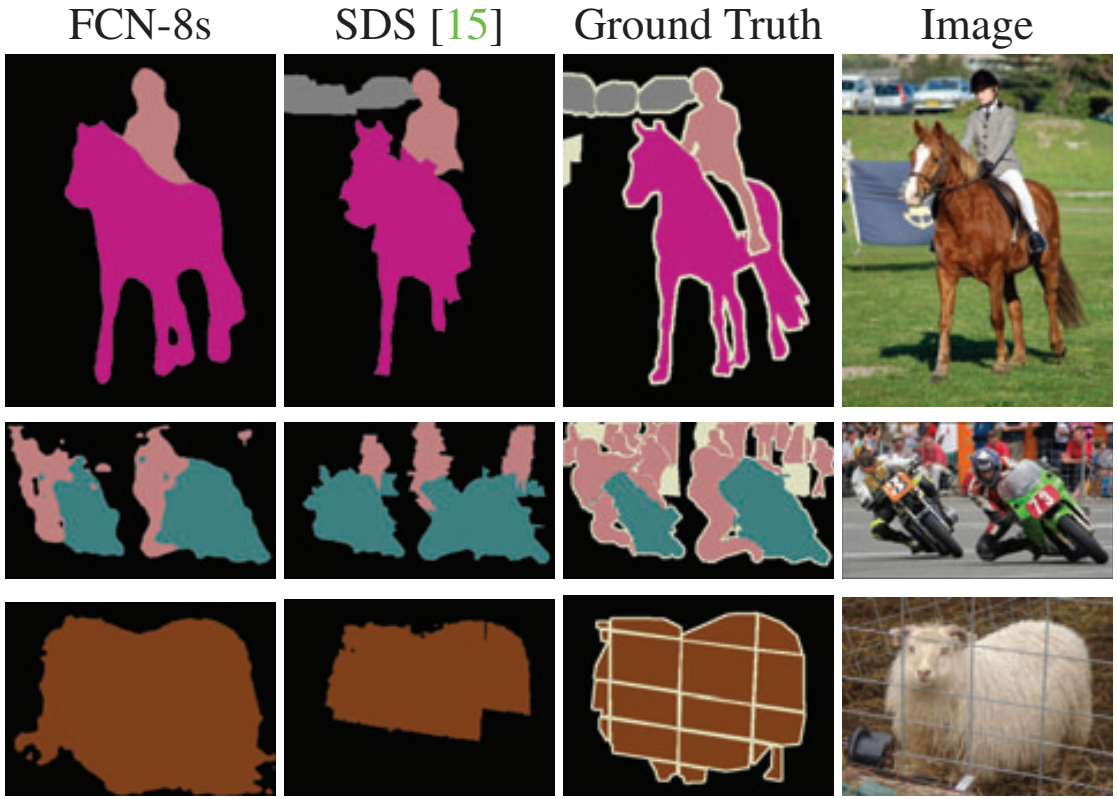


- learn to detect objects as a single classification and regression task, without scanning the image or detecting candidate regions
- first object detector to operate at 45fps

Redmon, Divvala, Girshick, Farhadi. CVPR 2016. You Only Look Once: Unified, Real-Time Object Detection.

semantic segmentation

[Long et al. 2015]



- learn to upsample
- apply to pixel-dense prediction tasks

instance segmentation and pose estimation

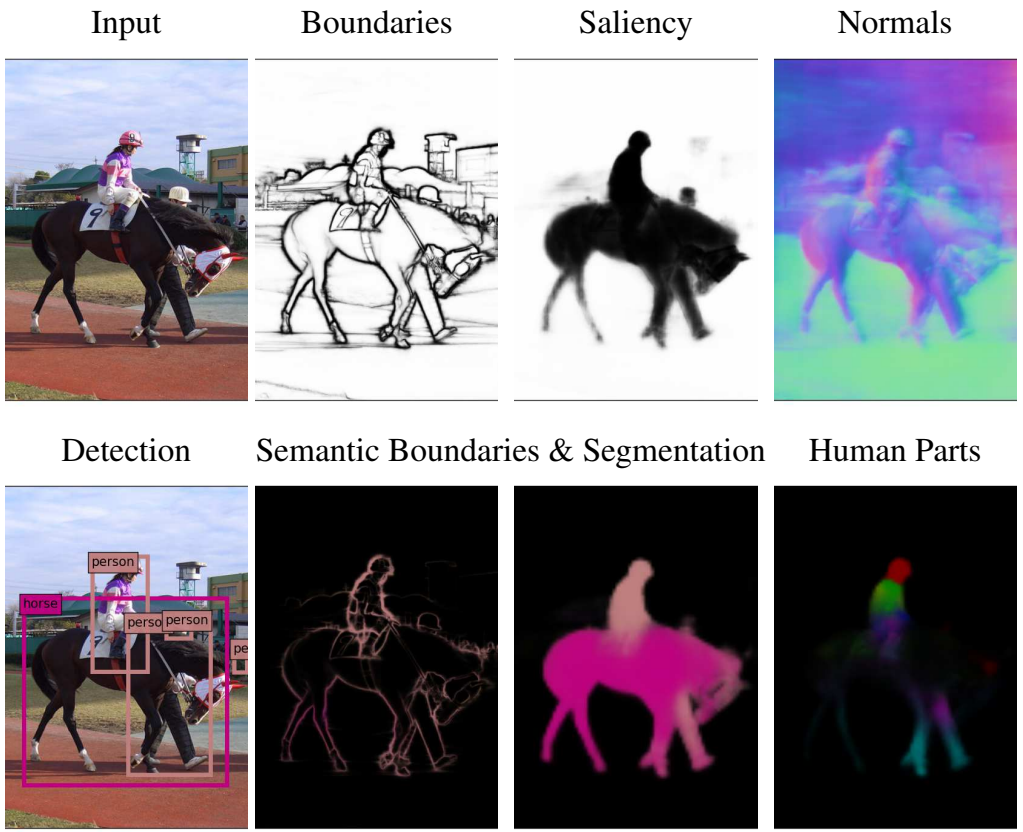
[He et al. 2017]

- semantic segmentation per detected region
- pose estimation as regression

He, Gkioxari, Dollar, Girshick. ICCV 2017. Mask R-CNN.

multi-task learning

[Kokkinos 2017]



- learn several vision tasks with a joint network architecture including task-specific skip layers

Kokkinos. CVPR 2017. Ubernet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory.

geometric matching

[Rocco et al. 2017]

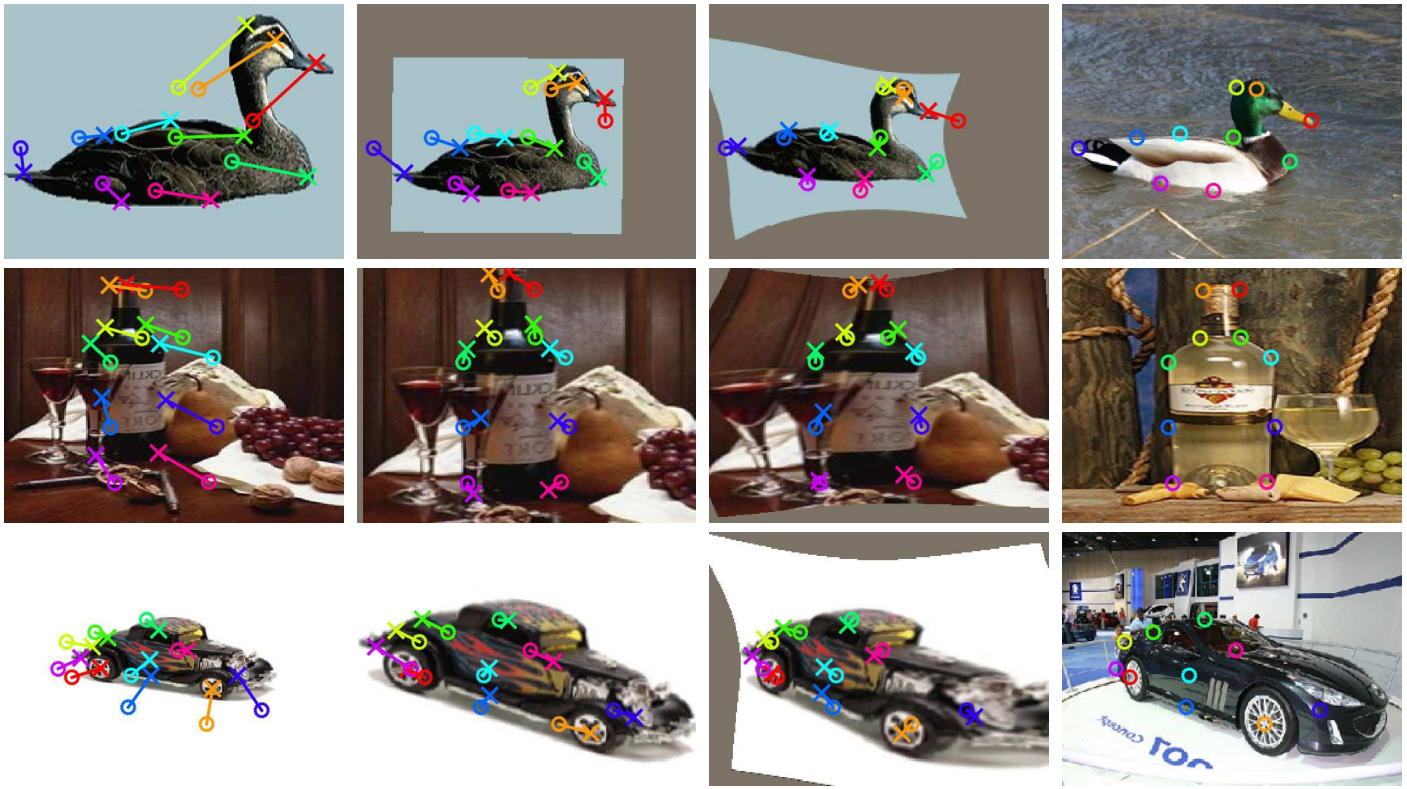


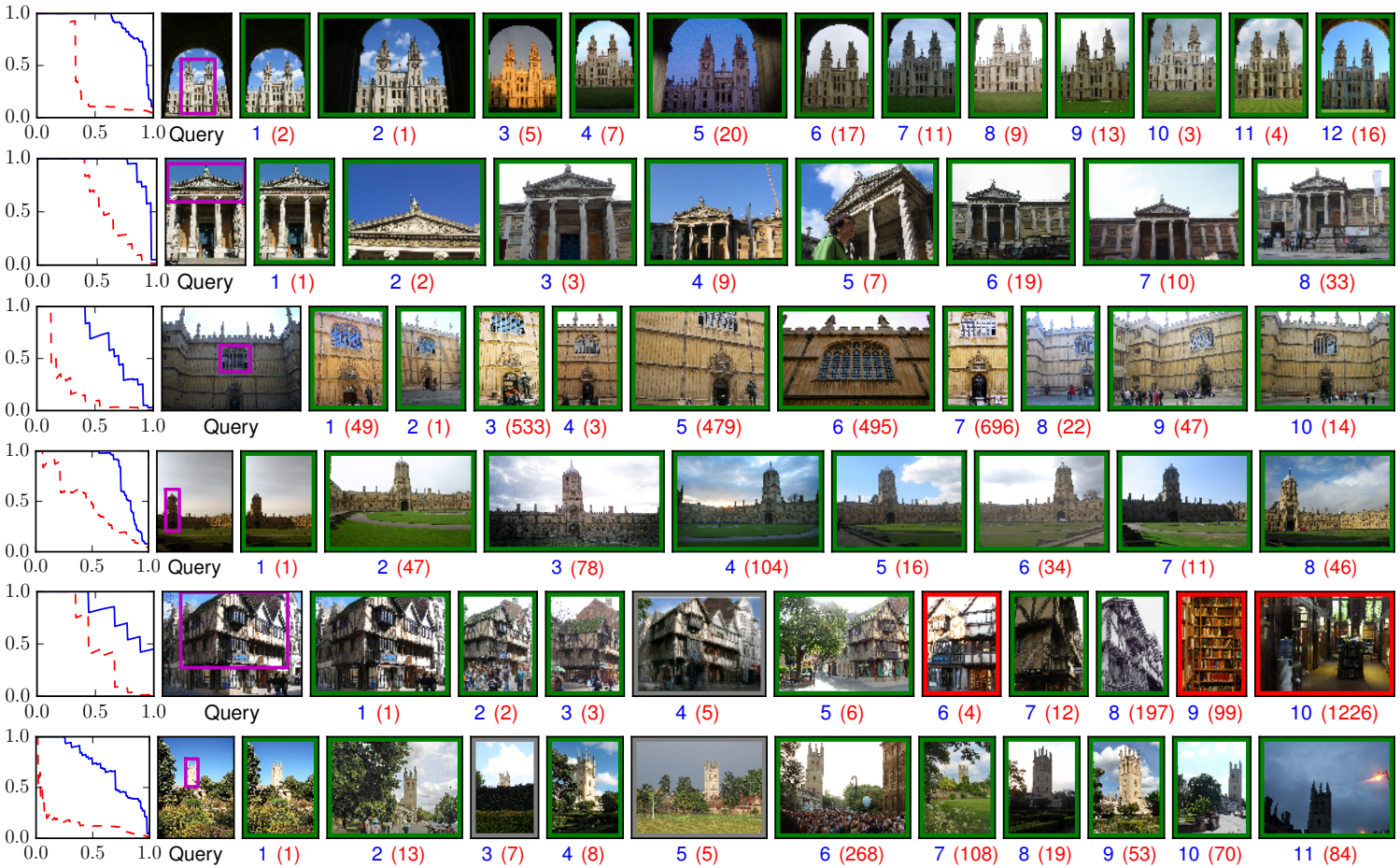
Image A | Aligned A (affine) | Aligned A (affine+TPS) | Image B

- mimic the standard steps of feature extraction, matching and simultaneous inlier detection and model parameter estimation
- still trainable end-to-end

Rocco, Arandjelovic, Sivic. CVPR 2017. Convolutional Neural Network Architecture for Geometric Matching.

image retrieval

[Gordo et al. 2016]

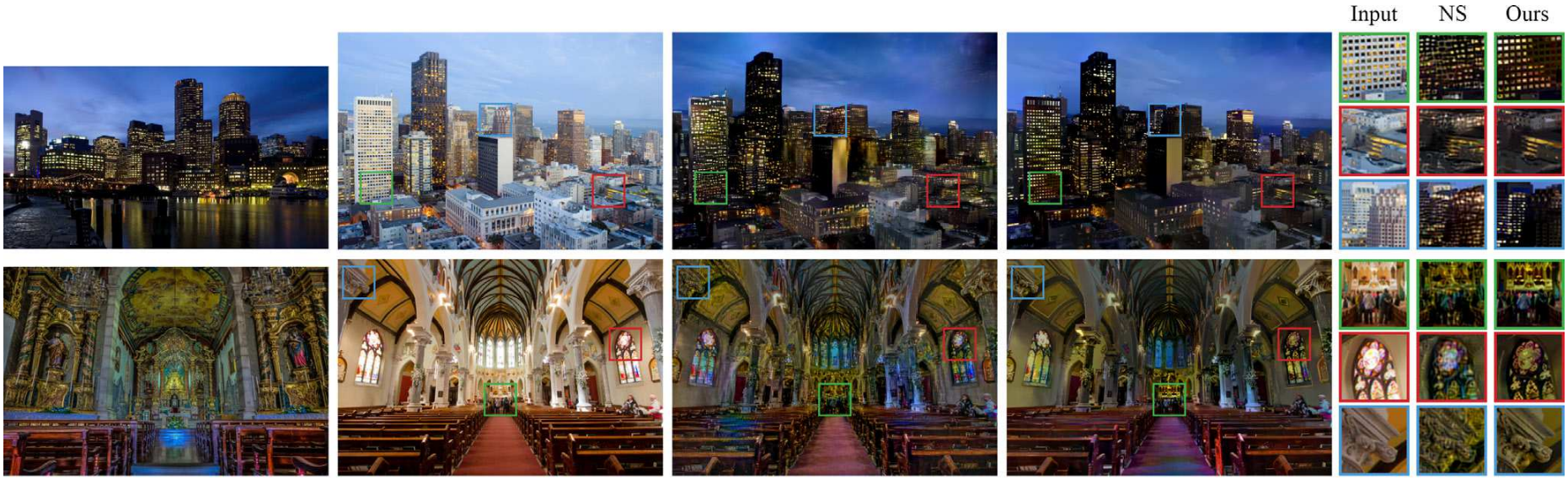


- learn to match
- apply as generic feature extractor

Gordo, Almazan, Revaud, Larlus. ECCV 2016. Deep Image Retrieval: Learning Global Representations for Image Search.

photorealistic style transfer

[Luan et al. 2017]



(a) Reference style image (b) Input image (c) Neural Style (distortions) (d) Our result (e) Insets

- generate same scene as input image
- transfer style from reference image
- photorealism regularization

Luan, Paris, Shechtman, Bala. CVPR 2017. Deep Photo Style Transfer.

image captioning

[Vinyals et al. 2017]

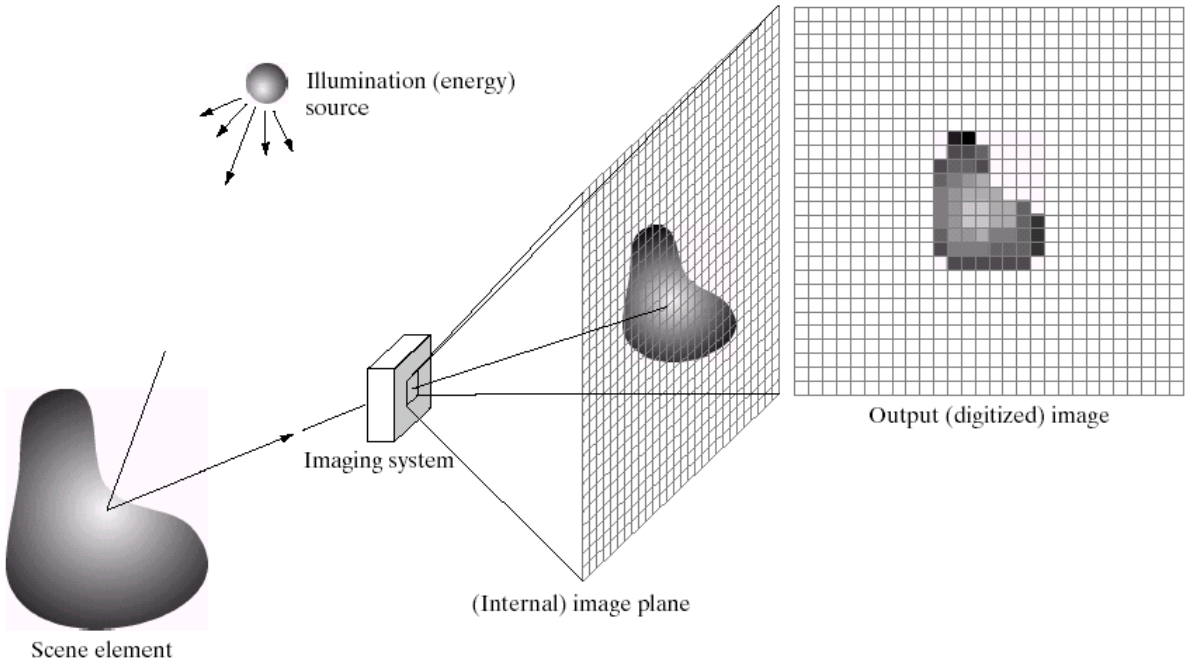
- image description by deep CNN
- language generation by RNN

Vinyals, Toshev, Bengio and Erhan. PAMI 2017. Show and Tell: Lessons Learned From the 2015 MSCOCO Image Captioning Challenge.

Digital Image Processing

Digital Image Processing

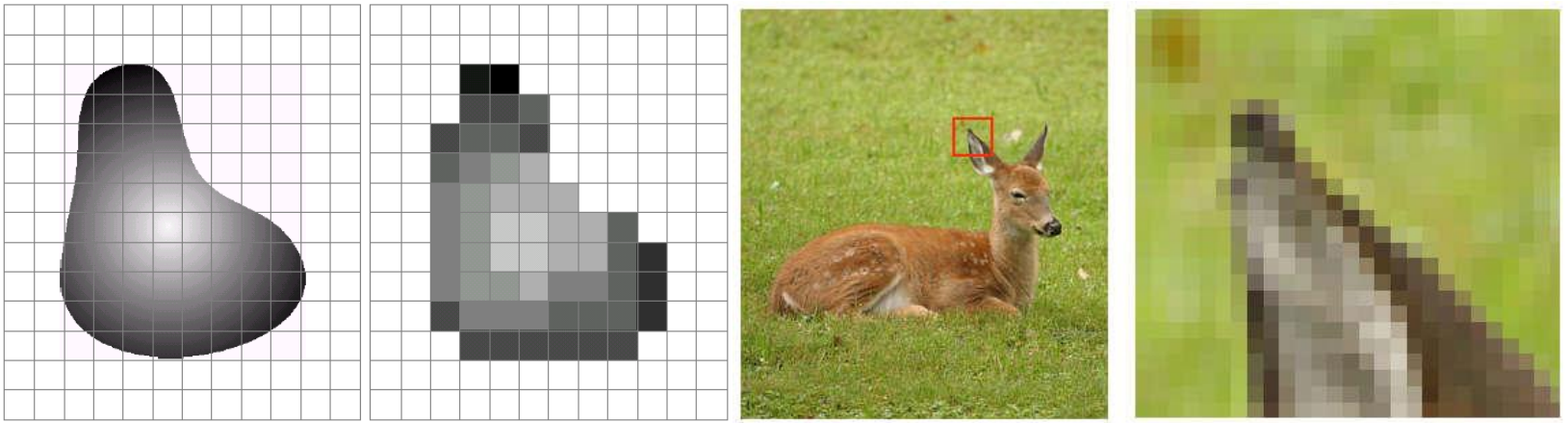
A digital image is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels



Gonzalez and Woods, Digital Image Processing, 2002.

Digital Image Processing

- Pixel values typically represent gray levels, colours, heights, opacities, etc.
- digitization implies that a digital image is an approximation of a real scene



Digital Image Processing

Digital image processing focuses on two major tasks

– Improvement of pictorial information for human interpretation
– Processing of image data for storage, transmission and representation for autonomous machine perception

Some argument about where image processing ends and fields such as image analysis and computer vision start

Digital Image Processing

The continuum from image processing to computer vision can be broken up into low-, mid- and high-level processes

- Low Level Process:
  Input: Image - Output: Image
  Examples: Noise removal, image sharpening
- Mid Level Process
  Input: Image - Output: Attributes
  Examples: Object recognition, segmentation
- High Level Process
  Input: Attributes - Output: Understanding
  Examples: Scene understanding, autonomous navigation

Digital Image Processing history

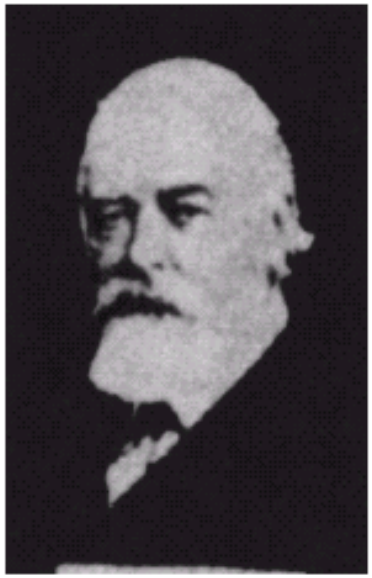
Early 1920s: One of the first applications of digital imaging was in the newspaper industry

– The Bartlane cable picture transmission service

– Images were transferred by submarine cable between London and New York

– Pictures were coded for cable transfer and reconstructed at the receiving end on a telegraph printer

Digital Image Processing history

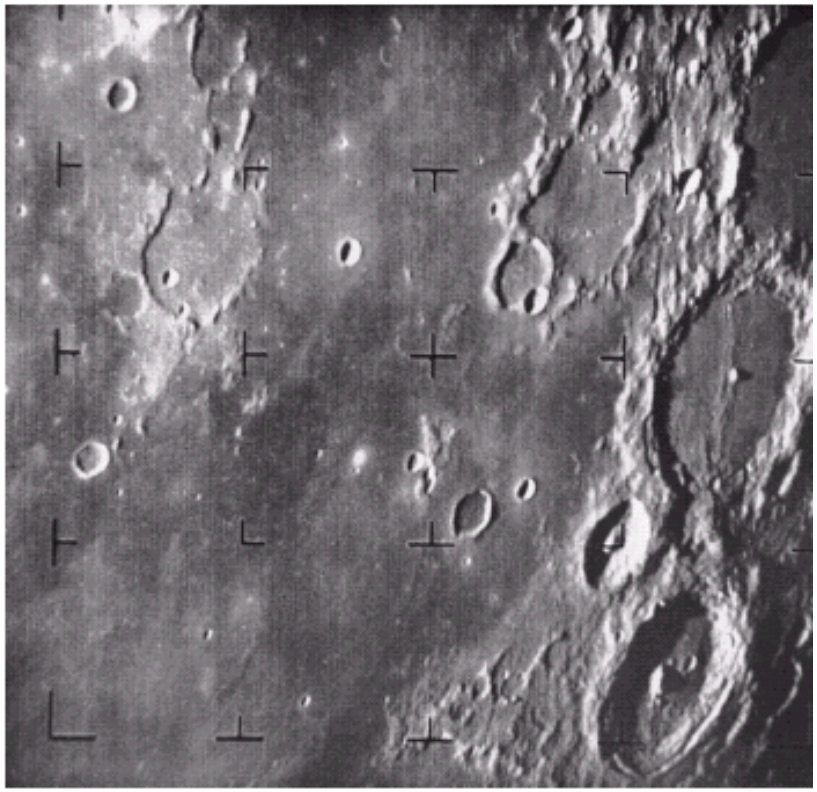


Mid to late 1920s: Improvements to the Bartlane system resulted in higher quality images

– New reproduction processes based on photographic techniques

– Increased number of tones in reproduced images

Digital Image Processing history



1960s: Improvements in computing technology and the onset of the space race led to a surge of work in digital image processing

– 1964: Computers used to improve the quality of images of the moon taken by the Ranger 7 probe

– Such techniques were used in other space missions including the Apollo landings

Digital Image Processing history

1970s: Digital image processing begins to be used in medical applications – 1979: Sir Godfrey N. Hounsfield Prof. Allan M. Cormack share the Nobel Prize in medicine for the invention of tomography, the technology behind Computerised Axial Tomography (CAT) scans

Digital Image Processing history

1980s - Today: The use of digital image processing techniques has exploded and they are now used for all kinds of tasks in all kinds of areas

- Image enhancement/restoration
- Artistic effects
- Medical visualisation
- Industrial inspection
- Law enforcement
- Human computer interfaces

Fourier

Fourier transform

- time (or space) $\rightarrow$ frequency

$$X(f) = \int x(t) e^{-j2\pi ft} \mathrm{d}t$$

- frequency $\rightarrow$ time (or space)

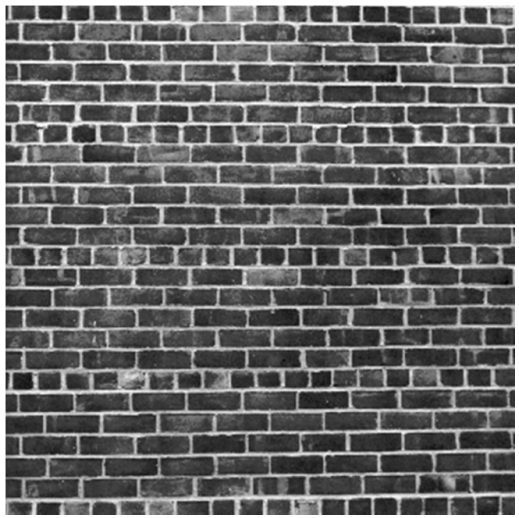
$$x(t) = \int X(f) e^{-j2\pi ft} \mathrm{d}f$$

- measurements

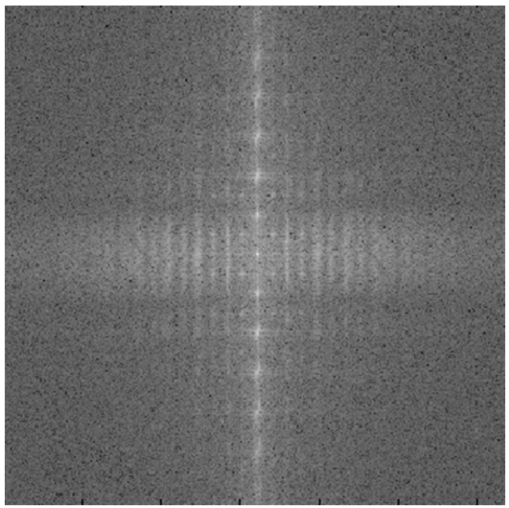
bar (+) bar (−) grating

Example 2D Fourier transform

Image with periodic structure



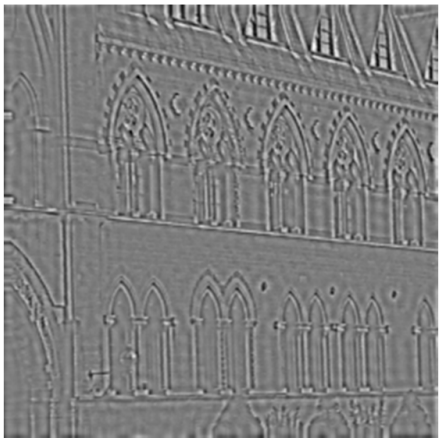
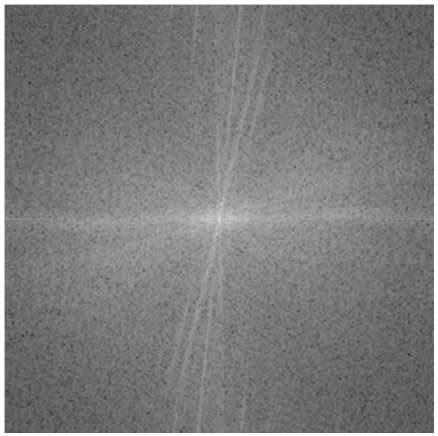
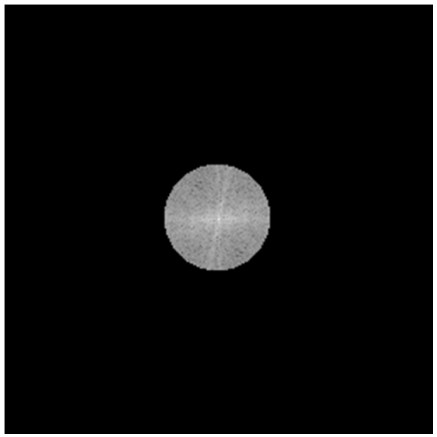
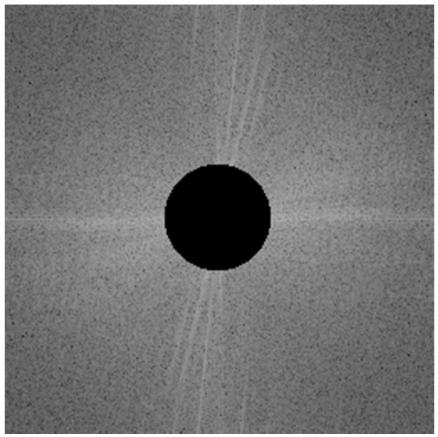
$f(x,y)$



$|F(u,v)|$

FT has peaks at spatial frequencies of repeated texture

Example: action of filters on a real image

	original	low pass	high pass
$f(x,y)$			
$\lvert F(u,v)\rvert$			

Convolution

Filtering vs convolution in 2D

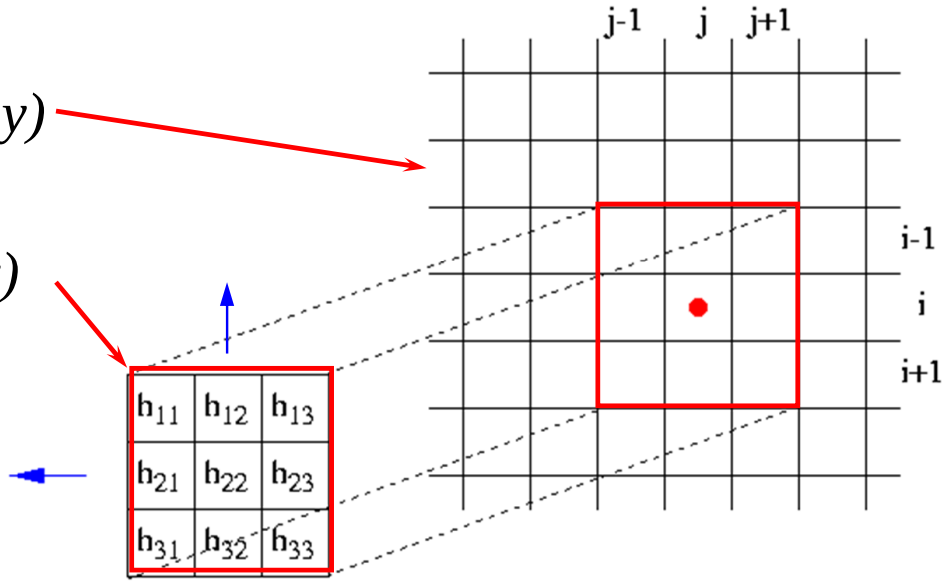
convolution

$$g(x,y) = h(x,y) * f(x,y) = f(x,y) * h(x,y)$$
$$= \int\int f(u,v)h(x-u,y-v)\,du\,dv$$

filtering

image *f(x,y)*

filter / kernel *h(x,y)*



$$g(x,y) = h_{11}f(i-1,j-1) + h_{12}f(i-1,j) + h_{13}f(i-1,j+1) +$$
$$h_{21}f(i,j-1) + h_{22}f(i,j) + h_{23}f(i,j+1) +$$
$$h_{31}f(i+1,j-1) + h_{32}f(i+1,j) + h_{33}f(i+1,j+1)$$

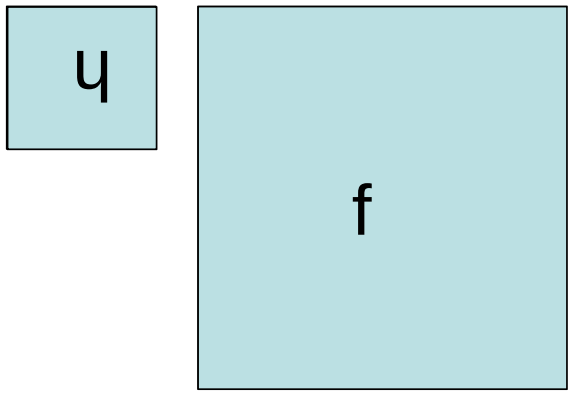
for convolution, reflect filter in x and y axes

Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

convolution with h



slide: K. Grauman

Filtering

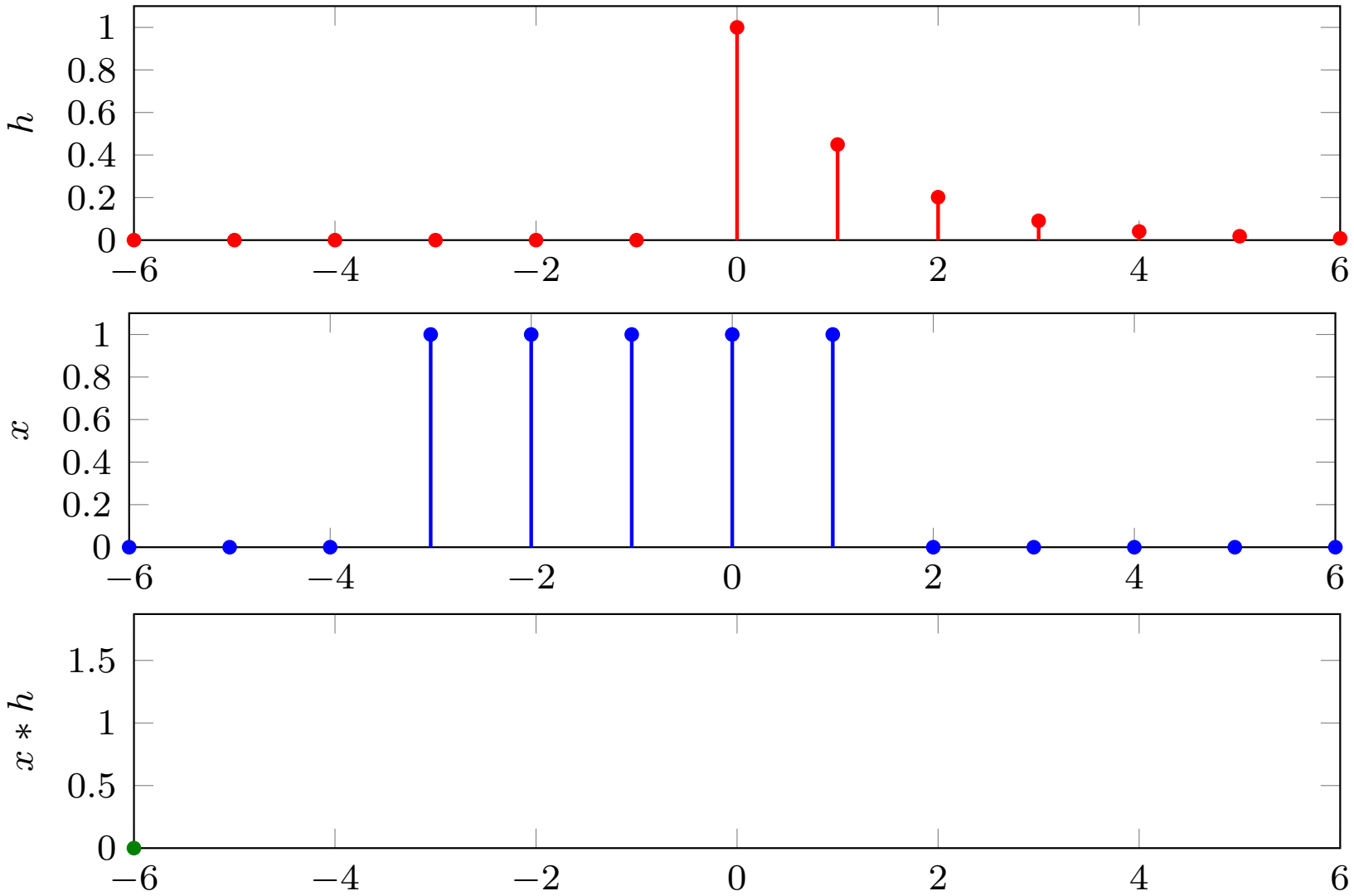
Applying filters:

1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9

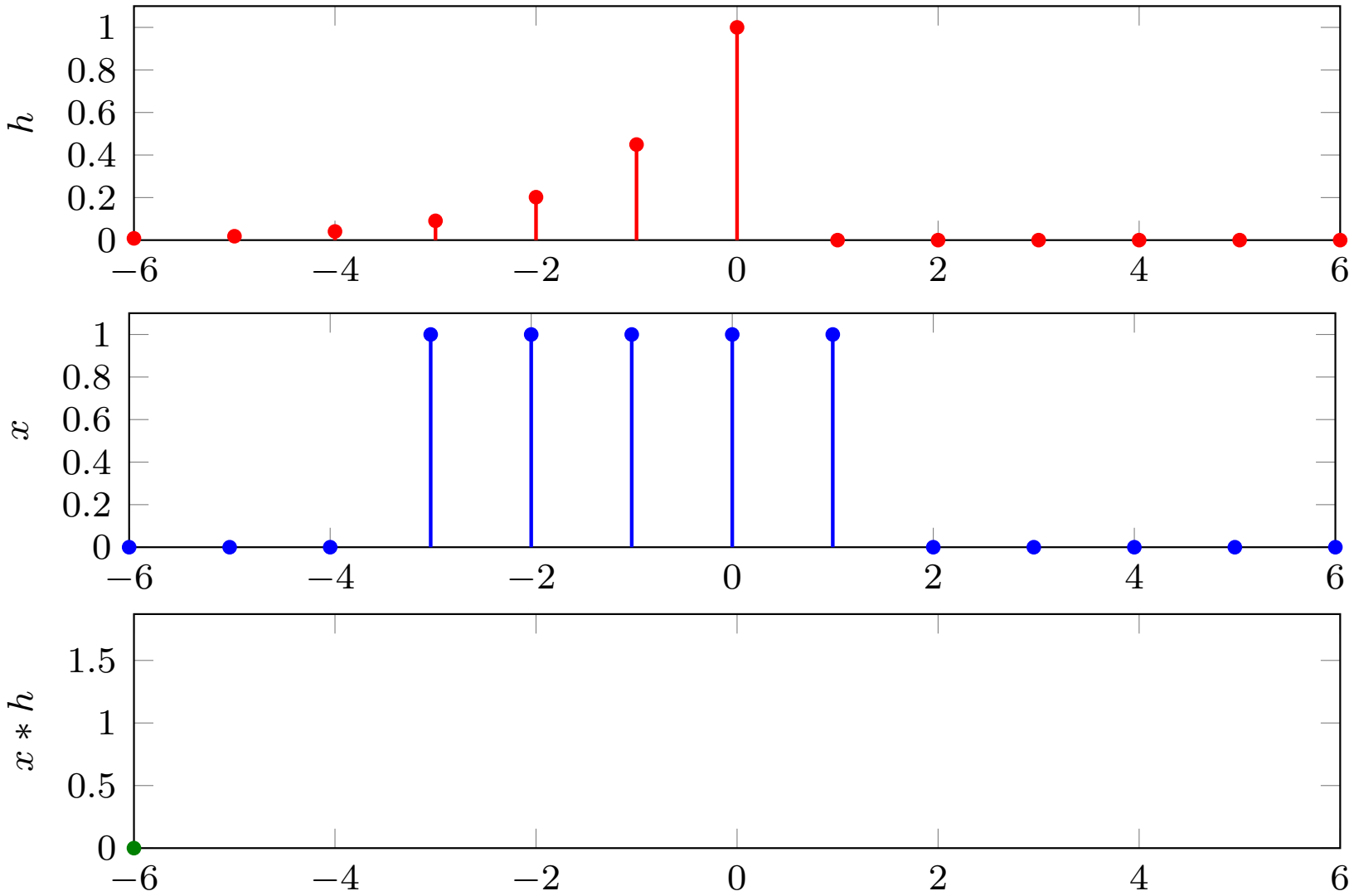
-1 0 1
-1 0 1
-1 0 1

Let's play with filters: http://setosa.io/ev/image-kernels/

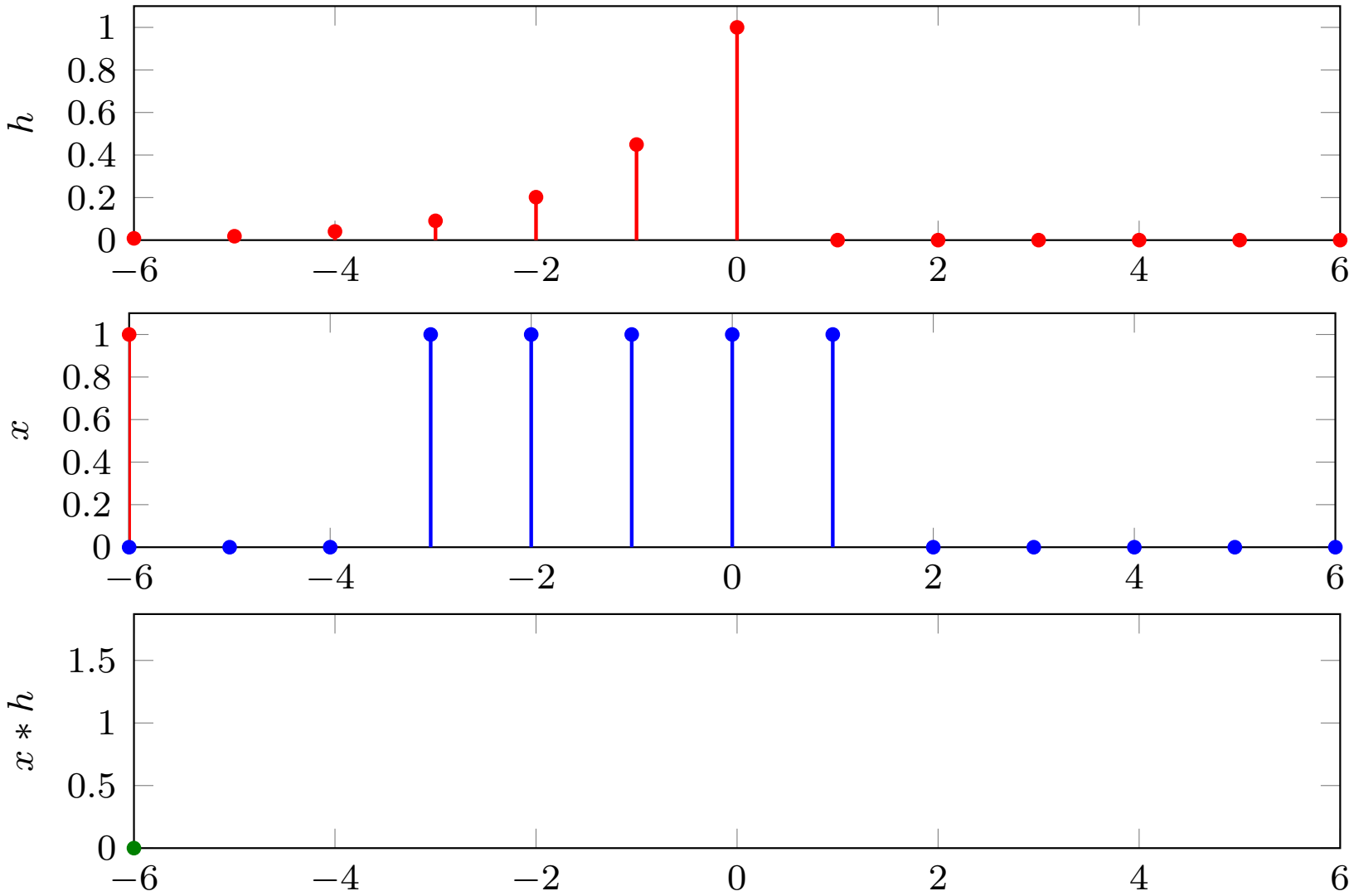
convolution



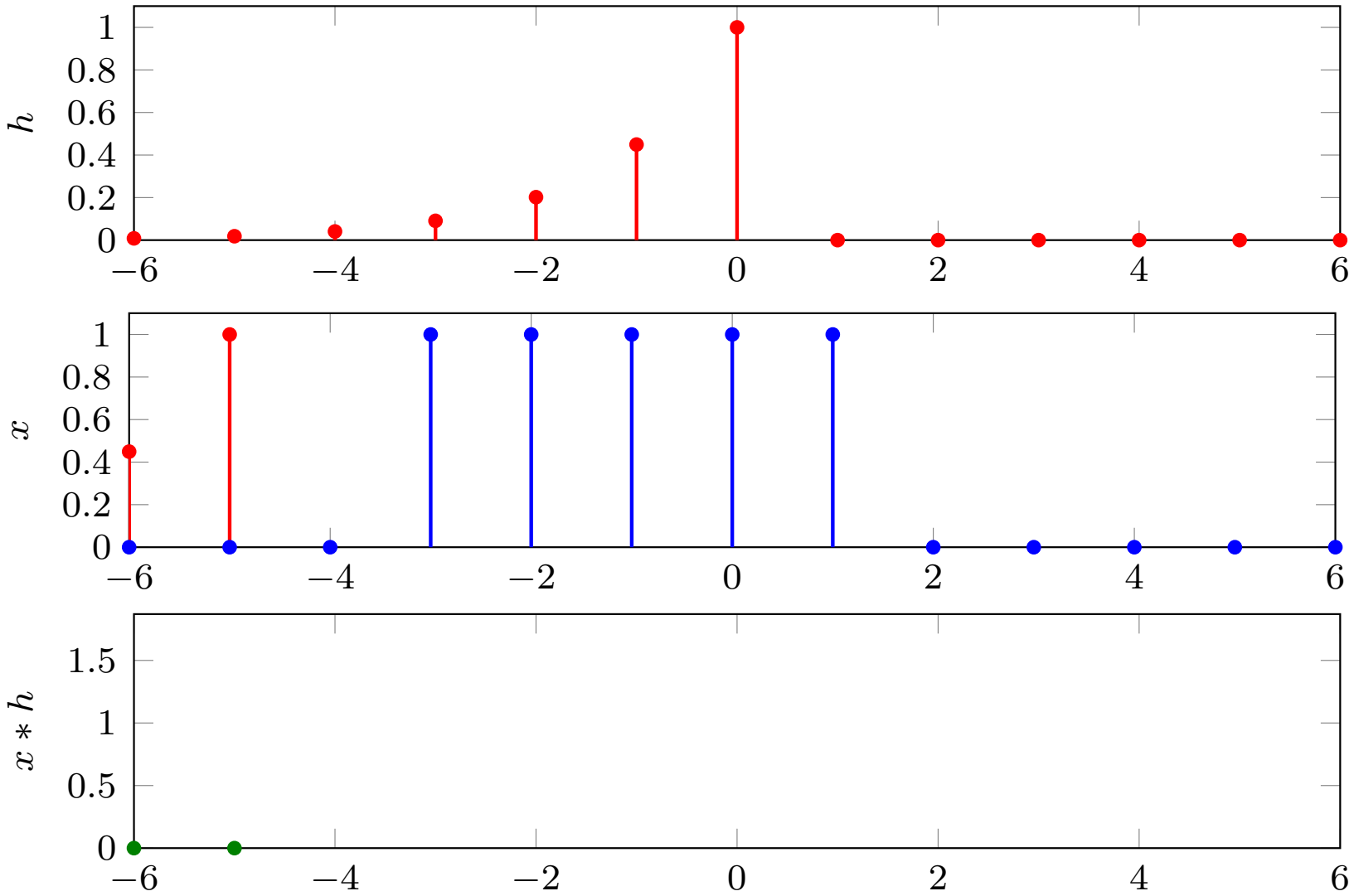
convolution



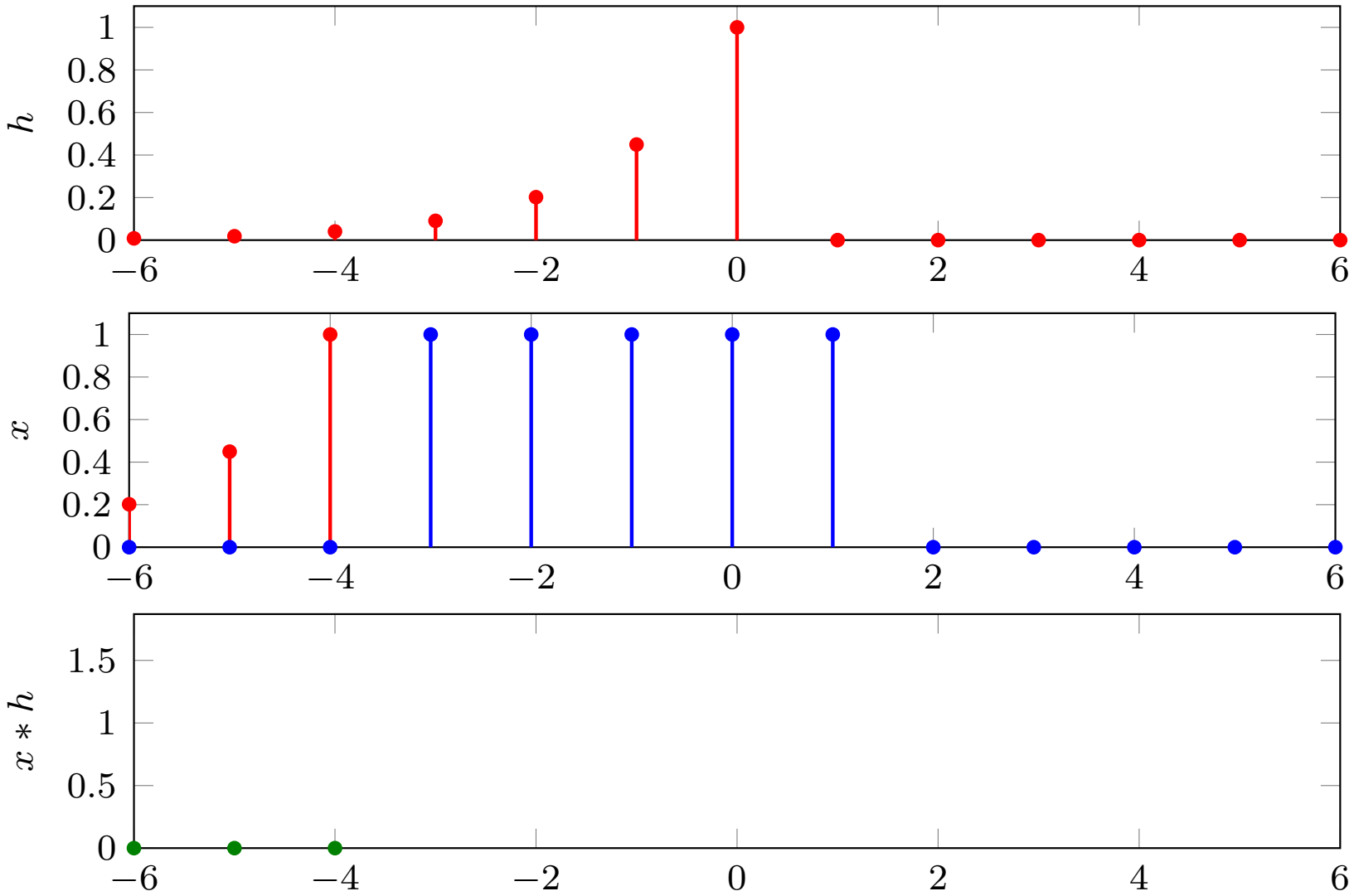
convolution



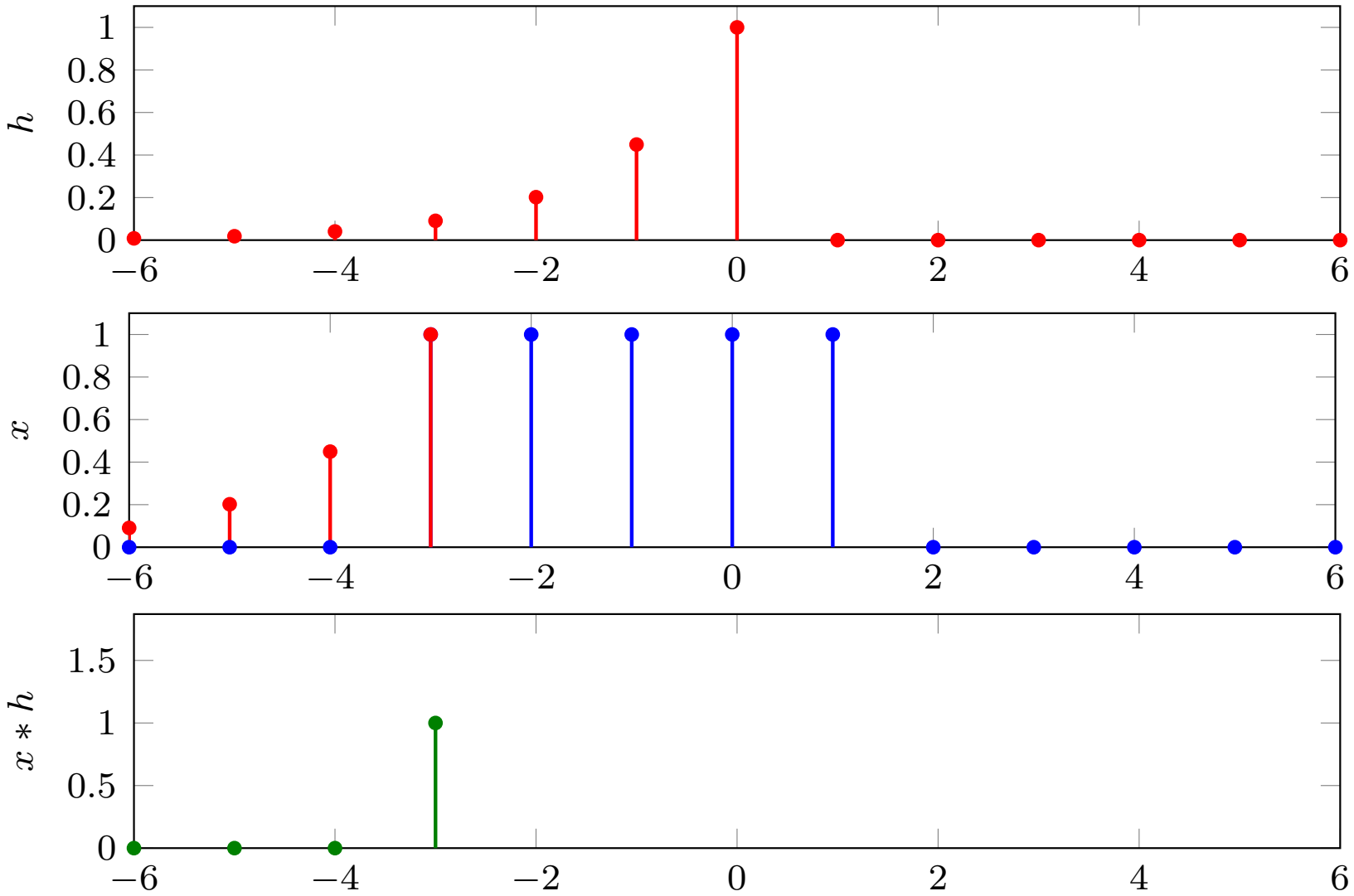
convolution



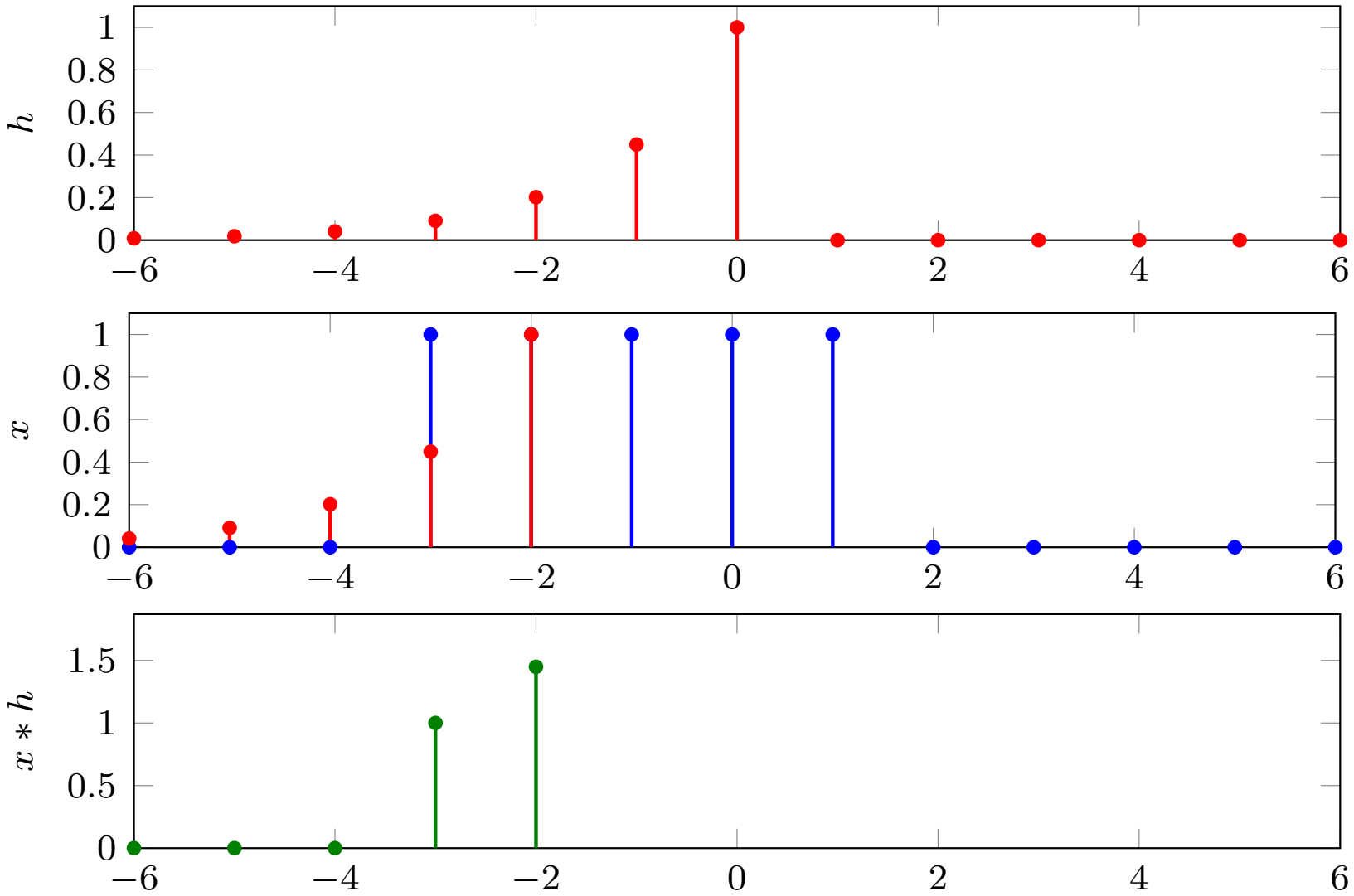
convolution



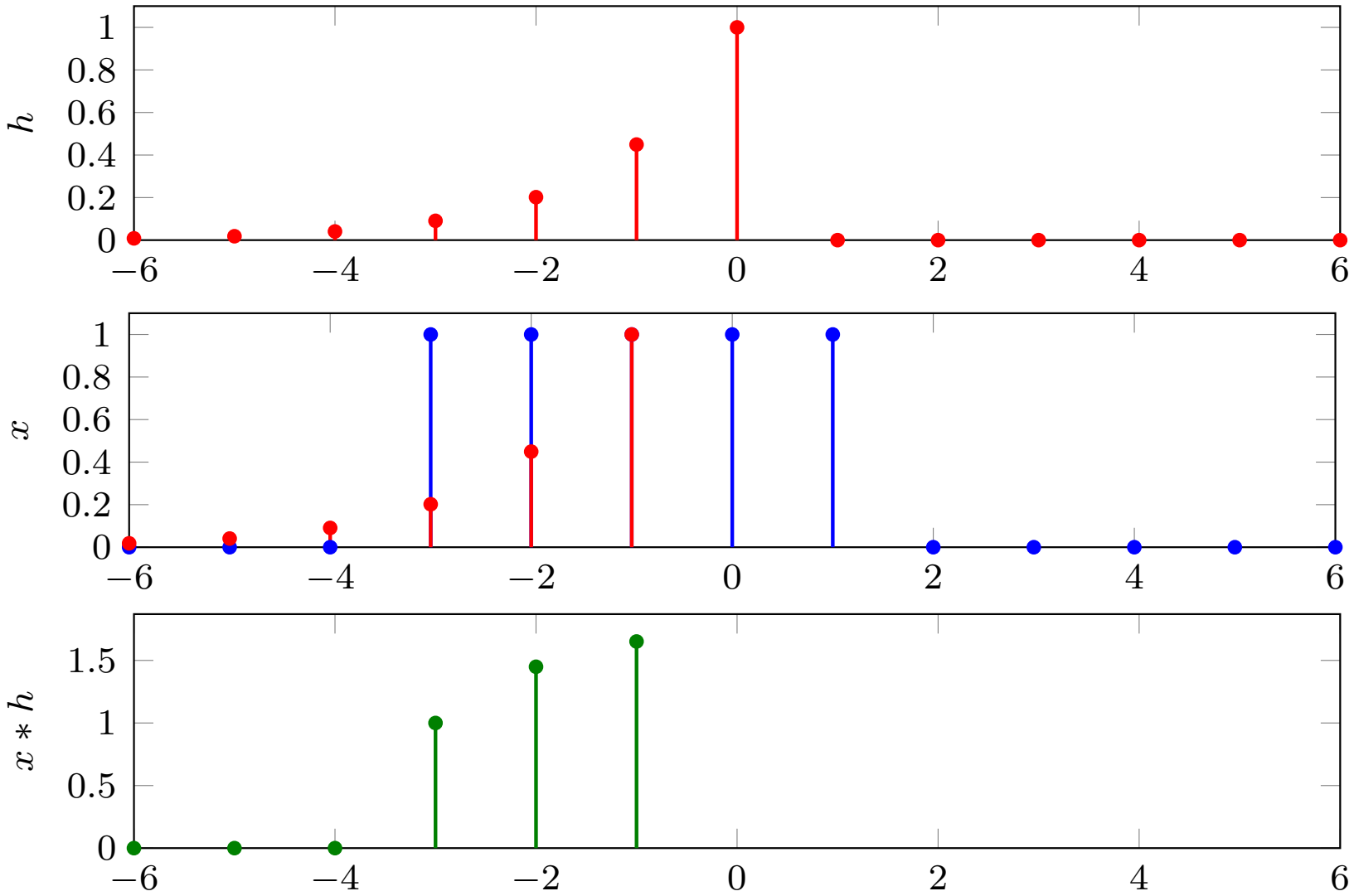
convolution



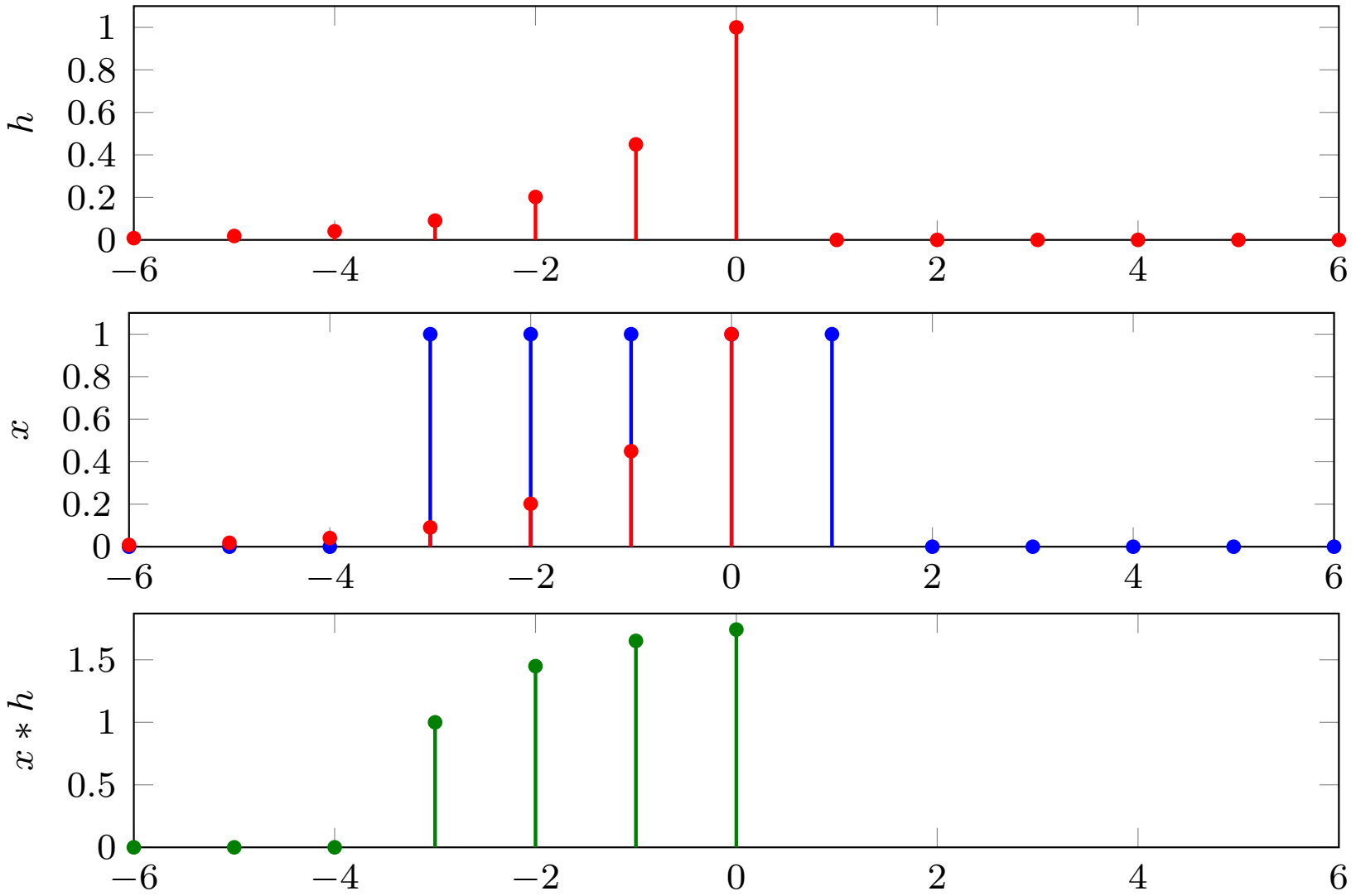
convolution



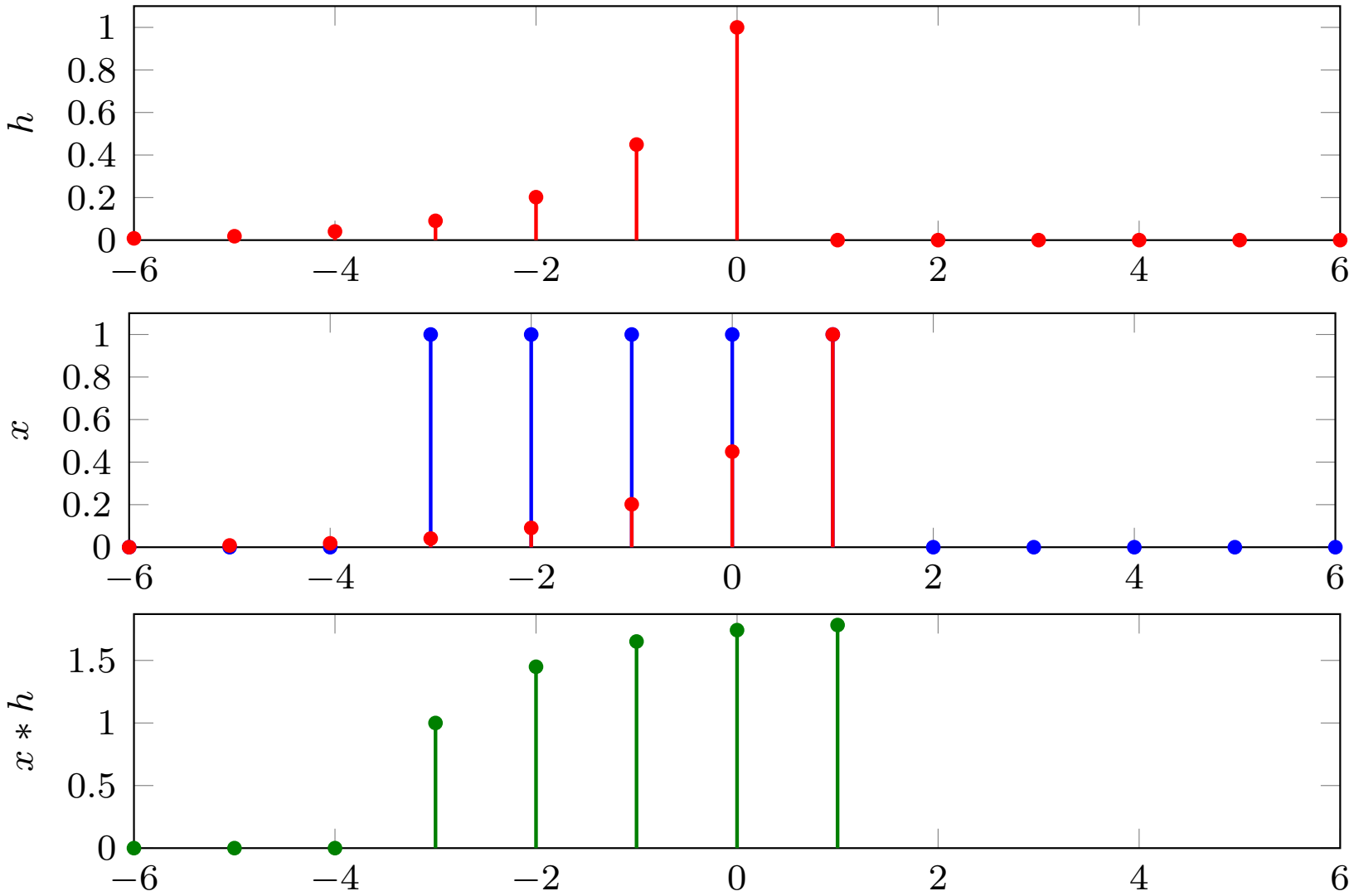
convolution



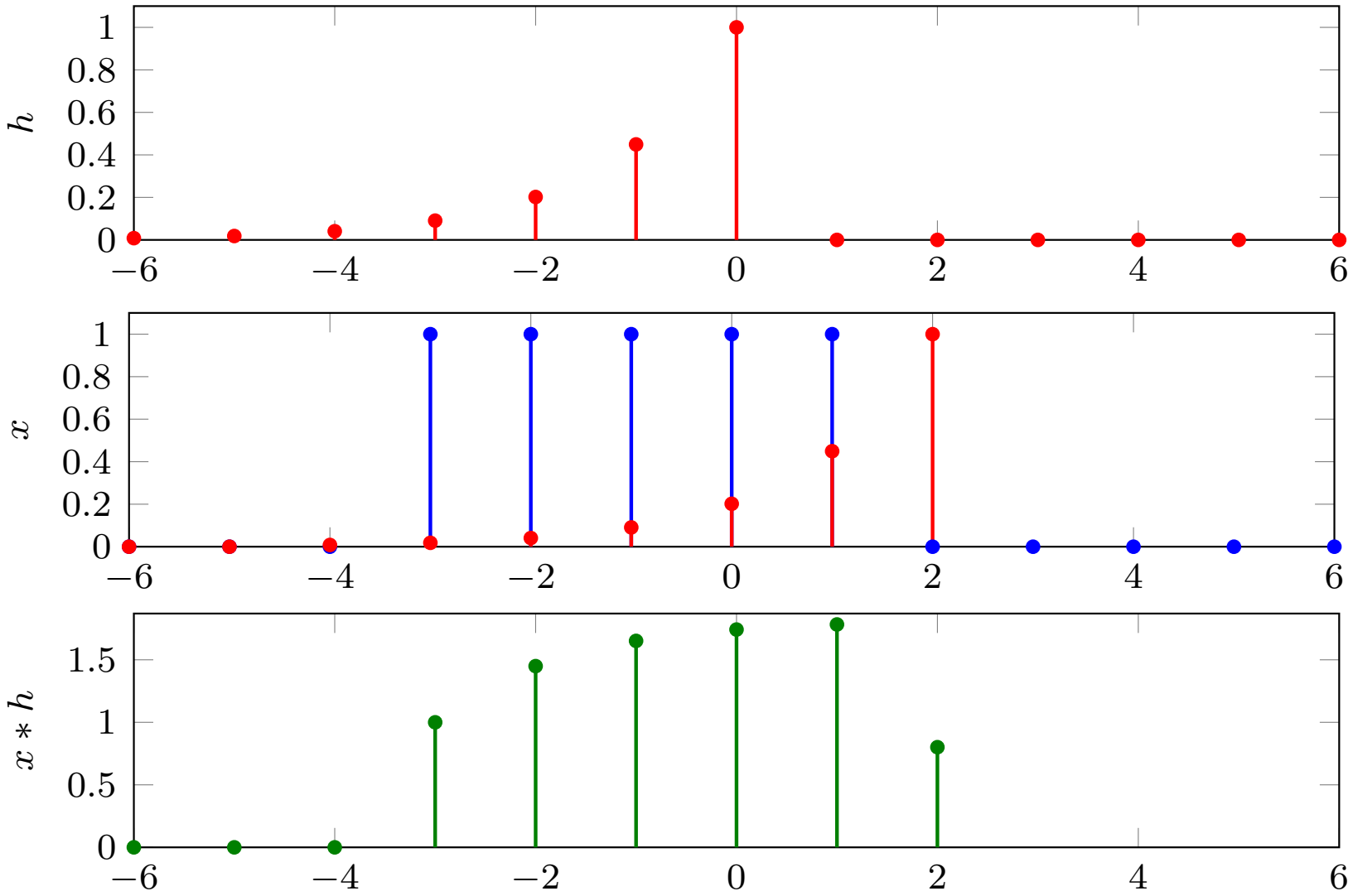
convolution



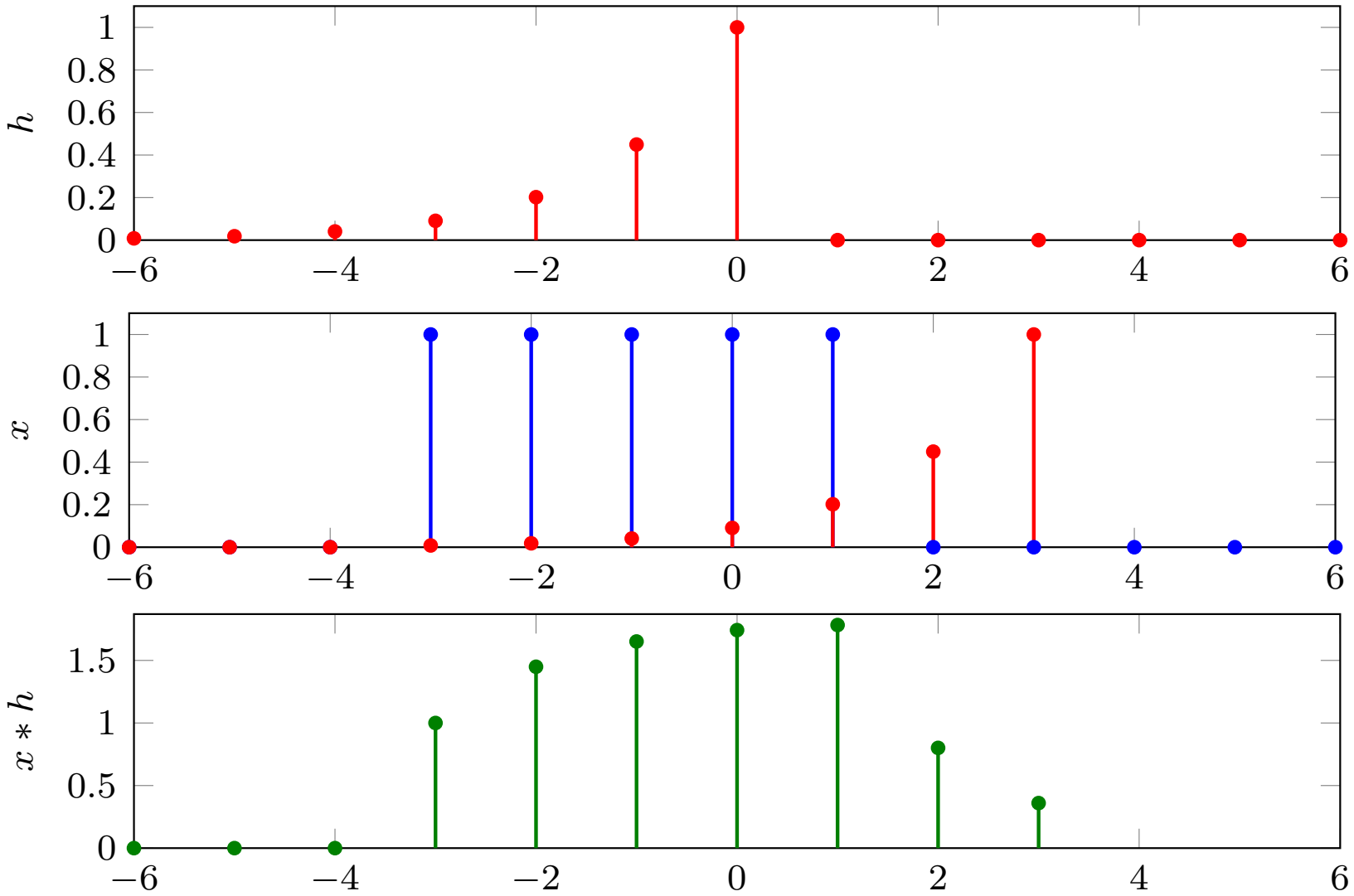
convolution



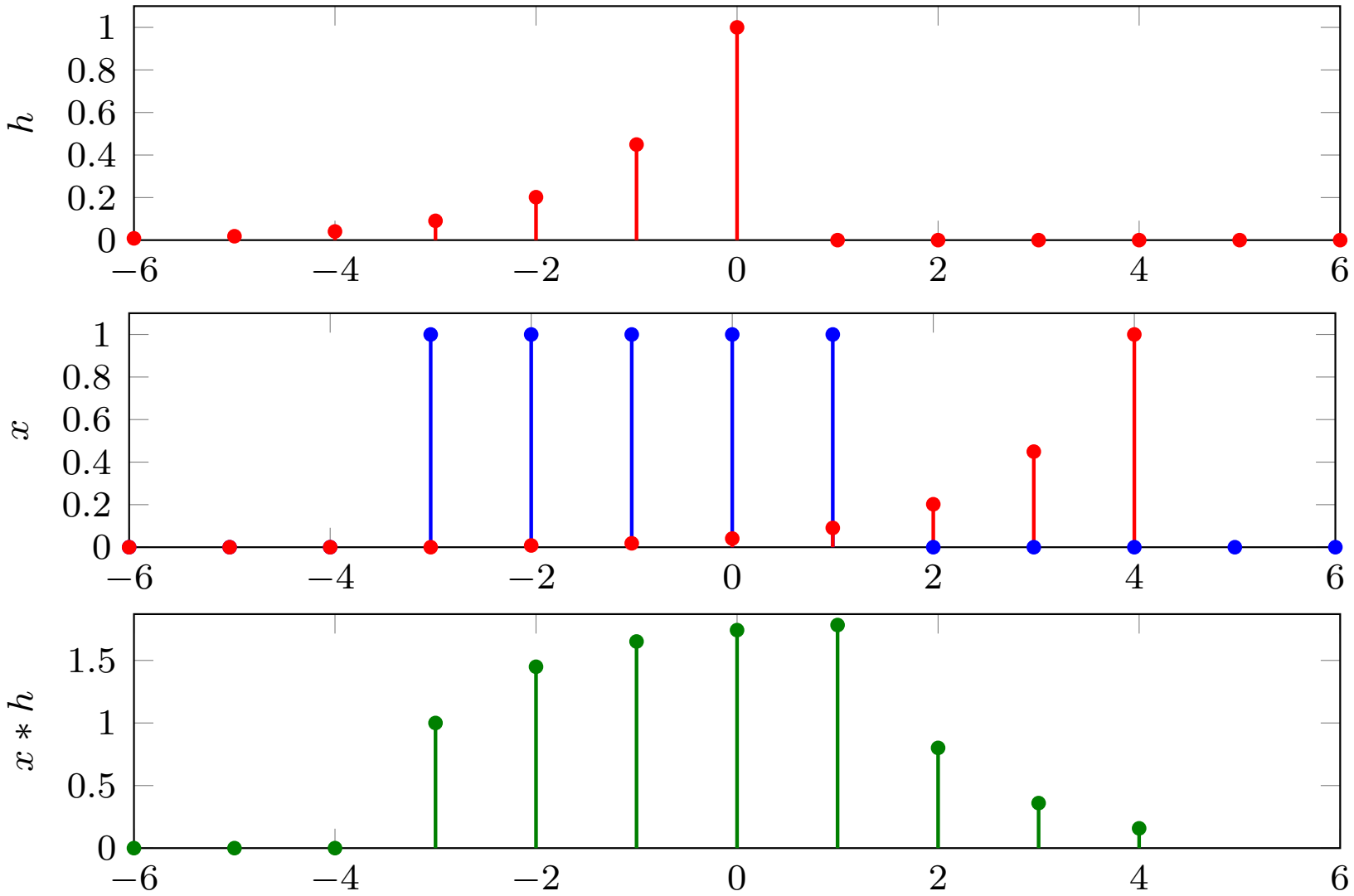
convolution



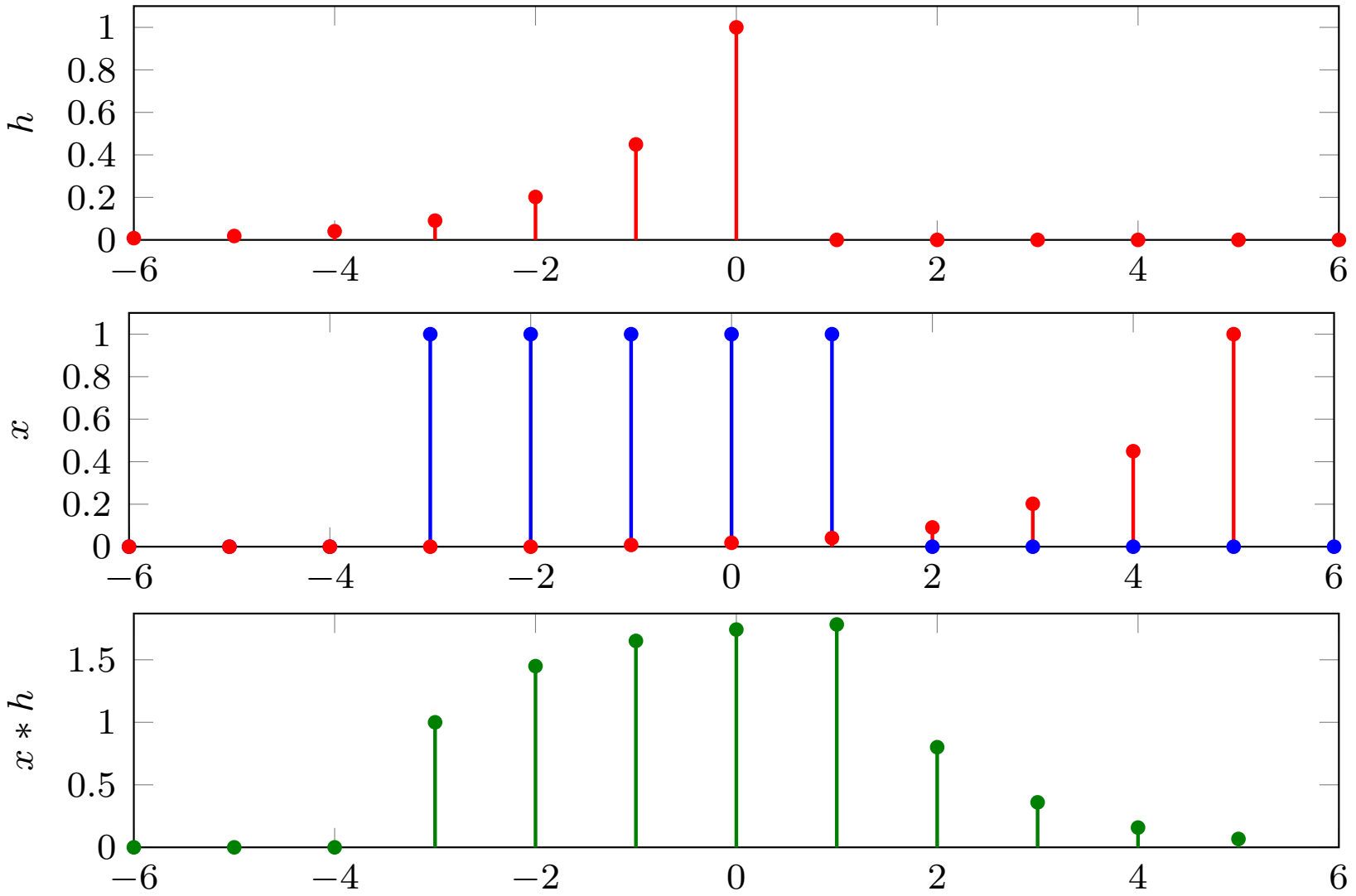
convolution



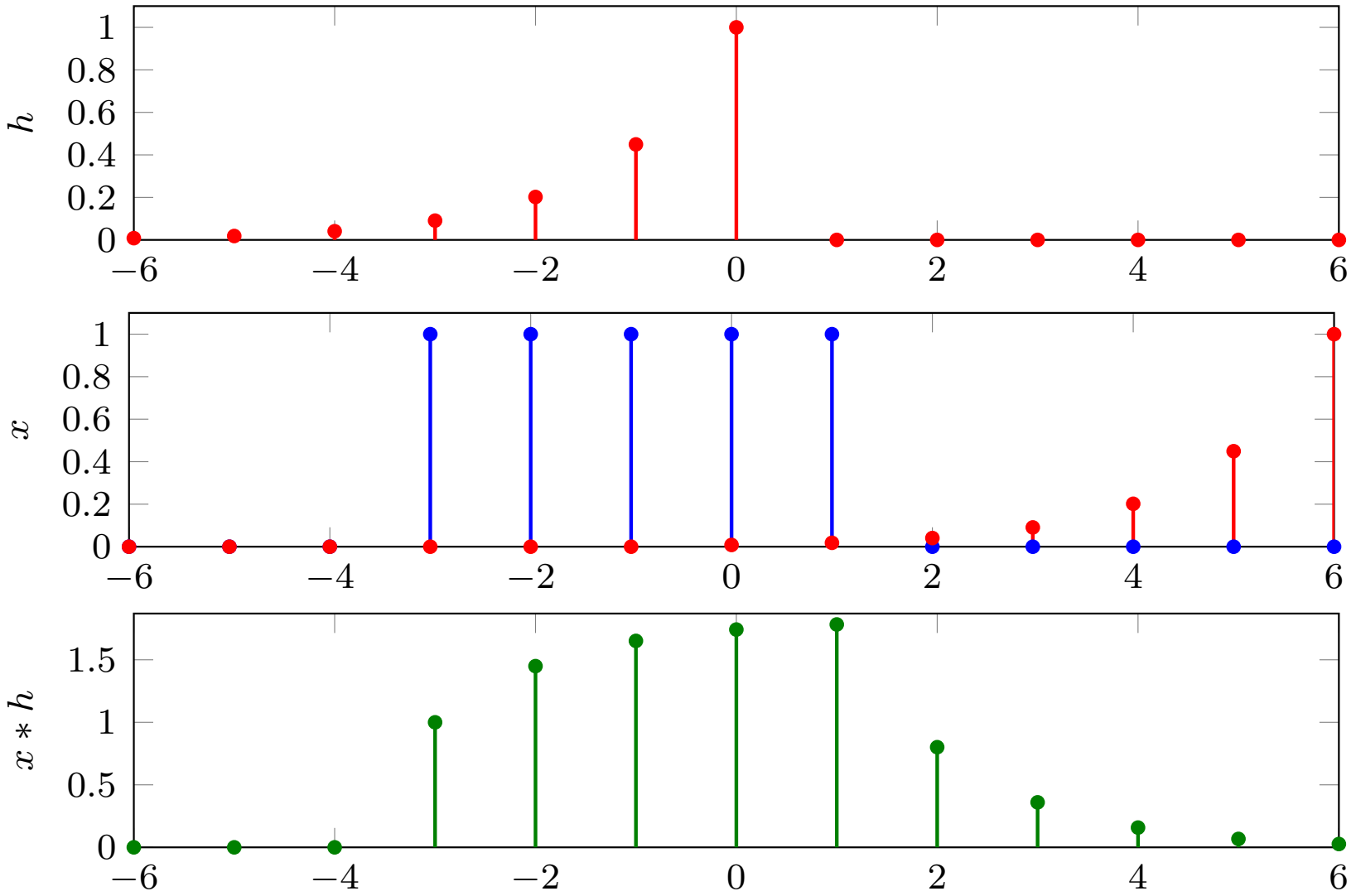
convolution



convolution



convolution



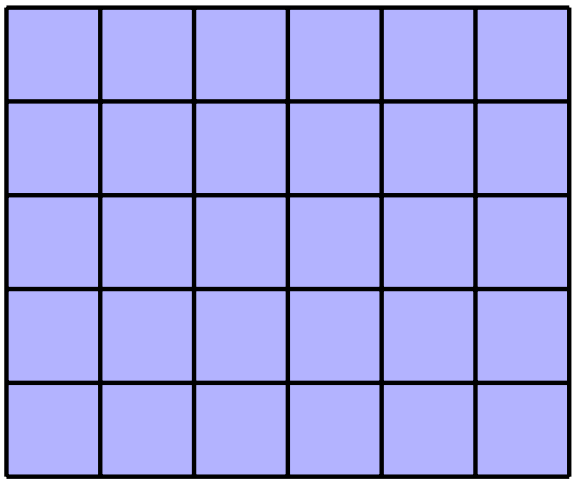
2d convolution

1	2	3
4	5	6
7	8	9

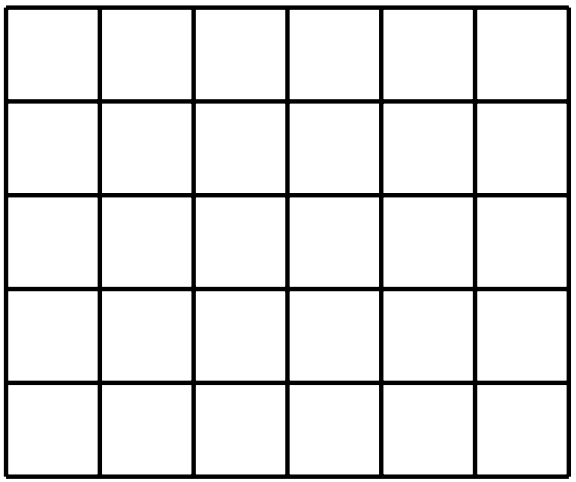
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

2d convolution

1	2	3
4	5	6
7	8	9

h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

9	8	7			
6	5	4			
3	2	1			

x

$x * h$

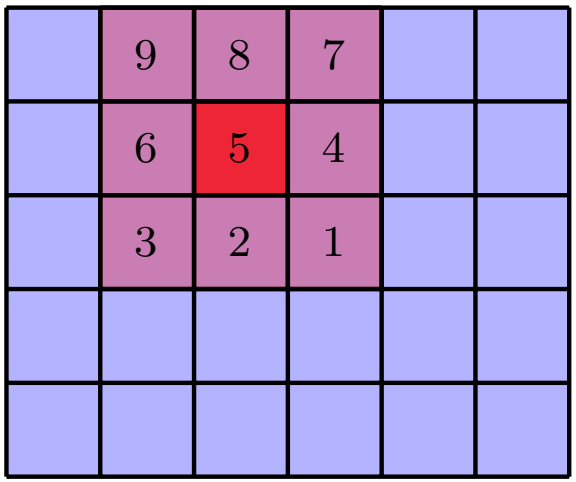
2d convolution

1	2	3
4	5	6
7	8	9

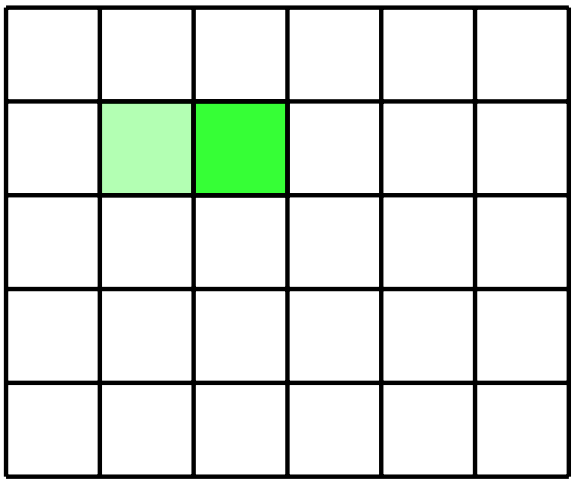
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

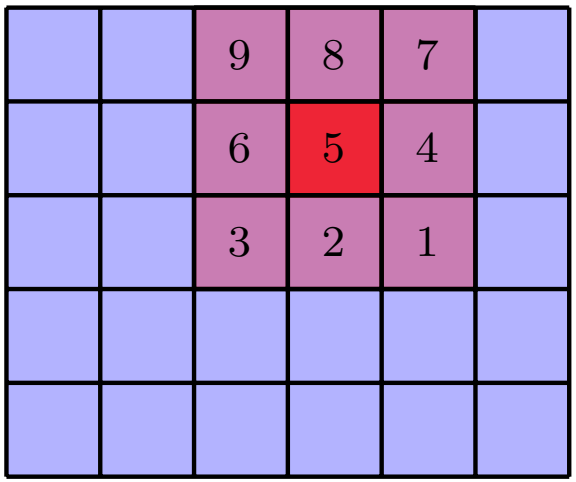
2d convolution

1	2	3
4	5	6
7	8	9

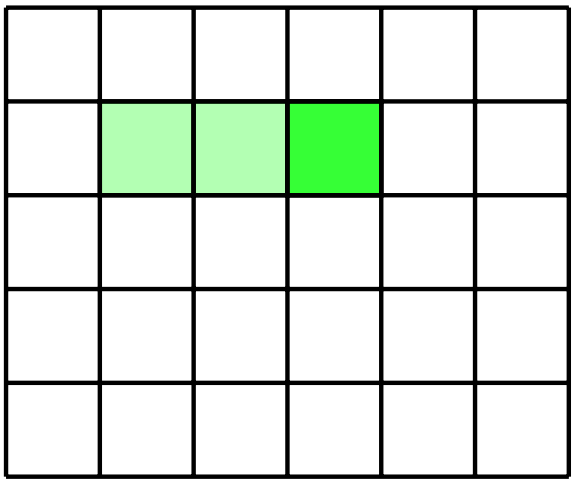
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

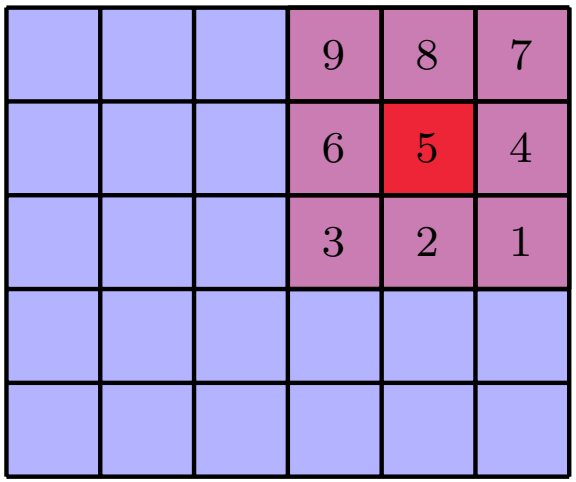
2d convolution

1	2	3
4	5	6
7	8	9

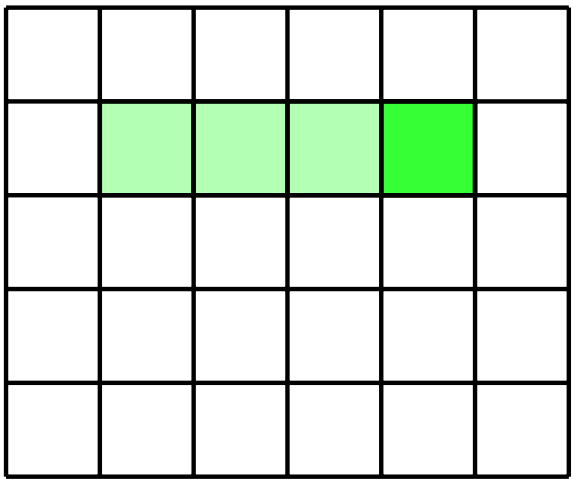
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

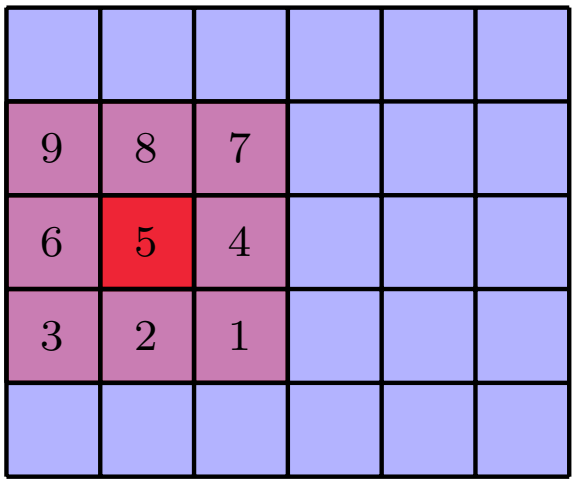
2d convolution

1	2	3
4	5	6
7	8	9

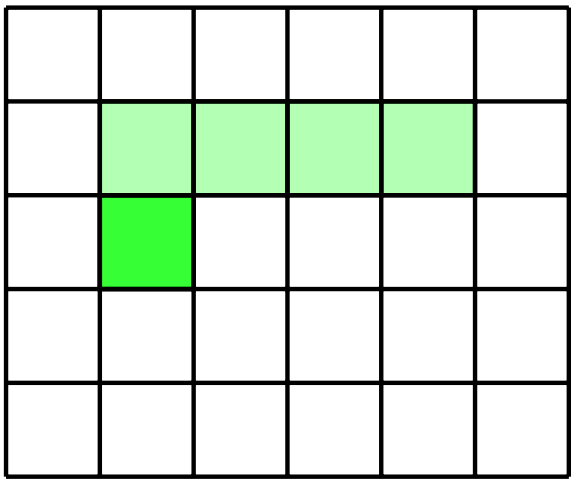
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

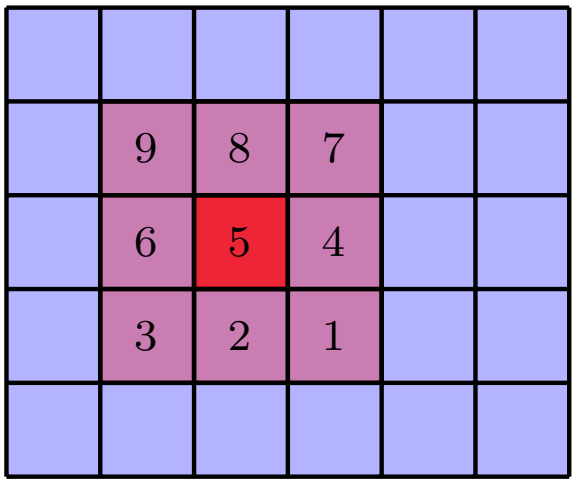
2d convolution

1	2	3
4	5	6
7	8	9

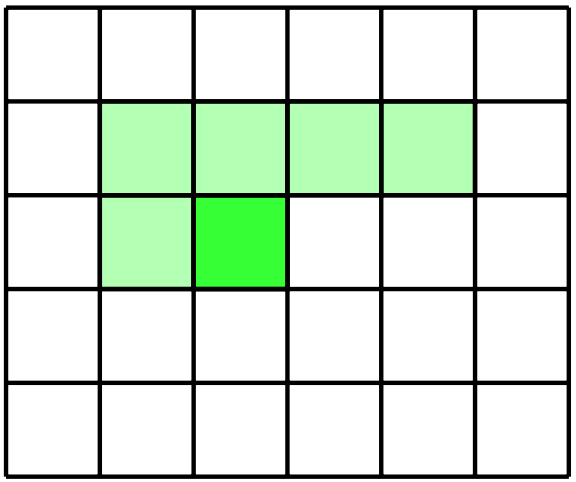
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

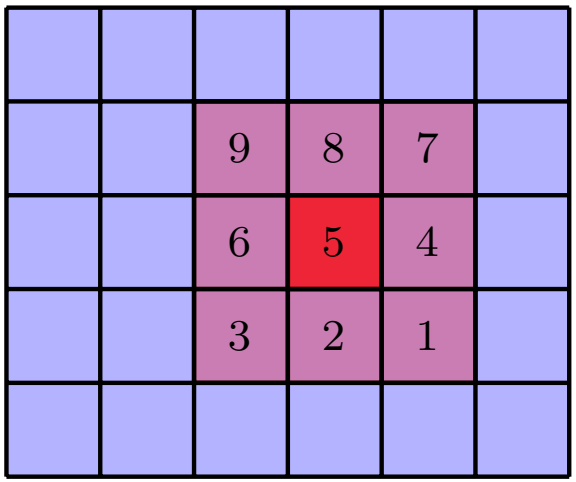
2d convolution

1	2	3
4	5	6
7	8	9

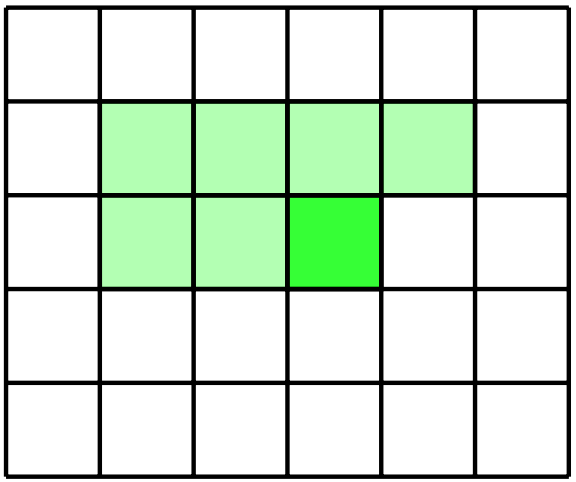
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

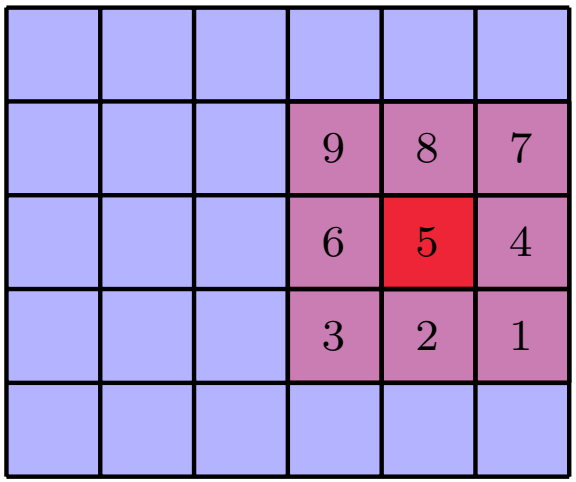
2d convolution

1	2	3
4	5	6
7	8	9

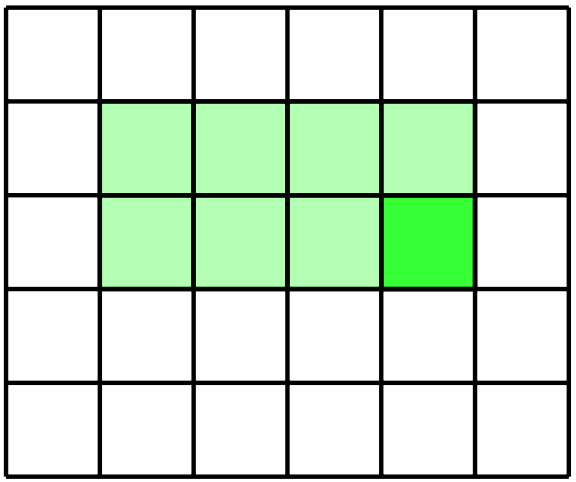
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

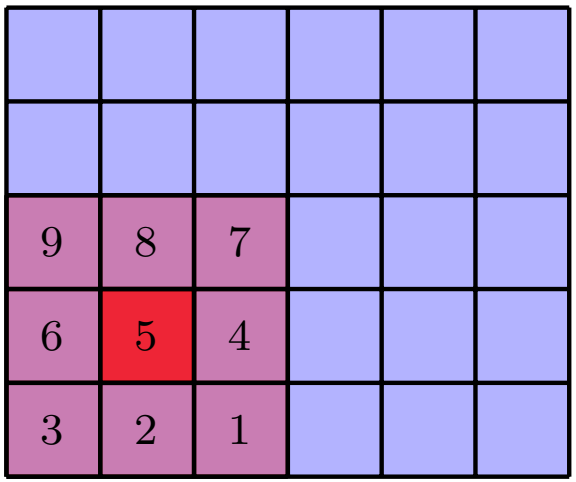
2d convolution

1	2	3
4	5	6
7	8	9

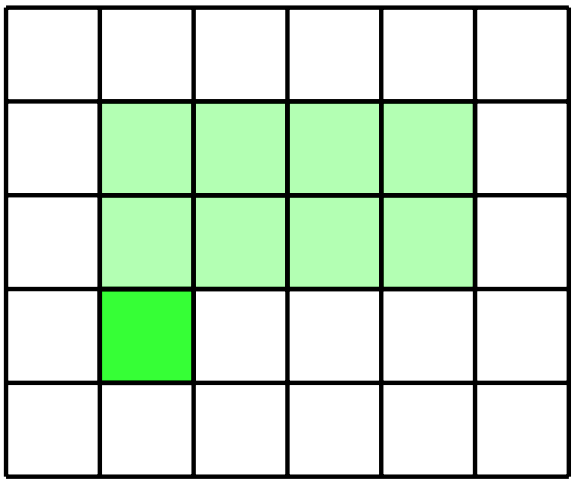
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

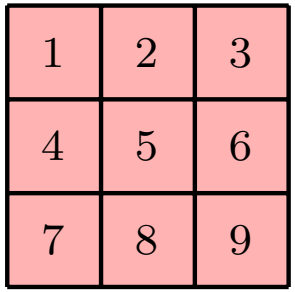


x



$x * h$

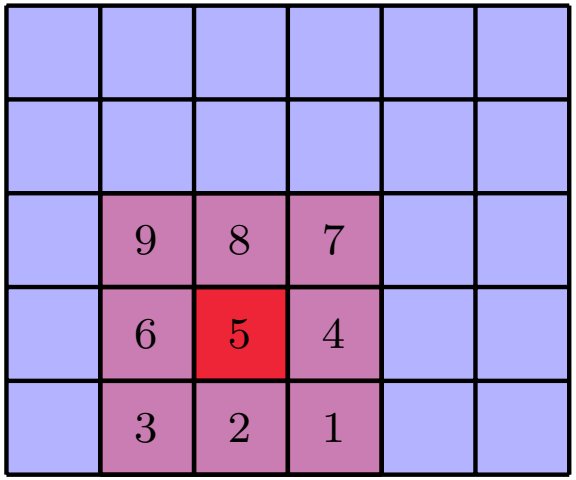
2d convolution



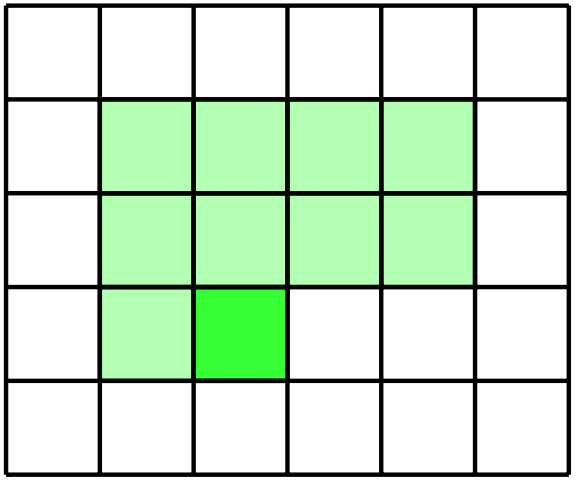
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

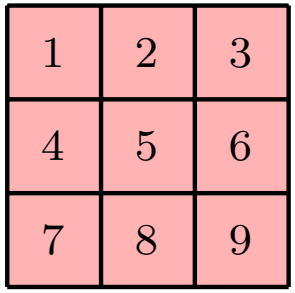


x



$x * h$

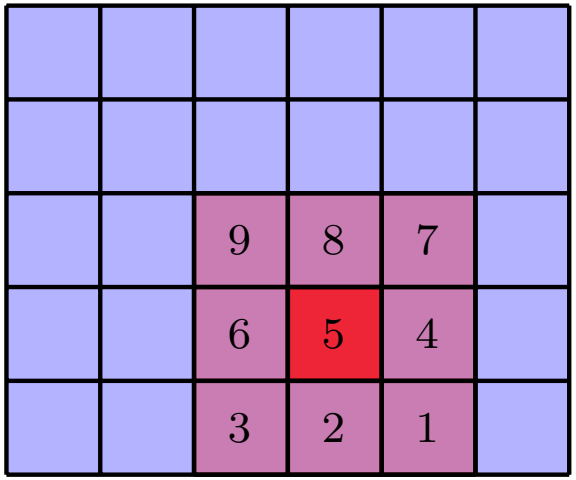
2d convolution



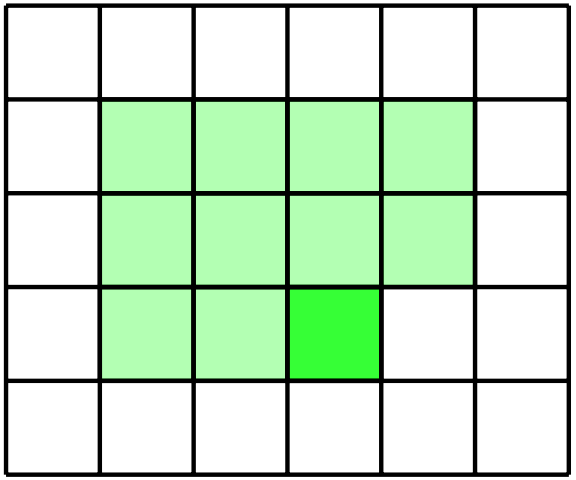
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

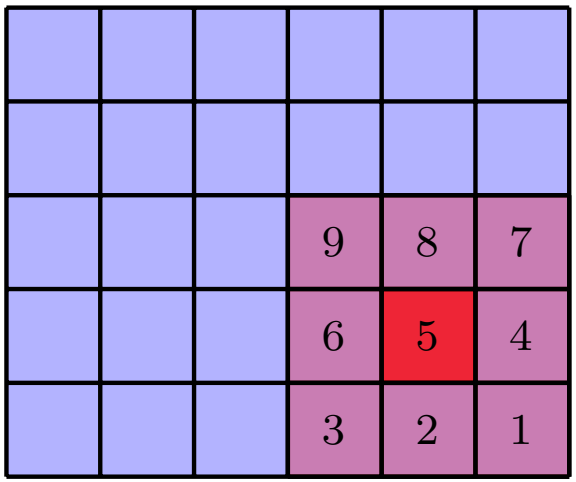
2d convolution

1	2	3
4	5	6
7	8	9

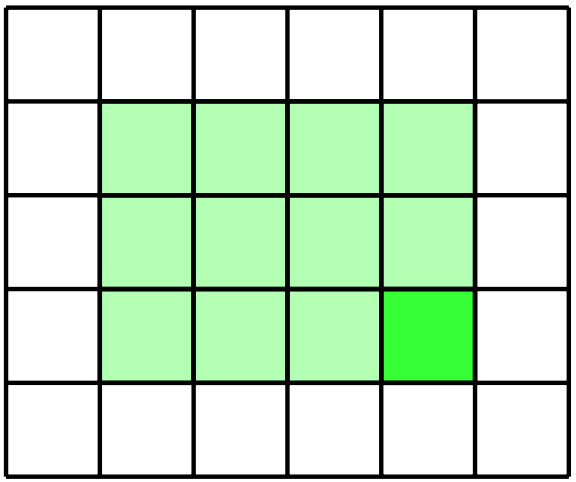
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

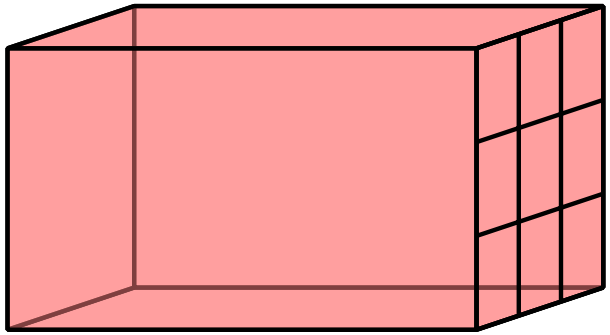


x



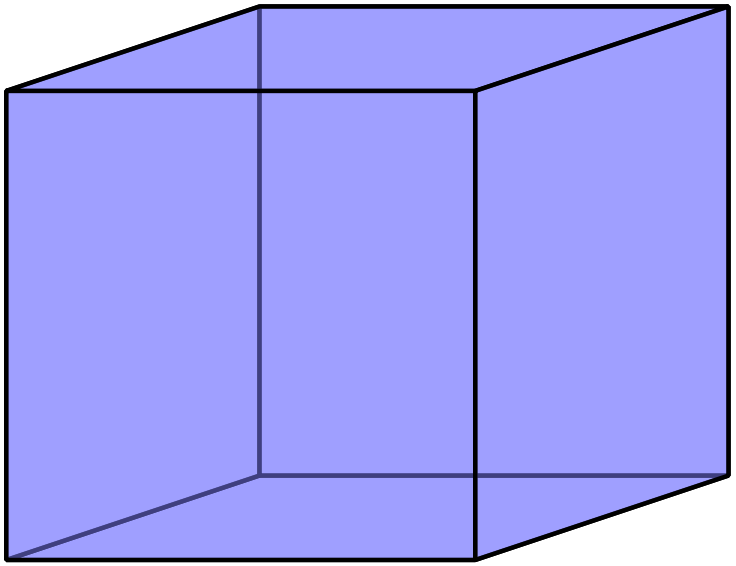
$x * h$

convolution on feature maps

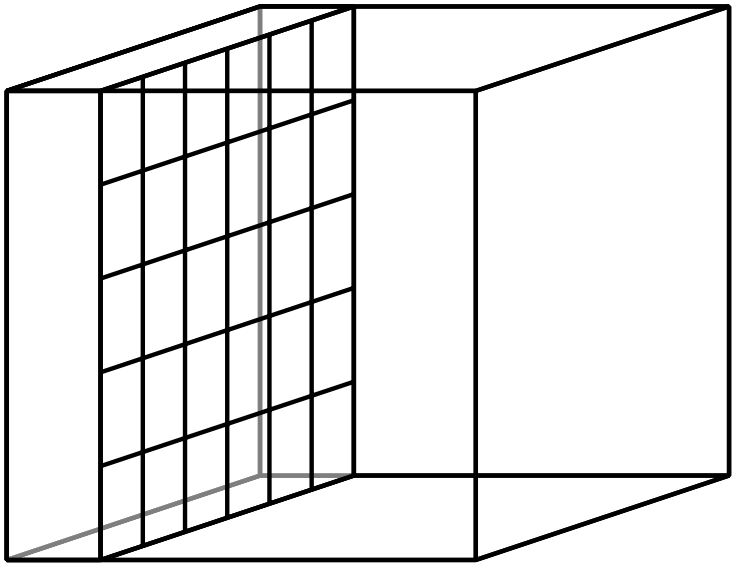


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

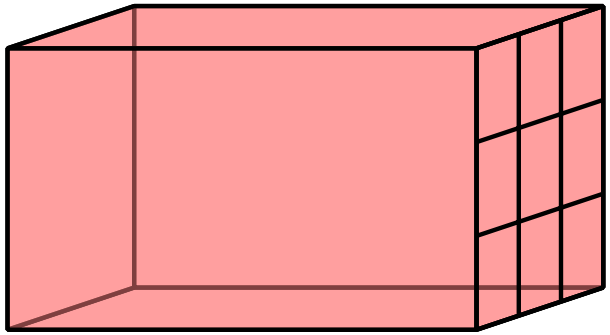


input $\mathbf{x}$



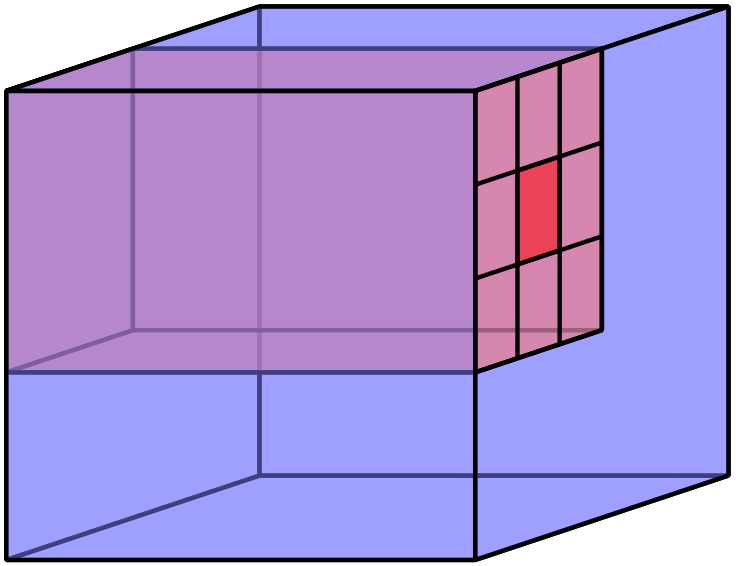
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

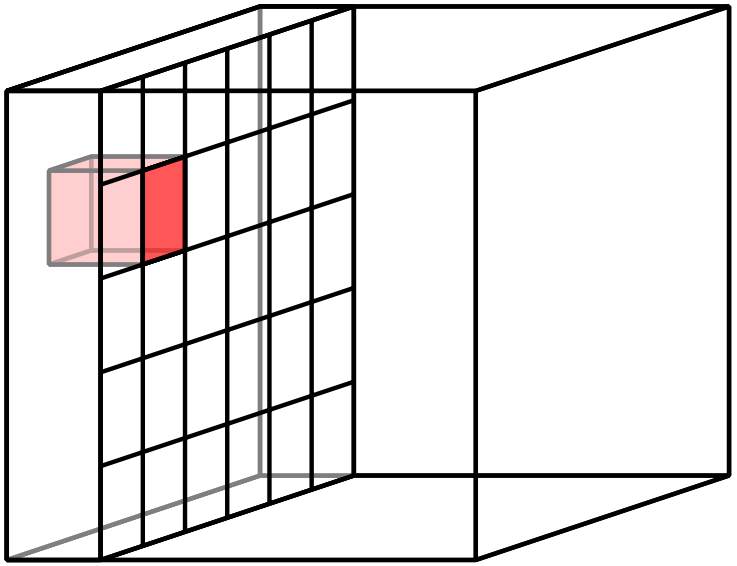


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

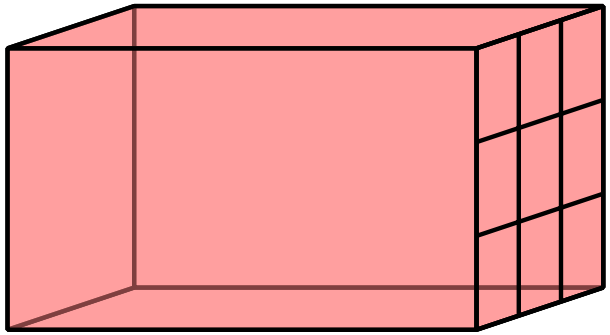


input $\mathbf{x}$



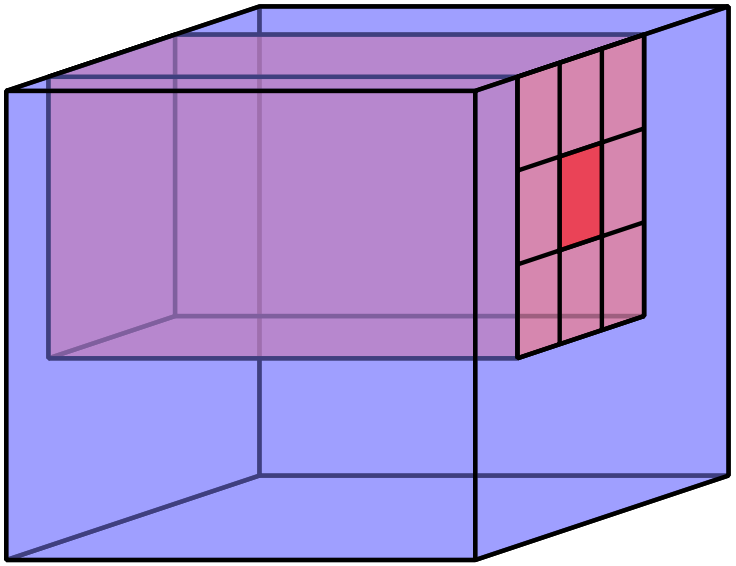
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

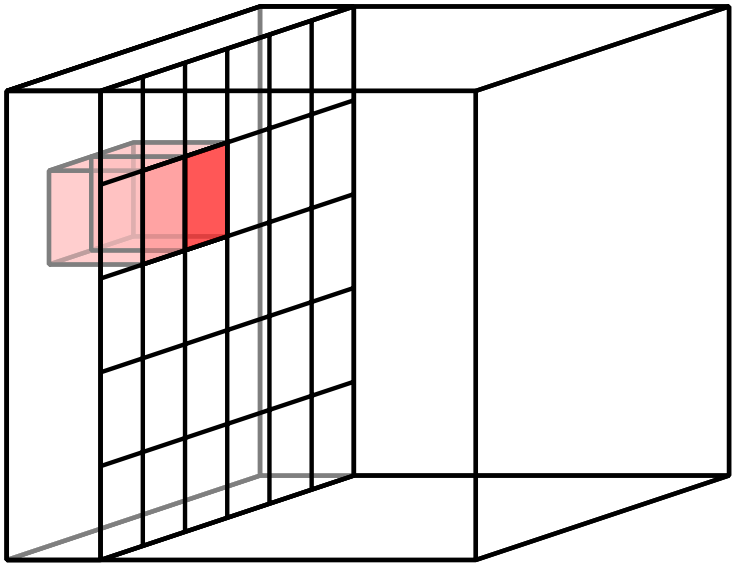


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

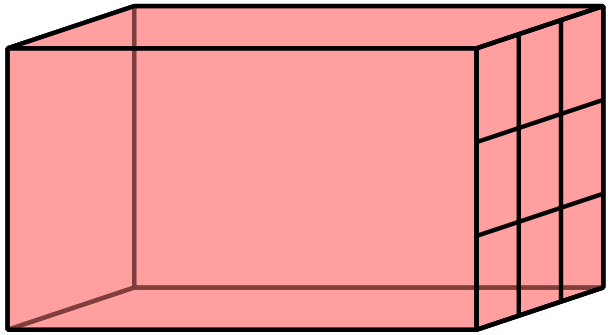


input $\mathbf{x}$



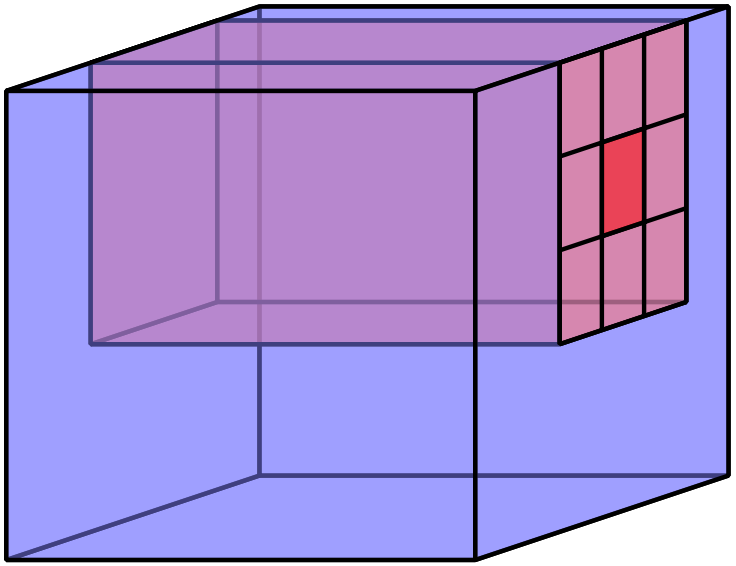
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

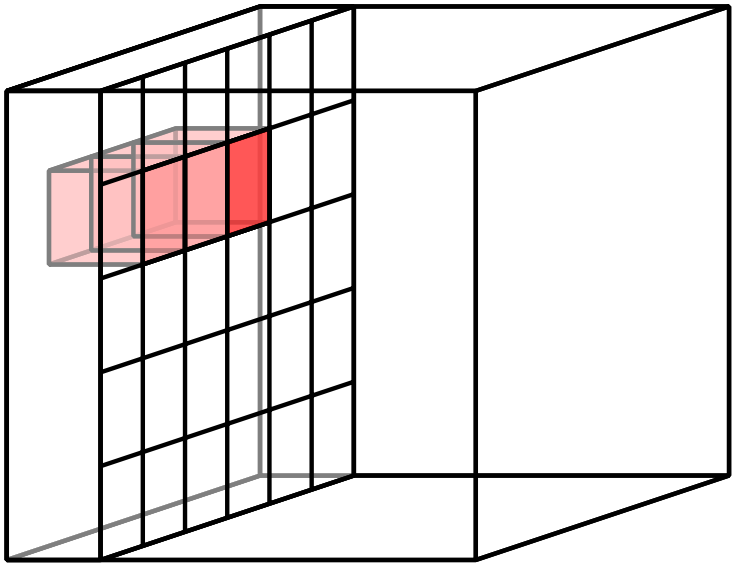


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

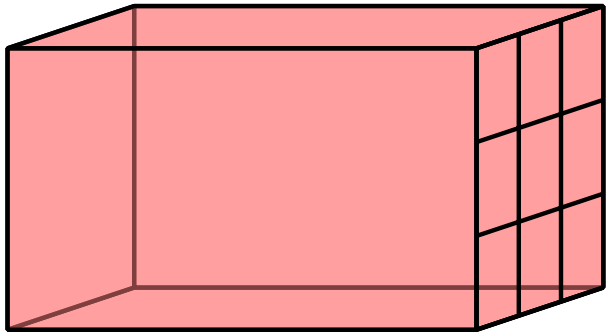


input $\mathbf{x}$



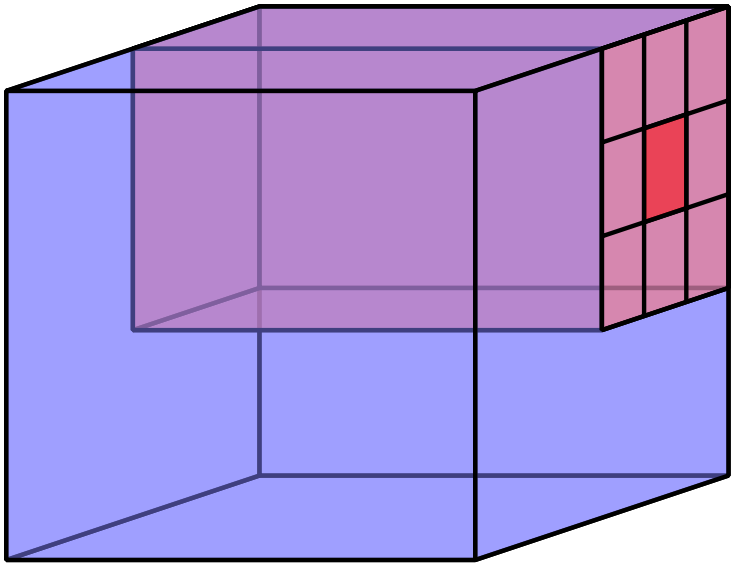
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

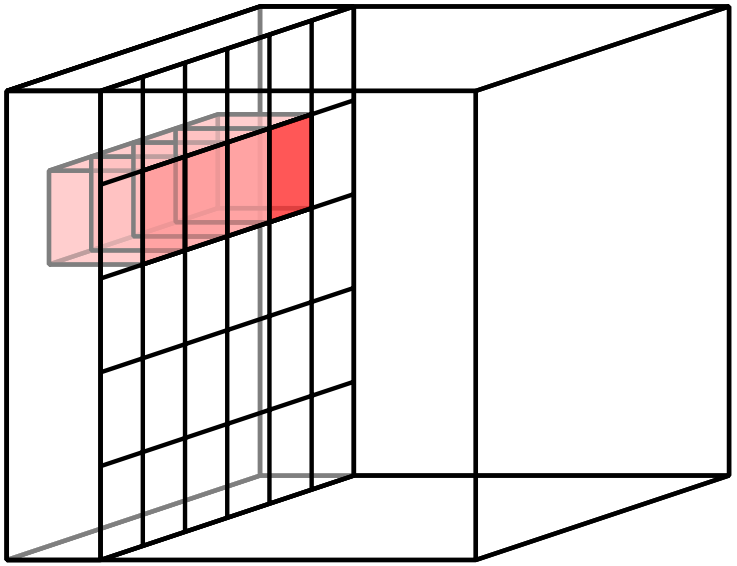


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

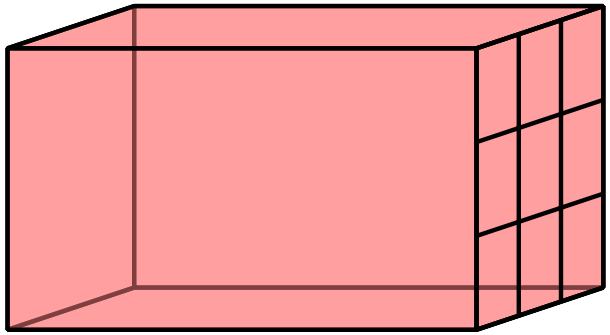


input $\mathbf{x}$



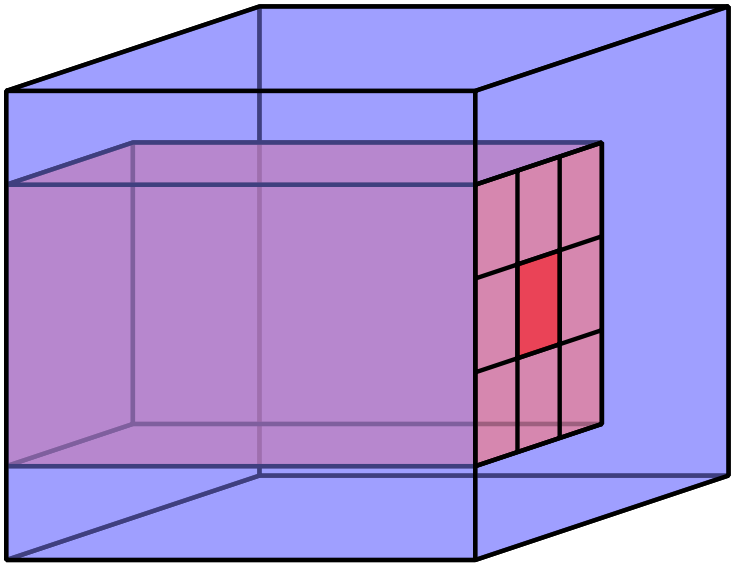
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

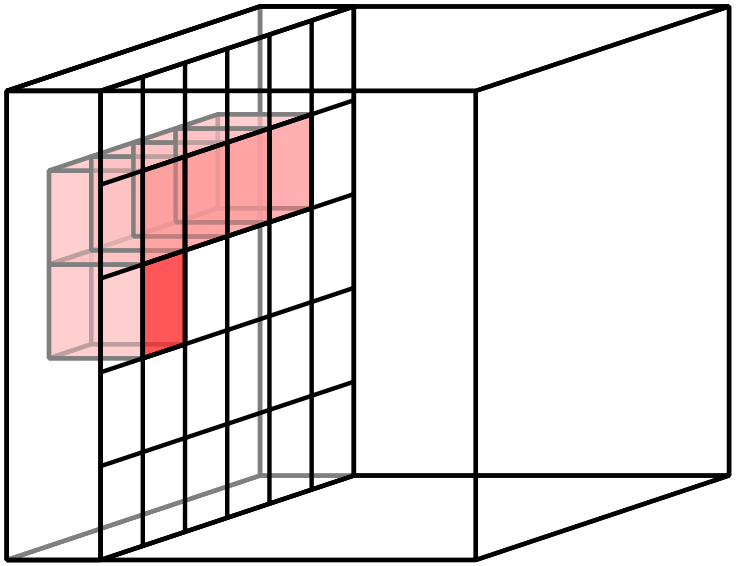


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

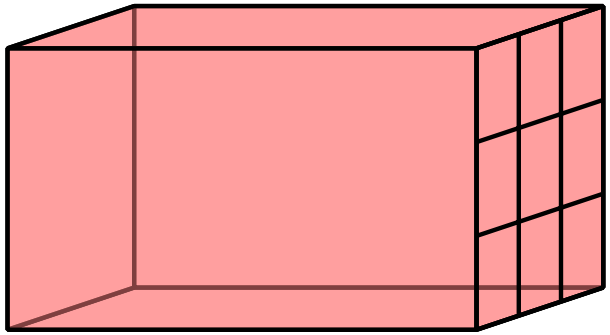


input $\mathbf{x}$



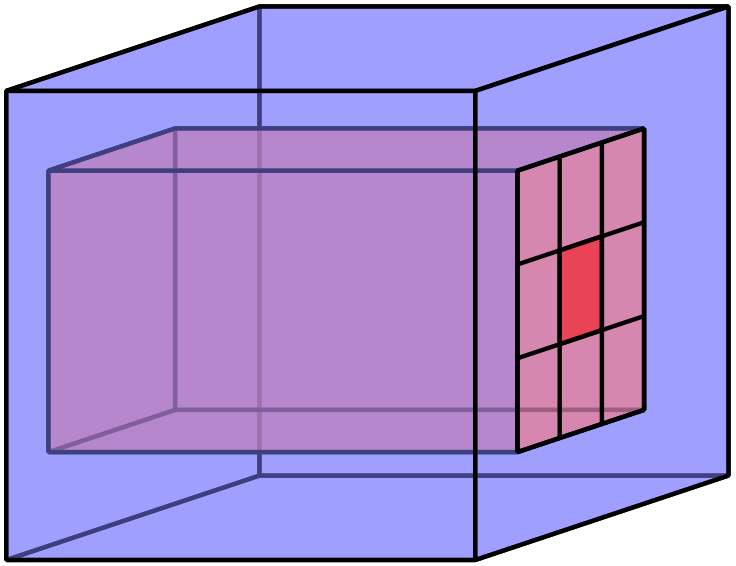
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

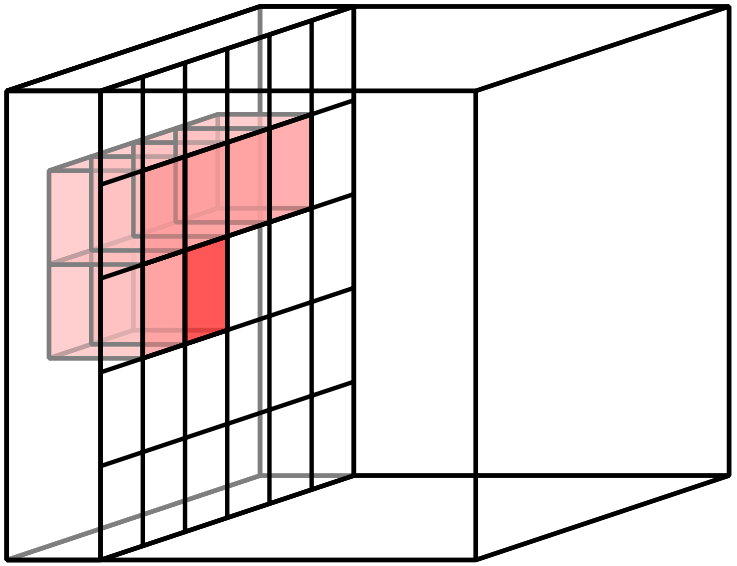


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

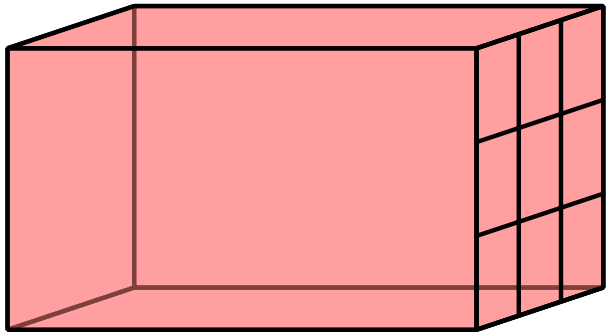


input $\mathbf{x}$



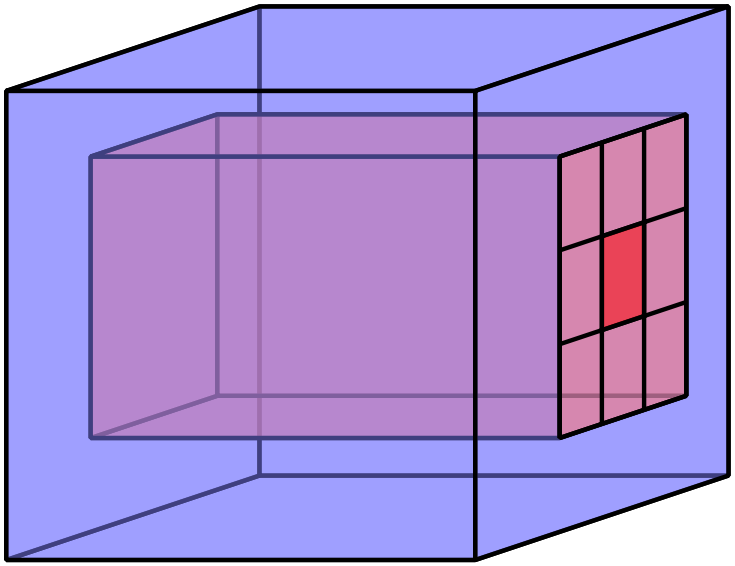
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

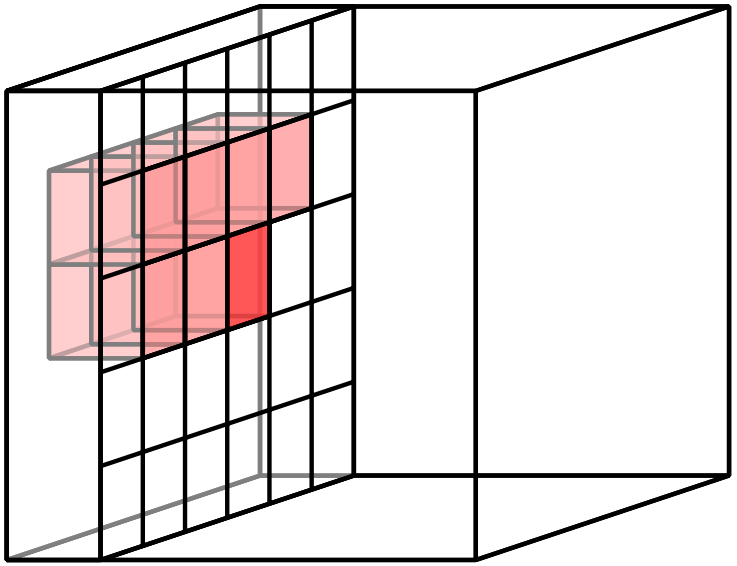


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

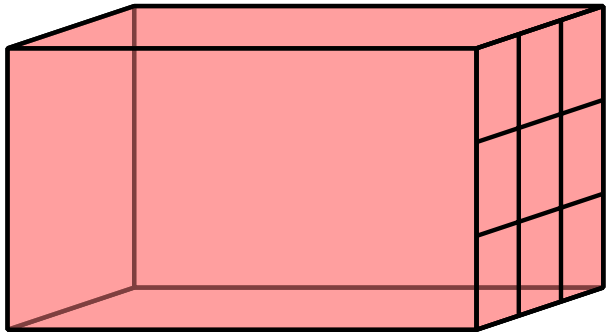


input $\mathbf{x}$



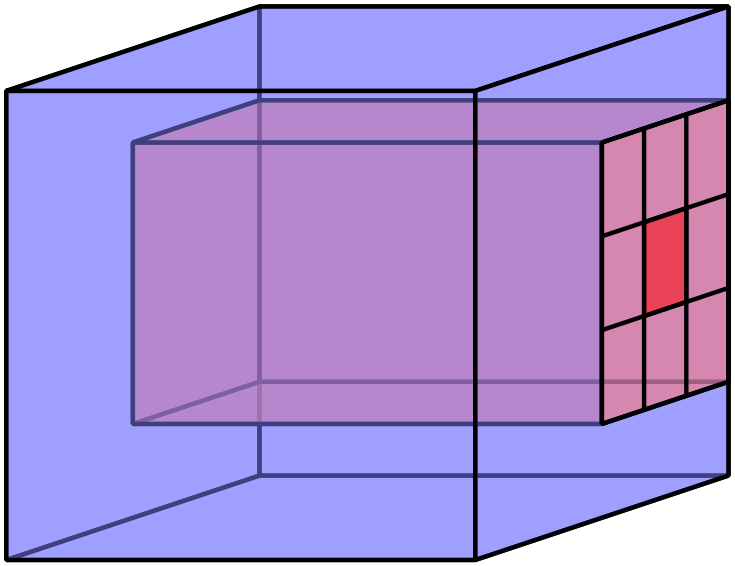
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

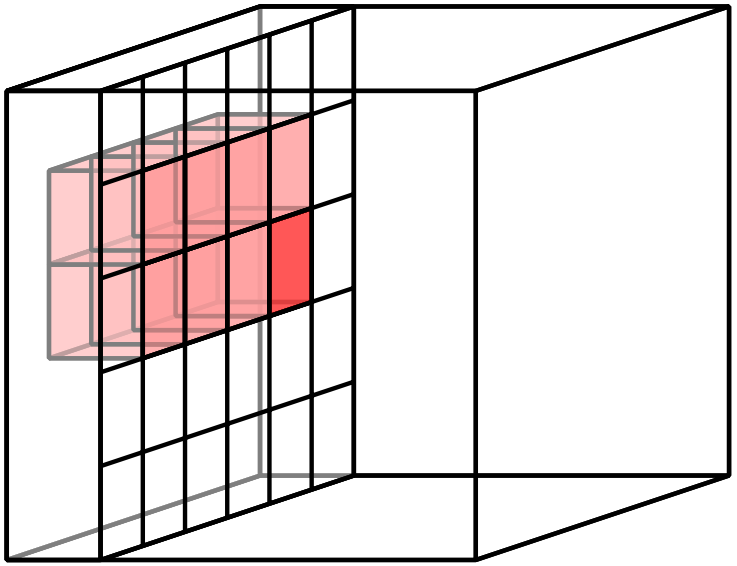


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

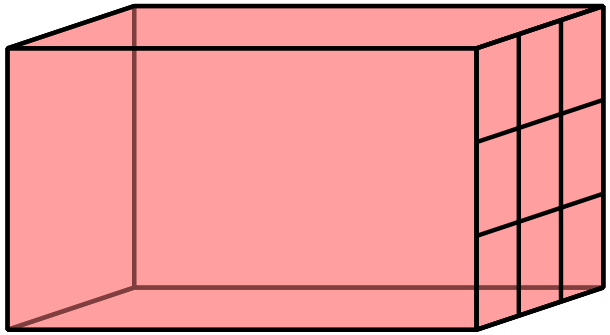


input $\mathbf{x}$



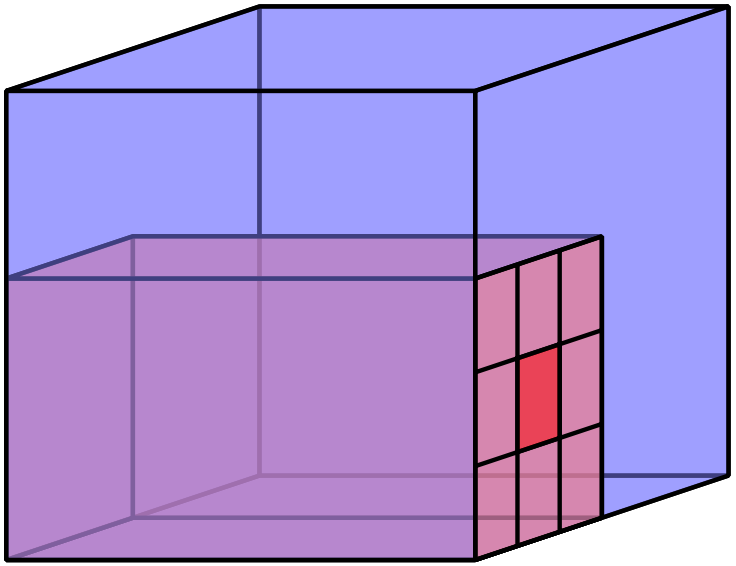
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

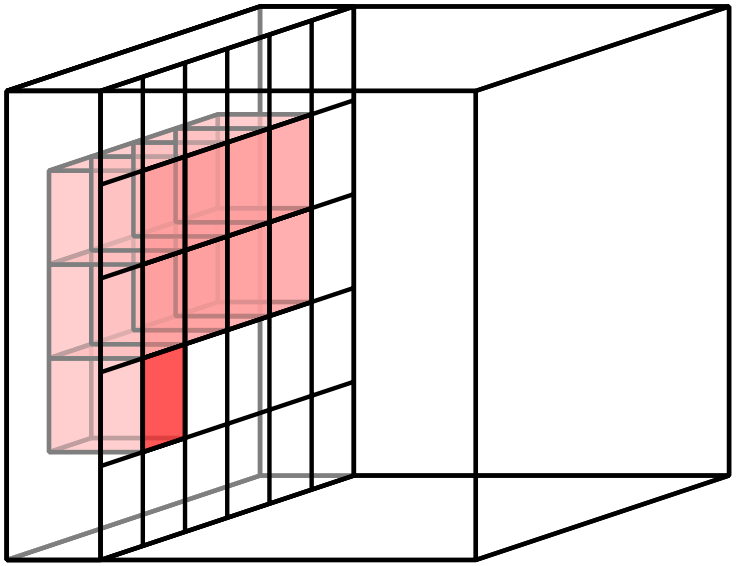


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

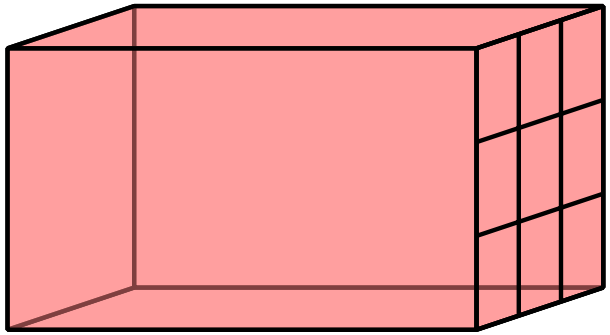


input $\mathbf{x}$



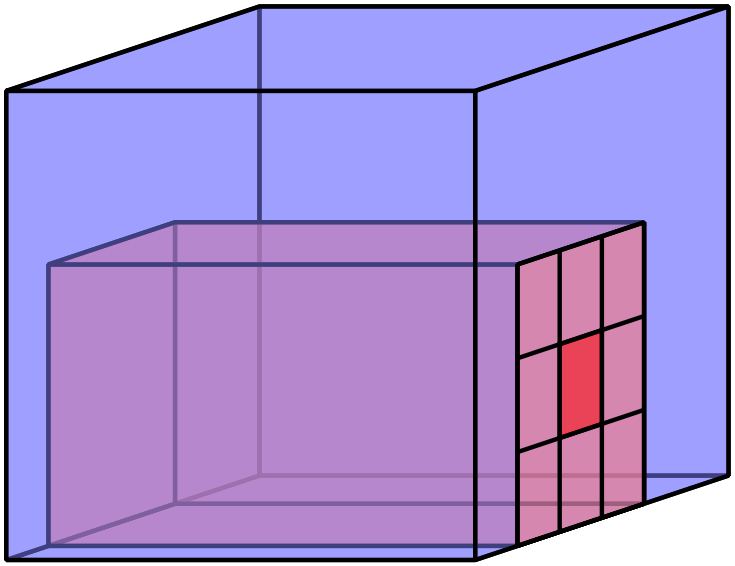
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

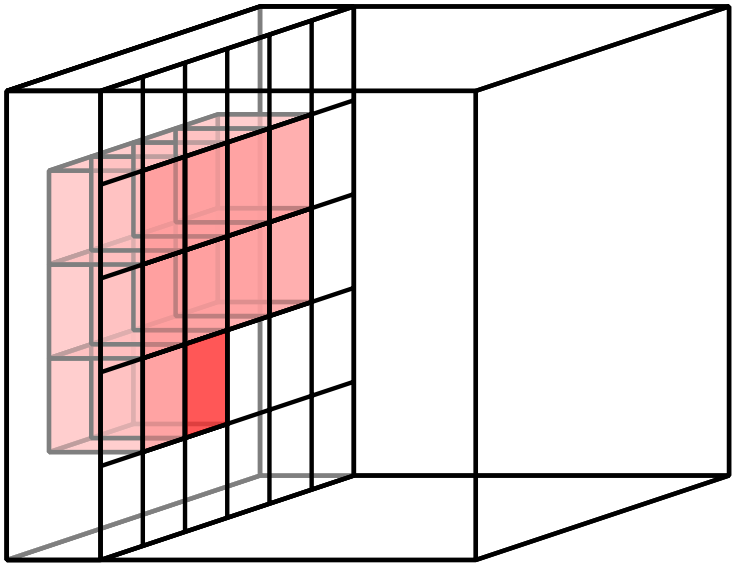


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

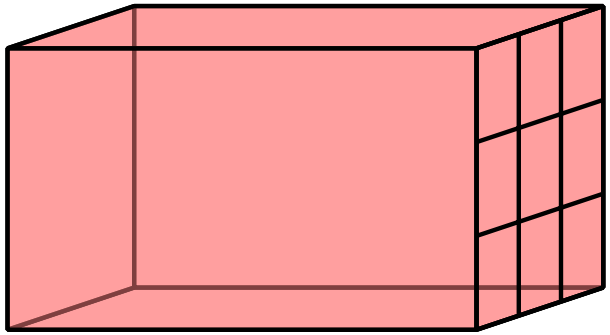


input $\mathbf{x}$



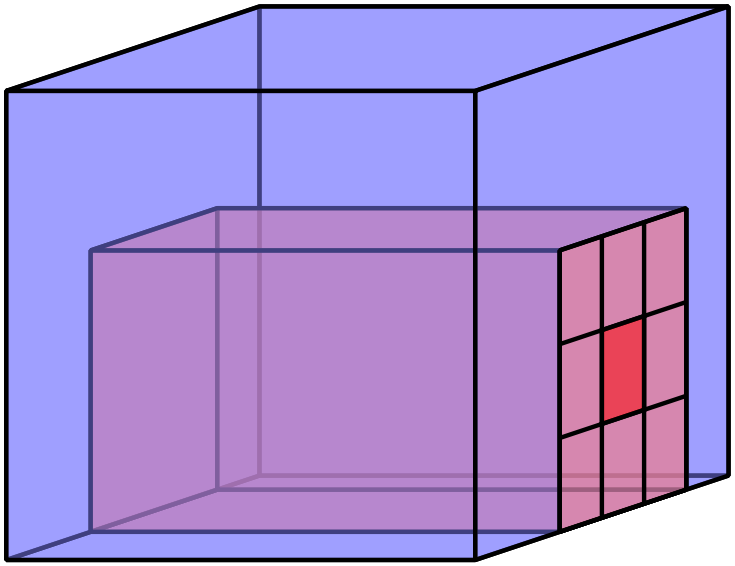
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

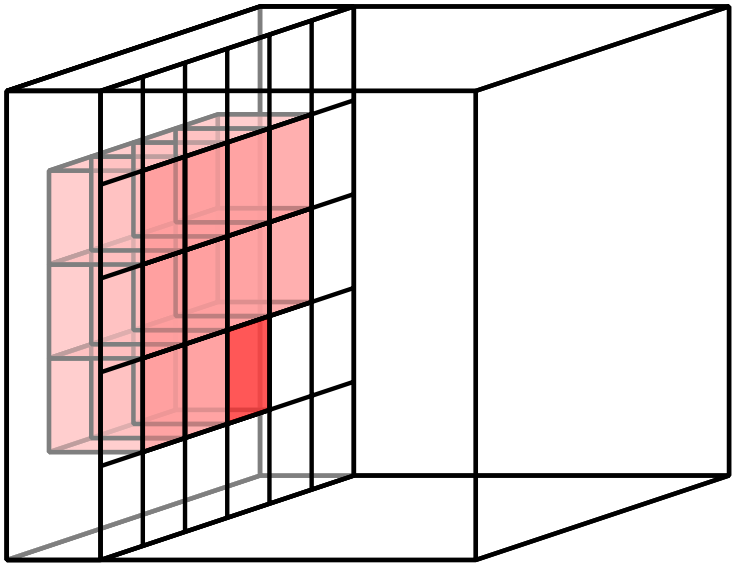


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

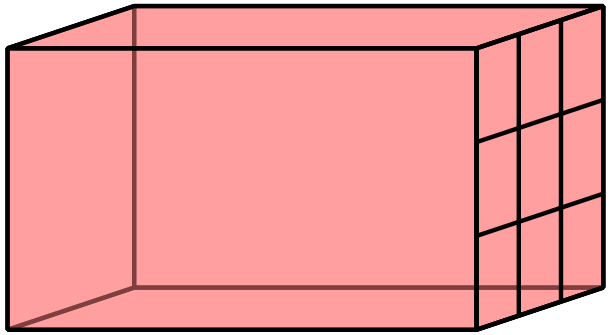


input $\mathbf{x}$



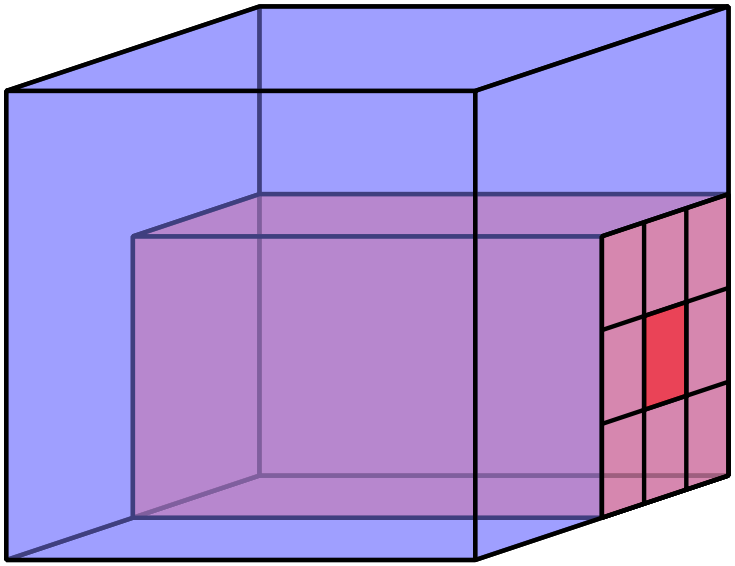
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

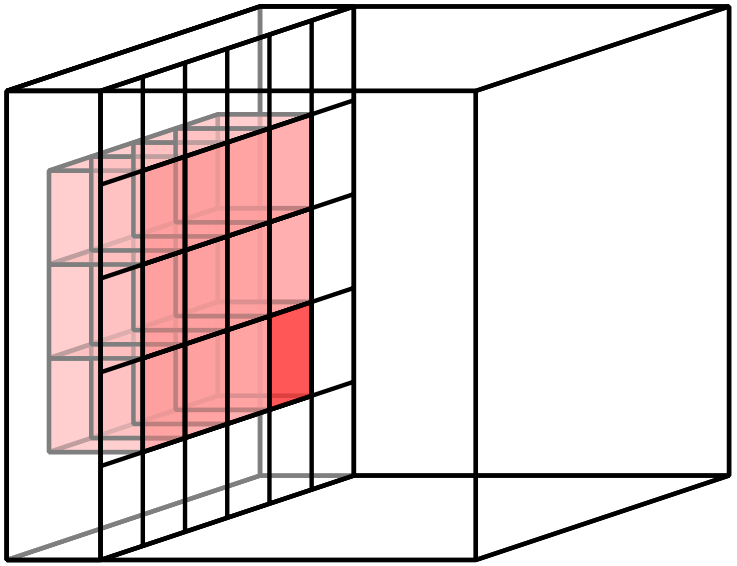


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

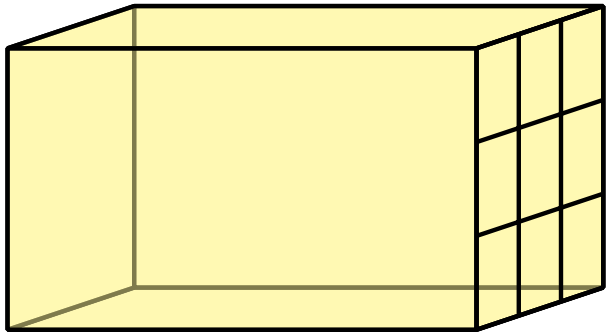


input $\mathbf{x}$



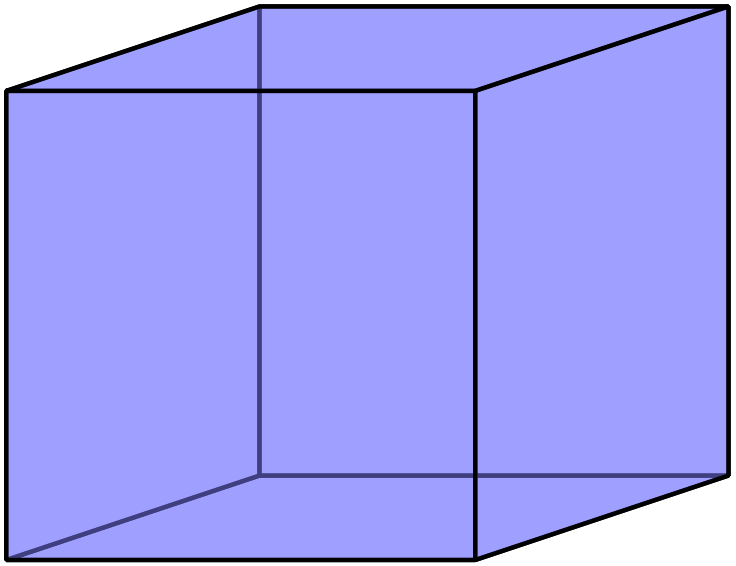
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

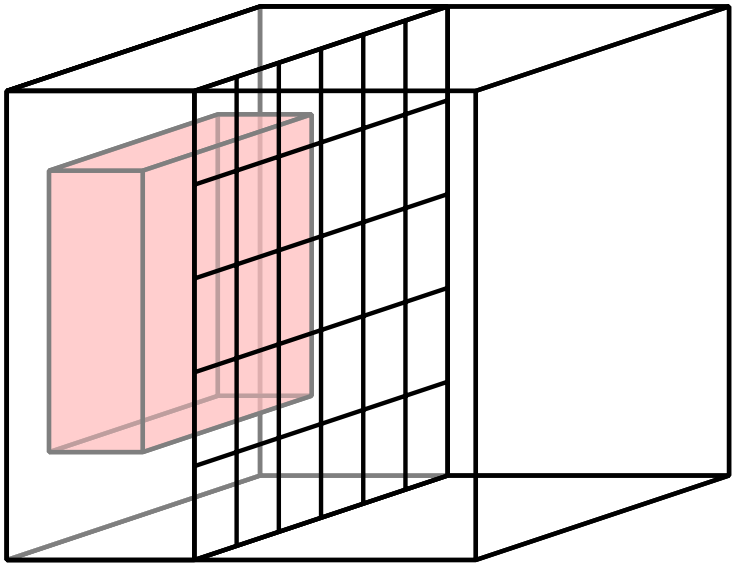


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

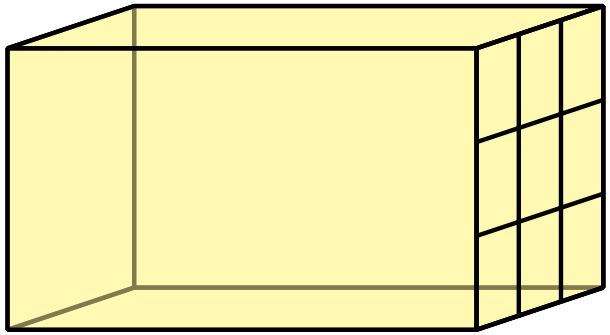


input $\mathbf{x}$



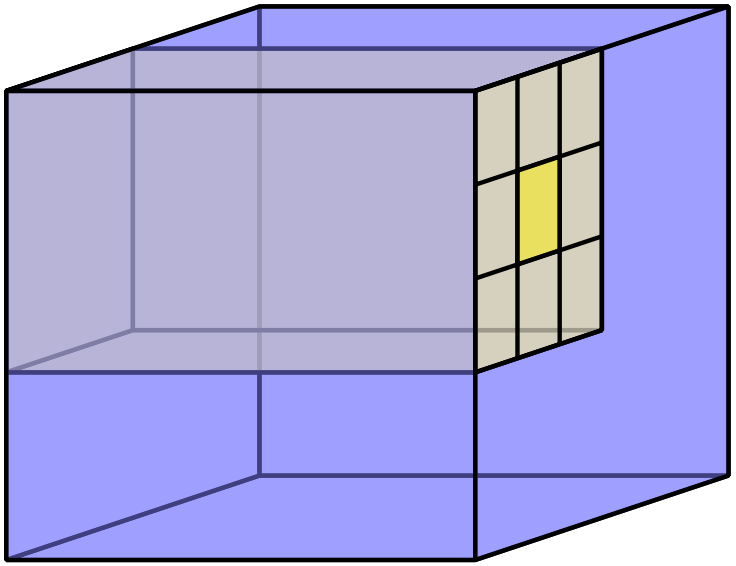
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

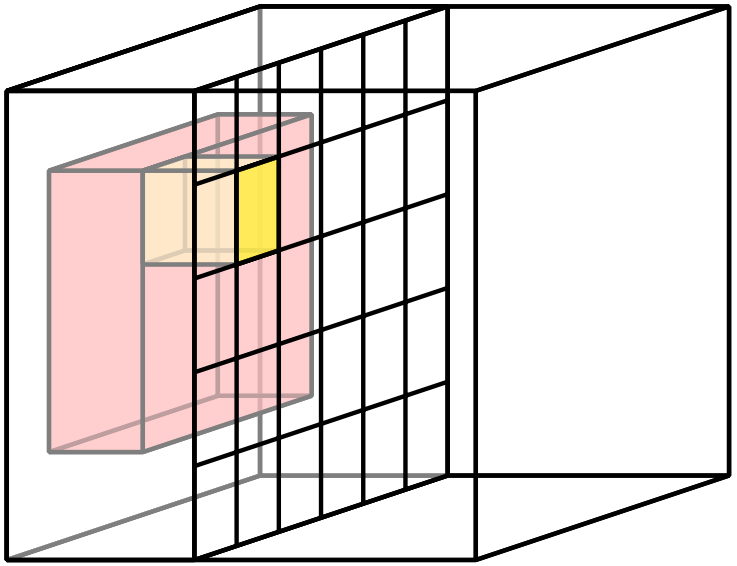


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

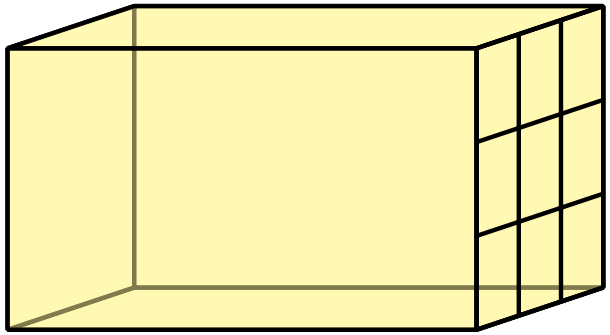


input $\mathbf{x}$



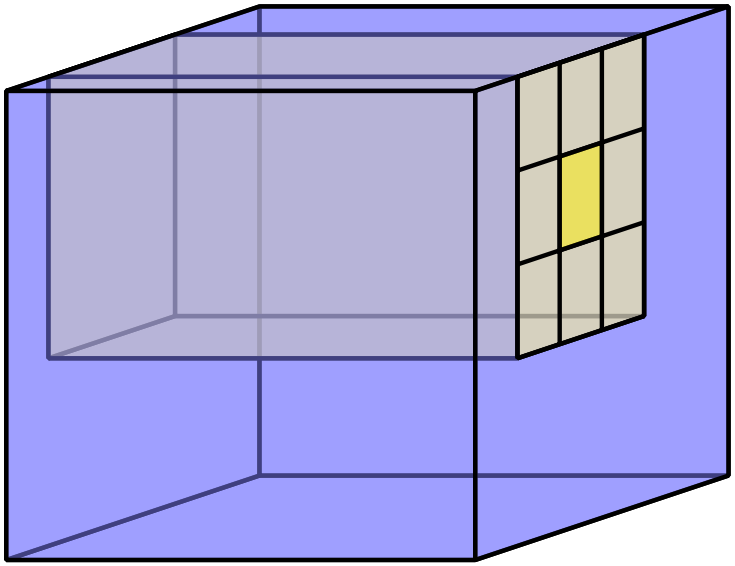
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

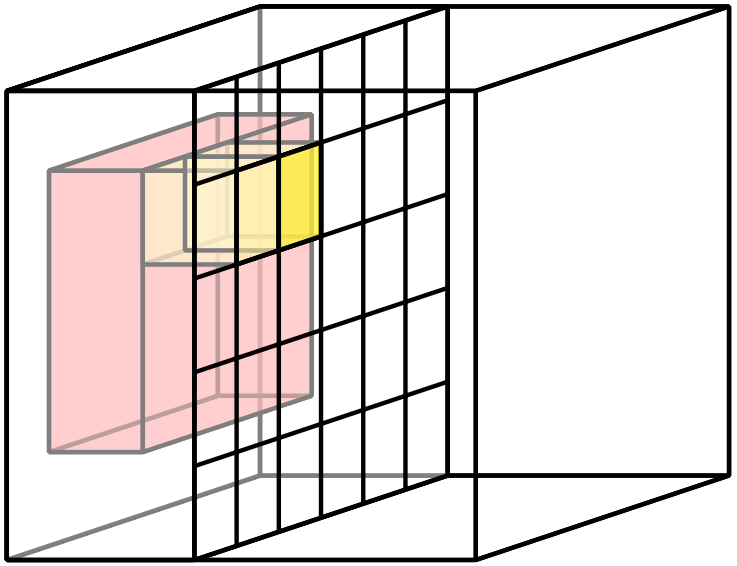


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

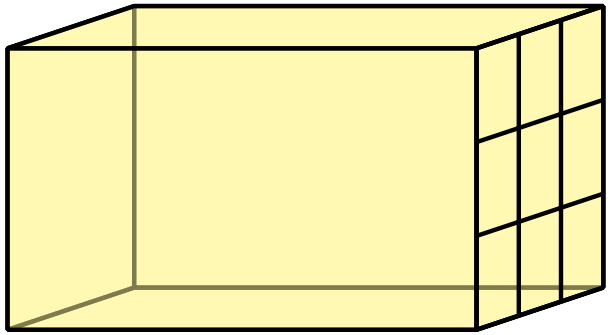


input $\mathbf{x}$



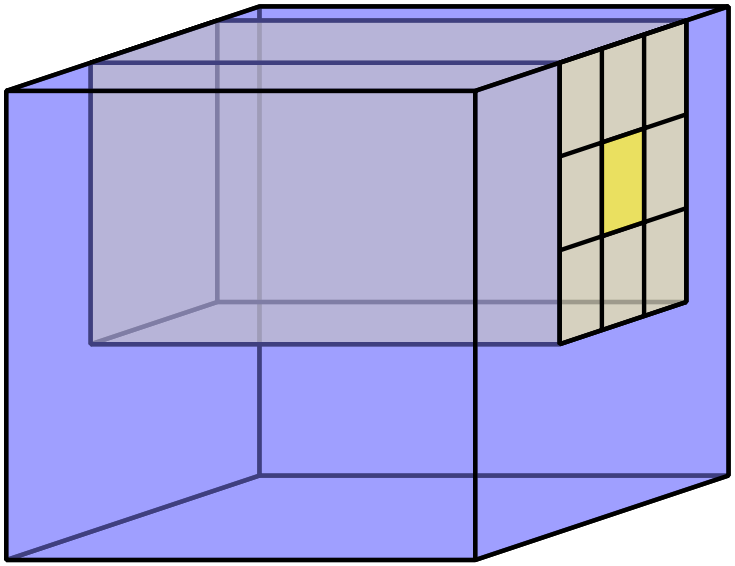
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

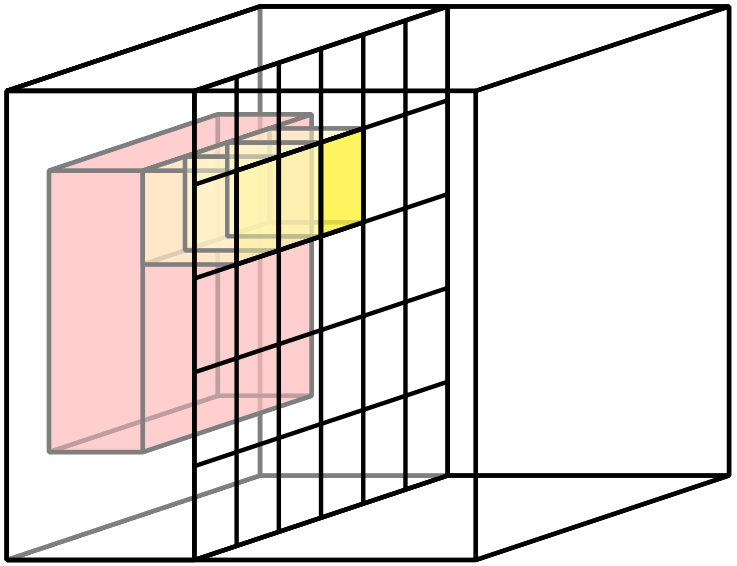


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

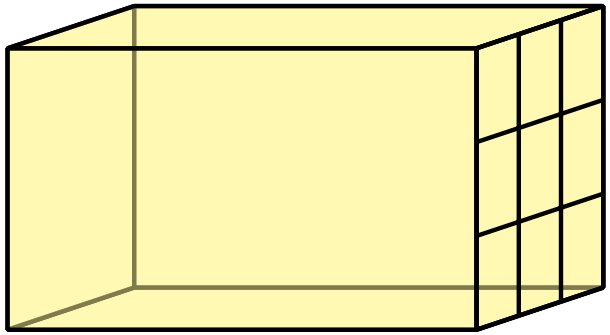


input $\mathbf{x}$



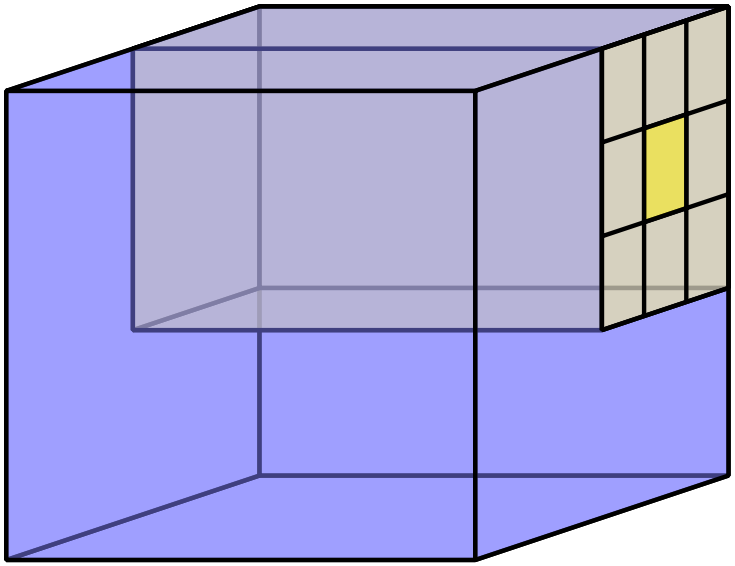
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

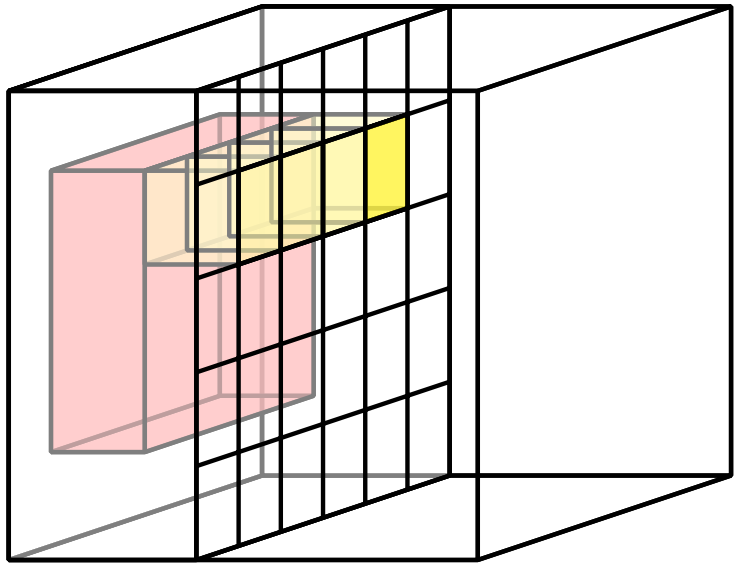


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

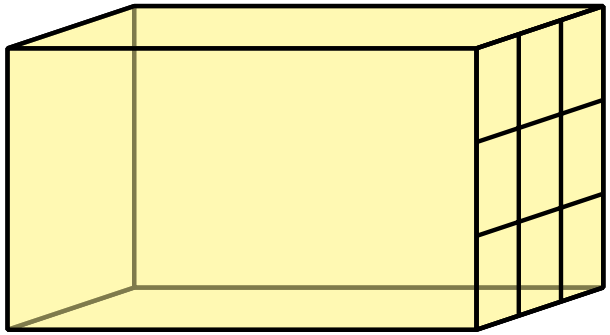


input $\mathbf{x}$



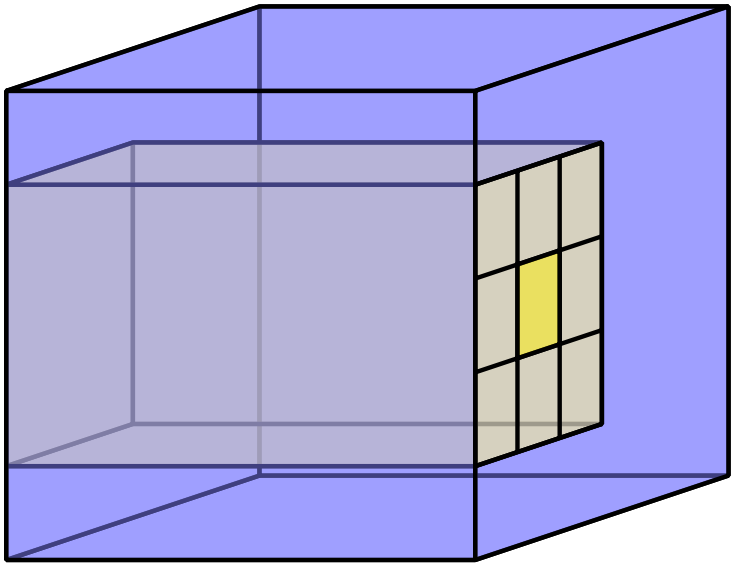
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

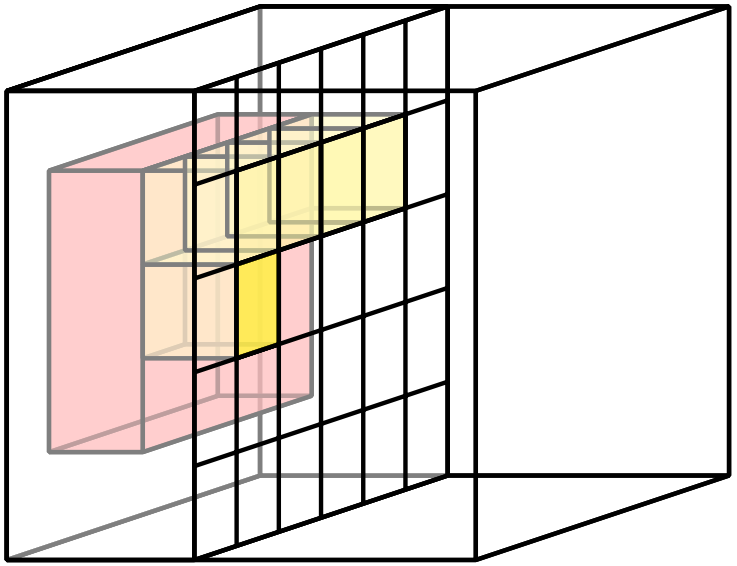


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

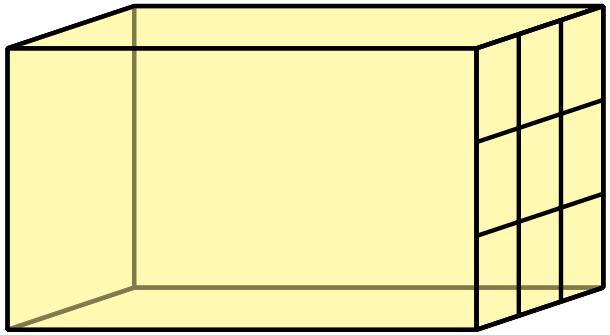


input $\mathbf{x}$



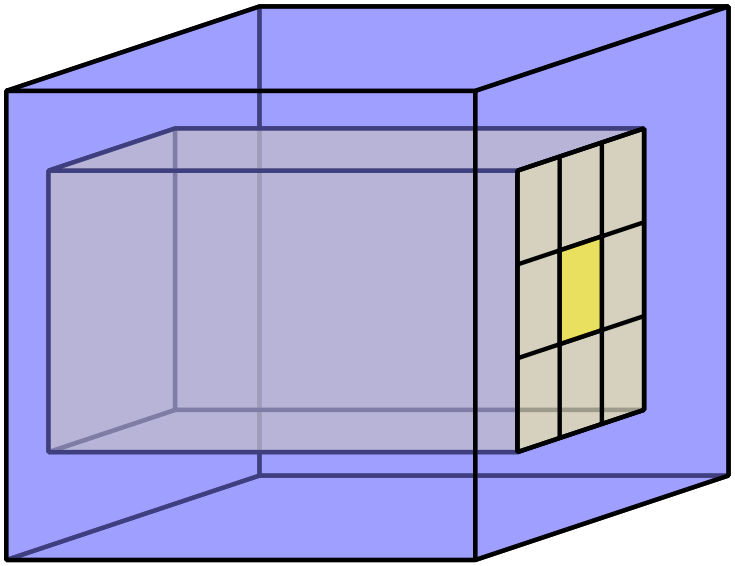
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

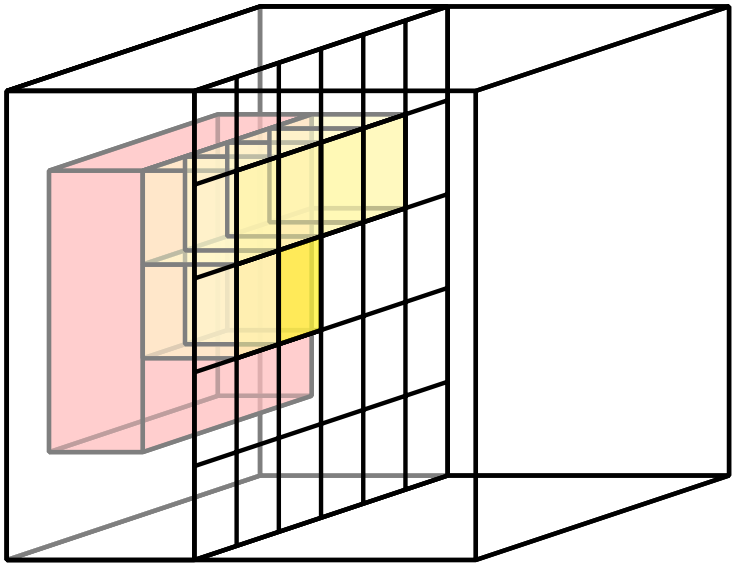


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

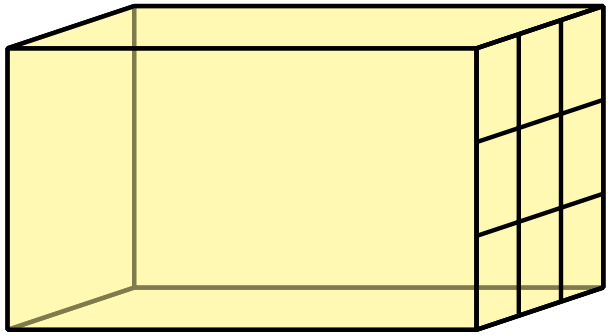


input $\mathbf{x}$



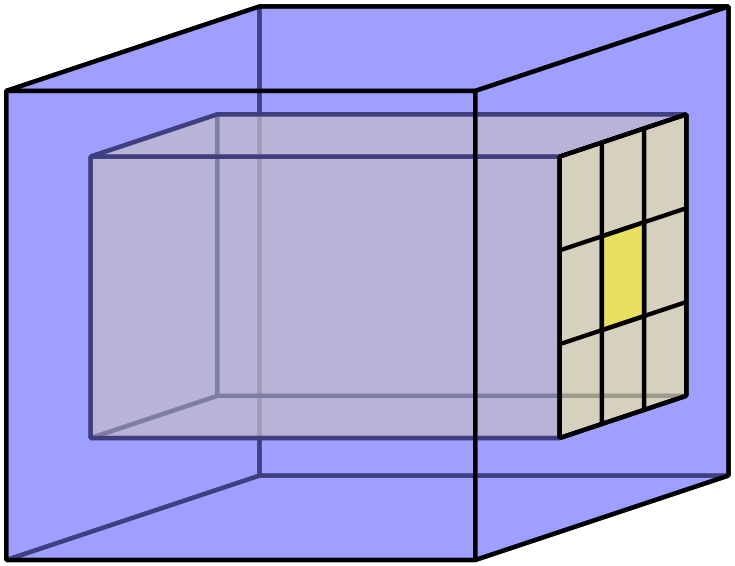
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

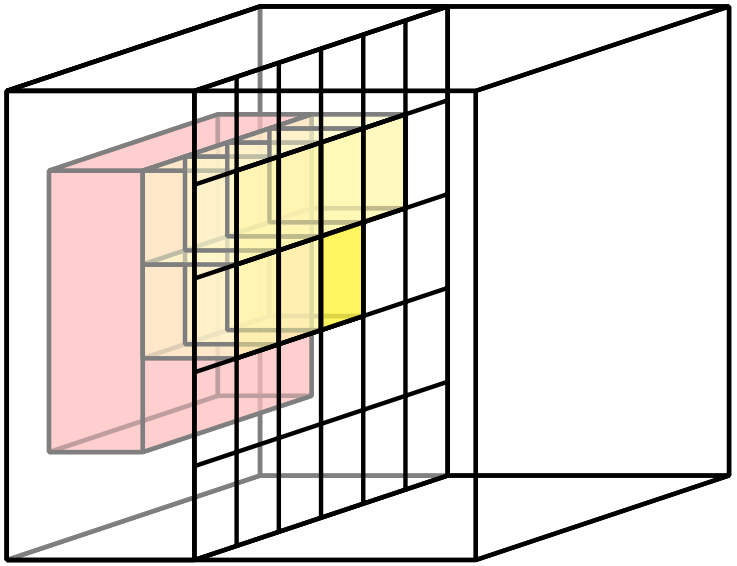


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

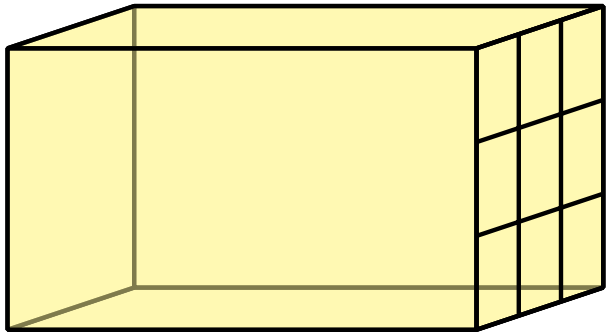


input $\mathbf{x}$



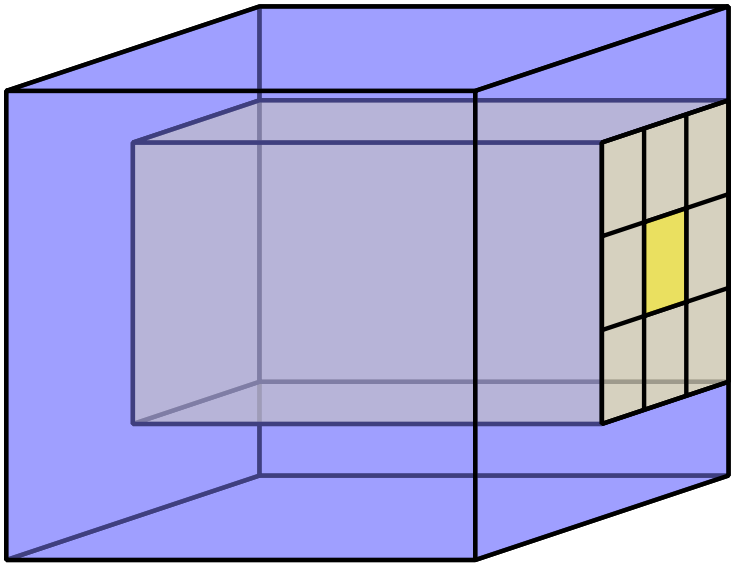
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

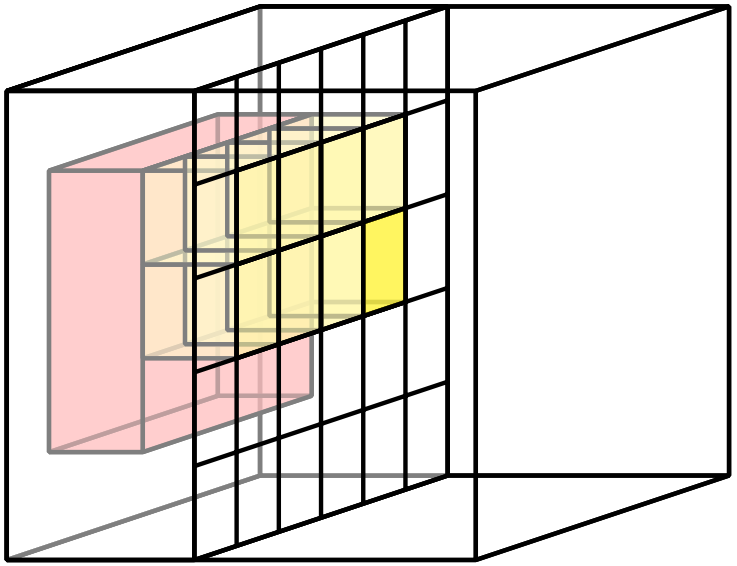


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

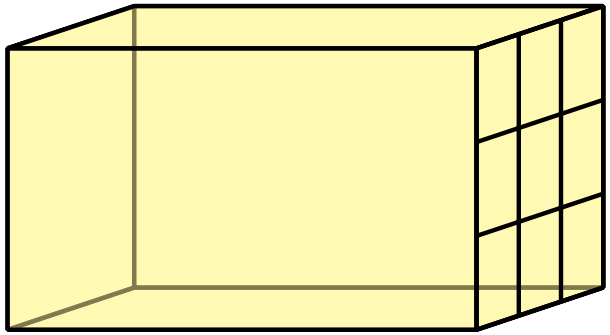


input $\mathbf{x}$



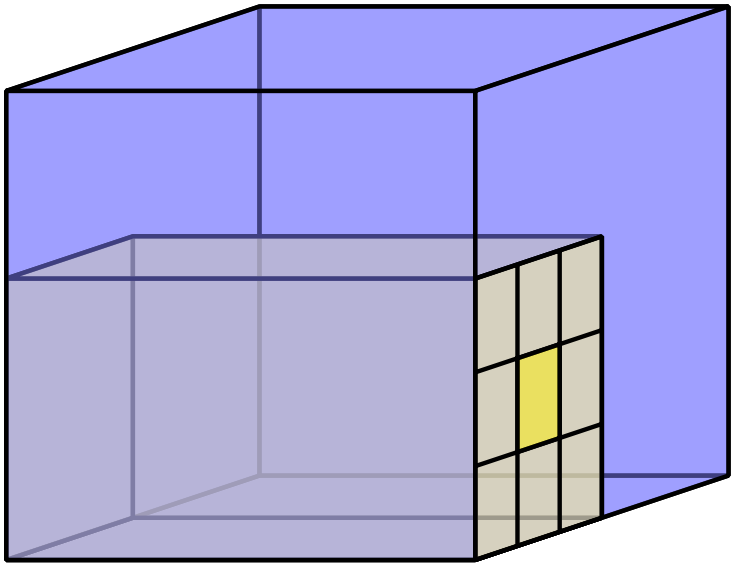
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

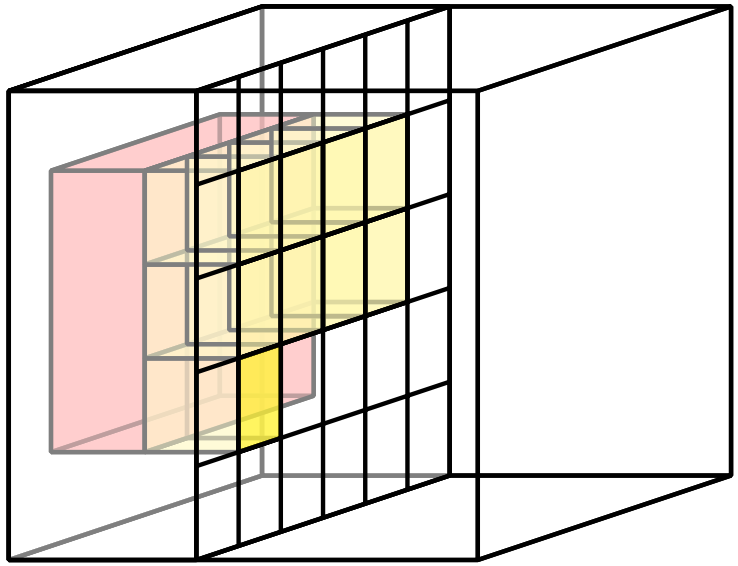


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

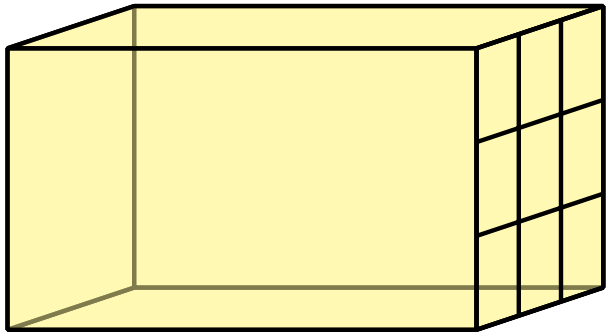


input $\mathbf{x}$



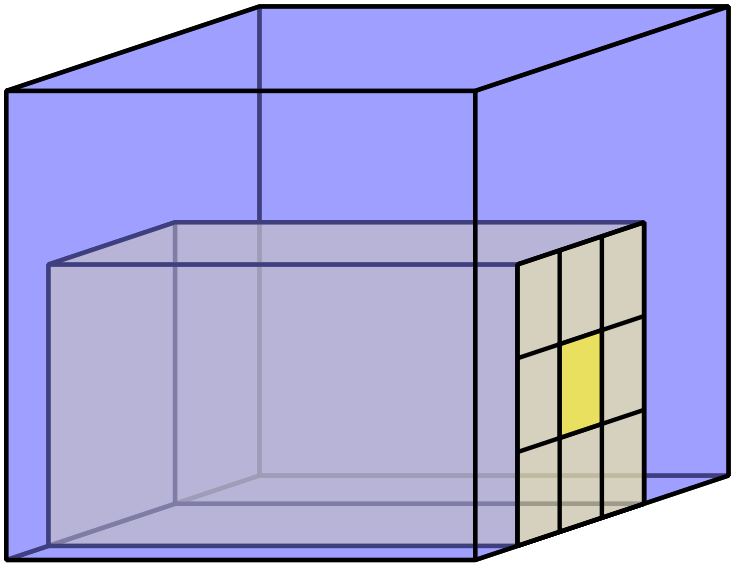
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

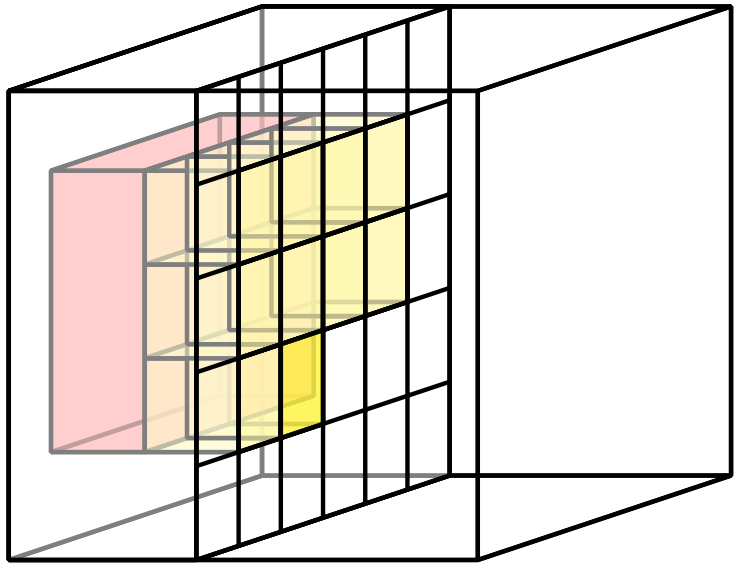


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

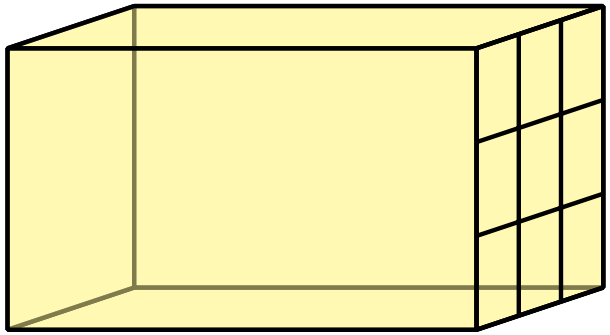


input $\mathbf{x}$



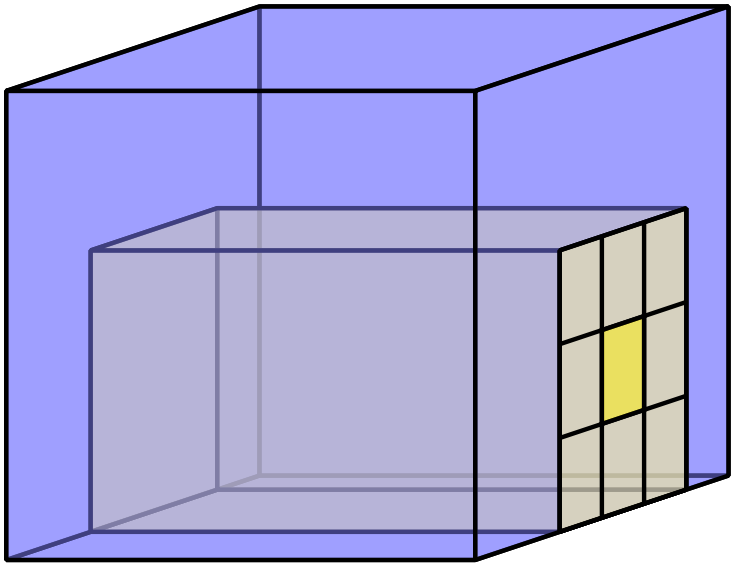
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

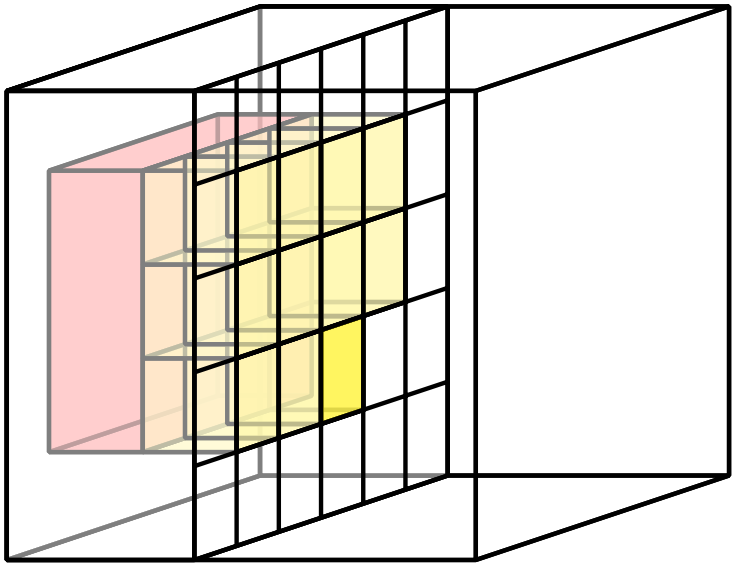


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

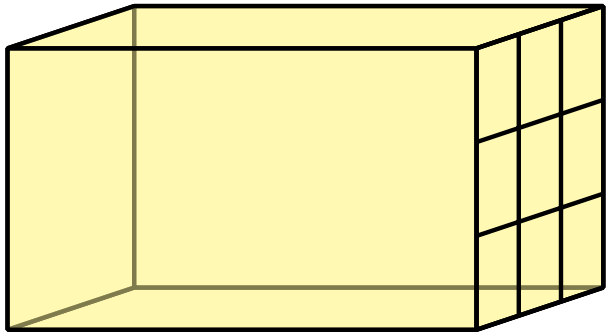


input $\mathbf{x}$



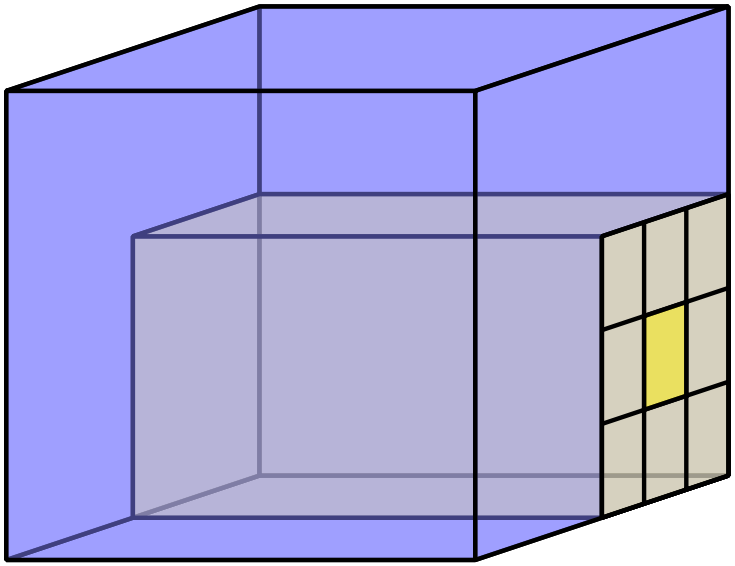
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

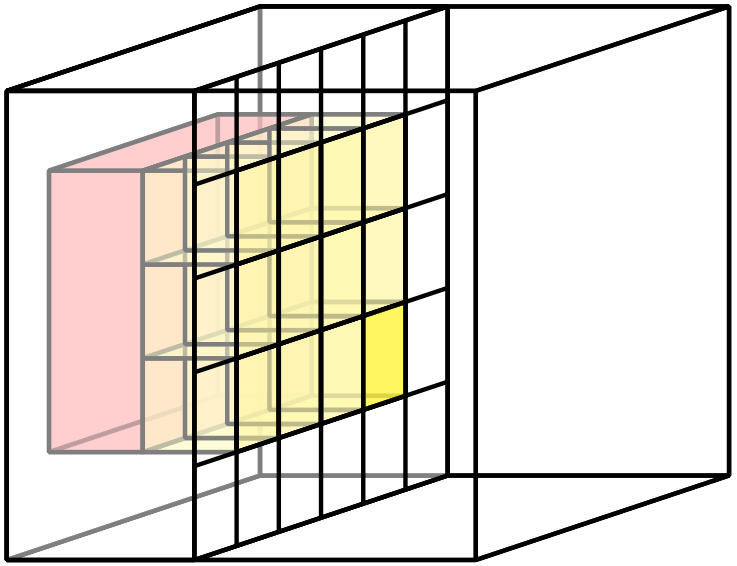


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

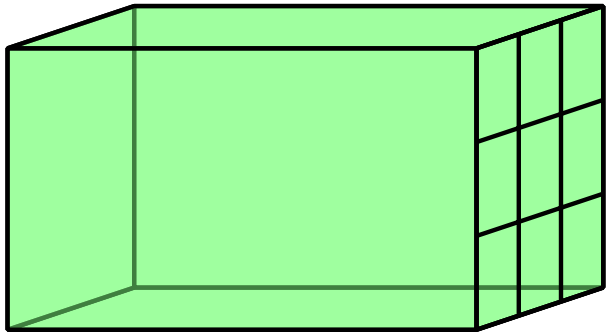


input $\mathbf{x}$



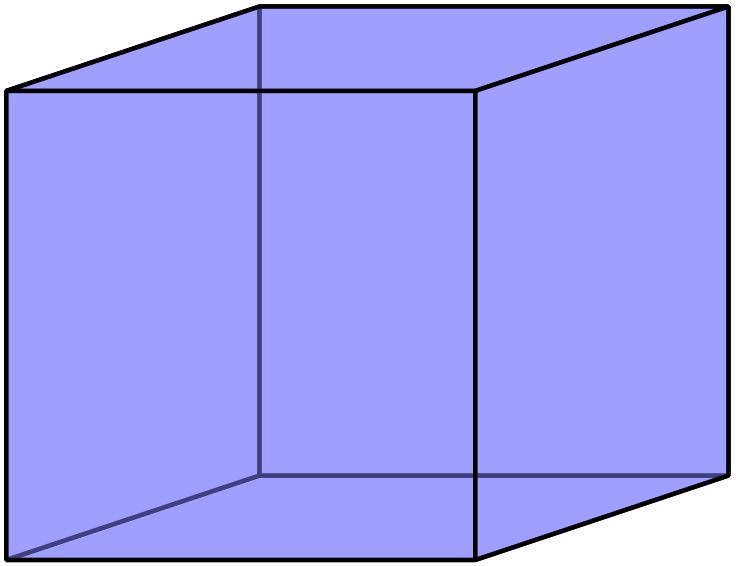
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

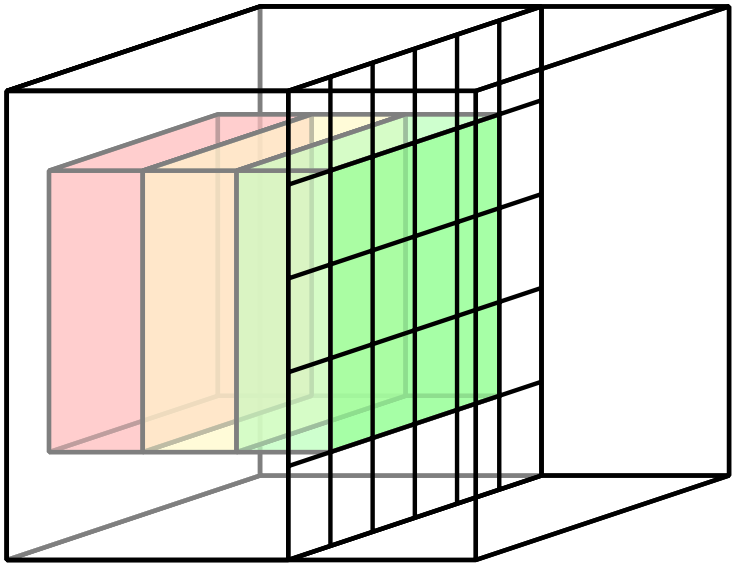


kernel $\mathbf{w}_3$

different kernel for
each output dimension



input $\mathbf{x}$

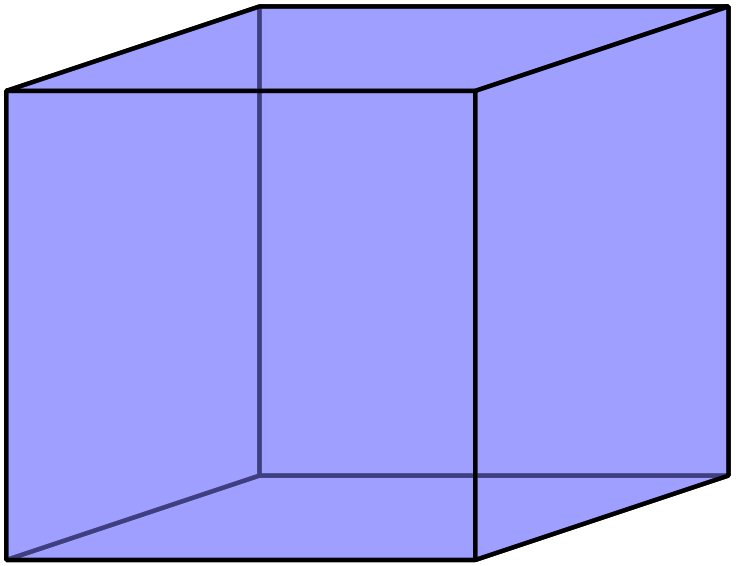


output $y_3 = h(\mathbf{w}_3^\top \star \mathbf{x} + b_3)$

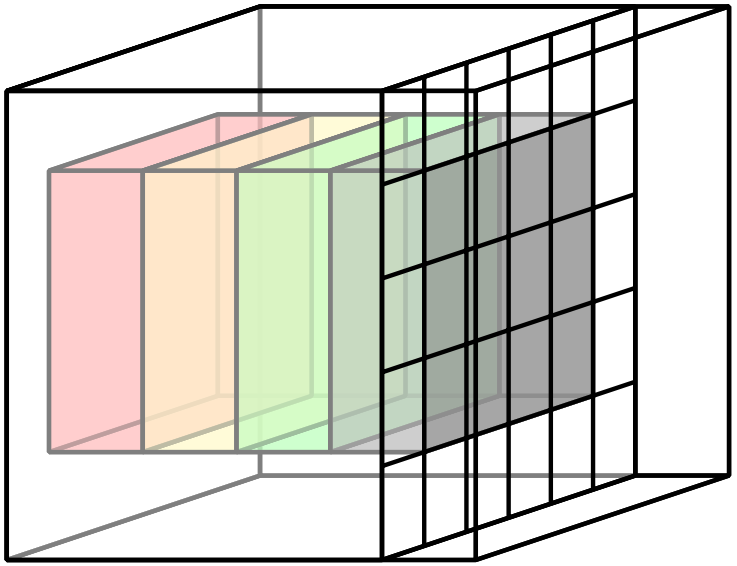
convolution on feature maps

kernel $\mathbf{w}_4$

different kernel for
each output dimension

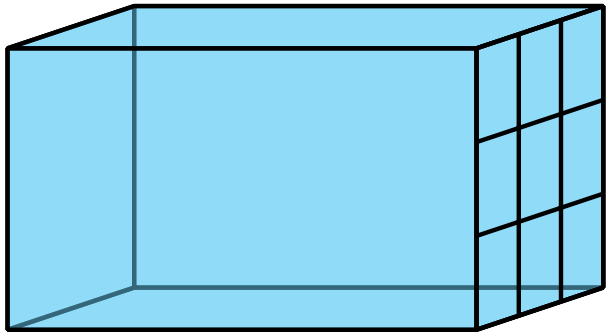


input $\mathbf{x}$



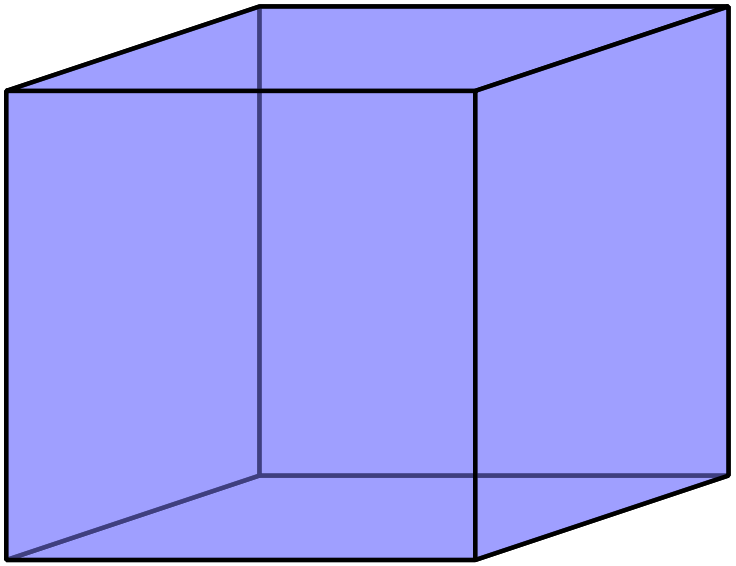
output $y_4 = h(\mathbf{w}_4^\top \star \mathbf{x} + b_4)$

convolution on feature maps

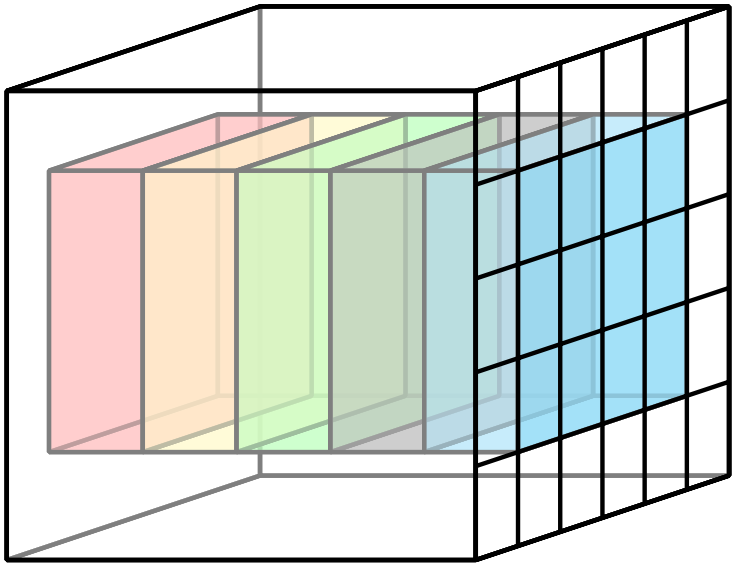


kernel $\mathbf{w}_5$

different kernel for
each output dimension

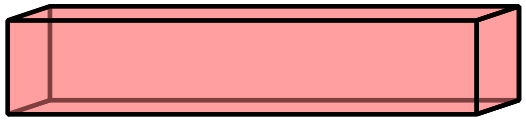


input $\mathbf{x}$



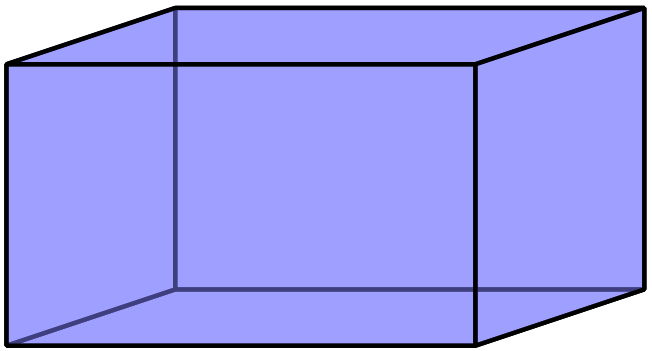
output $y_5 = h(\mathbf{w}_5^\top \star \mathbf{x} + b_5)$

1×1 convolution

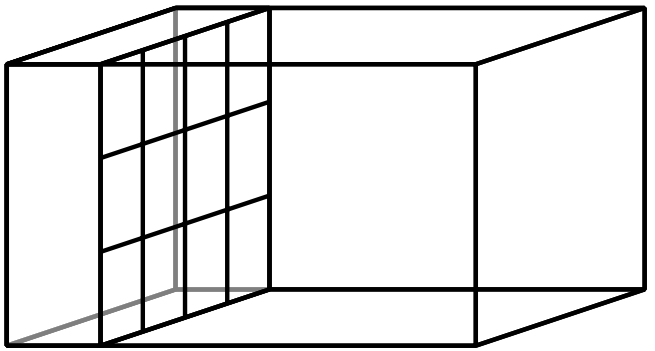


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

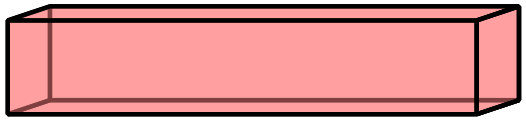


input $\mathbf{x}$



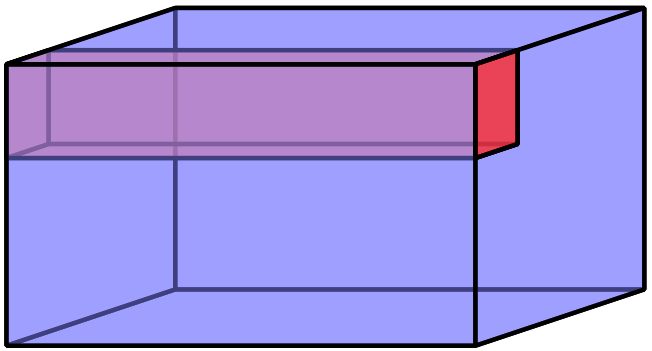
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

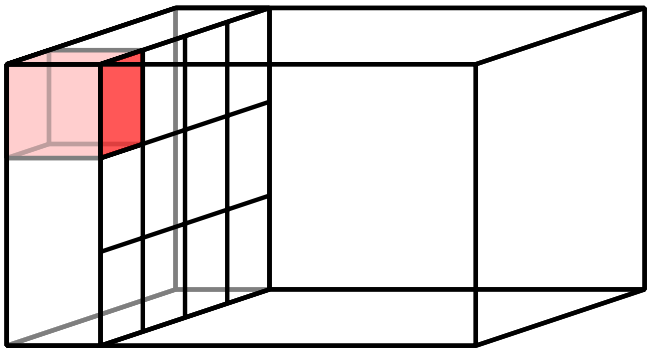


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

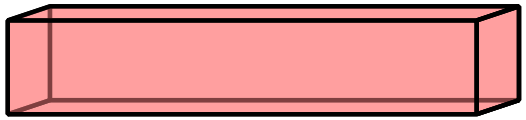


input $\mathbf{x}$



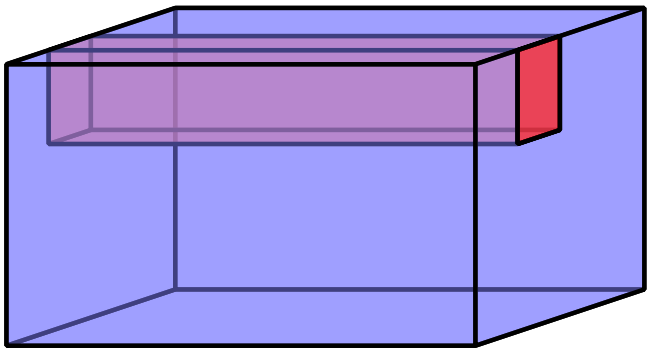
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

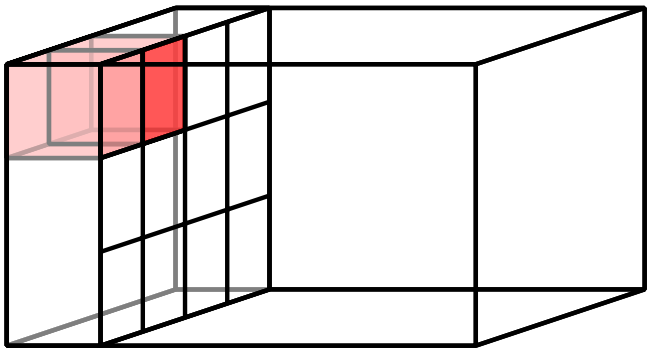


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

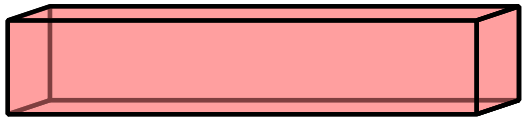


input $\mathbf{x}$



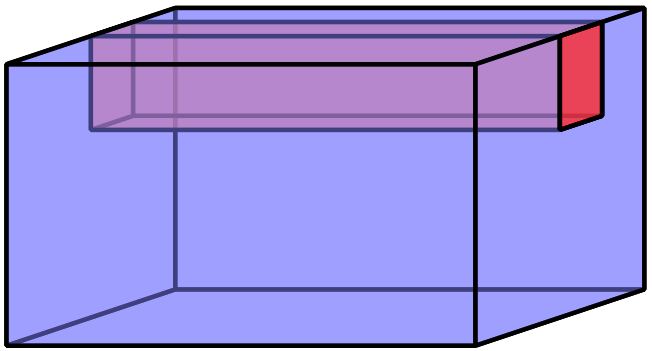
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

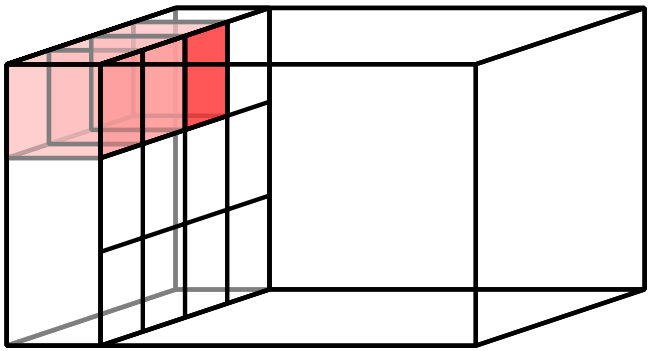


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

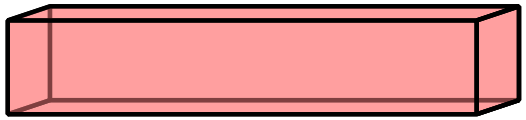


input $\mathbf{x}$



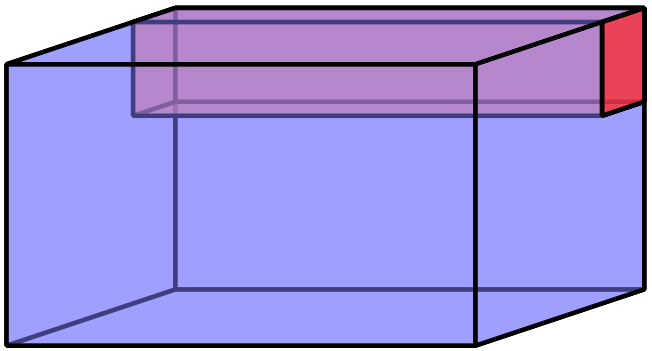
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

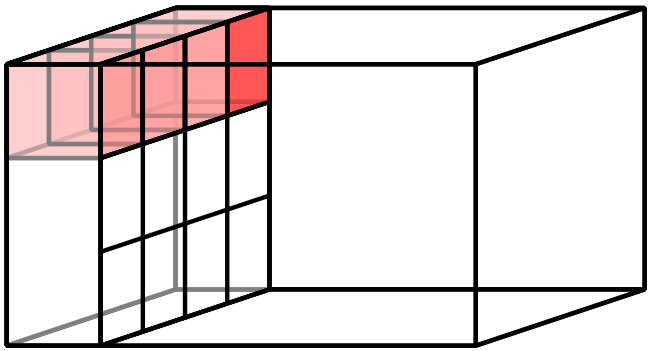


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

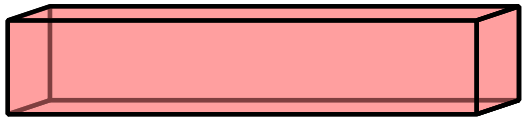


input $\mathbf{x}$



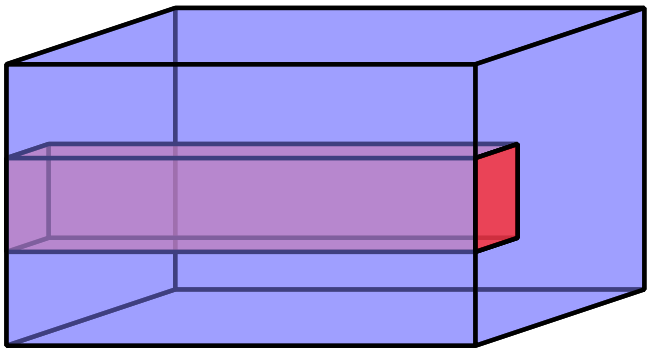
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

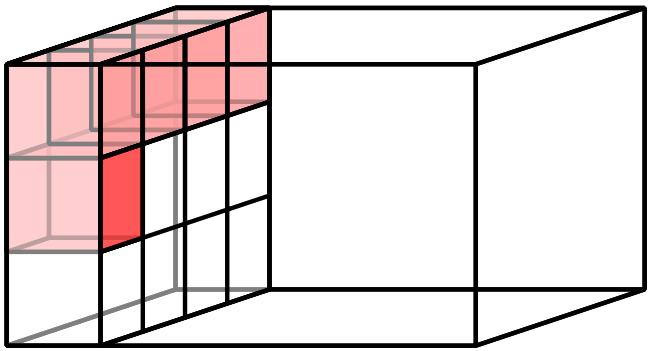


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

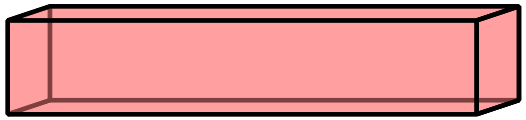


input $\mathbf{x}$



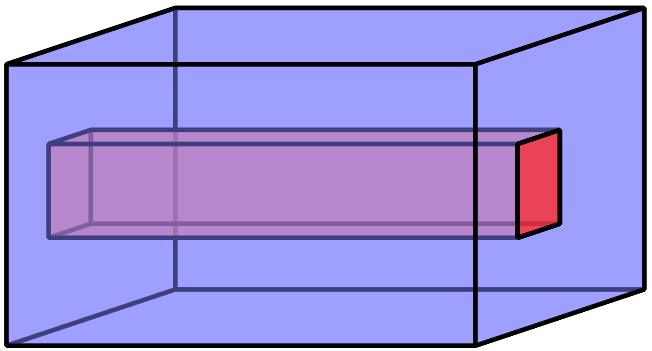
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

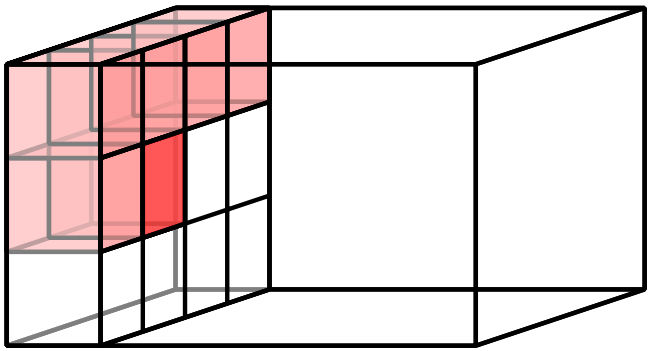


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

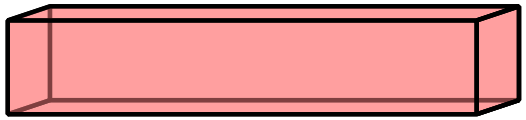


input $\mathbf{x}$



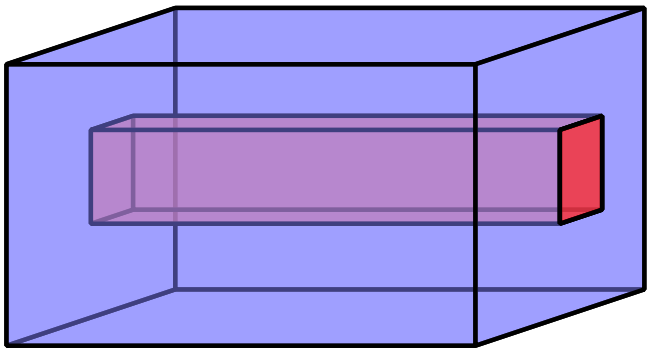
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

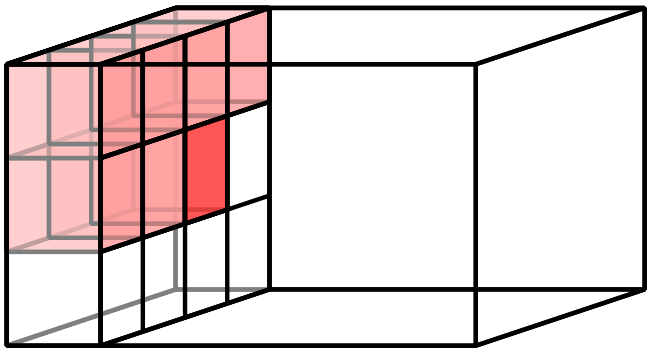


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

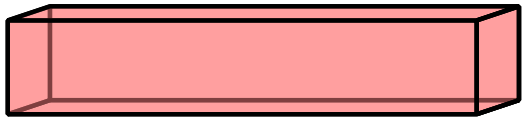


input $\mathbf{x}$



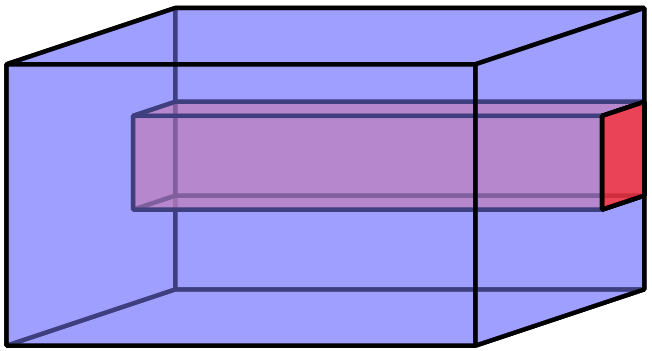
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

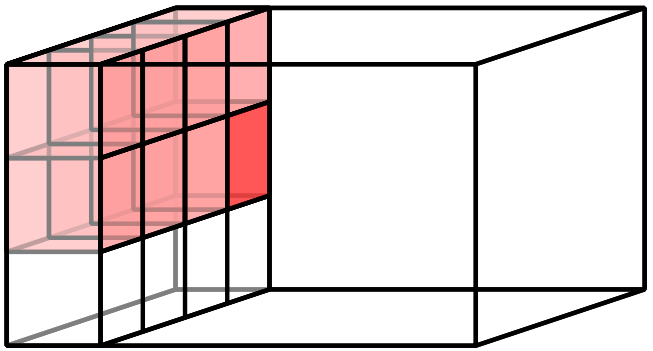


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

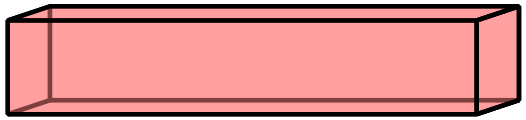


input $\mathbf{x}$



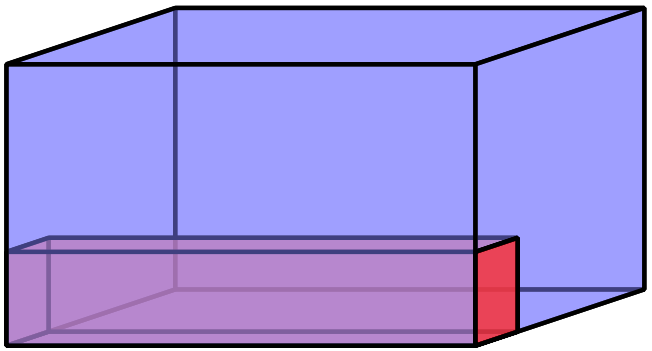
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

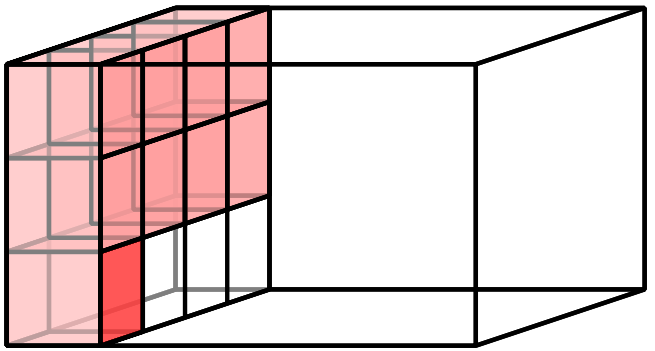


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

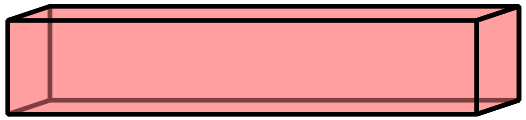


input $\mathbf{x}$



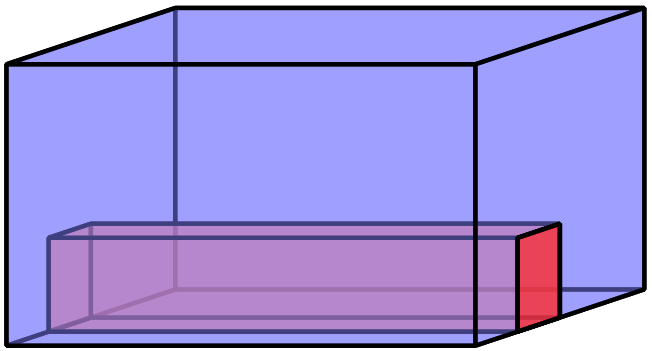
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

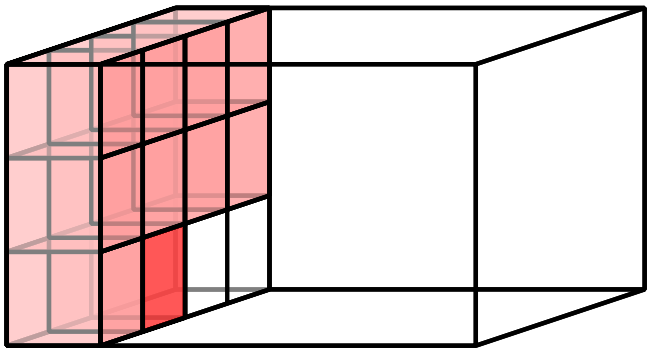


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

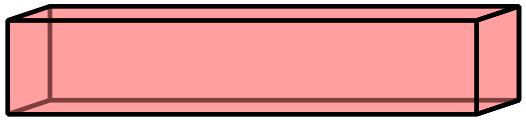


input $\mathbf{x}$



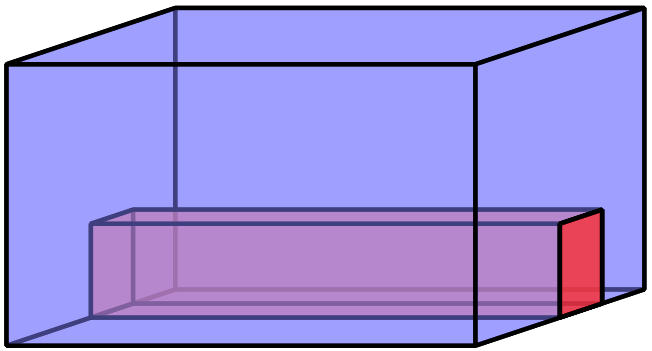
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

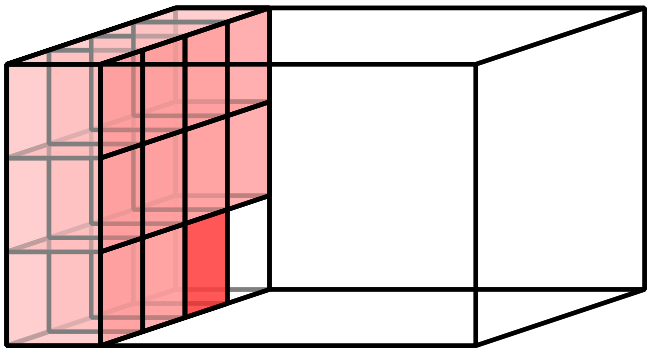


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

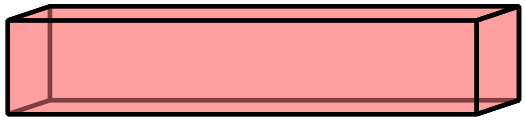


input $\mathbf{x}$



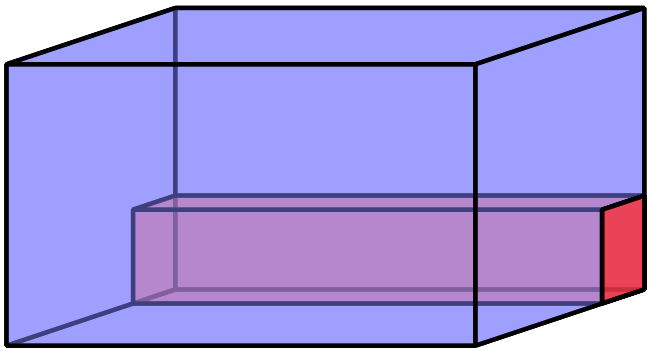
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

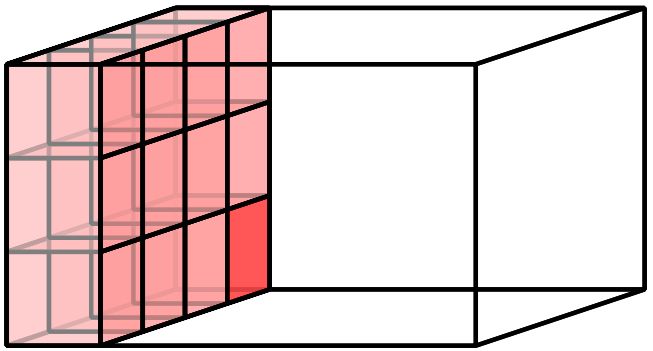


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

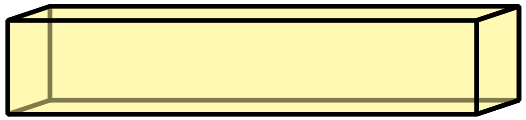


input $\mathbf{x}$



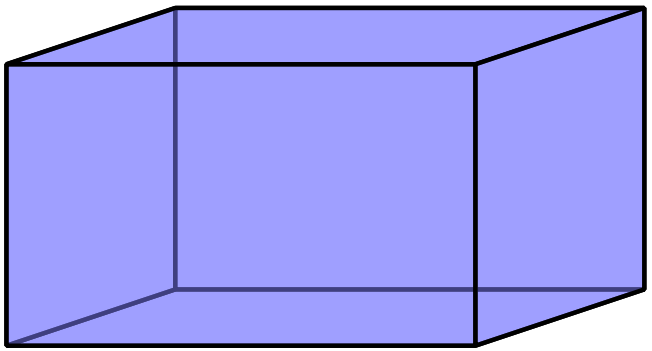
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

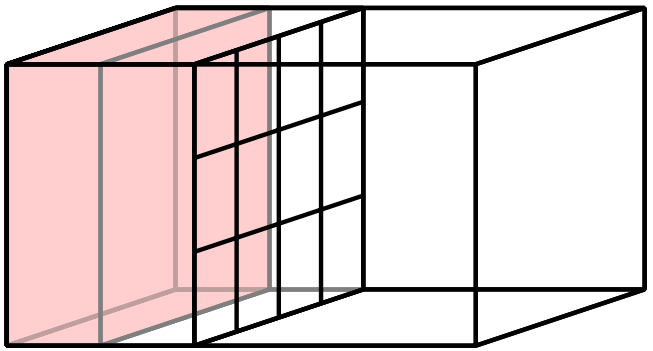


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

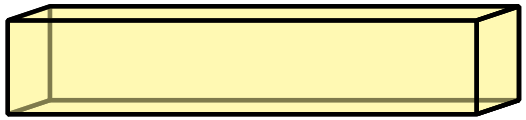


input $\mathbf{x}$



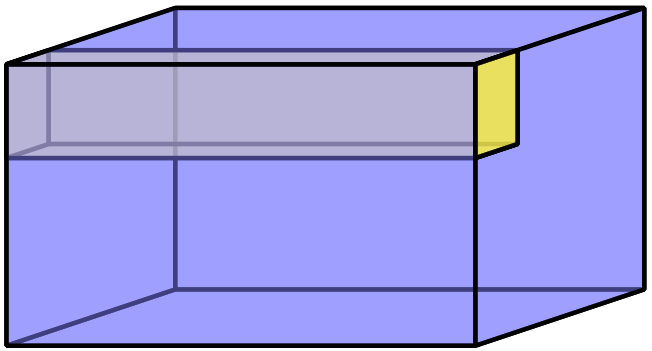
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

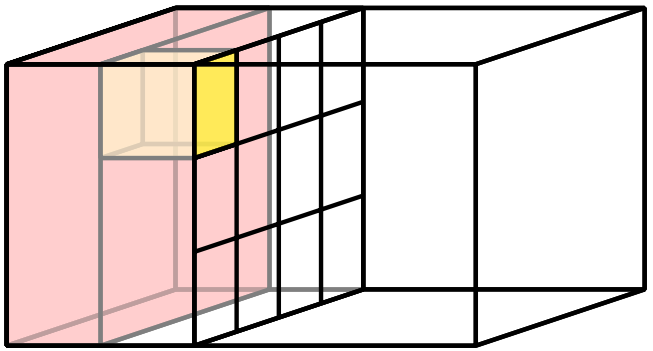


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

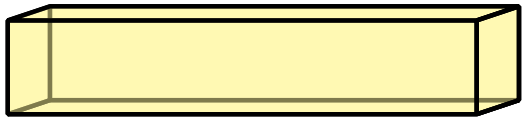


input $\mathbf{x}$



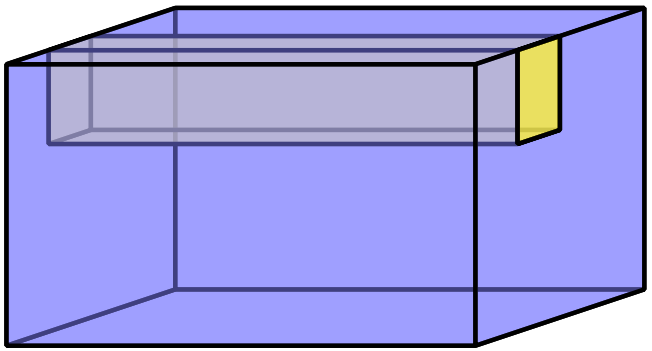
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

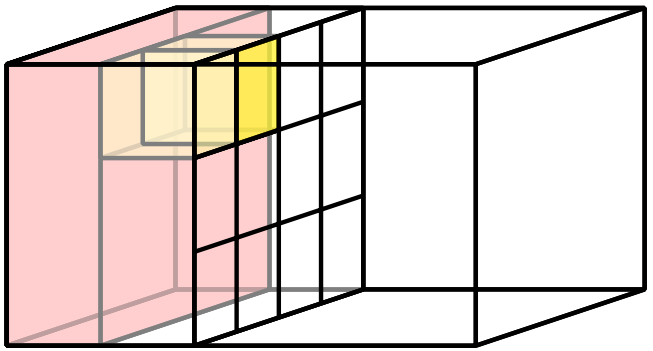


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

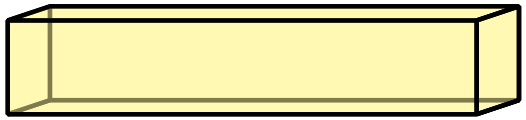


input $\mathbf{x}$



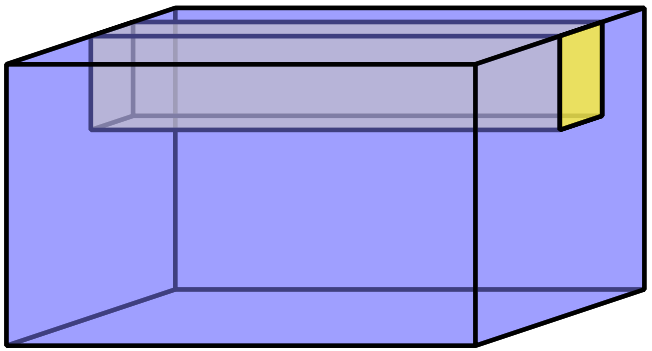
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

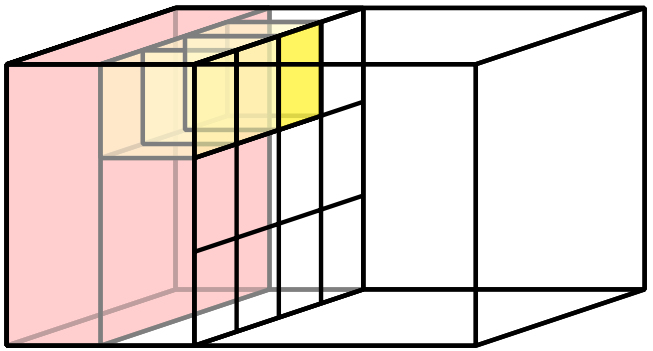


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

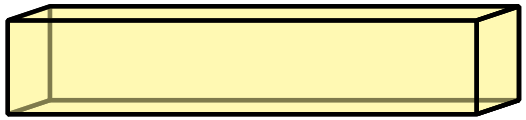


input $\mathbf{x}$



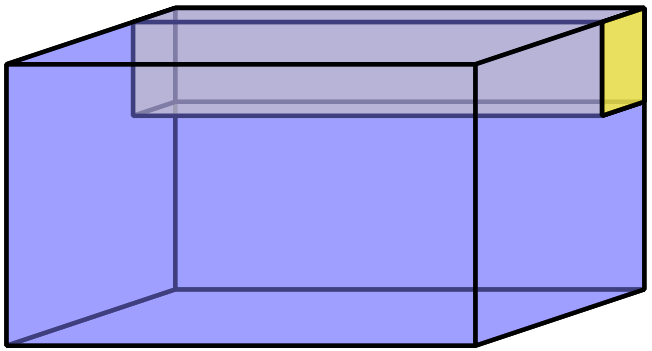
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

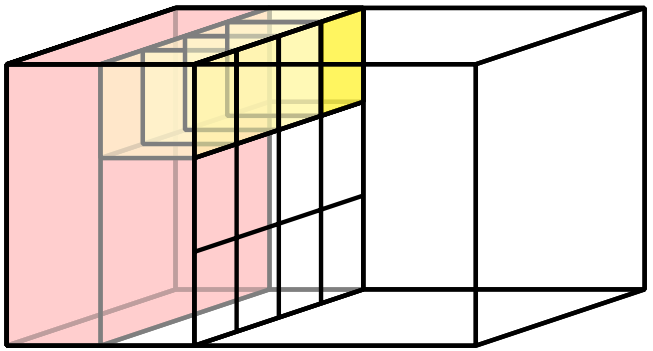


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

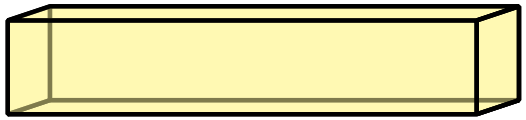


input $\mathbf{x}$



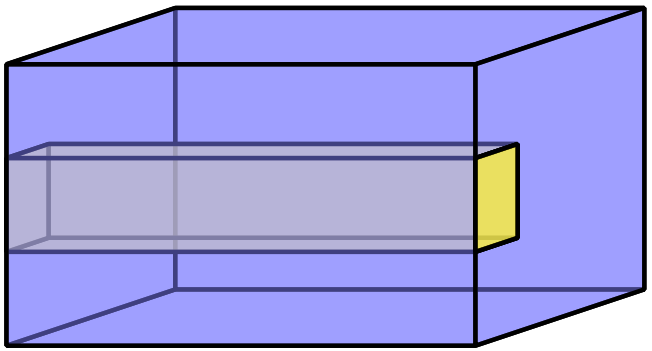
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 **convolution**

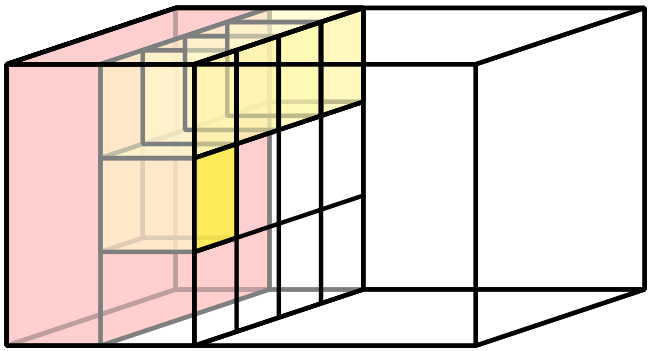


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

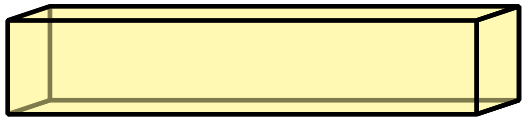


input $\mathbf{x}$



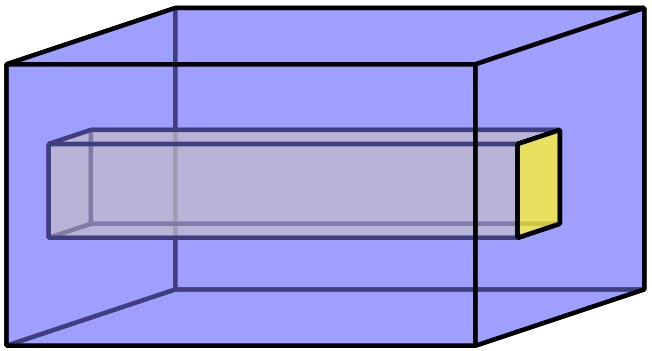
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

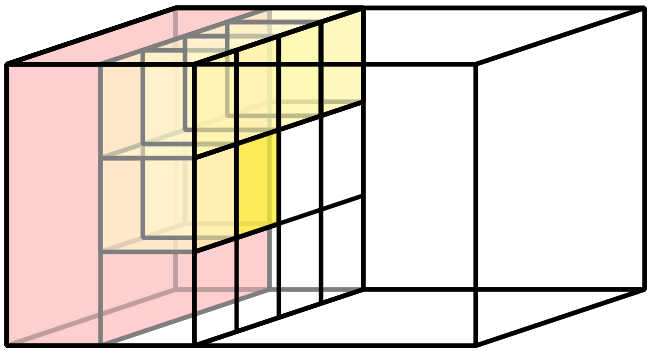


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

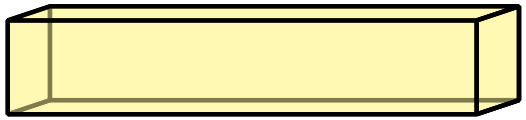


input $\mathbf{x}$



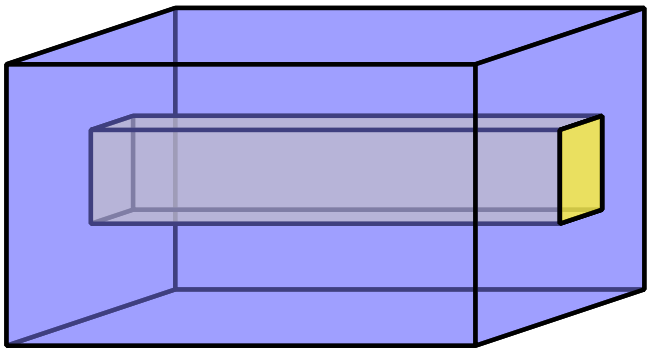
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

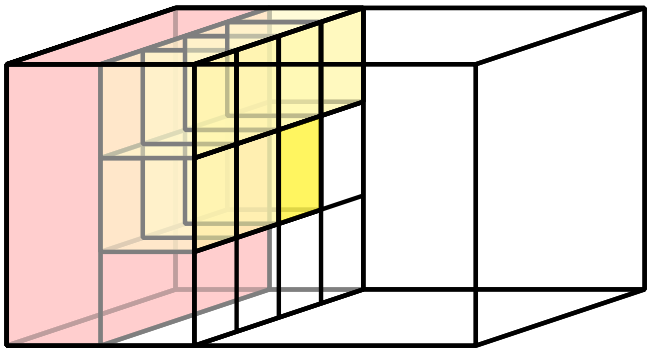


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

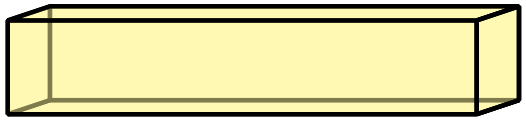


input $\mathbf{x}$



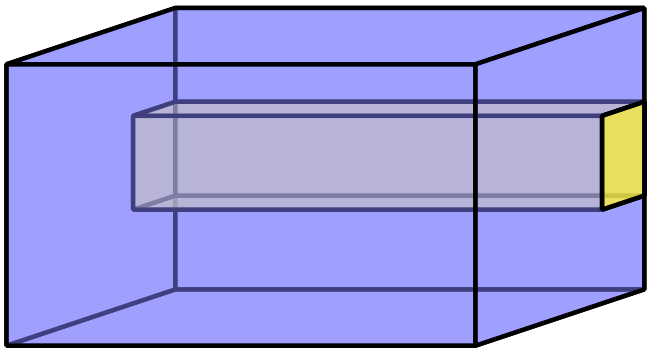
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

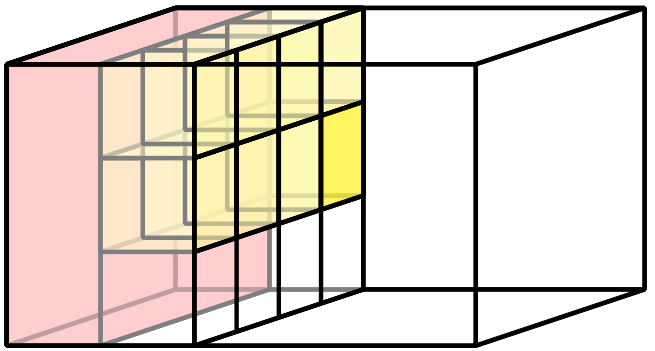


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

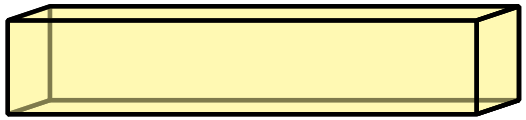


input $\mathbf{x}$



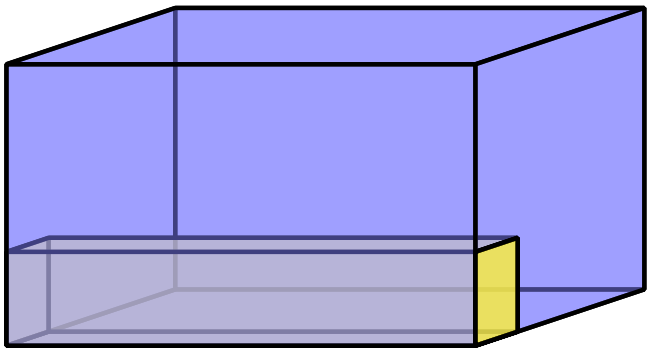
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

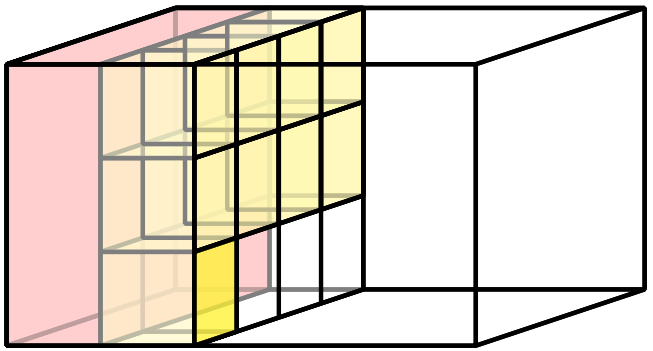


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

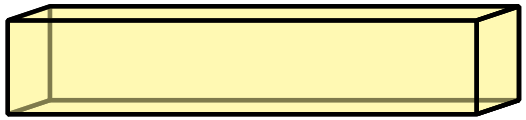


input $\mathbf{x}$



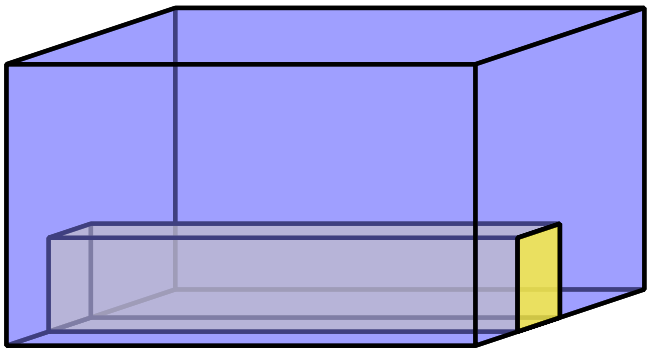
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

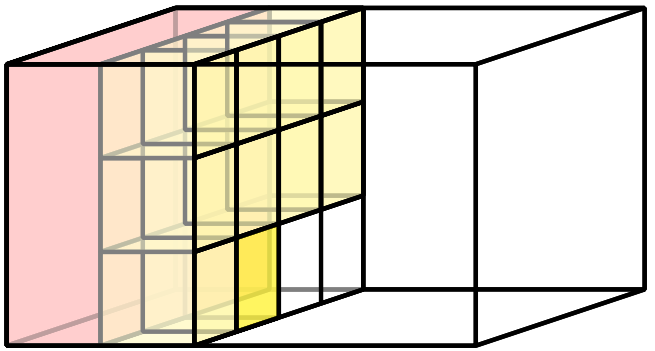


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

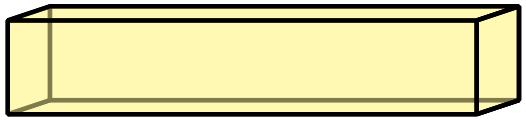


input $\mathbf{x}$



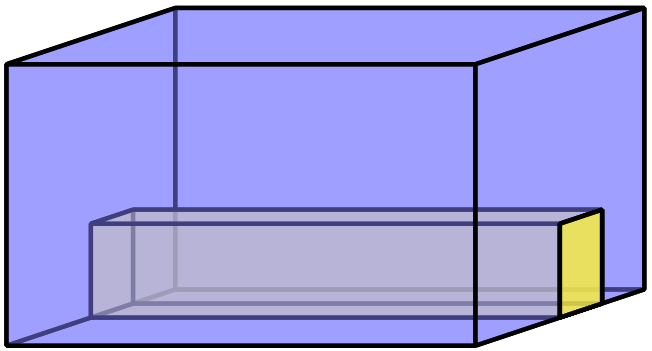
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

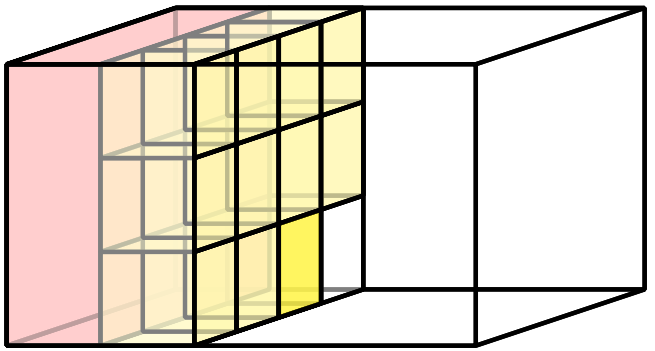


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

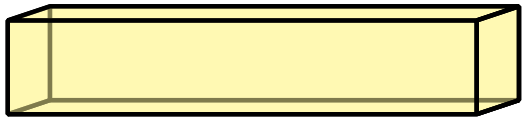


input $\mathbf{x}$



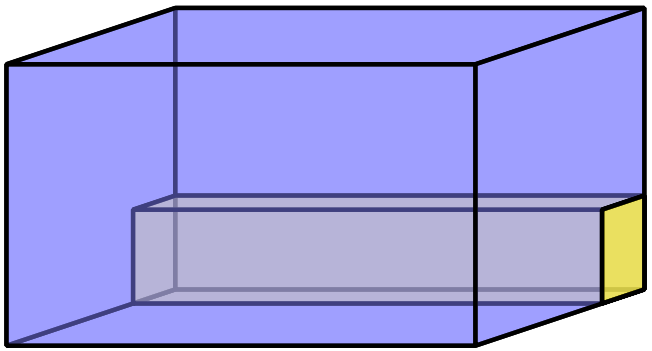
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

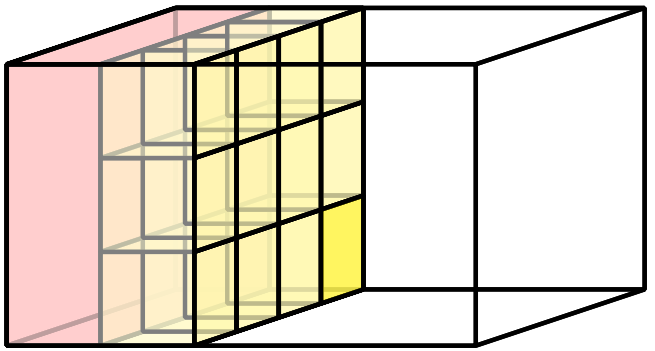


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

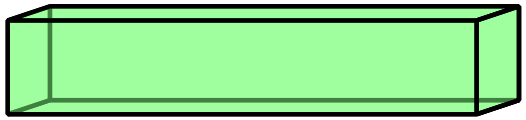


input $\mathbf{x}$



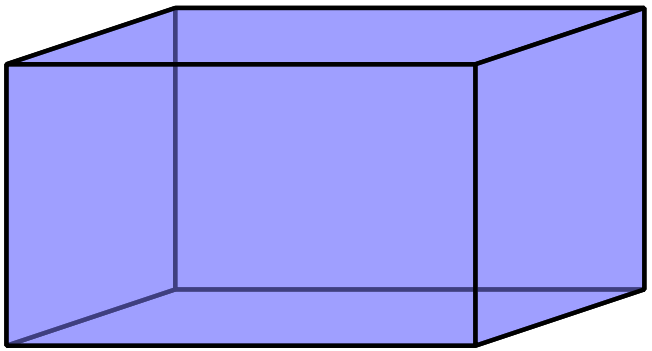
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 **convolution**

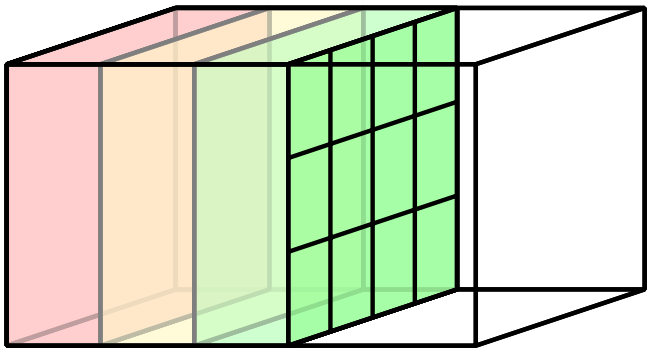


kernel $\mathbf{w}_3$

different kernel for
each output dimension



input $\mathbf{x}$

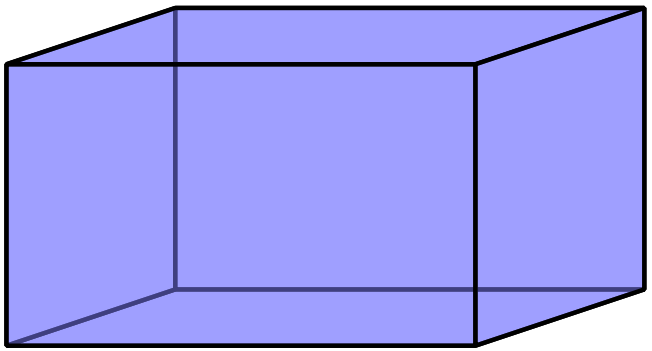


output $y_3 = h(\mathbf{w}_3^\top \star \mathbf{x} + b_3)$

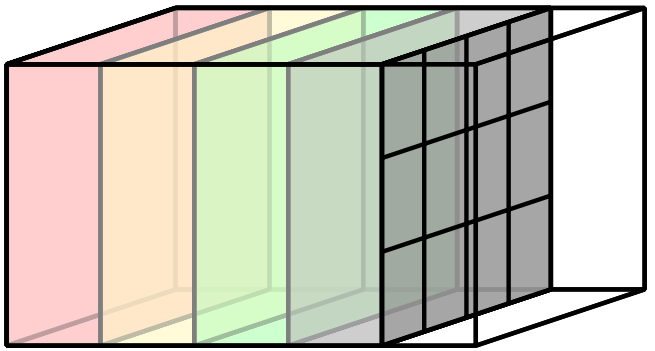
1×1 convolution

kernel $\mathbf{w}_4$

different kernel for
each output dimension



input $\mathbf{x}$

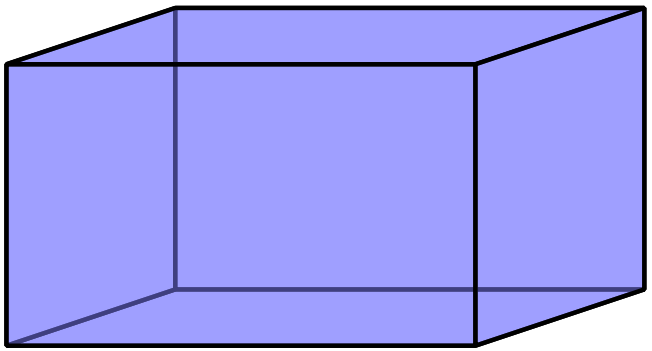


output $y_4 = h(\mathbf{w}_4^\top \star \mathbf{x} + b_4)$

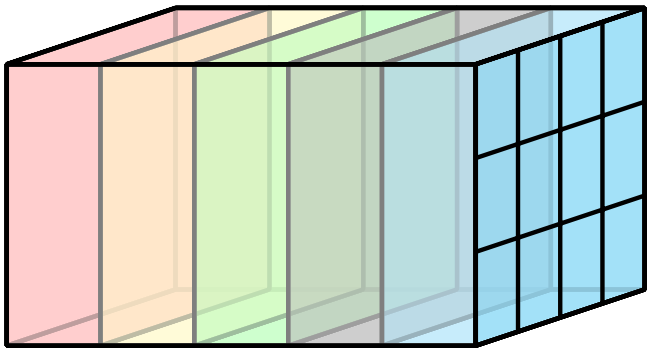
1×1 convolution

kernel $\mathbf{w}_5$

different kernel for
each output dimension



input $\mathbf{x}$



output $y_5 = h(\mathbf{w}_5^\top \star \mathbf{x} + b_5)$

Conclusion

Computer vision intro
Computer vision history and related fields
Image processing: Fourier, convolution, filtering