Outline

Convolutions: the basics

Convolutional networks

Interpreting convolutional networks

Common applications

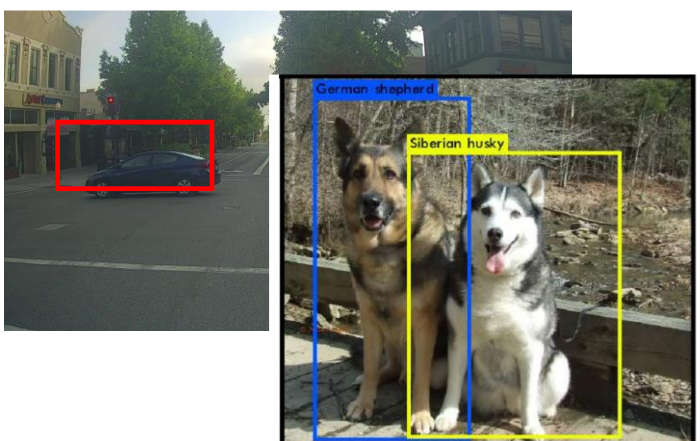
Computer Vision Problems

Image Classification

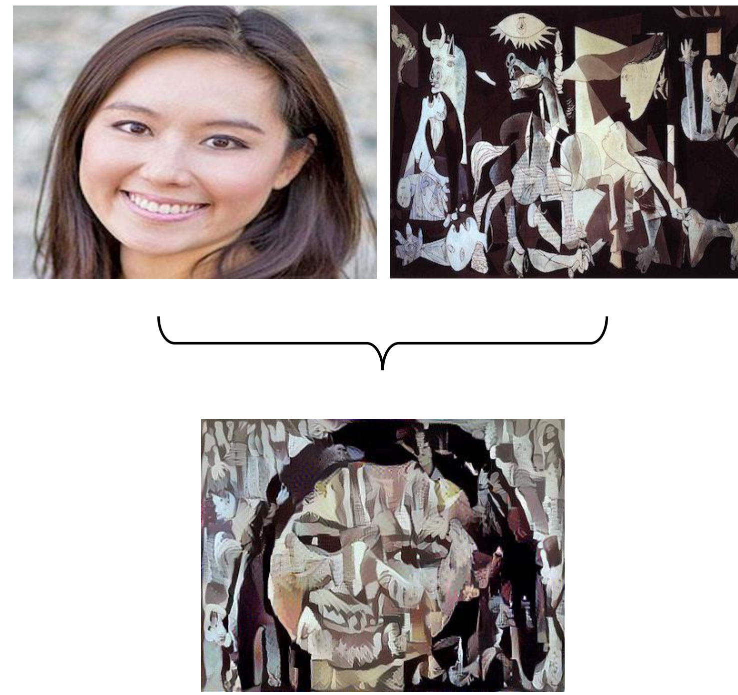
→ Cat? (0/1)

64x64

Object detection



Neural Style Transfer

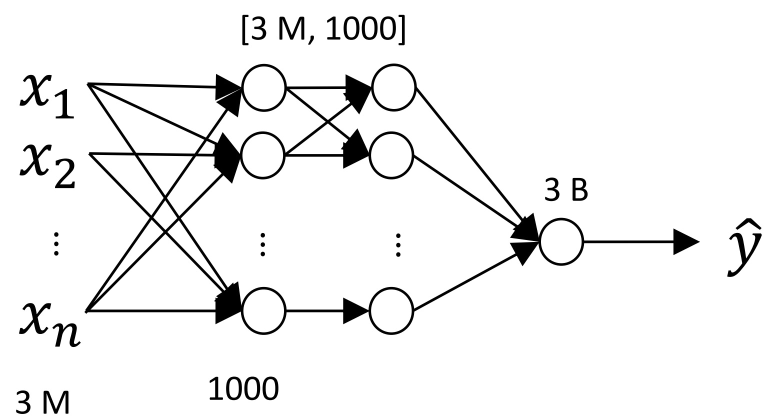


Deep Learning on large images

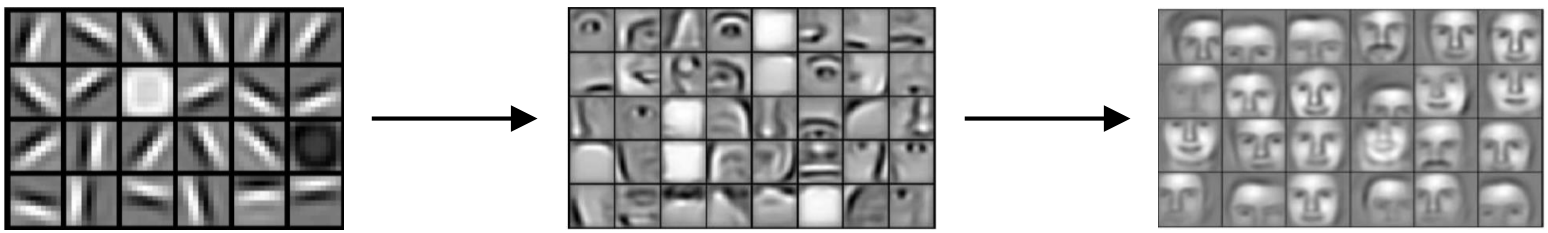
Cat? (0/1)

64x64x3 = 12.288 pixels

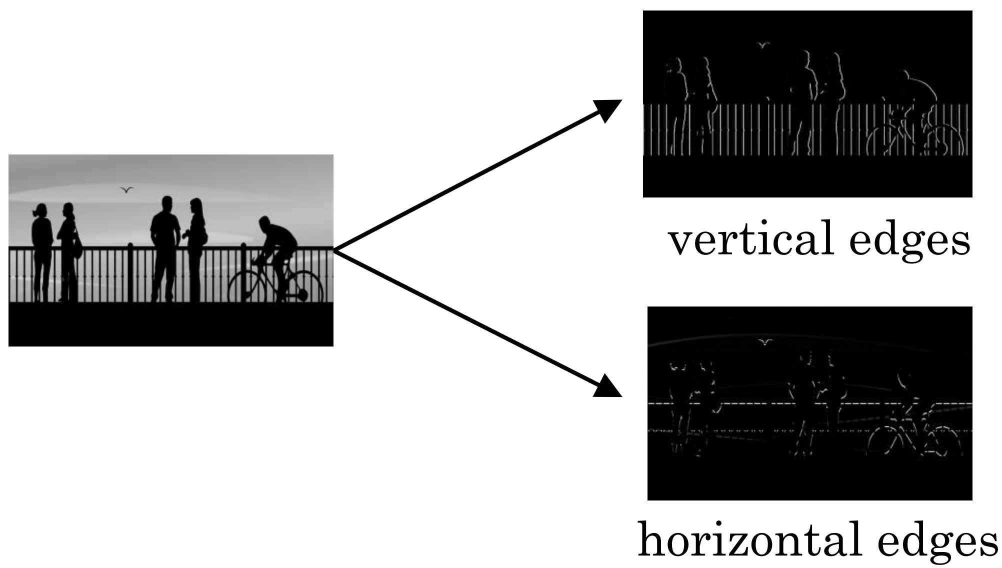
1000 x 1000 x 3 = 3 M pixels



Computer Vision Problem

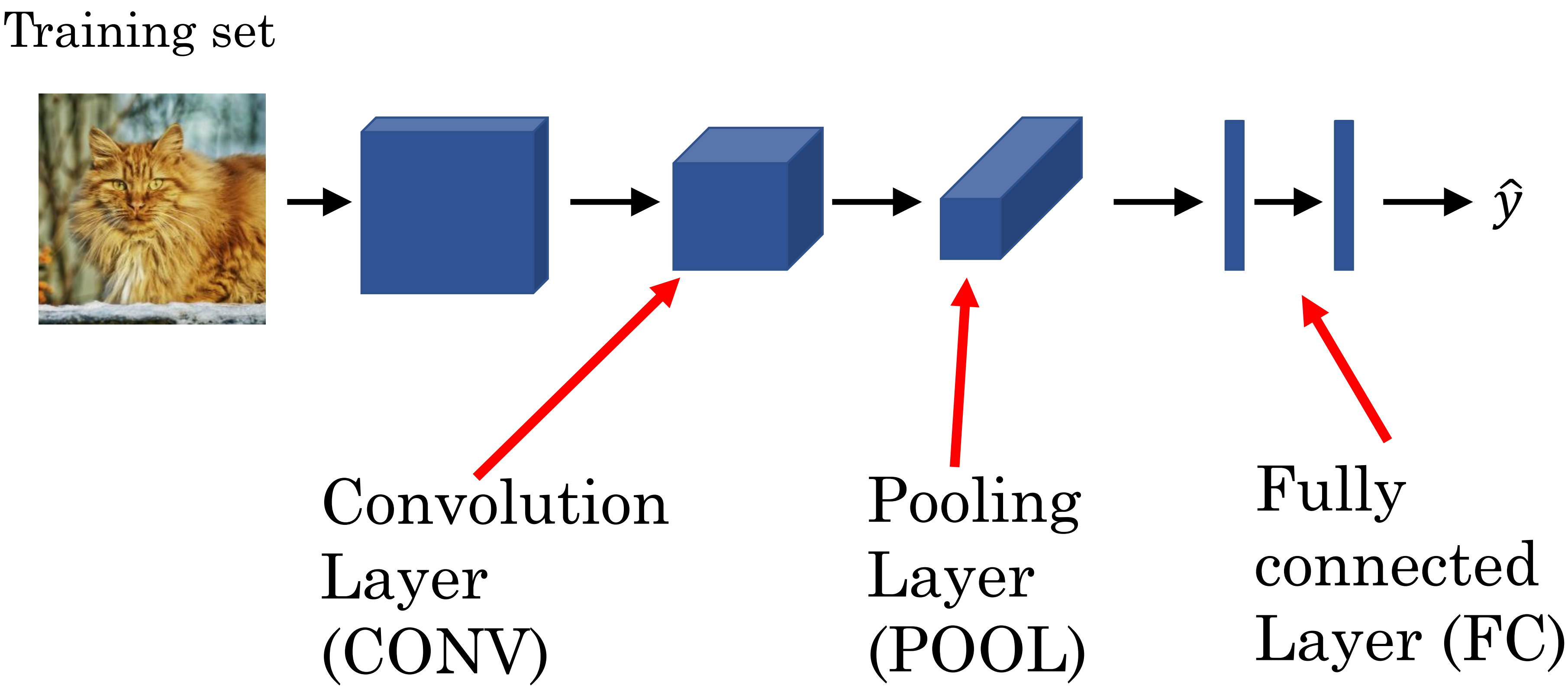


Detection of layers of Neural Networks



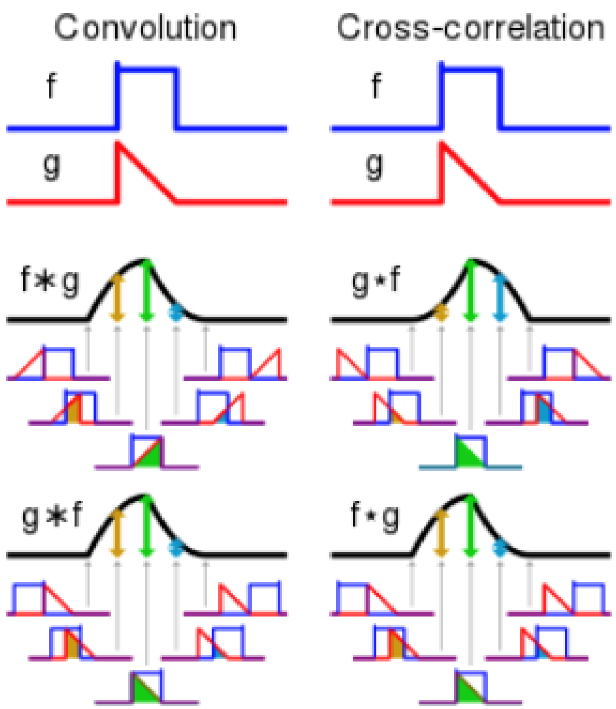
Convolutional Neural Network

Neural networks that include convolution operations



Convolution Layer

- A network layer that convolves its receptive input before passing it to the next layer.
- Most ML libraries implement convolutional layers as **cross-correlation layers**.



$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m, j+n)$$

Discrete cross-correlation: 2-D example

$$(I \star K)(0,0) = \begin{aligned} &I(0,0)K(0,0) + I(0,1)K(0,1) + I(0,2)K(0,2) + \\ &I(1,0)K(1,0) + I(1,1)K(1,1) + I(1,2)K(1,2) + \\ &I(2,0)K(2,0) + I(2,1)K(2,1) + I(2,2)K(2,2) + \end{aligned}$$

I

1	0	0	1	2
0	0	0	3	0
0	1	2	1	1
1	1	3	0	0
3	0	0	0	1

$\star$

K

0	0	1
0	2	0
1	1	0

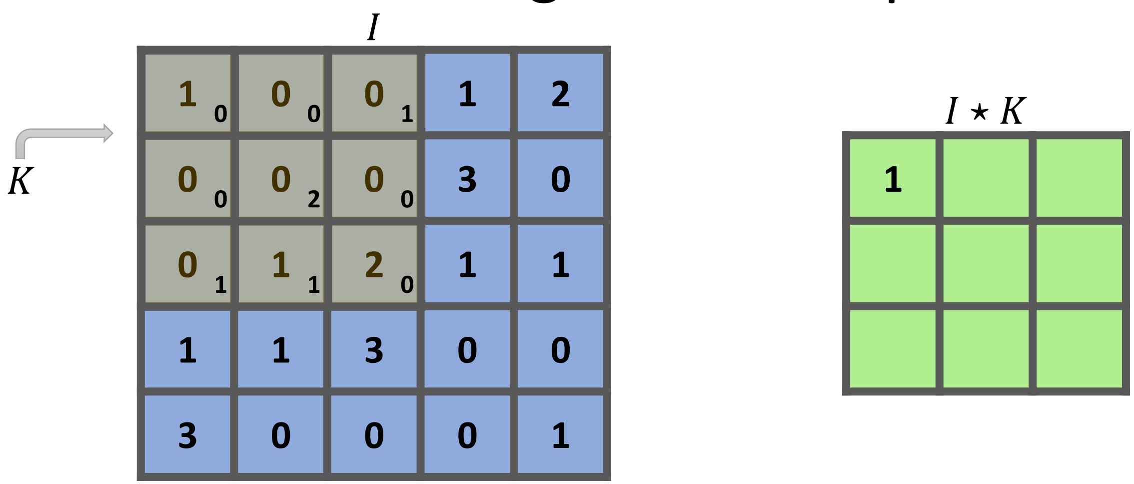
$$(K \star I)(0,0) = \begin{aligned} &1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + \\ &0 \cdot 0 + 0 \cdot 2 + 0 \cdot 0 + \\ &0 \cdot 1 + 1 \cdot 1 + 2 \cdot 0 + \end{aligned}$$

$$= \quad 1$$

$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m, j+n)$$

Discrete cross-correlation: 2-D example

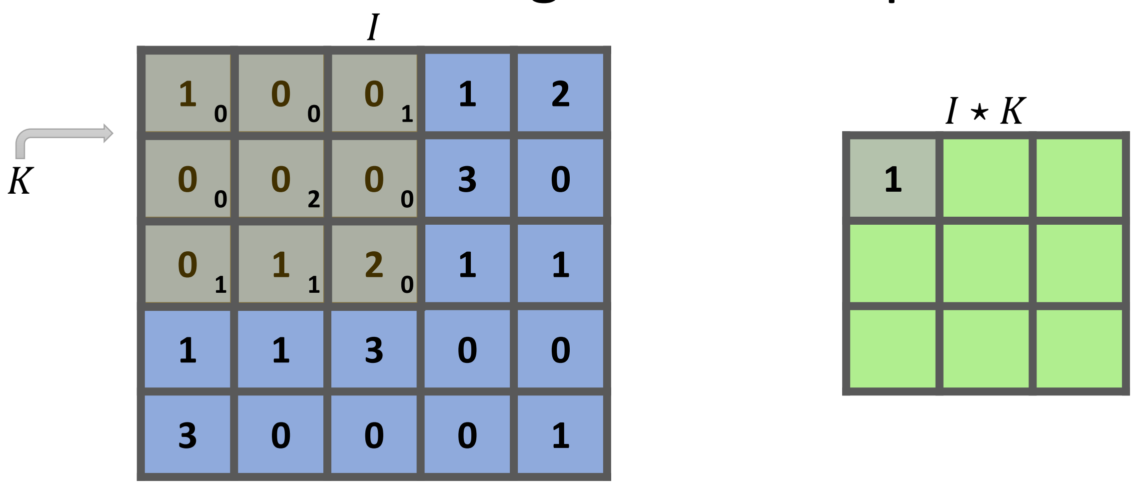
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_{m}\sum_{n} I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

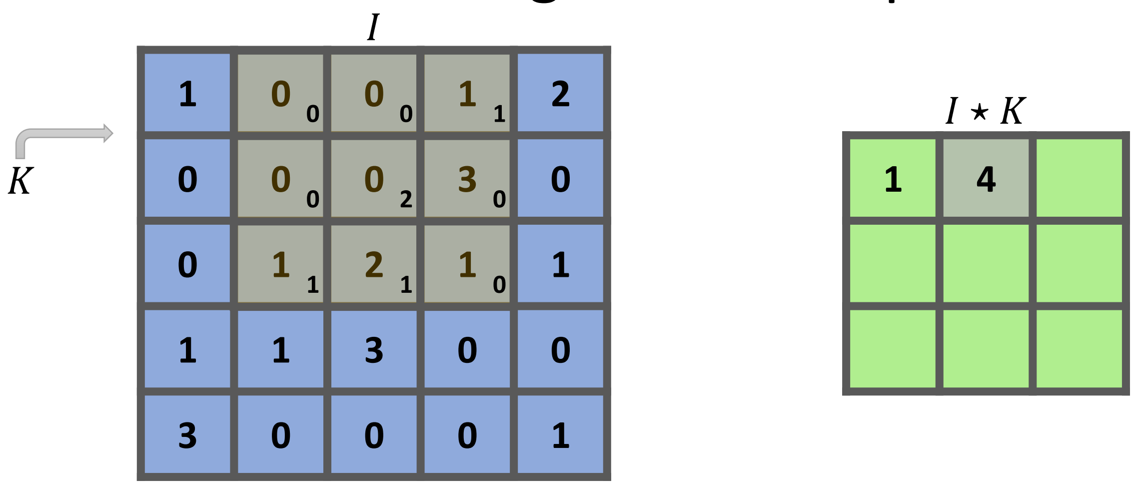
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_{m}\sum_{n} I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

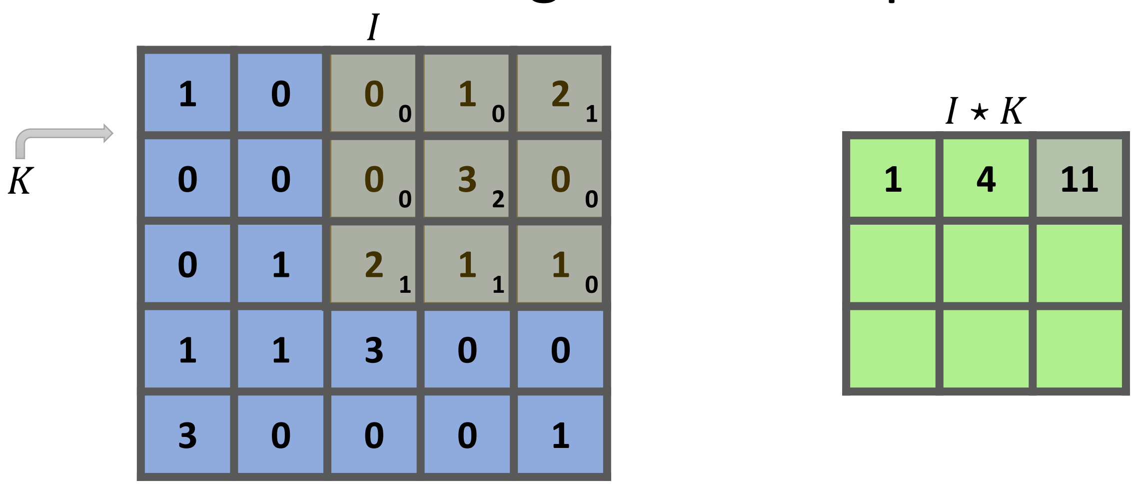
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_{m}\sum_{n} I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

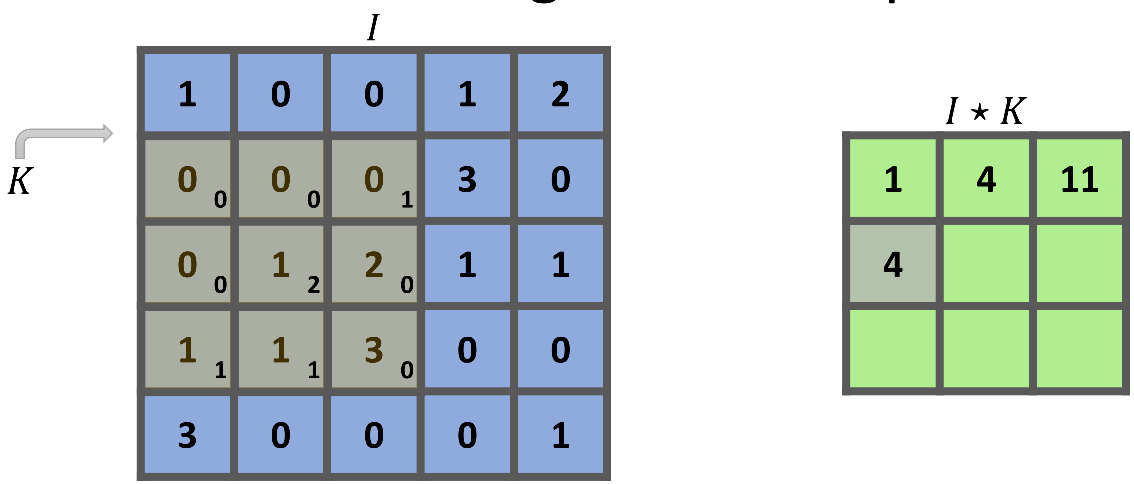
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

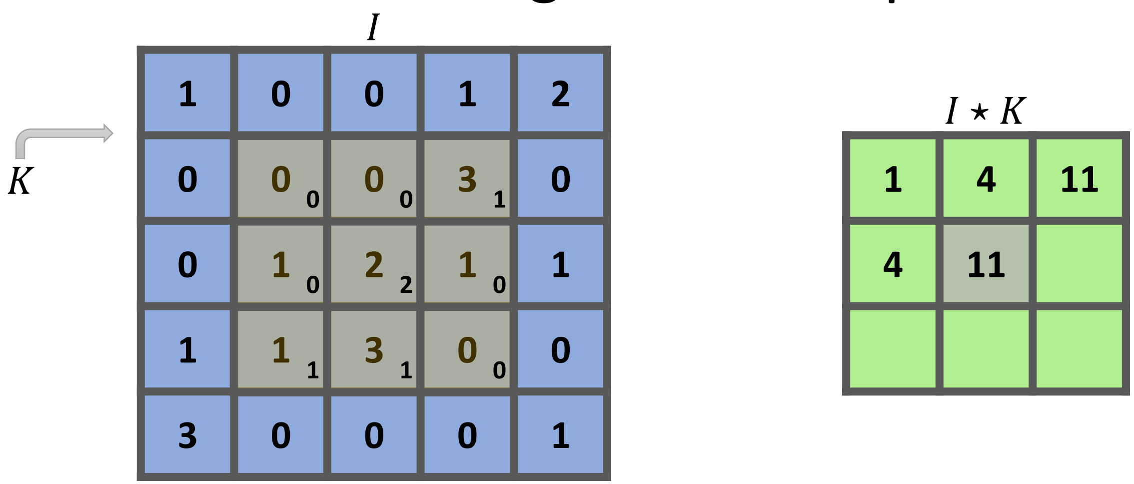
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m, j+n)$$

Discrete cross-correlation: 2-D example

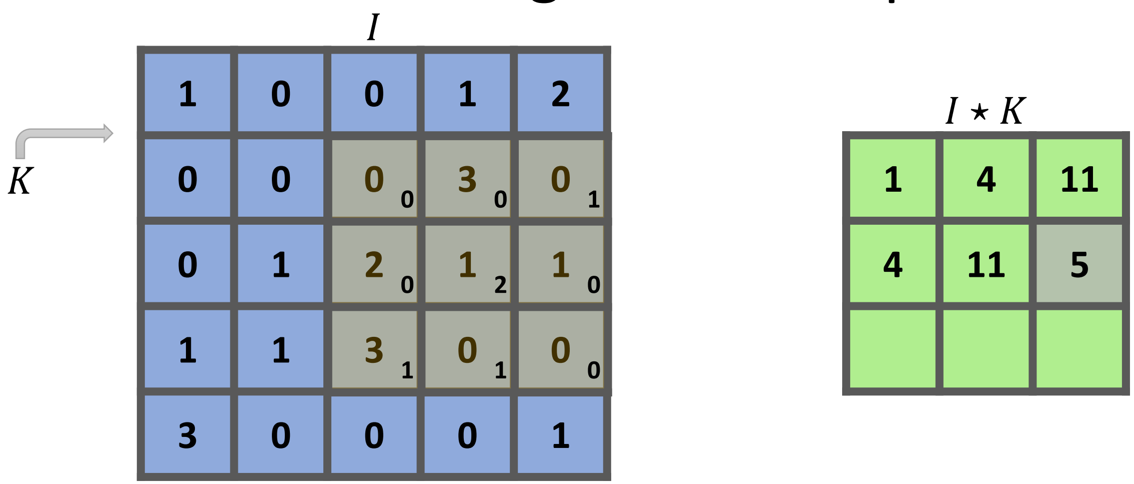
Can be viewed as a "sliding window" operation:



$$(I \star K)(i,j) == \sum_{m} \sum_{n} I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

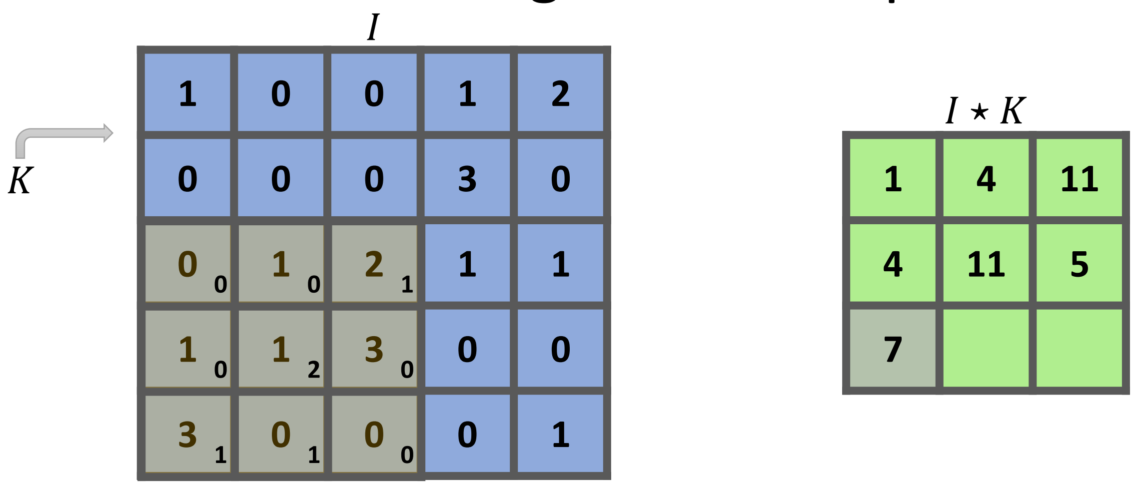
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_{m}\sum_{n} I(m,n)K(i+m, j+n)$$

Discrete cross-correlation: 2-D example

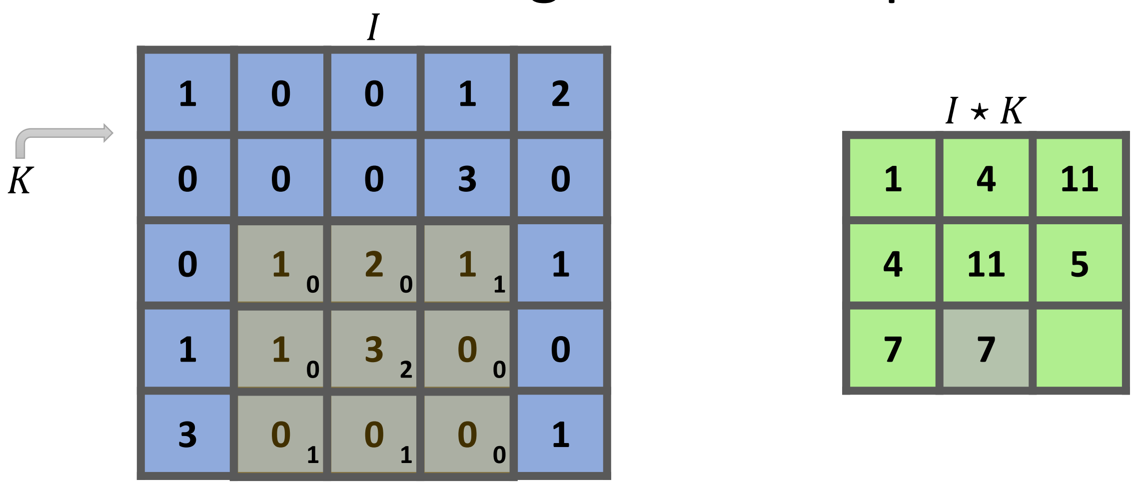
Can be viewed as a "sliding window" operation:



$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m,j+n)$$

Discrete cross-correlation: 2-D example

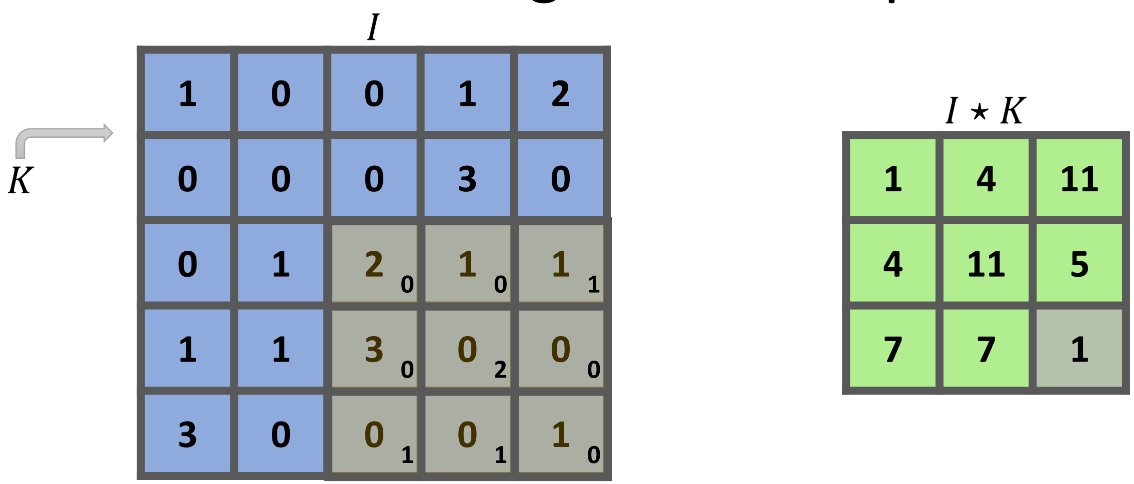
Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_{m} \sum_{n} I(m,n) K(i+m, j+n)$$

Discrete cross-correlation: 2-D example

Can be viewed as a “sliding window” operation:



$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m,j+n)$$

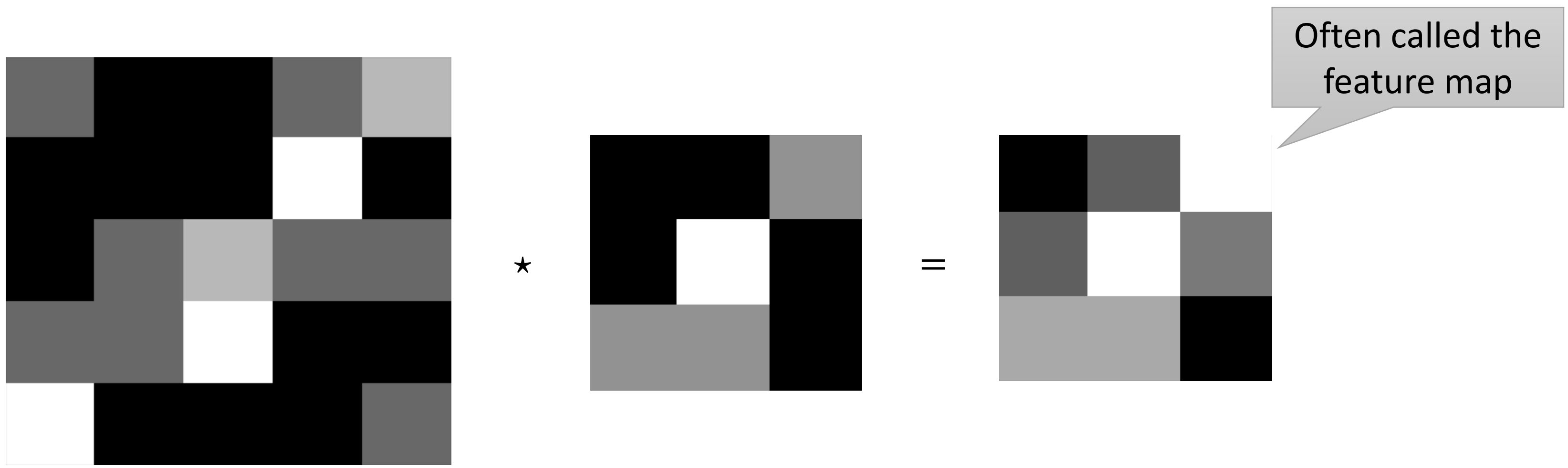
Discrete cross-correlation: 2-D example

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \star \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 11 \\ 4 & 11 & 5 \\ 7 & 7 & 1 \end{bmatrix}$$

Often called the feature map

$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m,j+n)$$

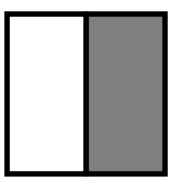
Discrete cross-correlation: 2-D example



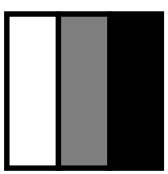
$$(I \star K)(i,j) == \sum_m \sum_n I(m,n)K(i+m, j+n)$$

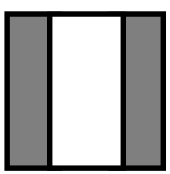
Vertical edge detection

$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline \end{array}$$



*





Vertical edge detection examples

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$

Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

$*$

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

$=$

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Discrete cross-correlation: 2-D example

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \star \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 11 \\ 4 & 11 & 5 \\ 7 & 7 & 1 \end{bmatrix}$$

Something's not quite right…

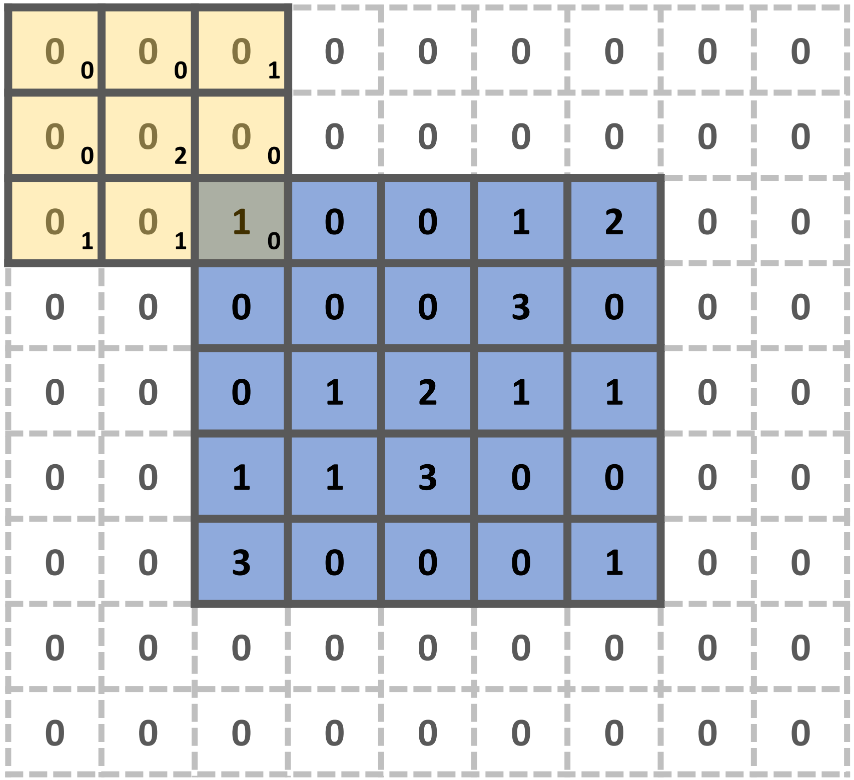
$$(I \star K)(i,j) == \sum_{m} \sum_{n} I(m,n)K(i+m, j+n)$$

Cross-correlation: padding

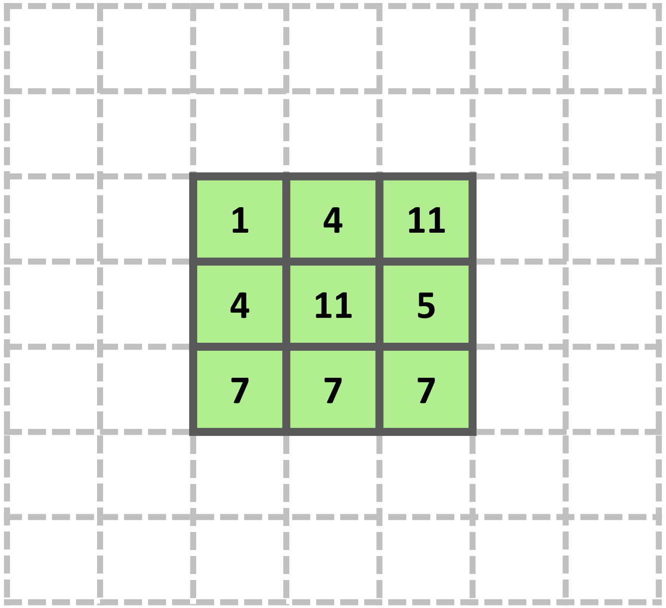
Adding extra pixels outside the image

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

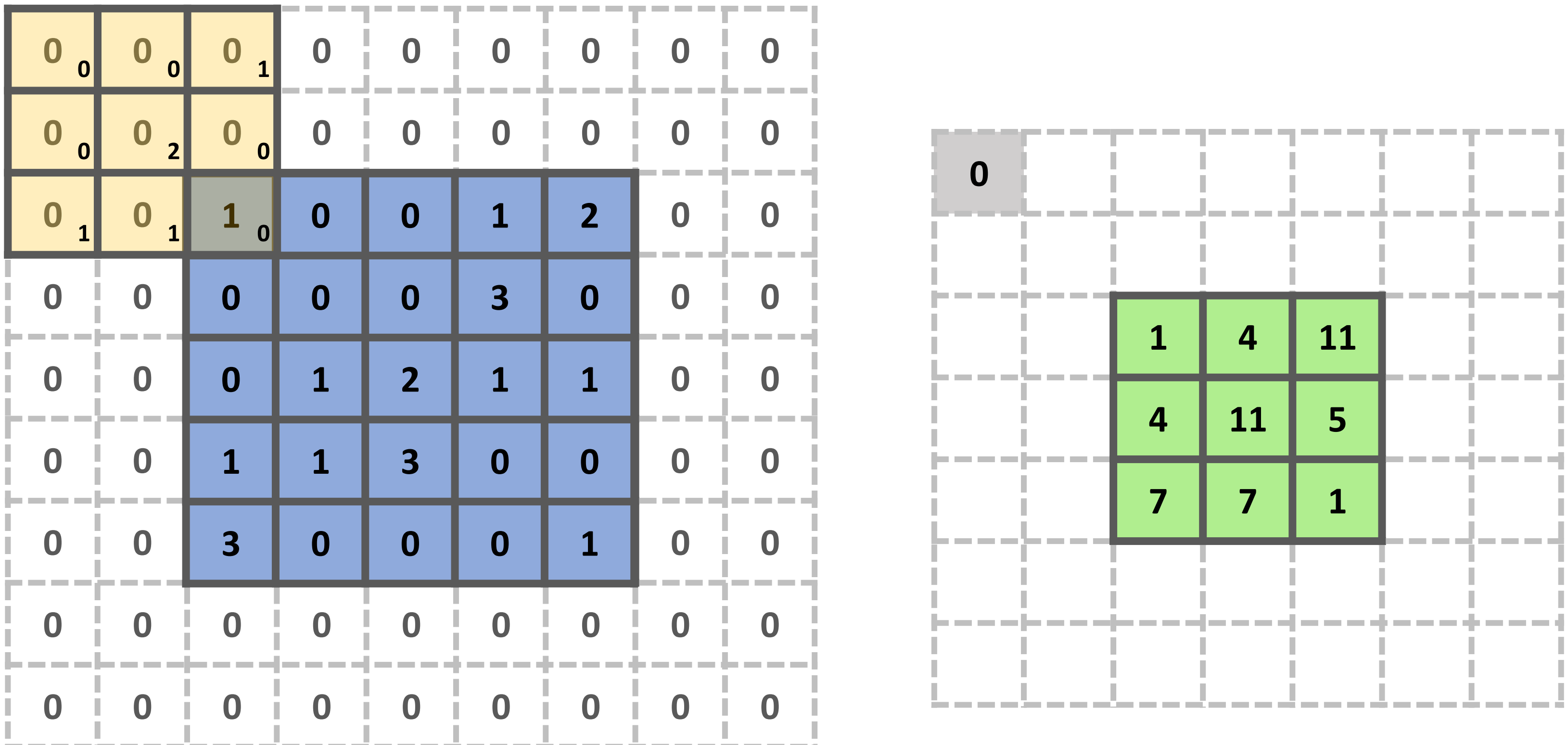
Discrete cross-correlation: padding



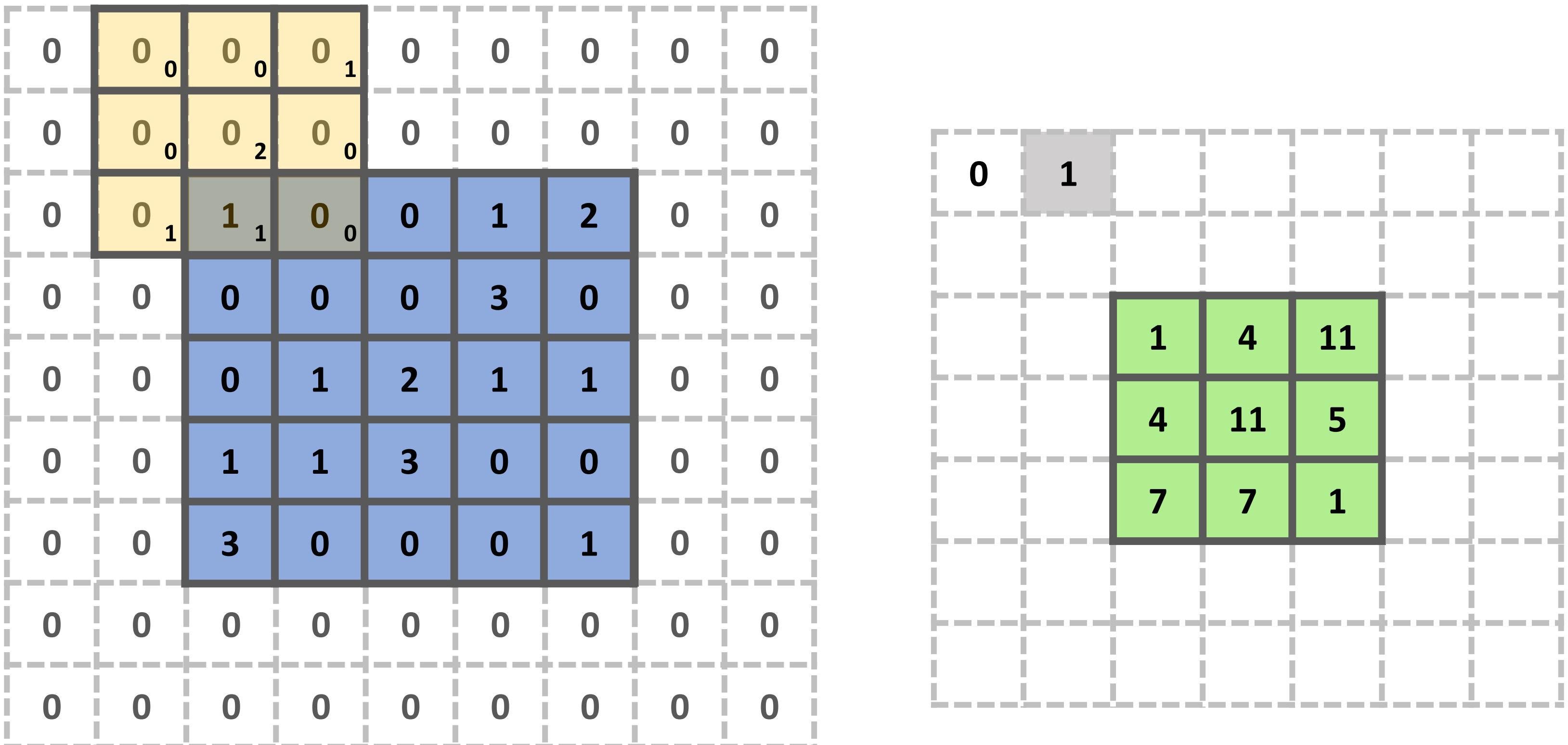
Technically, our signals have infinite extent… we solve by padding with zeros



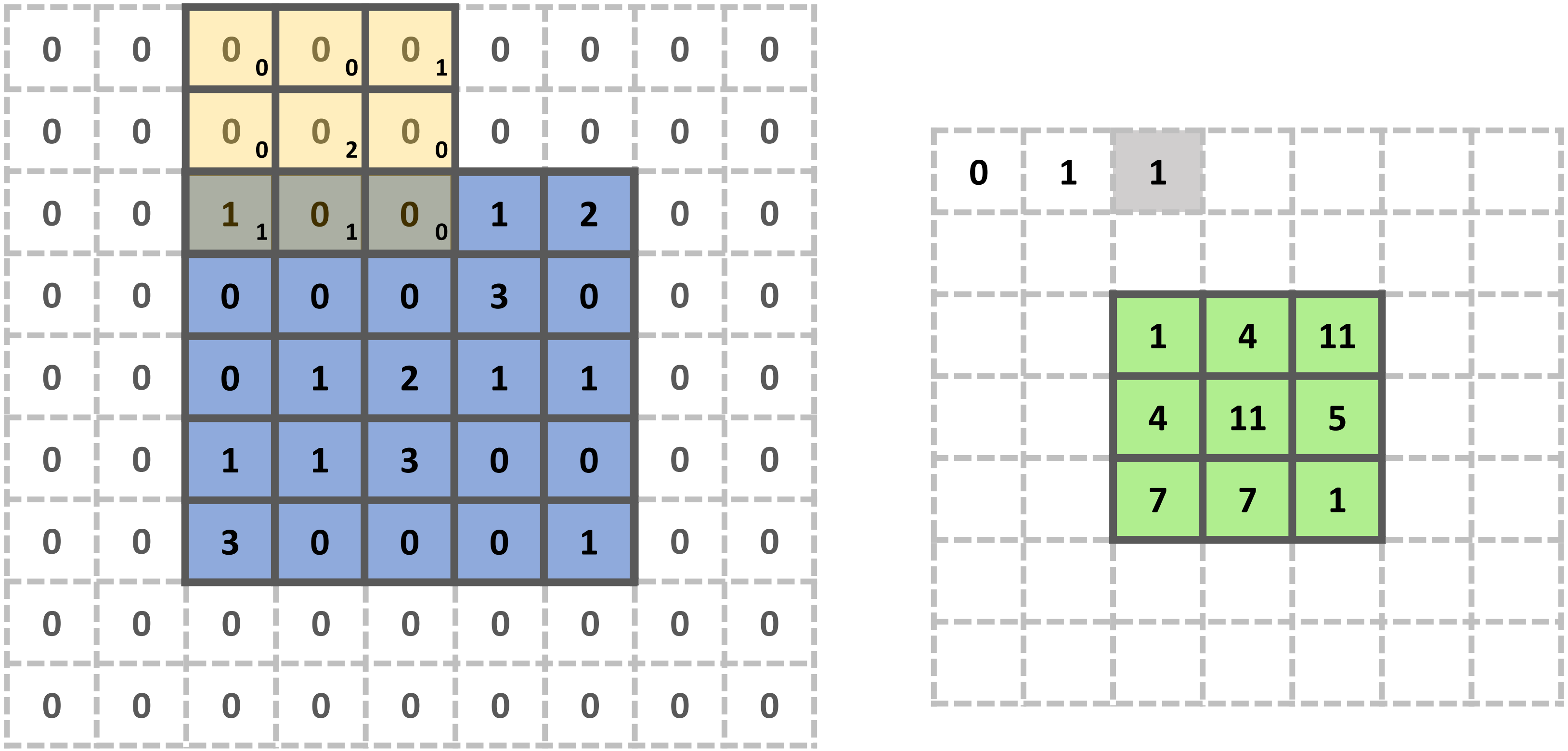
Discrete cross-correlation: padding



Discrete cross-correlation: padding



Discrete cross-correlation: padding



Discrete cross-correlation: padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	2	0	0
0	0	0	0	0	3	0	0	0
0	0	0	1	2	1	1	0	0
0	0	1	1	3	0	0	0	0
0	0	3	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Normally we want the output to maintain the input size

0	1	1	0	1	3	2
0	2	0	0	5	7	0
1	0	1	4	11	2	1
0	1	4	11	5	2	0
0	6	7	7	1	1	1
1	7	3	0	0	2	0
3	0	0	0	1	0	0

Discrete cross-correlation: padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	2	0	0
0	0	0	0	0	3	0	0	0
0	0	0	1	2	1	1	0	0
0	0	1	1	3	0	0	0	0
0	0	3	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Normally we want the output to maintain the input size

0	1	1	0	1	3	2
0	2	0	0	5	7	0
1	0	1	4	11	2	1
0	1	4	11	5	2	0
0	6	7	7	1	1	1
1	7	3	0	0	2	0
3	0	0	0	1	0	0

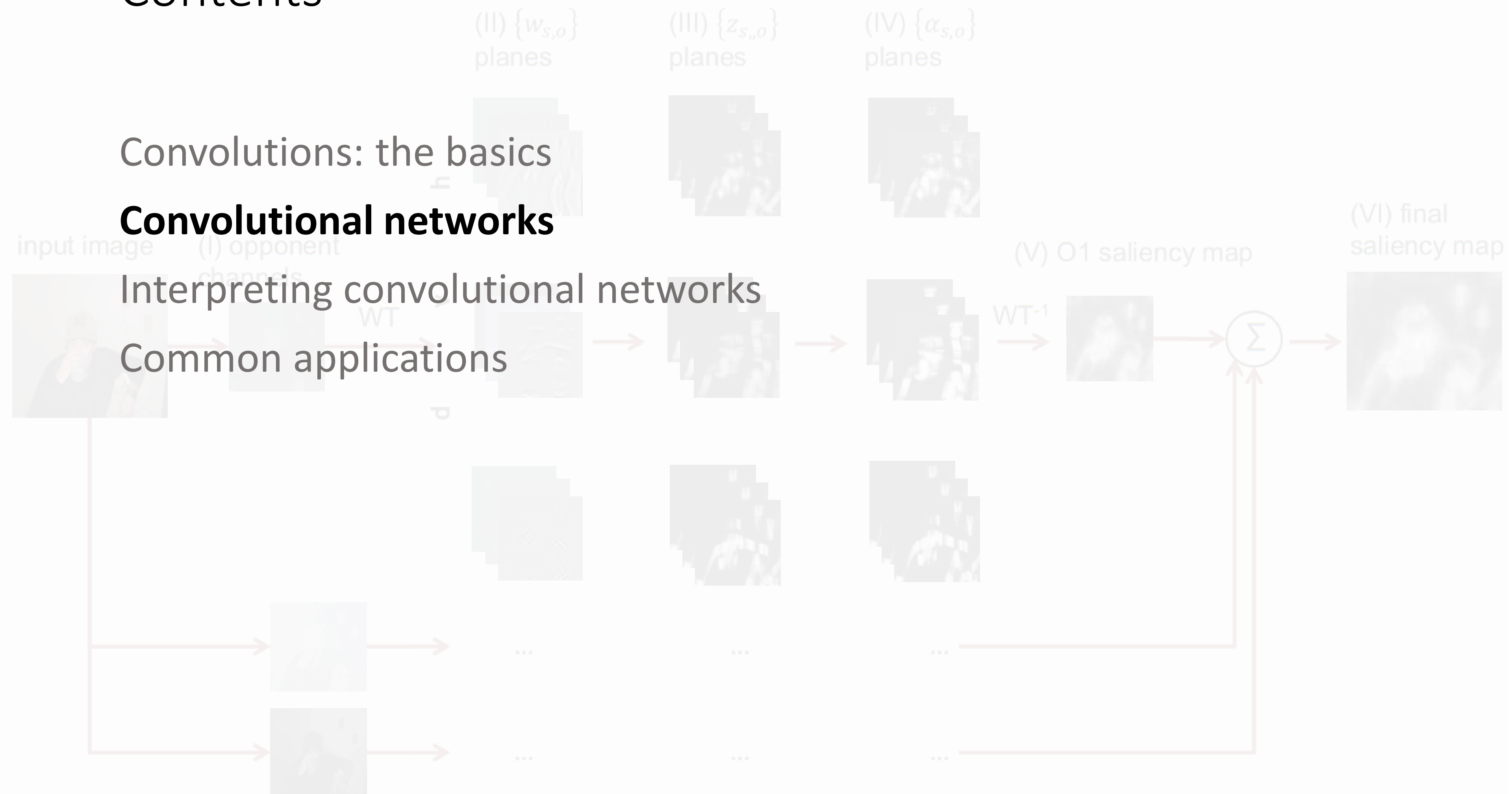
Contents

Convolutions: the basics

Convolutional networks

Interpreting convolutional networks

Common applications



Convolutional networks

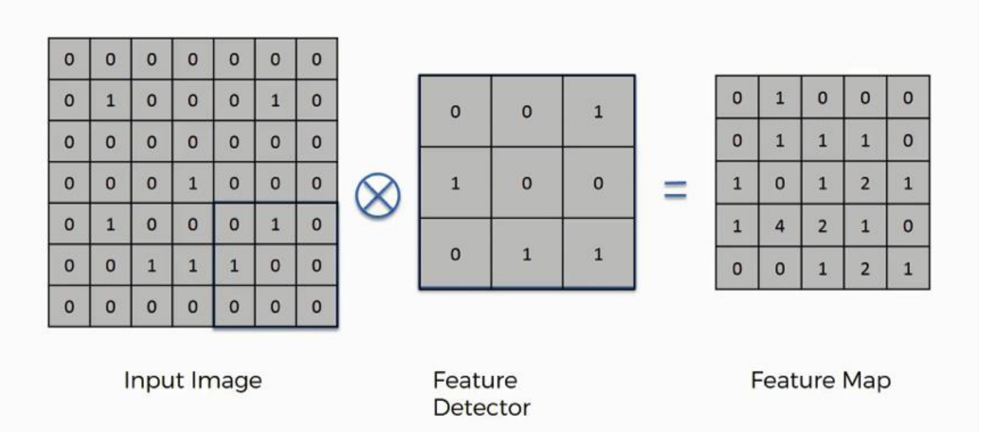
Reminder: Neural networks that include convolution operations

Can be used in place of dense matrix multiplication (i.e. fully-connected layers)

Motivations:

- Sparse connectivity
- Parameter sharing
- Translation equivariance
- Arbitrary input sizes

Why convolutions: Motivation



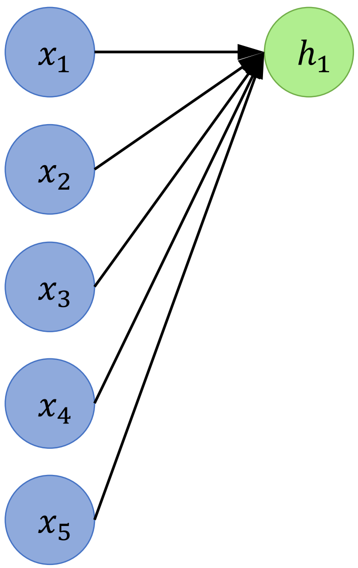
Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Translation Equivariance

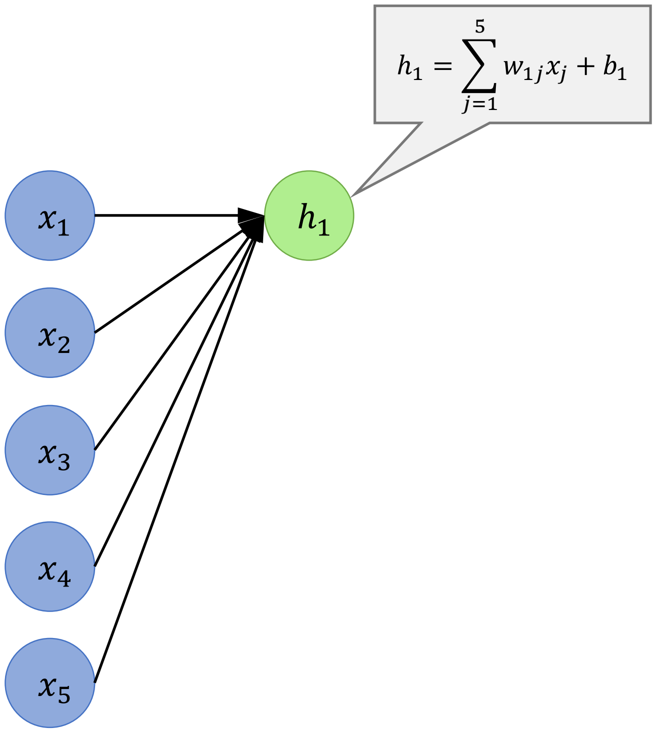
Arbitrary Input Sizes

Neural networks ⇒ Convolutional networks



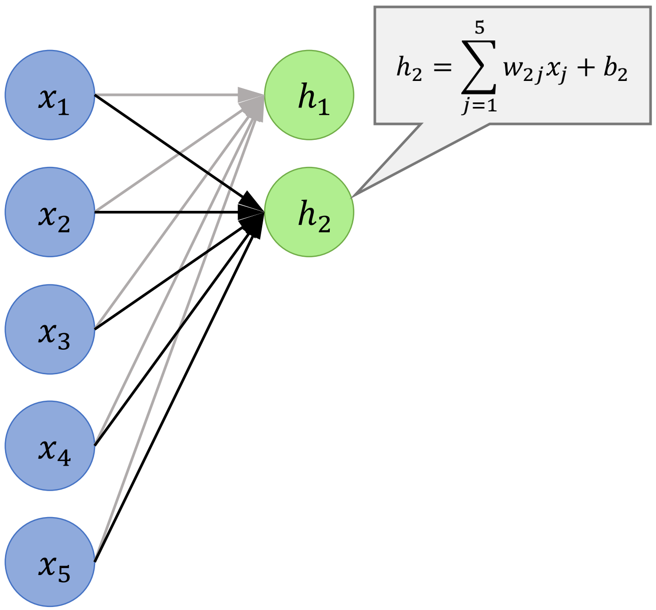
$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$$

Neural networks ⇒ Convolutional networks



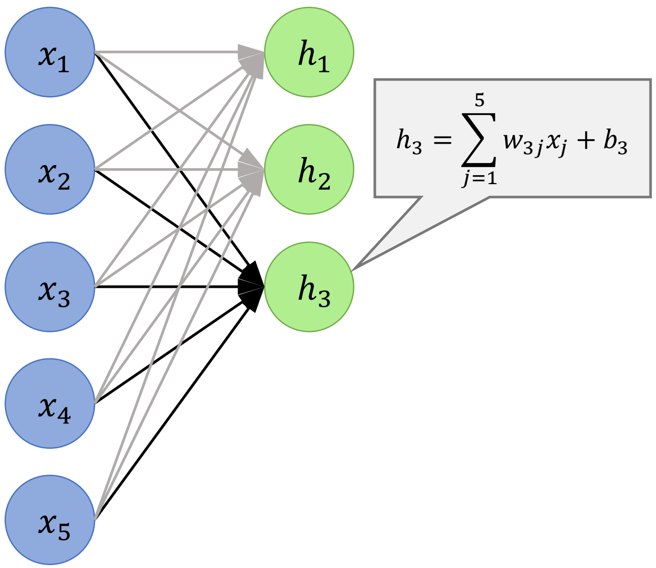
$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

Neural networks ⇒ Convolutional networks



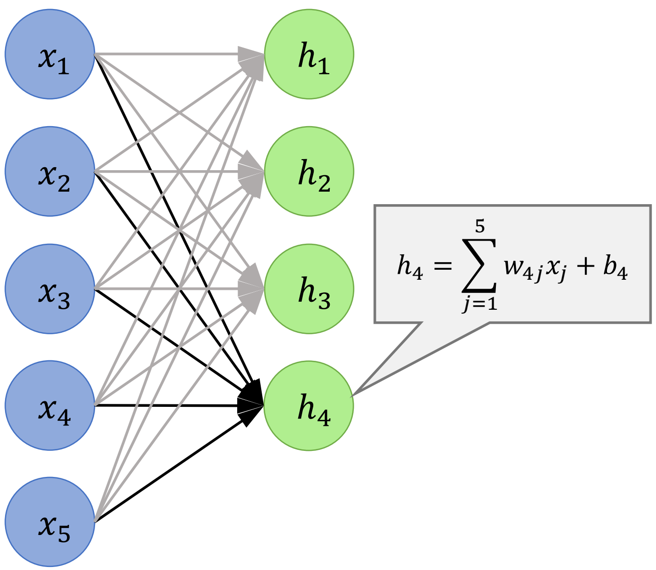
$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$; $h_i = \sum_j w_{ij} x_j + b_i$

Neural networks ⇒ Convolutional networks



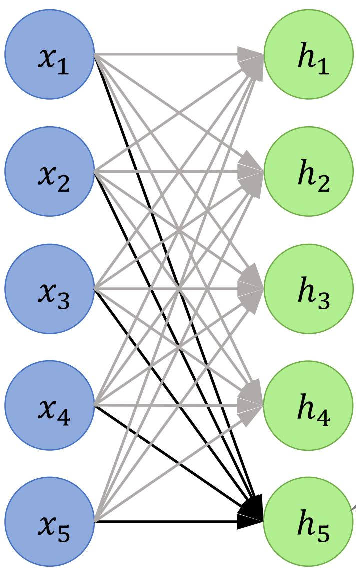
$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

Neural networks ⇒ Convolutional networks



$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

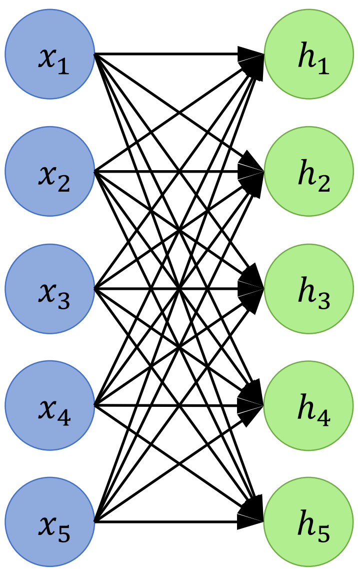
Neural networks ⇒ Convolutional networks



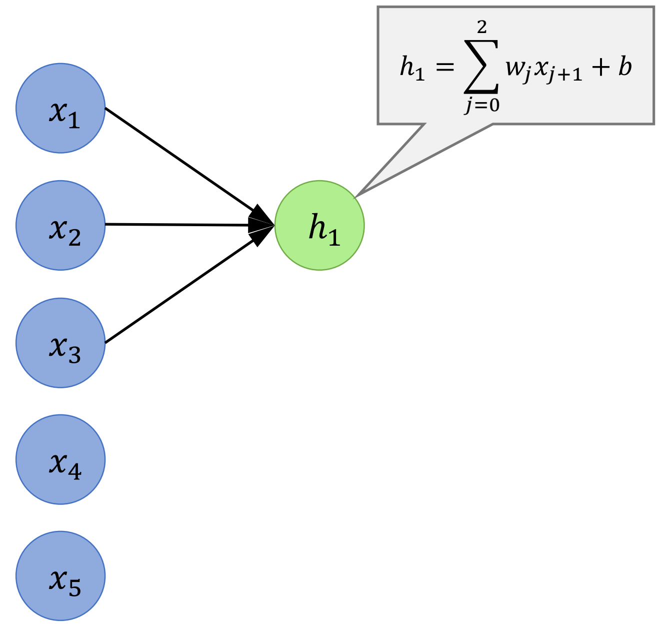
$$h_5 = \sum_{j=1}^{5} w_{5j} x_j + b_5$$

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$$

Neural networks ⇒ Convolutional networks

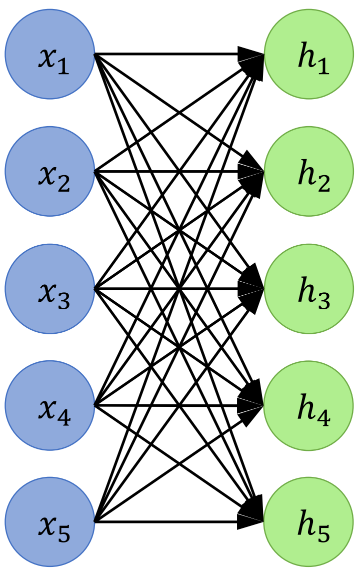


$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$$

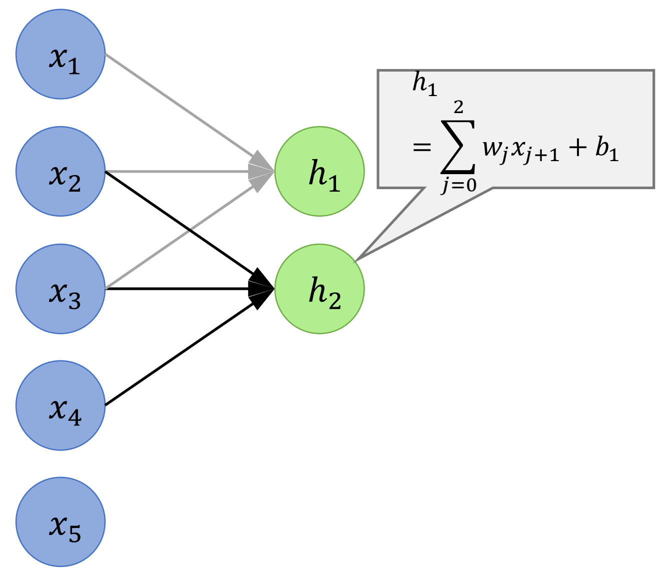


$$h_i = \sum_j w_j x_{j+i} + b$$

Neural networks ⇒ Convolutional networks

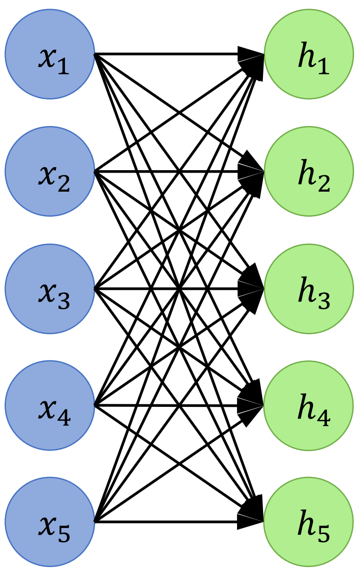


$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

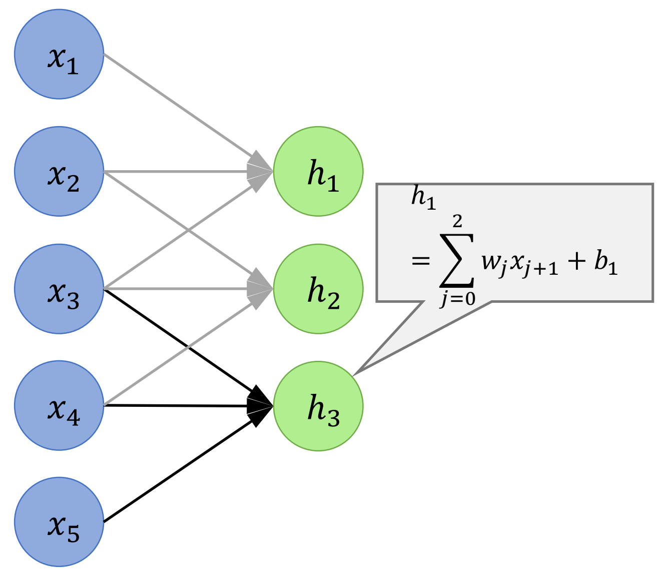


$h_i = \sum_j w_j x_{j+i} + b$

Neural networks ⇒ Convolutional networks

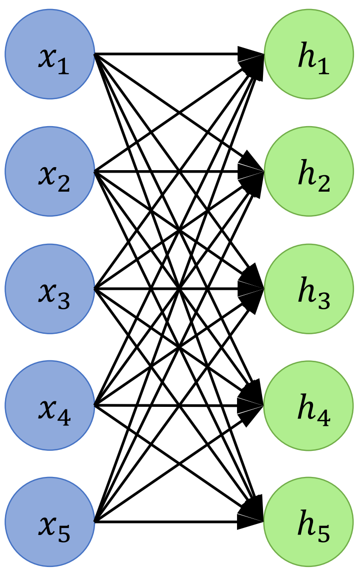


$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$



$h_i = \sum_j w_j x_{j+i} + b$

Neural networks ⇒ Convolutional networks

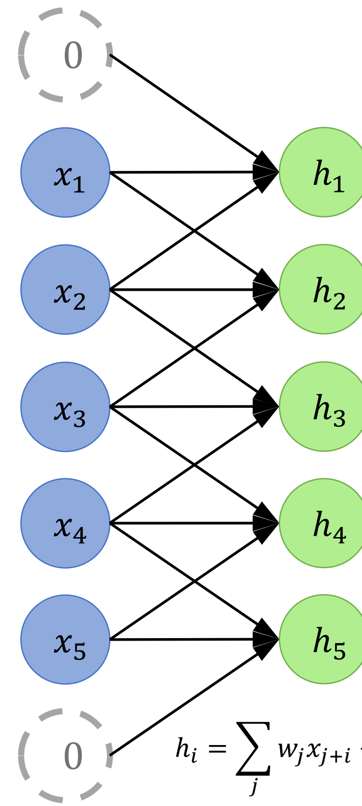


$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

dense connectivity vs sparse connectivity

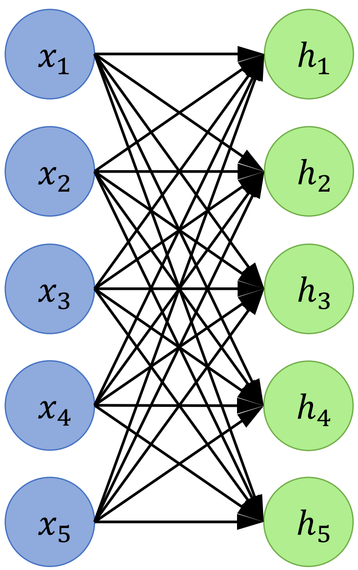
In our example:
5x5 multiplications vs 5x3 multiplications

Sparse connectivity scales better
e.g. 25x25 vs 25x3



$h_i = \sum_j w_j x_{j+i} + b$

Neural networks ⇒ Convolutional networks

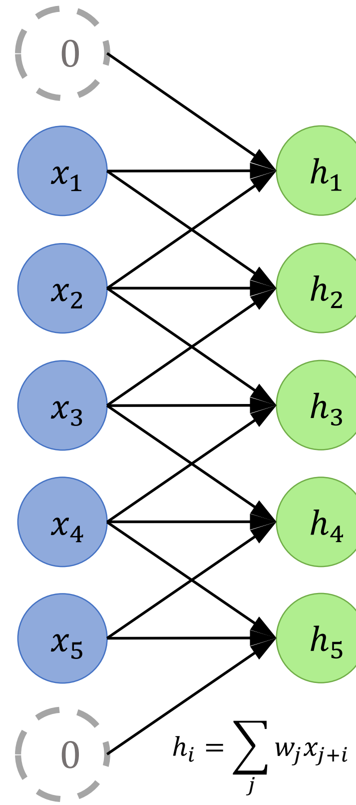


$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b};\ h_i = \sum_j w_{ij} x_j + b_i$

unshared vs shared weights

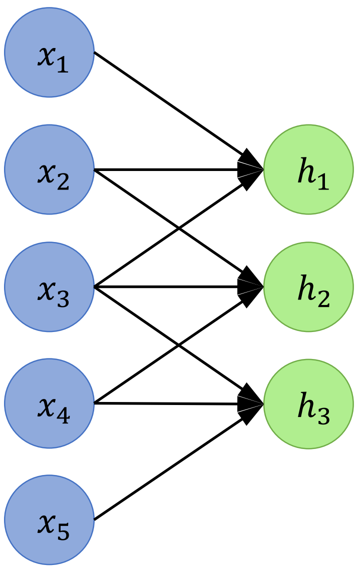
In our example:
5x5 vs 3 weights

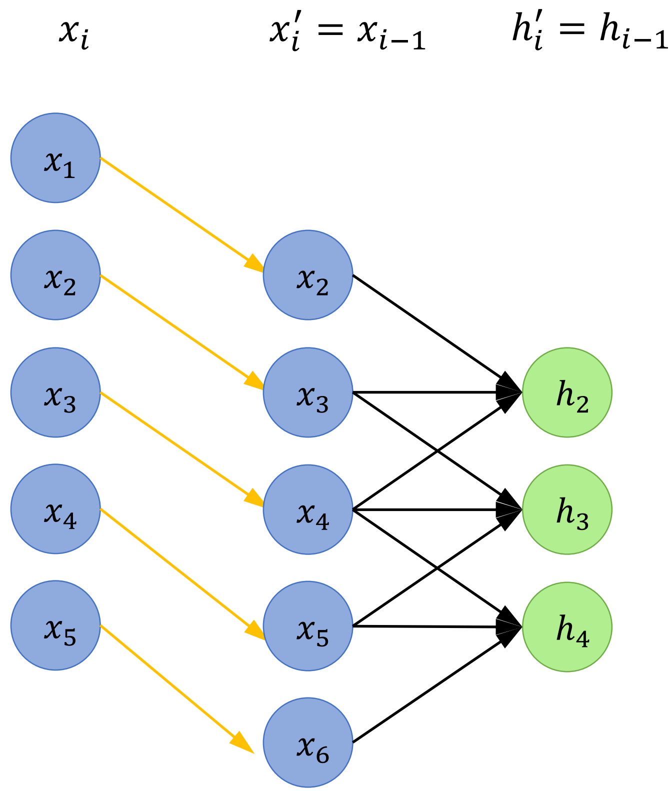
shared weights scale way better:
e.g. 25x25 vs 3



$h_i = \sum_j w_j x_{j+i} + b$

Translation equivariance





Translation equivariance

Why is translation equivariance useful?

Helps make predictions **translation invariant**

CNN → “butterfly”

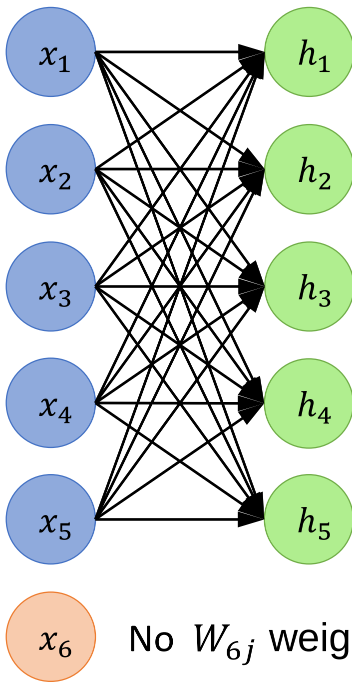
CNN → “butterfly”

CNN → “butterfly”

What other types of invariance could be useful?

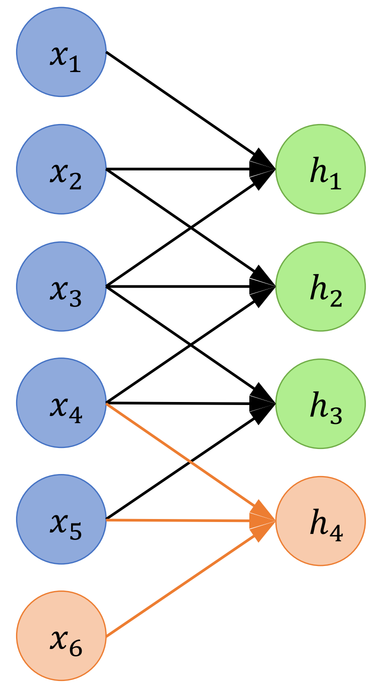
Arbitrary input sizes

$$h_i = \sum_j W_{ij} x_j + b_i$$



No W_{6j} weights!

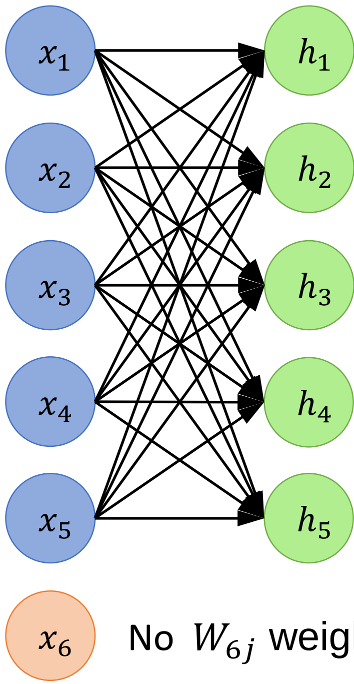
$$h_i = \sum_j w_j x_{j+i} + b$$



Arbitrary input sizes

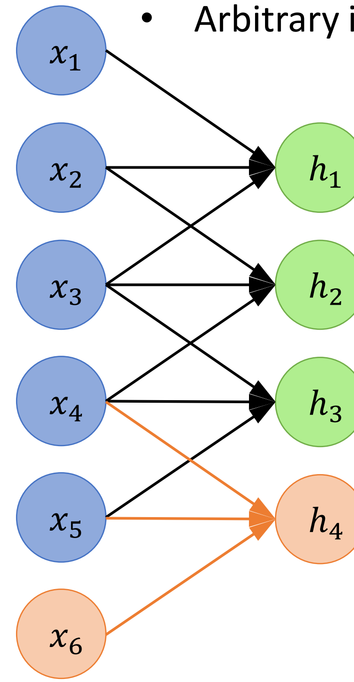
- Sparse connectivity
- Parameter sharing
- Translation equivariance
- Arbitrary input sizes

$$h_i = \sum_j W_{ij} x_j + b_i$$



No W_{6j} weights!

$$h_i = \sum_j w_j x_{j+i} + b$$



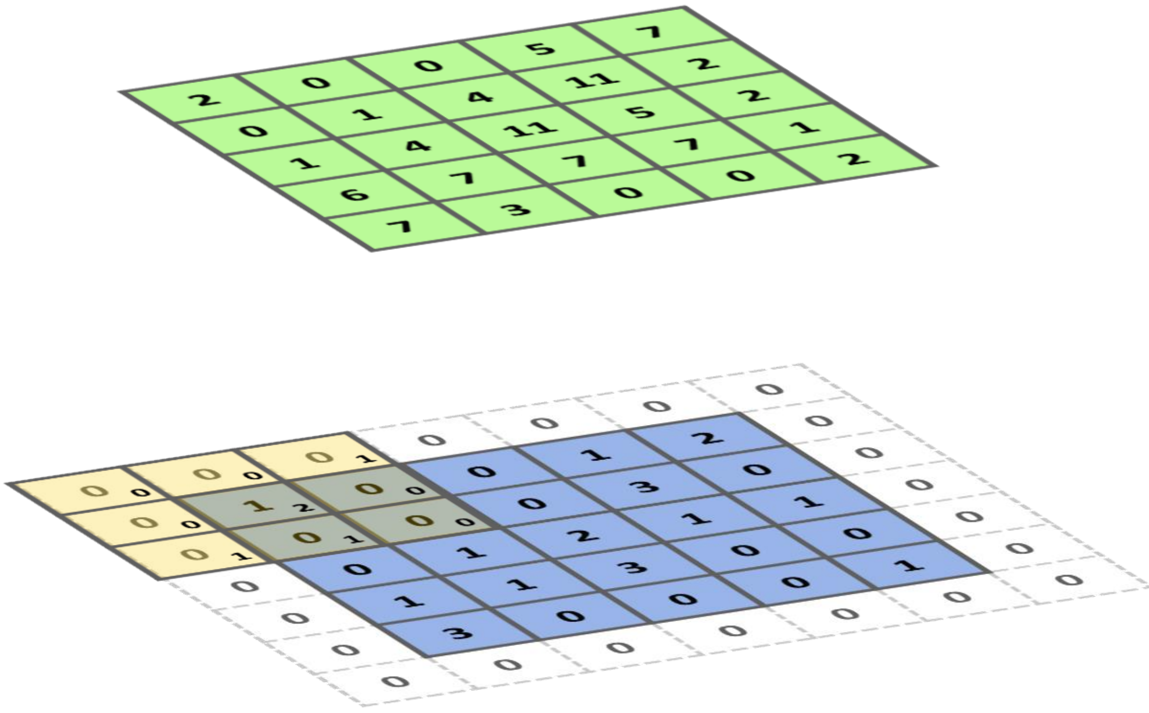
Strided convolution

$$\begin{bmatrix} 2 & 3 & 7 & 4 & 6 & 2 & 9 \\ 6 & 6 & 9 & 8 & 7 & 4 & 3 \\ 3 & 4 & 8 & 3 & 8 & 9 & 7 \\ 7 & 8 & 3 & 6 & 6 & 3 & 4 \\ 4 & 2 & 1 & 8 & 3 & 4 & 6 \\ 3 & 2 & 4 & 1 & 9 & 8 & 3 \\ 0 & 1 & 3 & 9 & 2 & 1 & 4 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 4 \\ 1 & 0 & 2 \\ -1 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 91 & 100 & 83 \\ 69 & 91 & 127 \\ 44 & 72 & 74 \end{bmatrix}$$

$$\text{Stride} = 2$$

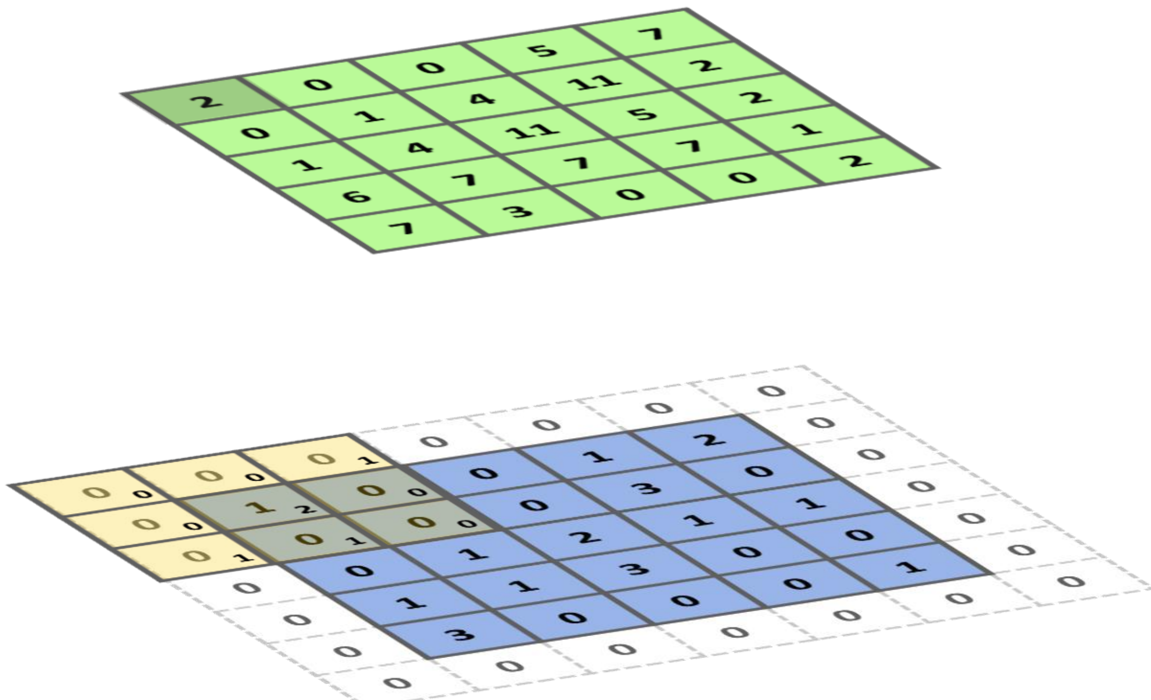
Stride is the number of pixels shifts over the input matrix (sliding step).

The effect of different strides



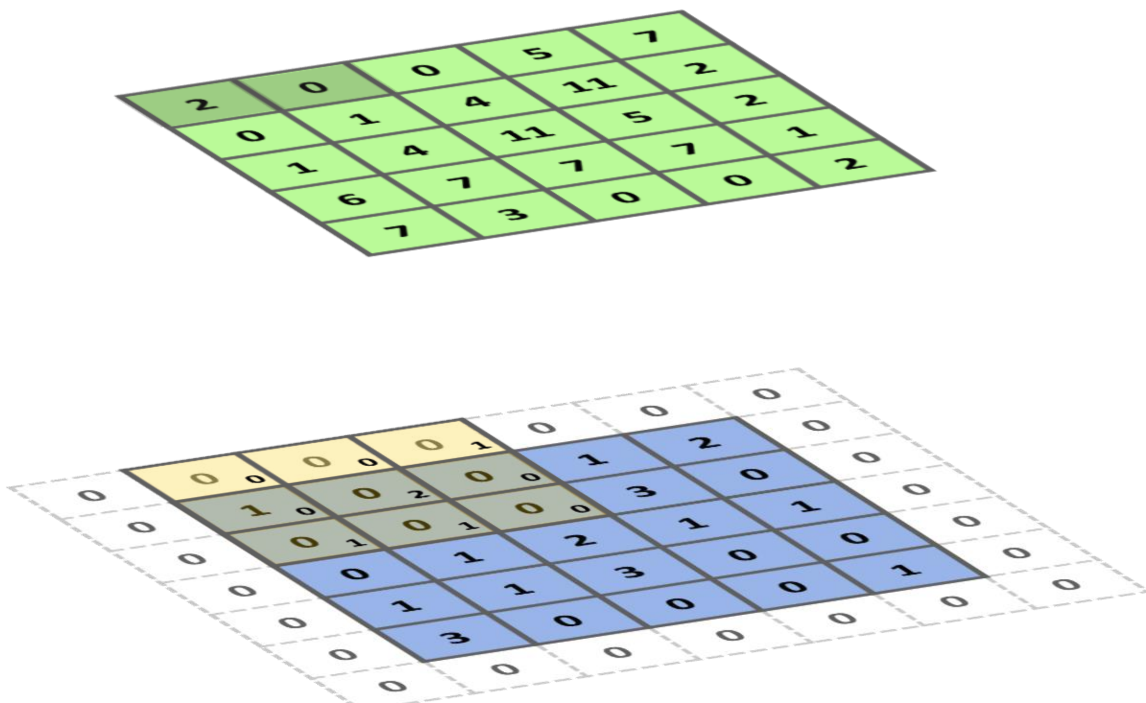
The effect of different strides

Stride: 1x1



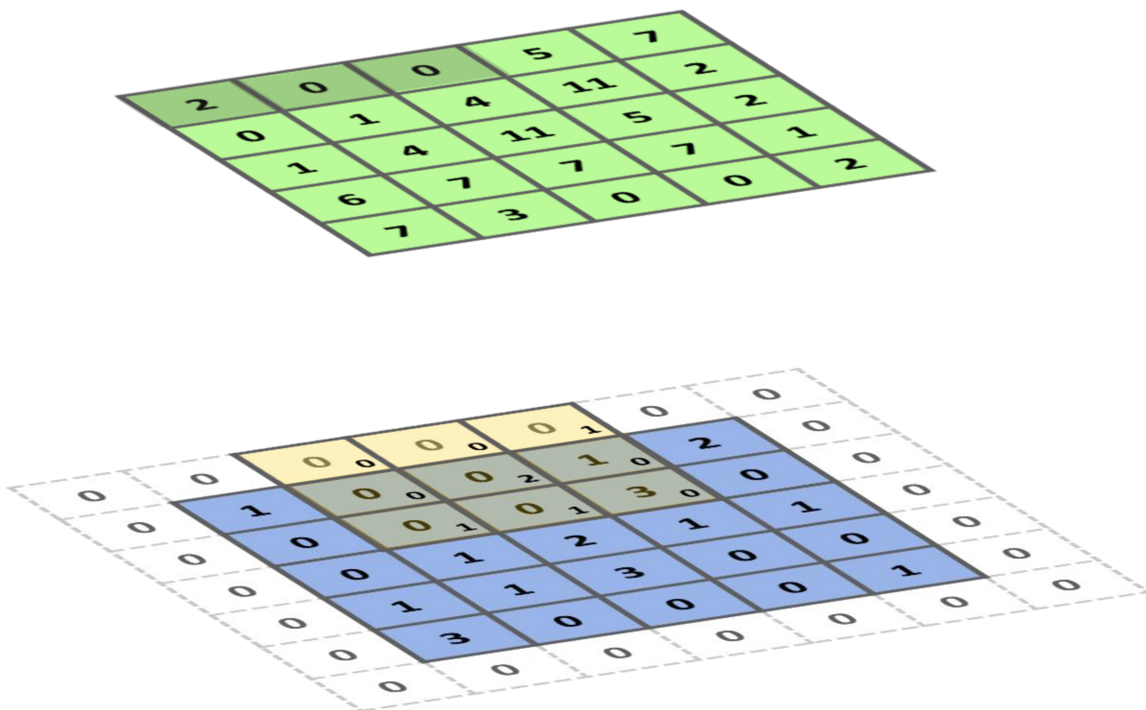
The effect of different strides

Stride: 1x1



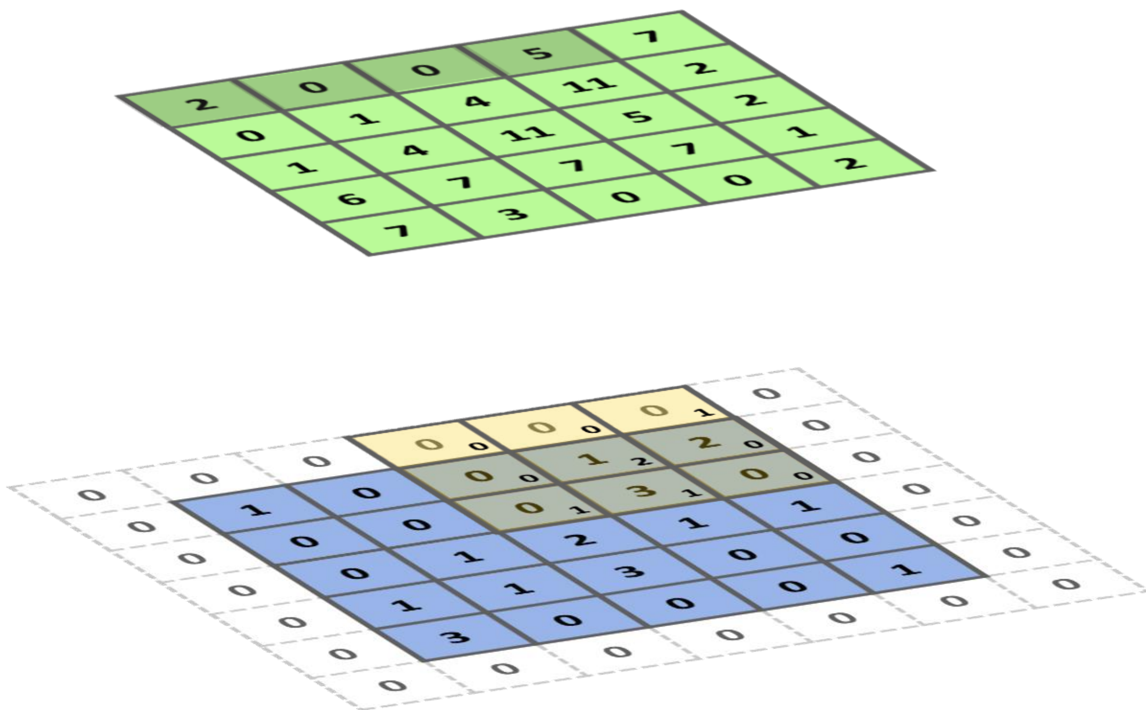
The effect of different strides

Stride: 1x1



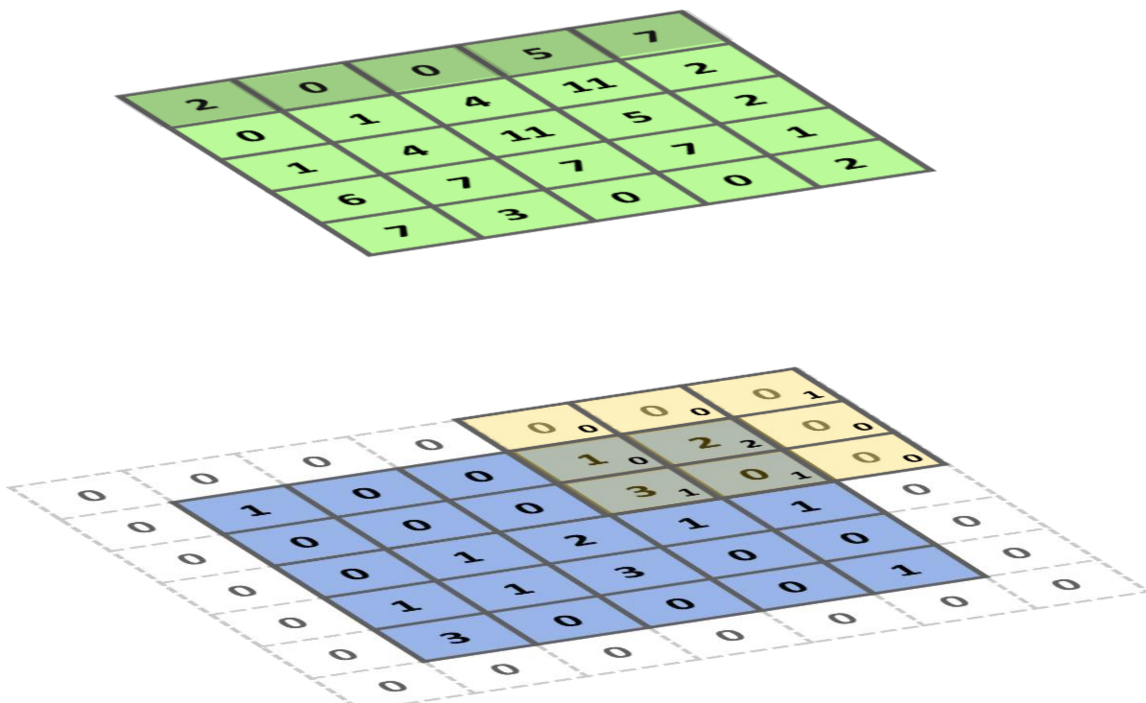
The effect of different strides

Stride: 1x1



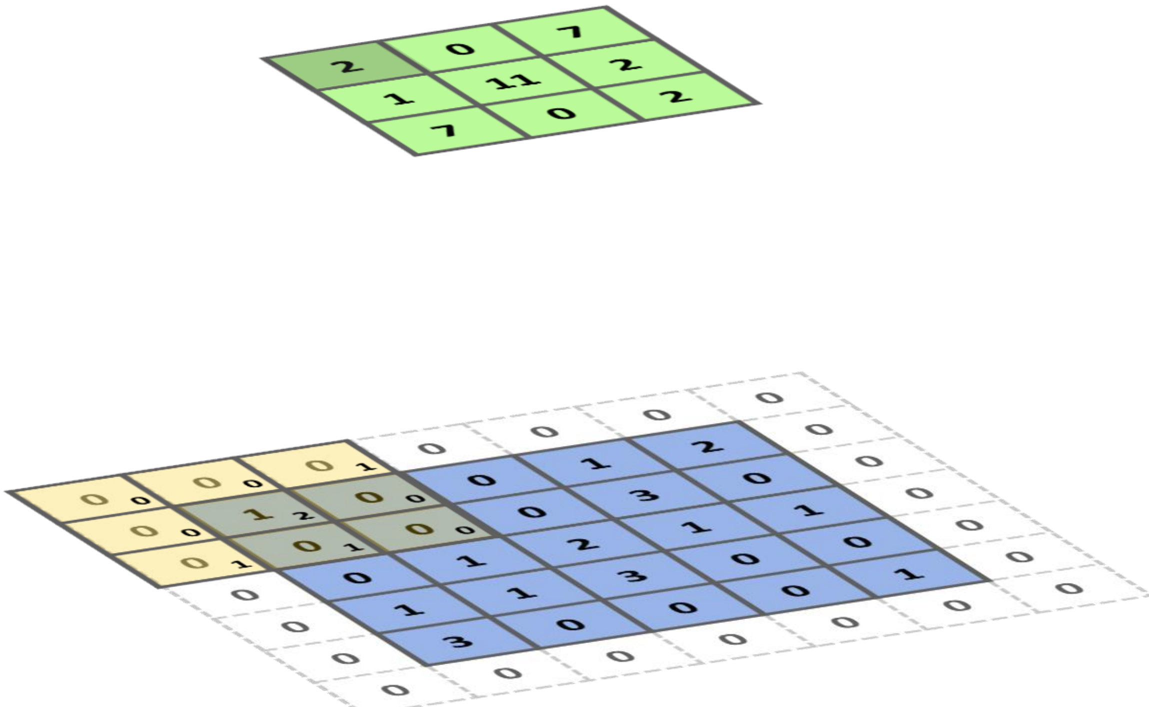
The effect of different strides

Stride: 1x1



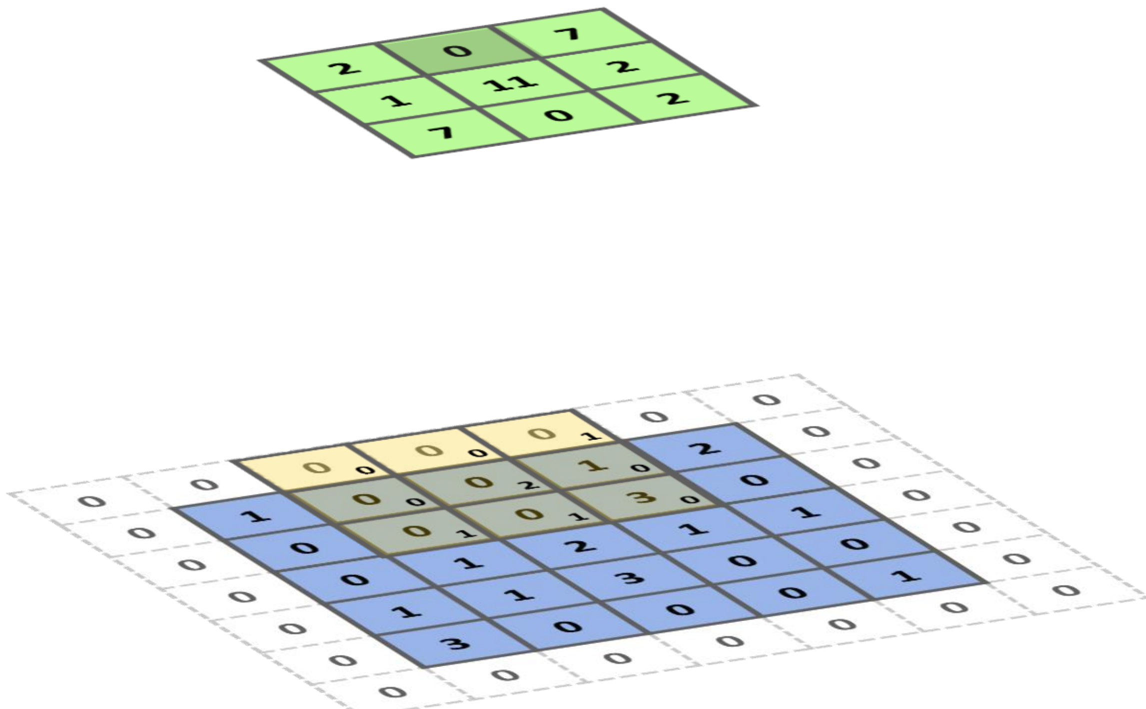
The effect of different strides

Stride: 2x2



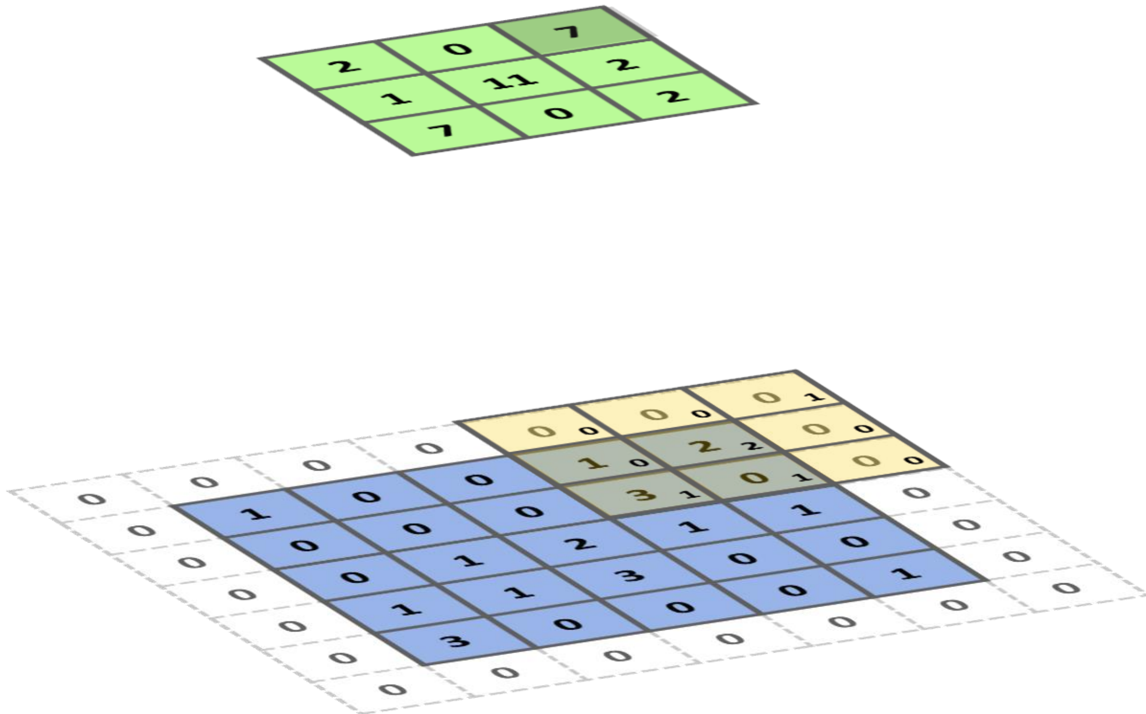
The effect of different strides

Stride: 2x2



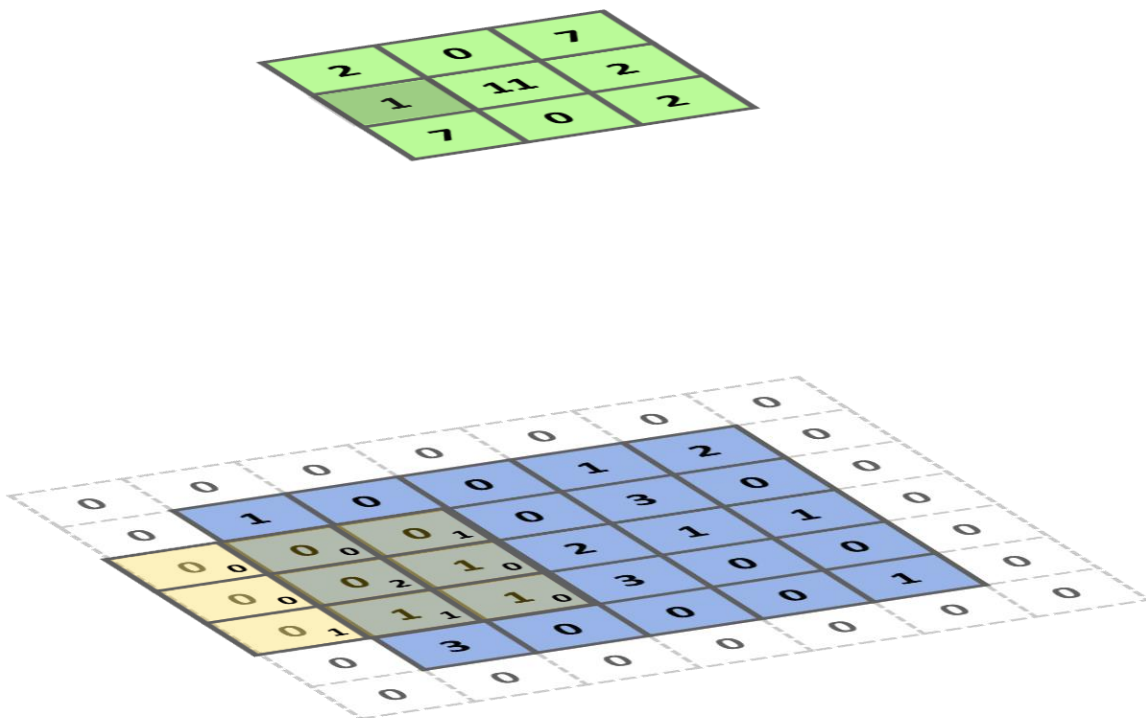
The effect of different strides

Stride: 2x2



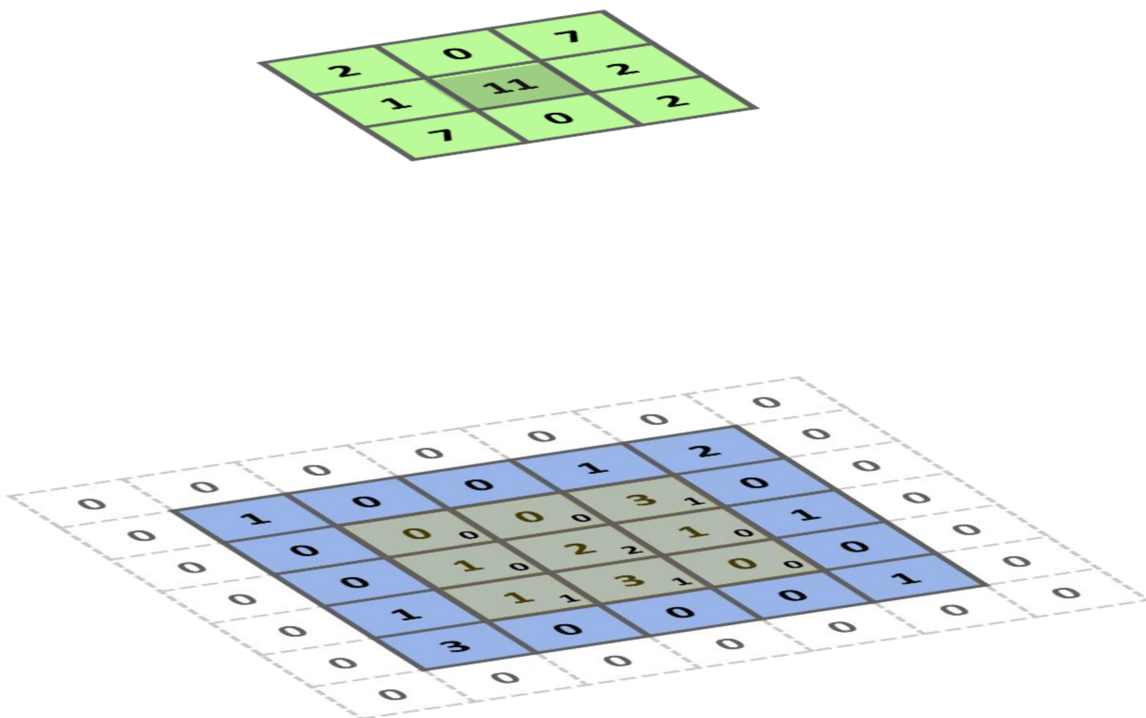
The effect of different strides

Stride: 2x2



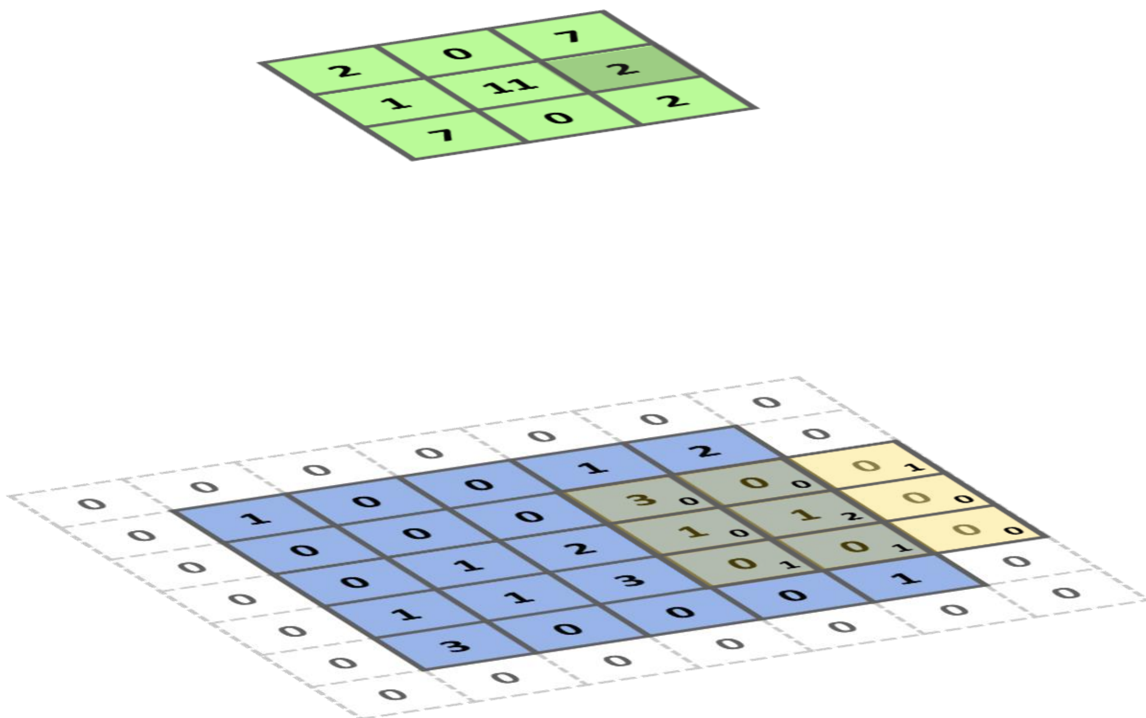
The effect of different strides

Stride: 2x2



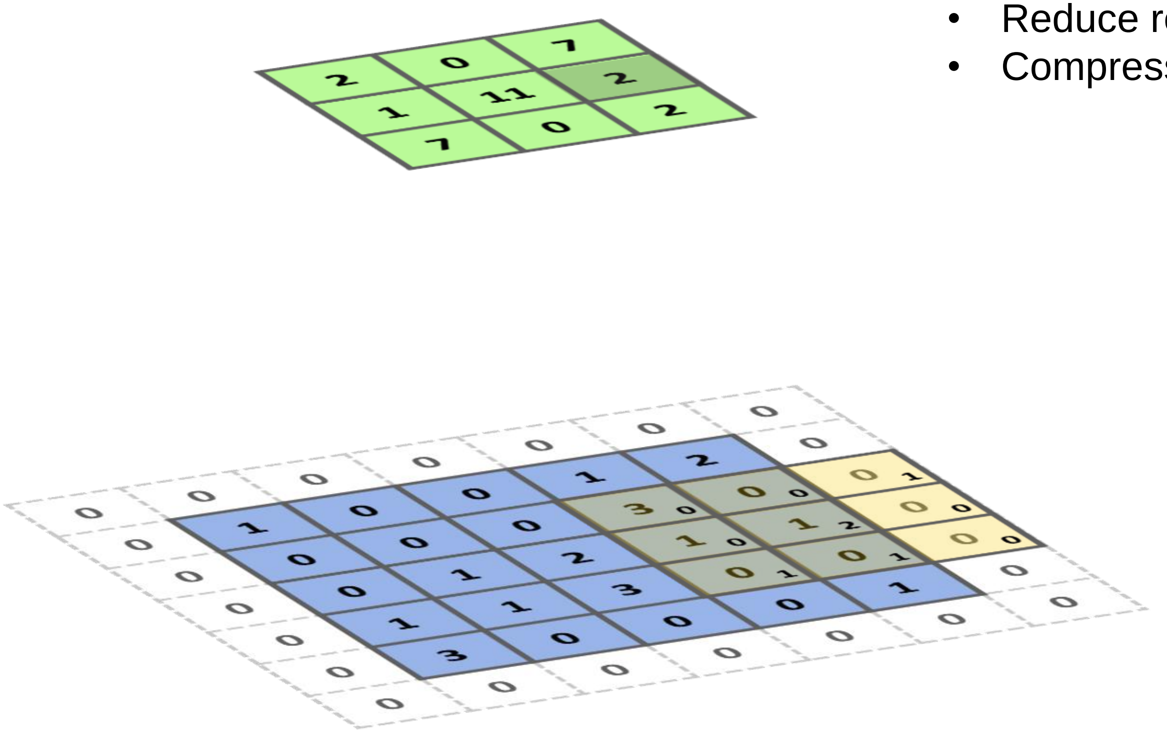
The effect of different strides

Stride: 2x2



The effect of different strides

Stride: 2x2



Why use stride > 1?

- Reduce redundancy
- Compress feature map

Summary of convolutions

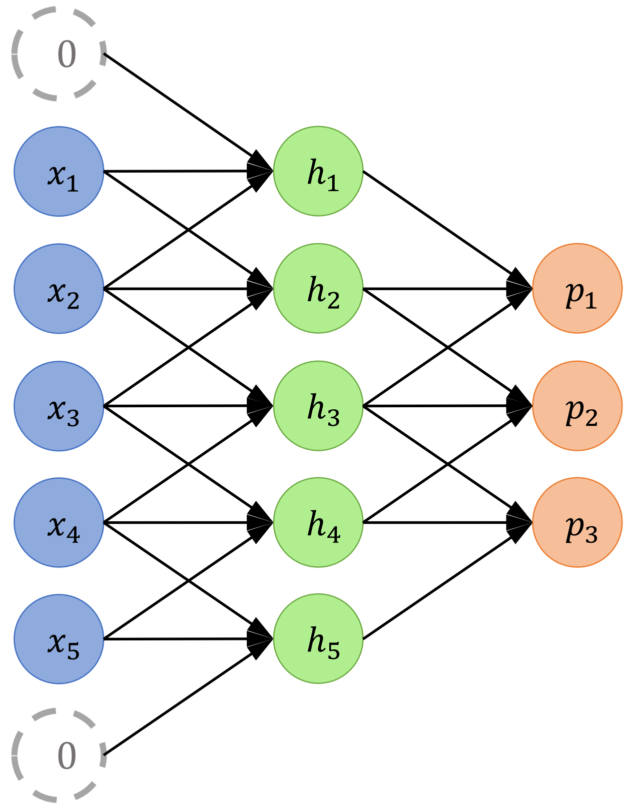
$n \times n$ image $\quad$ $f \times f$ filter

padding p $\quad$ stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

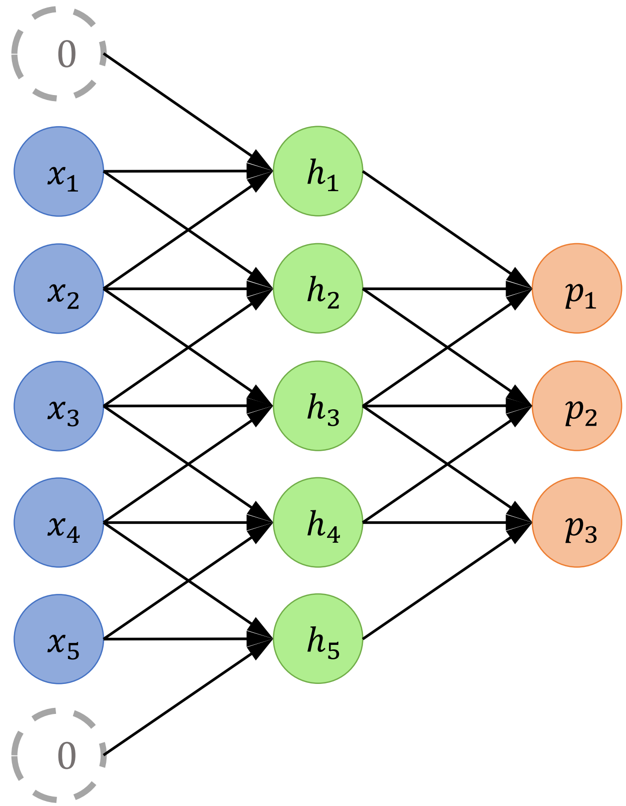
Pooling

Operation to aggregate or "summarize" sub-region of input.



Pooling

Operation to aggregate or "summarize" sub-region of input.



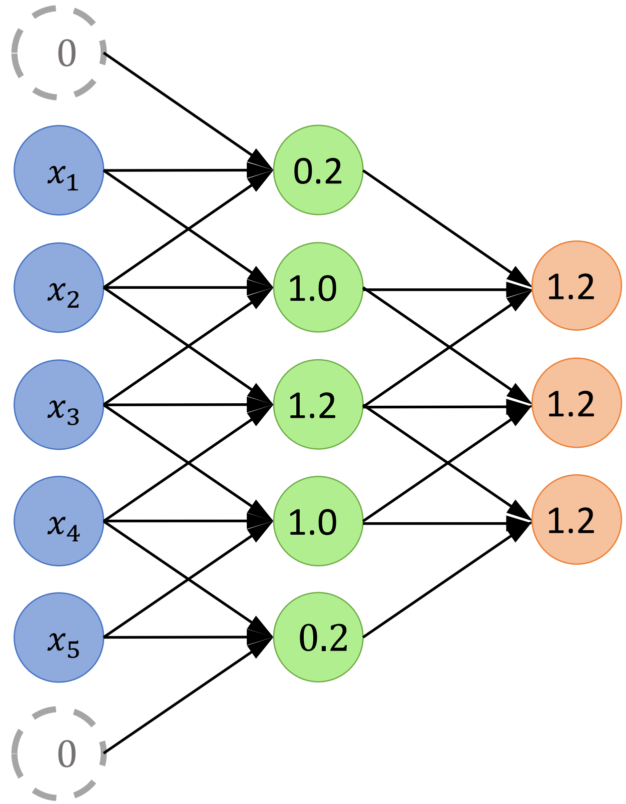
$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Pooling

Operation to aggregate or "summarize" sub-region of input.



$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

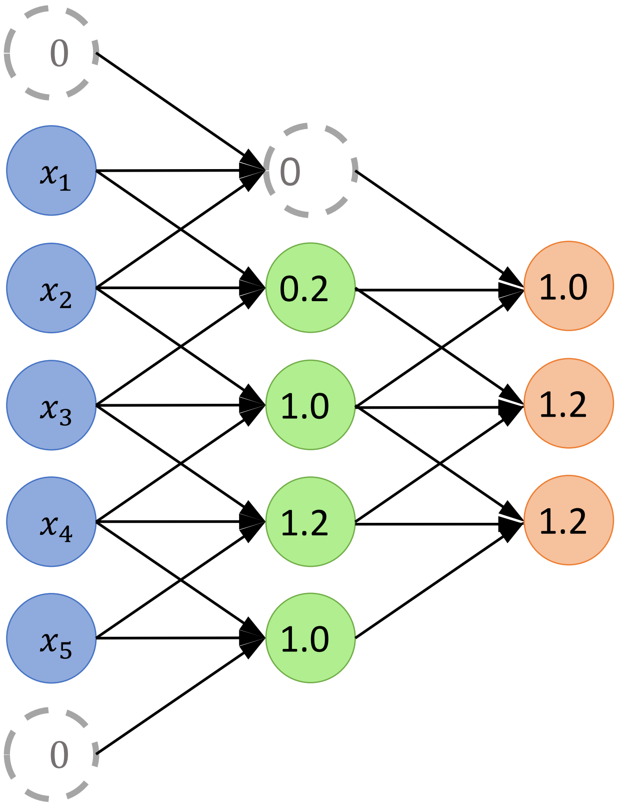
$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Pooling

Operation to aggregate or "summarize" sub-region of input.

Shift input by 1 position:

5/5 inputs change but only 1/3 pooled outputs



$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

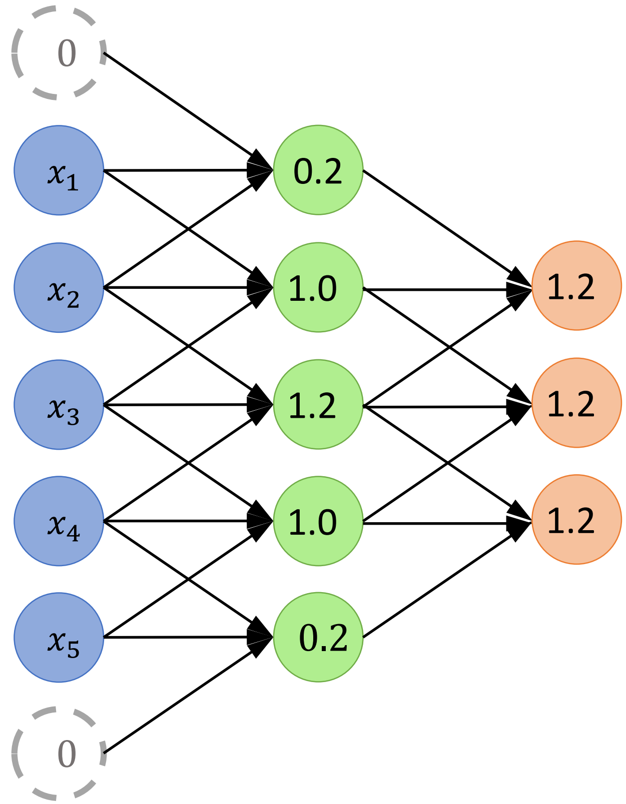
$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Pooling

Operation to aggregate or “summarize” sub-region of input.

Shift input by 1 position:

5/5 inputs change but only 1/3 pooled outputs



$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Pooling

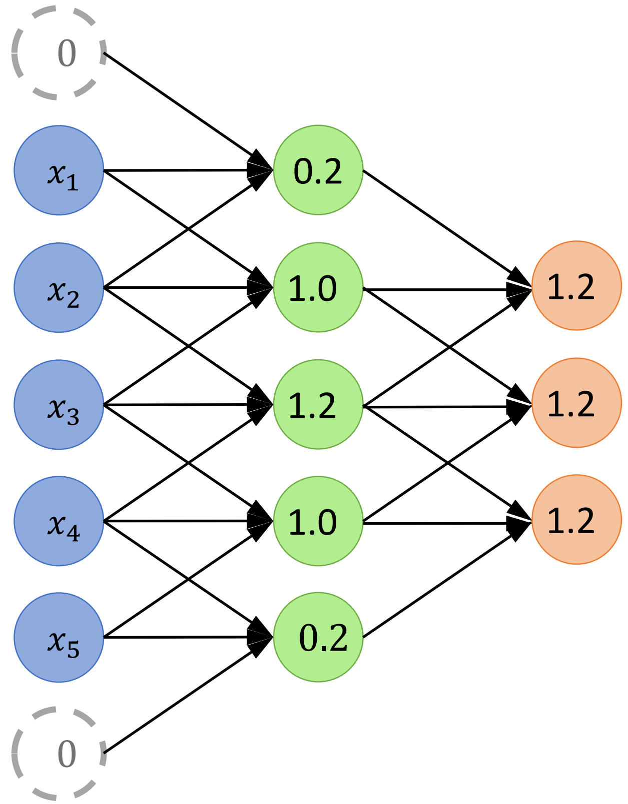
Operation to aggregate or "summarize" sub-region of input.

Shift input by 1 position:

5/5 inputs change but only 1/3 pooled outputs

Why is pooling useful?

- Adds **translation invariance** (useful when we care about "what" more than "where")
- Can be used to **compress signal** (useful to improve computational efficiency)



$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Pooling

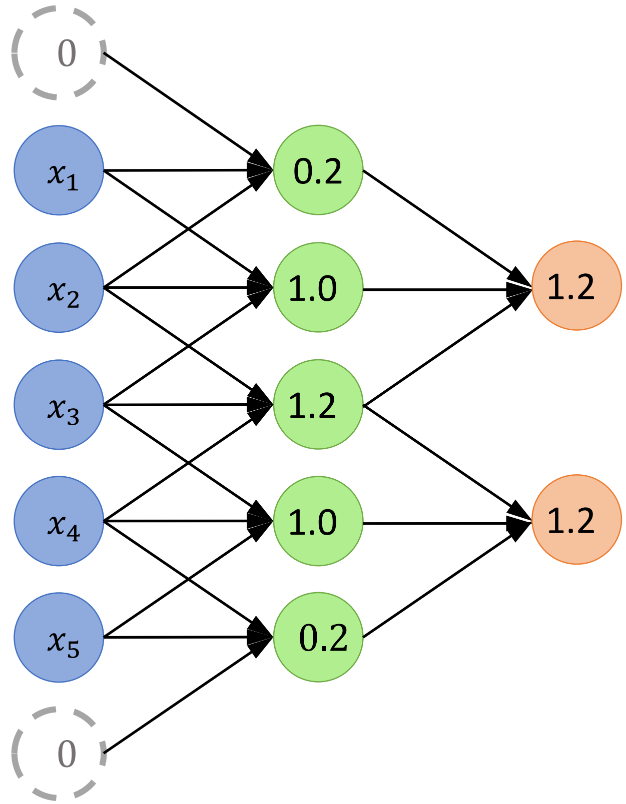
Operation to aggregate or "summarize" sub-region of input.

Shift input by 1 position:

5/5 inputs change but only 1/3 pooled outputs

Why is pooling useful?

- Adds **translation invariance** (useful when we care about "what" more than "where")
- Can be used to **compress signal** (useful to improve computational efficiency)



$$p = f(\boldsymbol{a})$$

$$f_{max}(\boldsymbol{a}) = \max_i(a_i)$$

$$f_{avg}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i$$

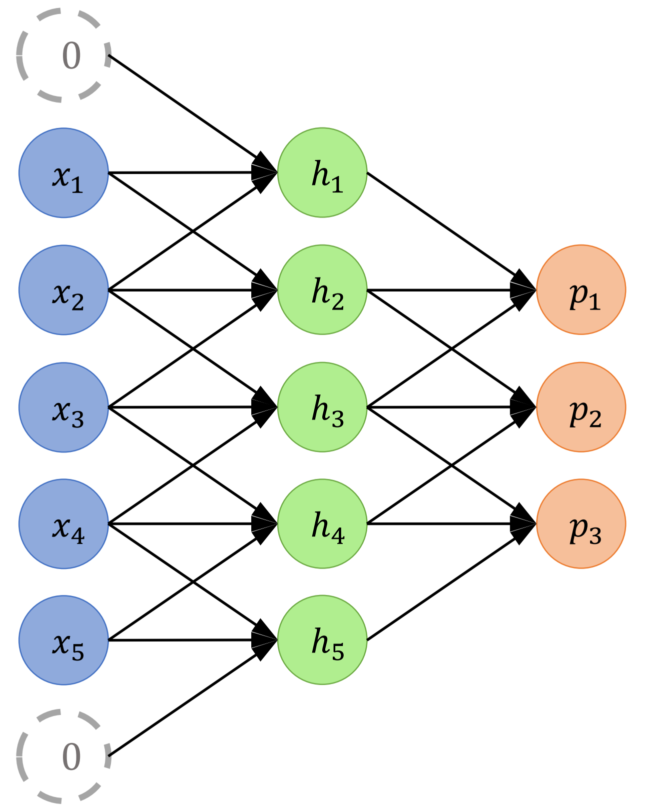
Receptive fields

Borrowed terminology from neuroscience:

⇒ stimulus region that impacts a neuron's firing

For neural networks:

⇒ Region of input signal that impacts node output

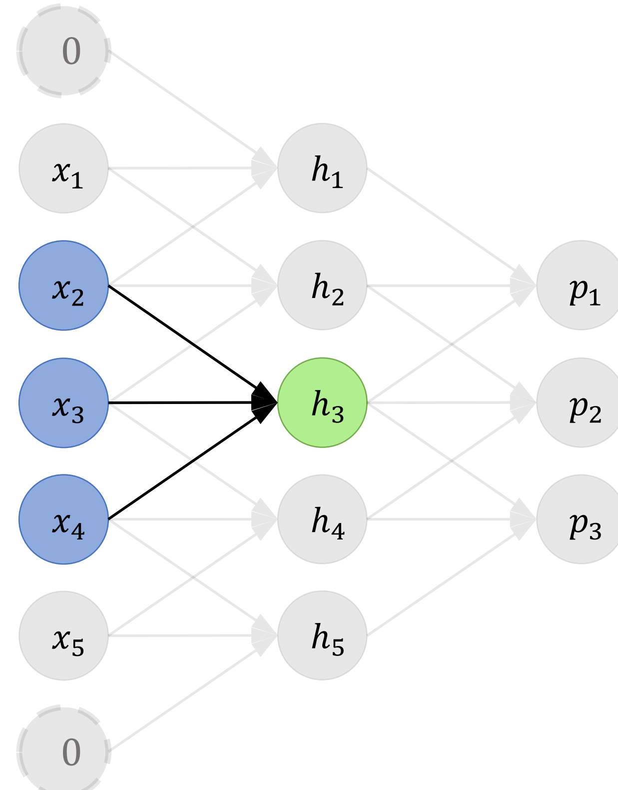


Borrowed terminology from neuroscience:

⇒ stimulus region that impacts neuronal firing

For neural networks:

⇒ Region of input signal that impacts node output

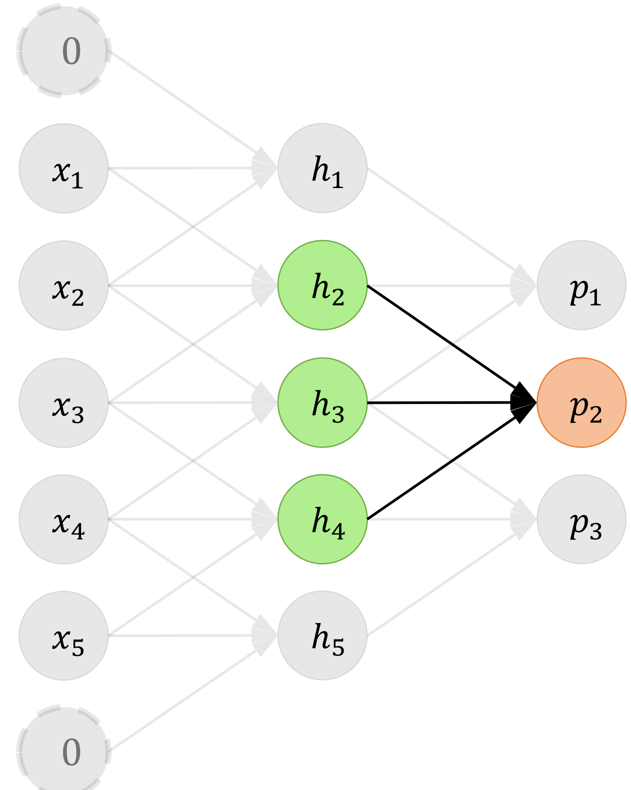


Borrowed terminology from neuroscience:

⇒ stimulus region that impacts neuronal firing

For neural networks:

⇒ Region of input signal that impacts node output



Receptive fields

Borrowed terminology from neuroscience:

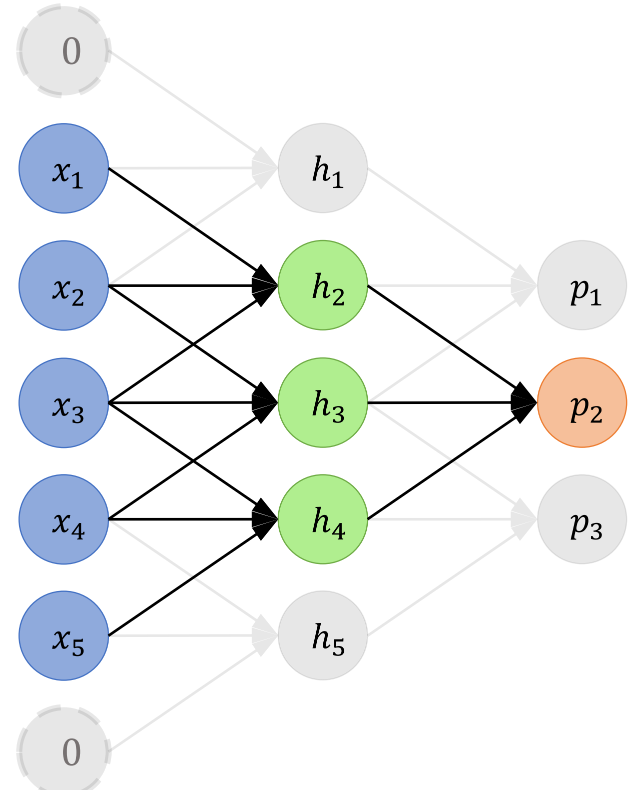
⇒ stimulus region that impacts neuronal firing

For neural networks:

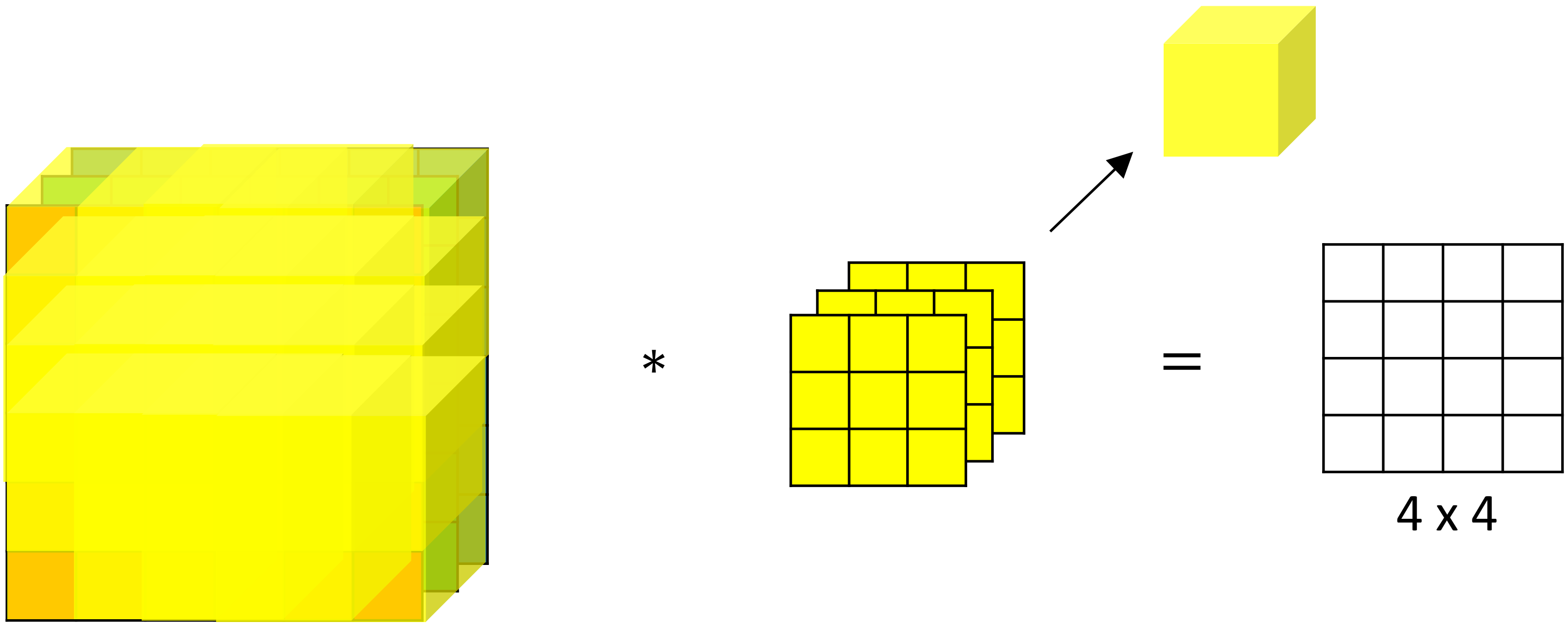
⇒ Region of input signal that impacts node output

Effective field size:

⇒ Region of **network** input signal that impacts node output



Convolutions on RGB image



Multiple filters