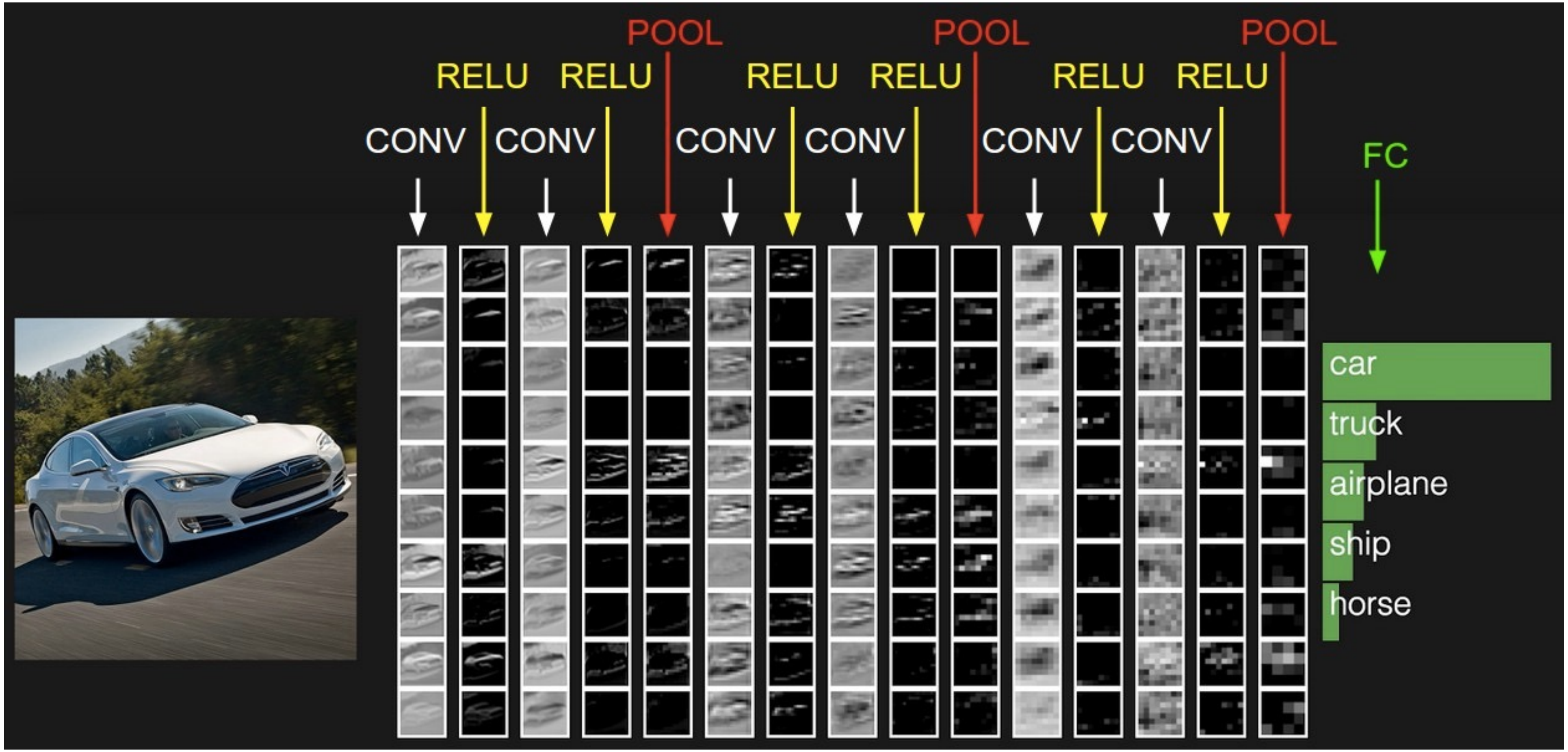
Convolutional neural networks

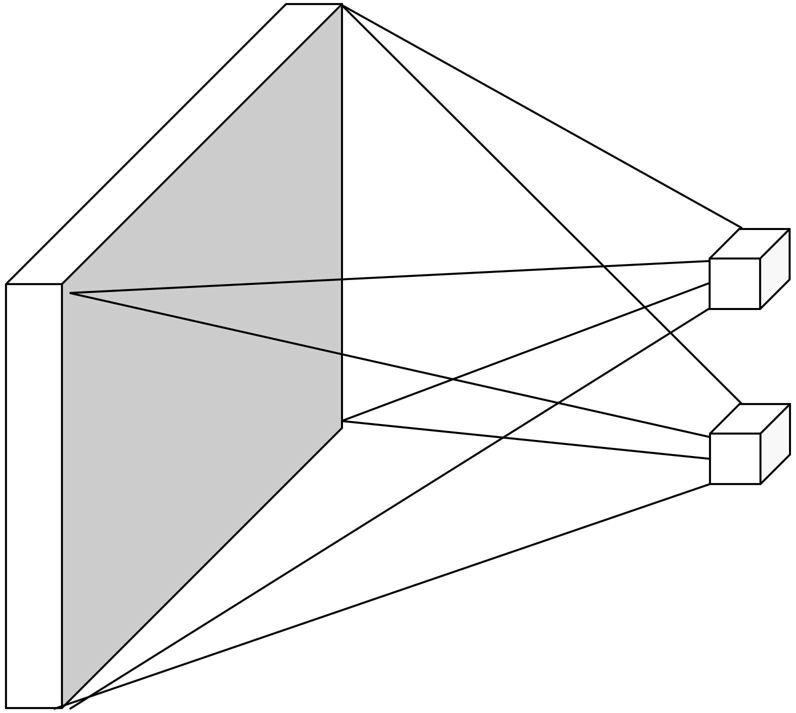


Many slides from Rob Fergus, Andrej Karpathy

Outline

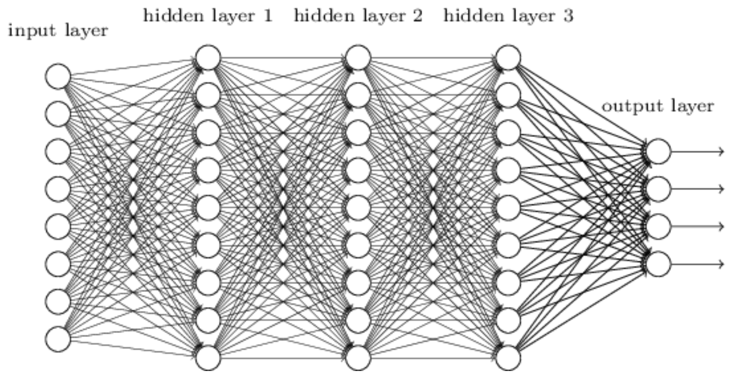
- Basic convolutional layer
  - Backward pass
- Max pooling layer

Let’s design a neural network for images



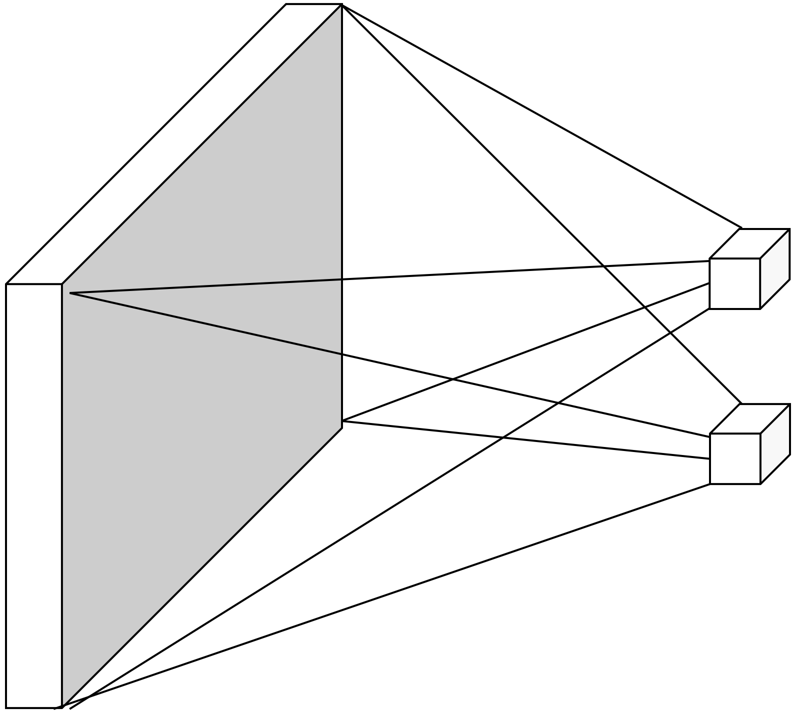
image

Fully connected layer



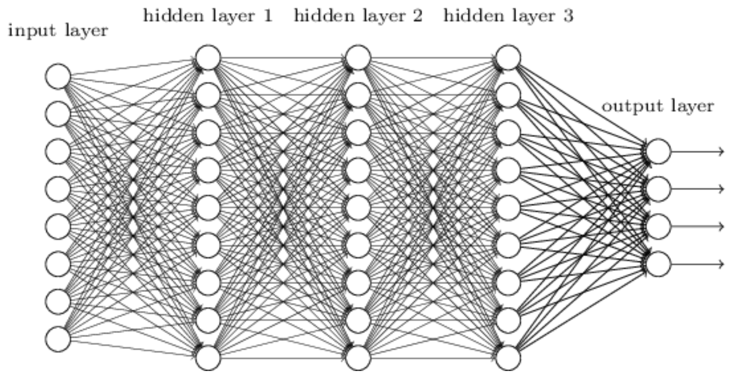
- This kind of design is known as *multi-layer perceptron* (MLP)

Let's design a neural network for images



image

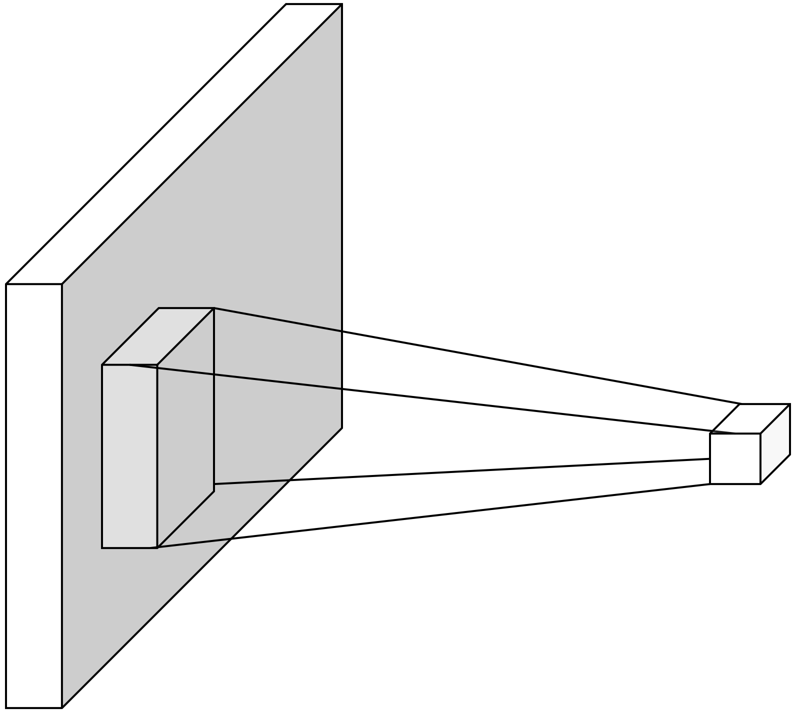
Fully connected layer



- This kind of design is known as *multi-layer perceptron* (MLP)
- What is wrong with this?

Recall: MLP as bank of whole-image templates

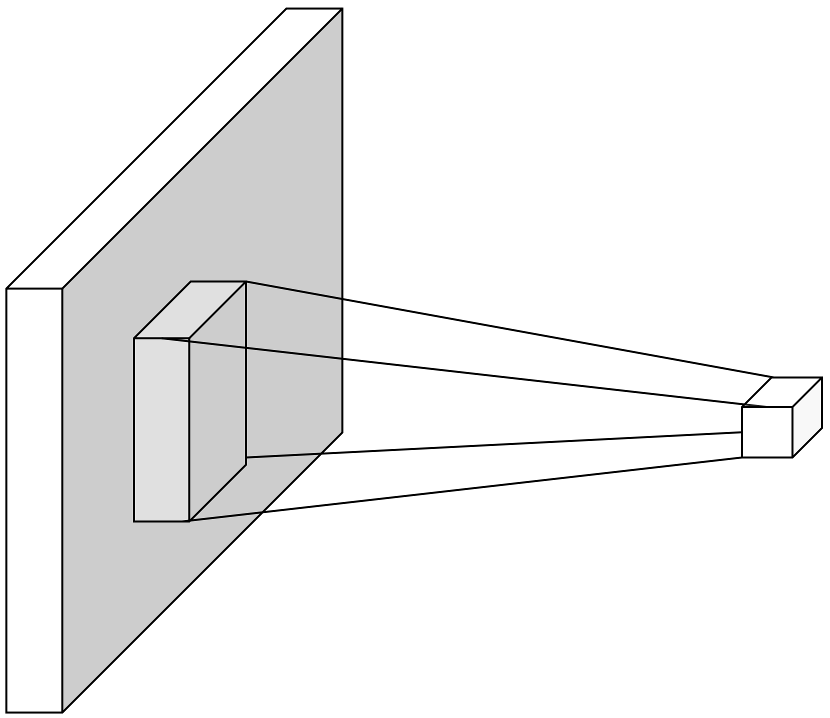
Convolutional architecture



image

- Let’s limit the *receptive fields* of units, tile them over the input image, and share their weights

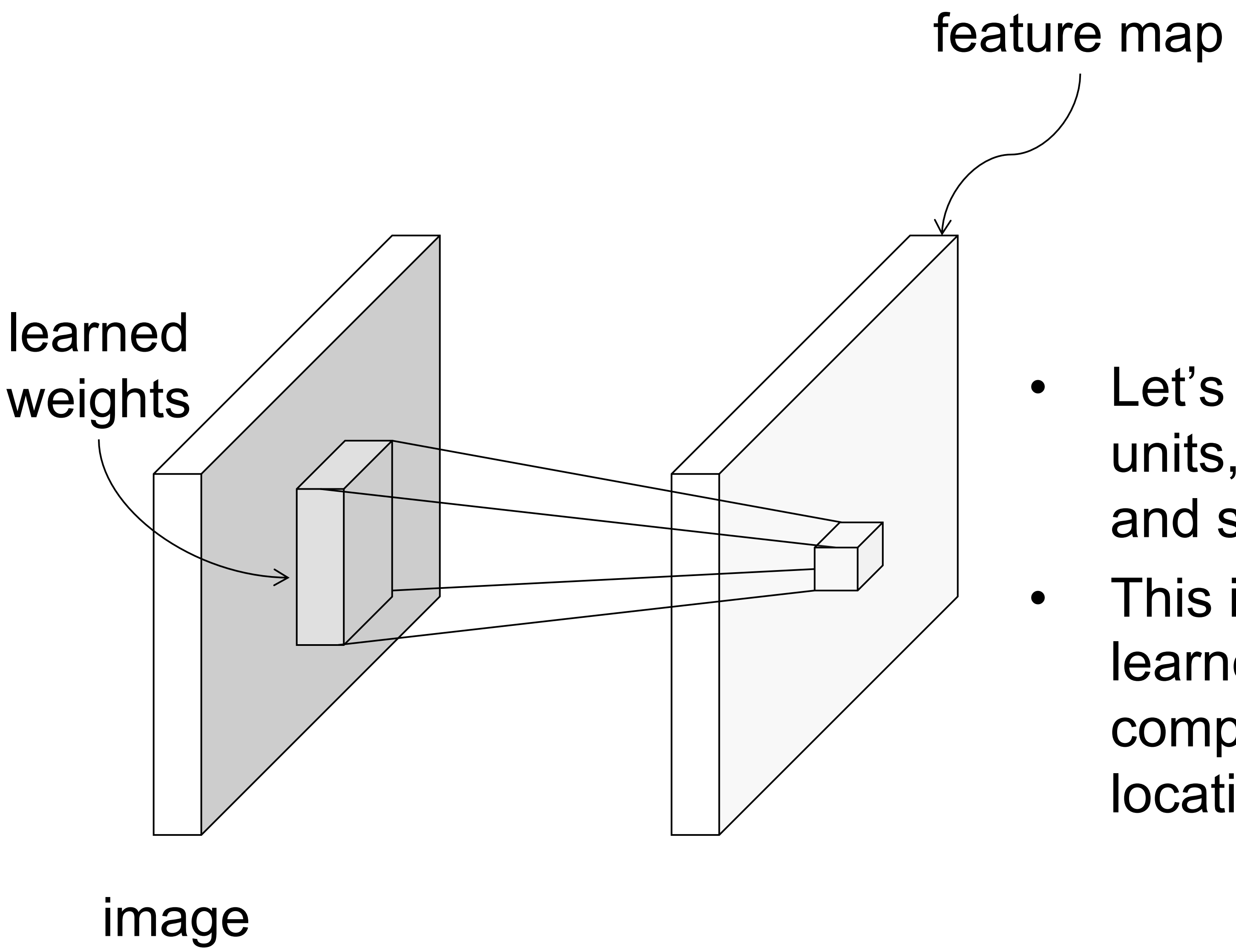
Convolutional architecture



image

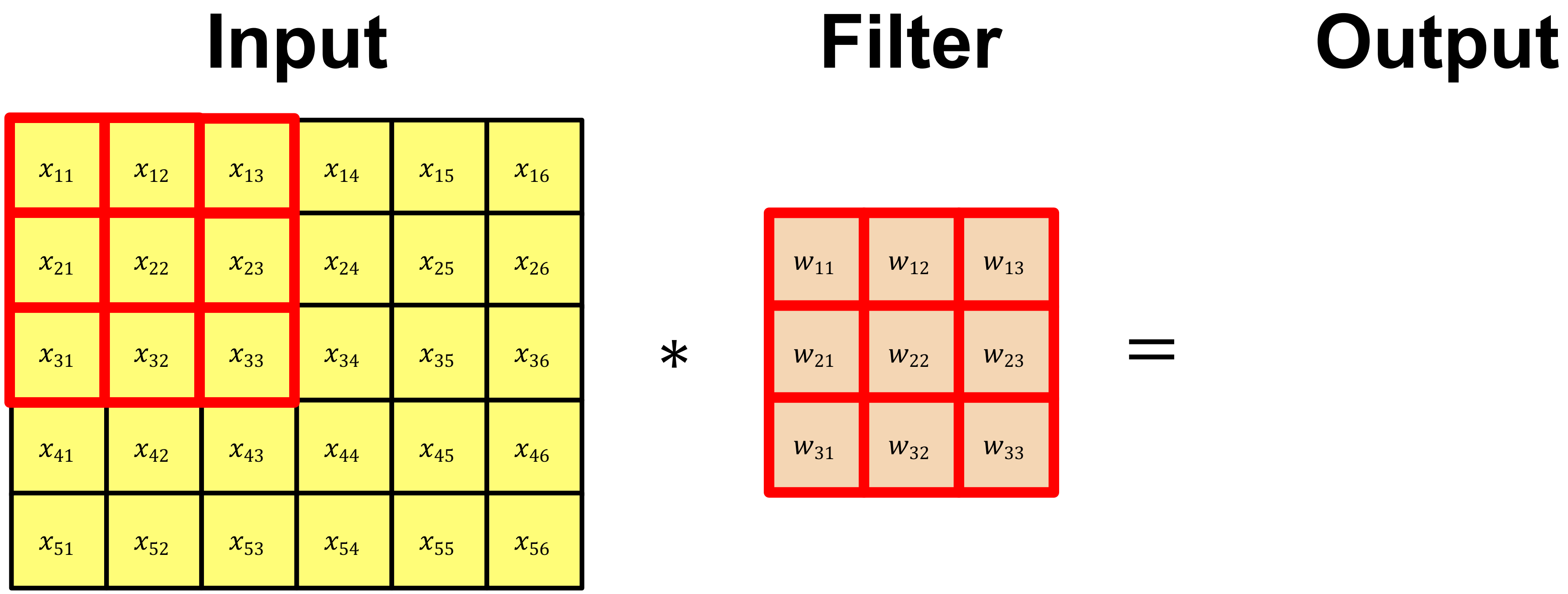
- Let’s limit the *receptive fields* of units, tile them over the input image, and share their weights

Convolutional architecture



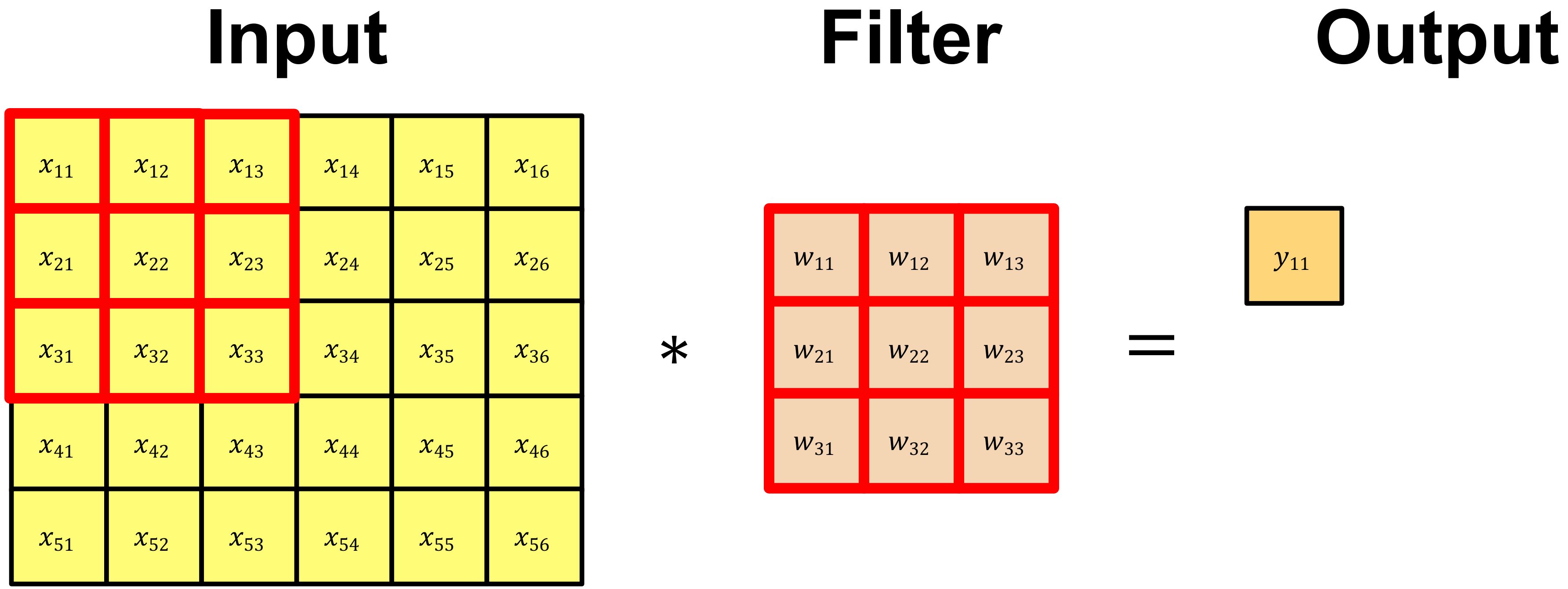
- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights
- This is equivalent to sliding the learned filter over the image, computing dot products at every location

Convolution example



Adapted from [D. Fouhey and J. Johnson]

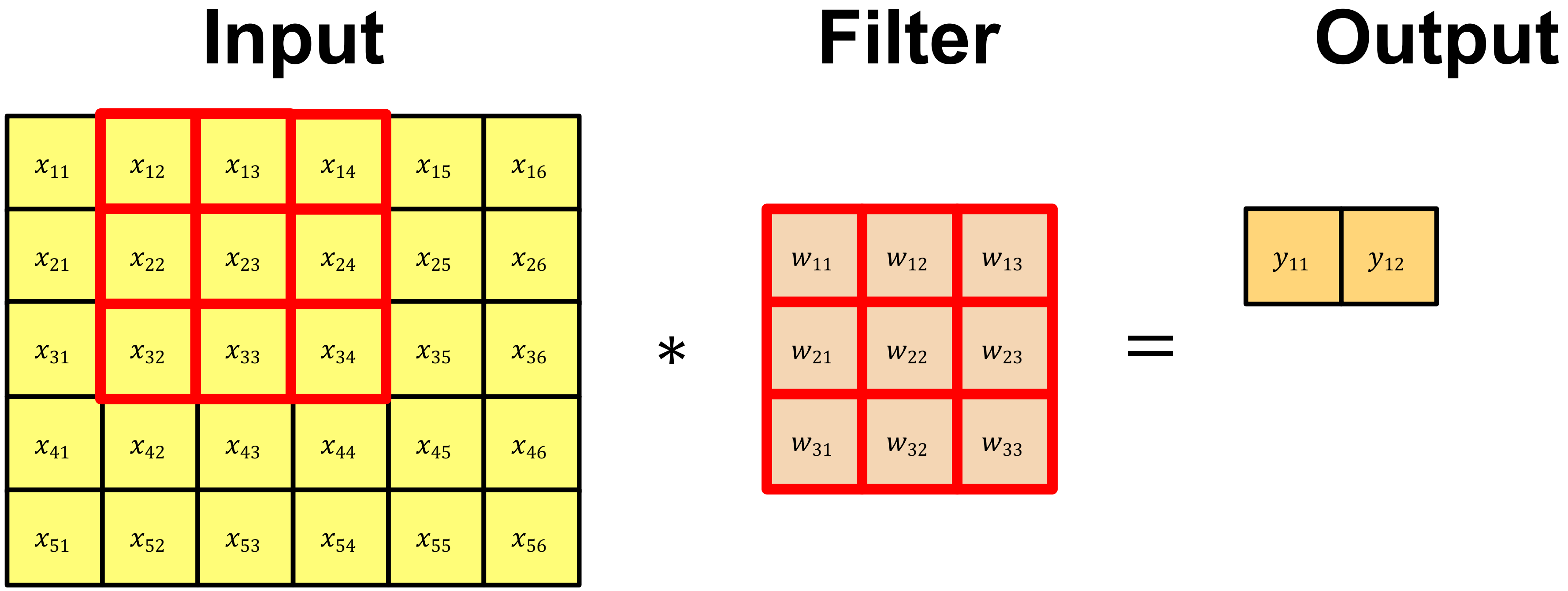
Convolution example



$$y_{11} = x_{11} \cdot w_{11} + x_{12} \cdot w_{12} + x_{13} \cdot w_{13} + \ldots + x_{33} \cdot w_{33}$$

Adapted from D. Fouhey and J. Johnson

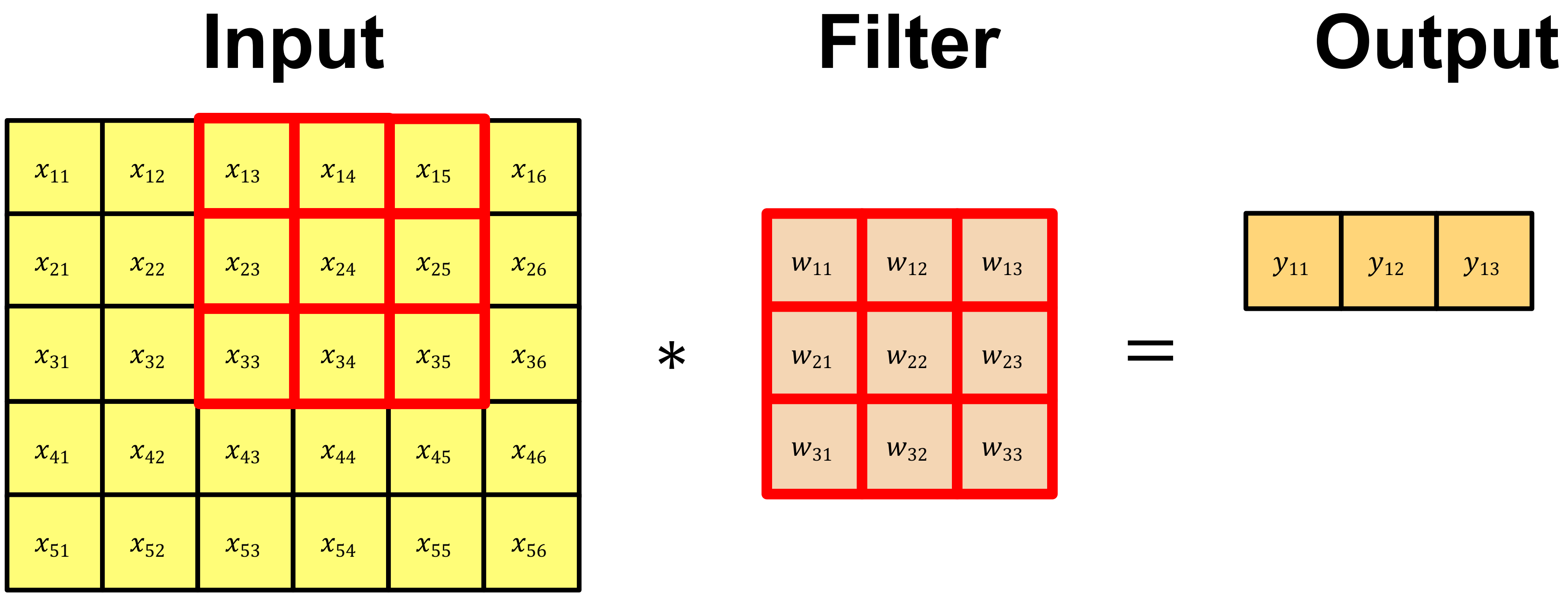
Convolution example



$$y_{12} = x_{12} \cdot w_{11} + x_{13} \cdot w_{12} + x_{14} \cdot w_{13} + \ldots + x_{34} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

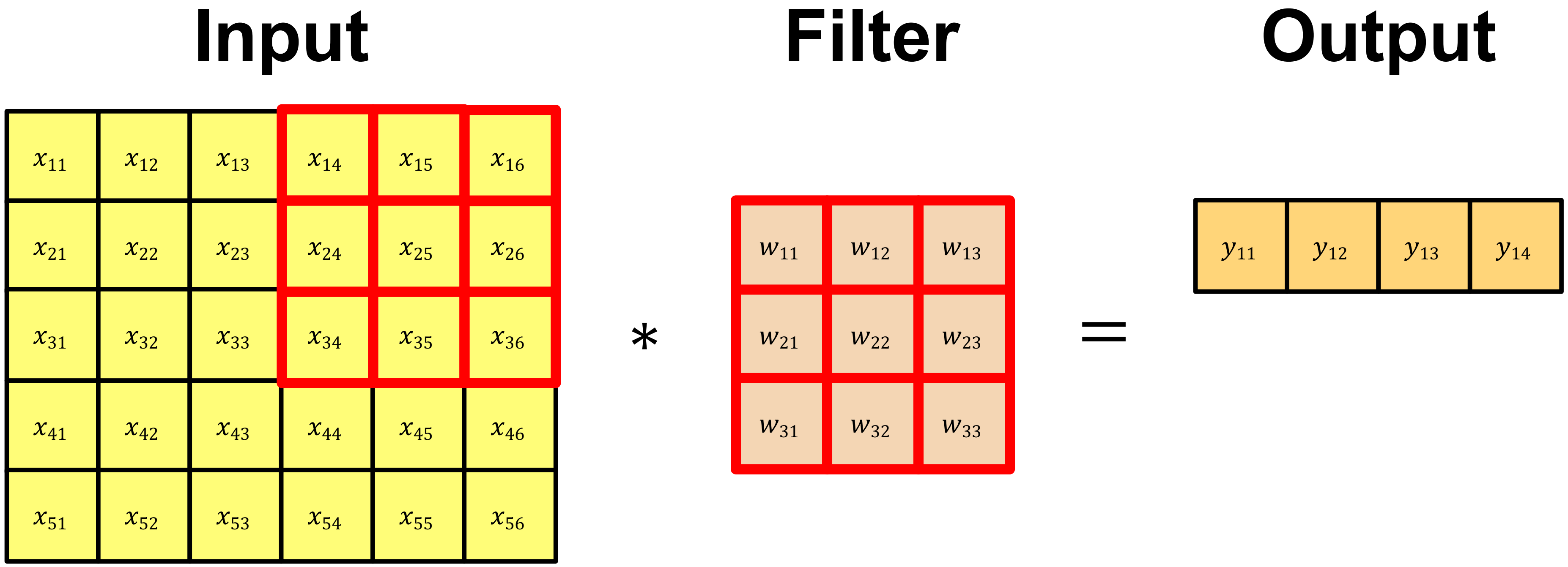
Convolution example



$$y_{13} = x_{13} \cdot w_{11} + x_{14} \cdot w_{12} + x_{15} \cdot w_{13} + \ldots + x_{35} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

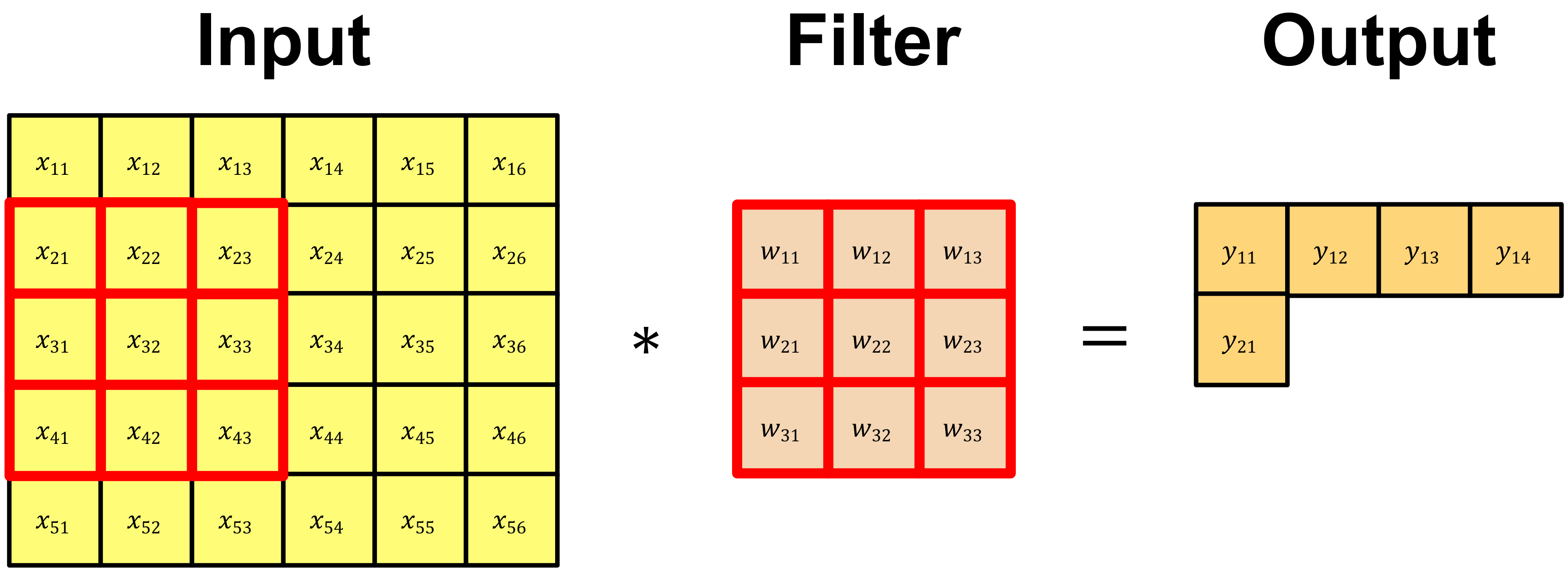
Convolution example



$$y_{14} = x_{14} \cdot w_{11} + x_{15} \cdot w_{12} + x_{16} \cdot w_{13} + \ldots + x_{36} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

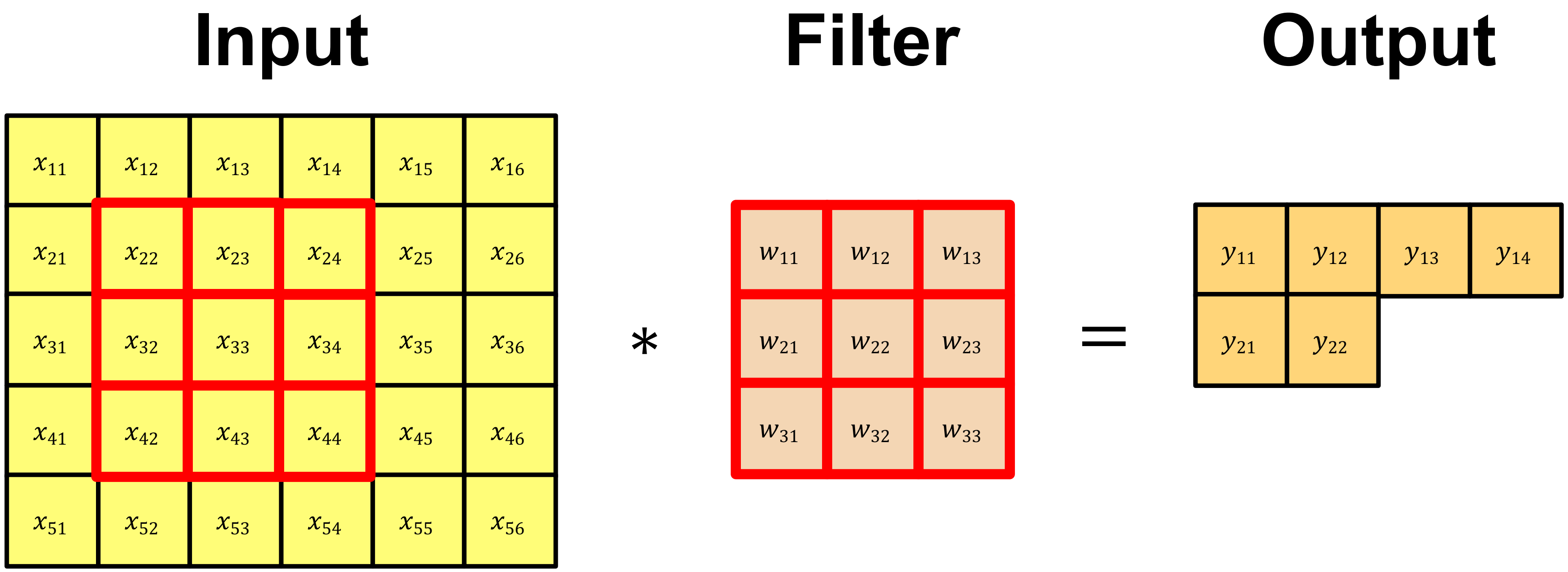
Convolution example



$$y_{21} = x_{21} \cdot w_{11} + x_{22} \cdot w_{12} + x_{23} \cdot w_{13} + \ldots + x_{43} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

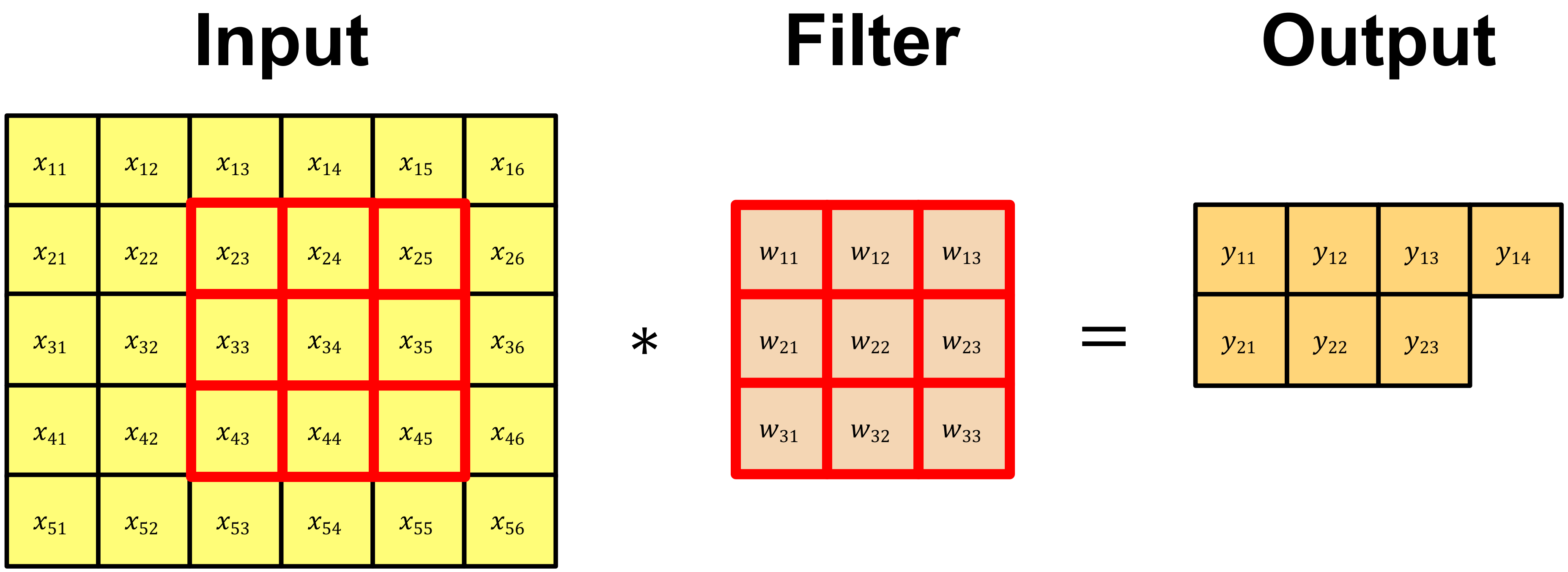
Convolution example



$$y_{22} = x_{22} \cdot w_{11} + x_{23} \cdot w_{12} + x_{24} \cdot w_{13} + \ldots + x_{44} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

Convolution example



$$y_{23} = x_{23} \cdot w_{11} + x_{24} \cdot w_{12} + x_{25} \cdot w_{13} + \ldots + x_{45} \cdot w_{33}$$

Adapted from [D. Fouhey and J. Johnson]

Convolution example

Input

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}

$*$

Filter

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

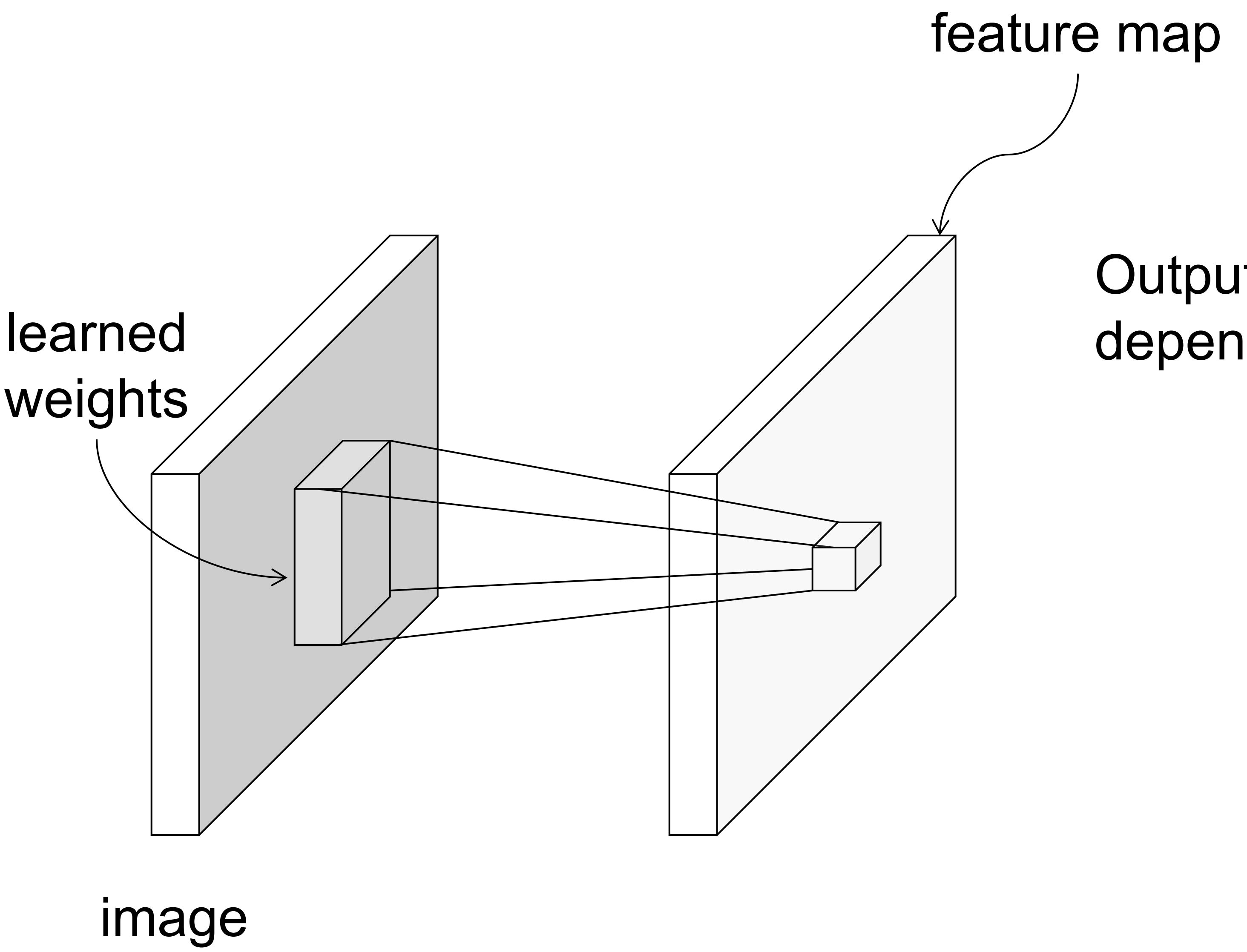
$=$

Output

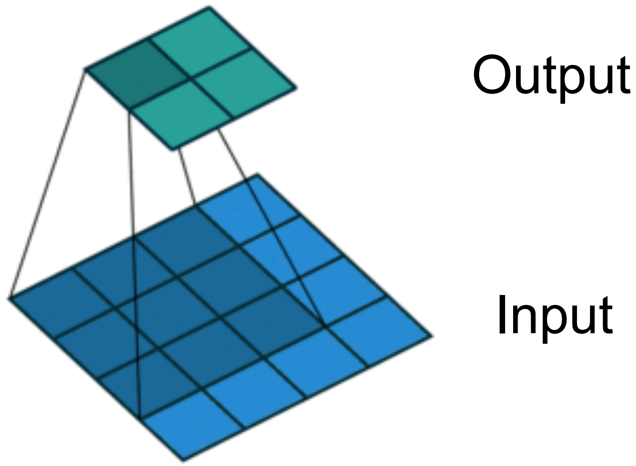
y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}

Adapted from [D. Fouhey and J. Johnson]

Convolutional architecture



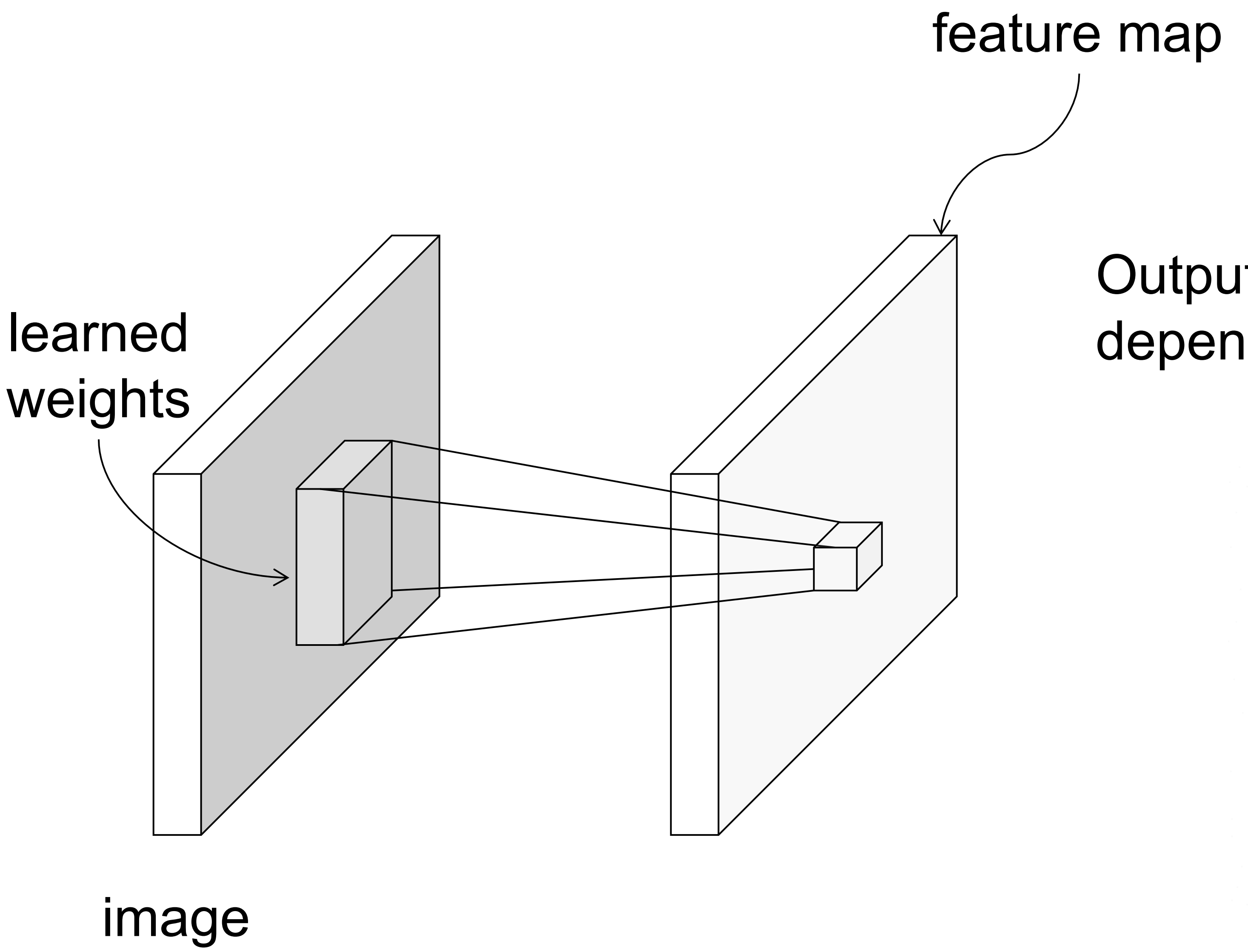
Output feature map resolution depends on *padding* and *stride*



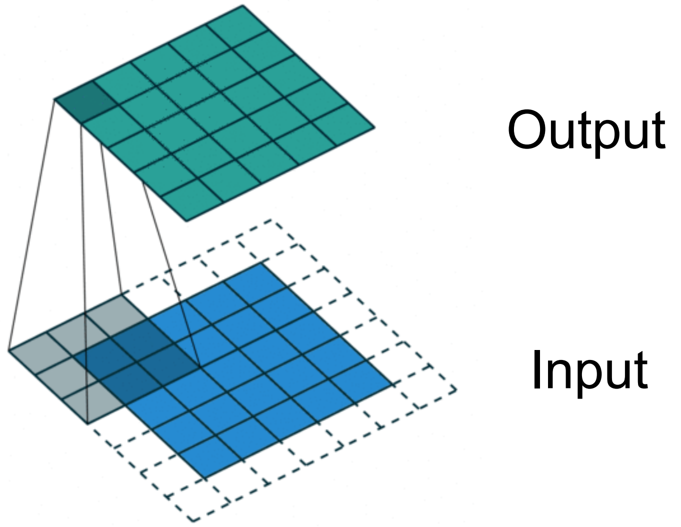
No padding, stride 1

[Animation source]

Convolutional architecture



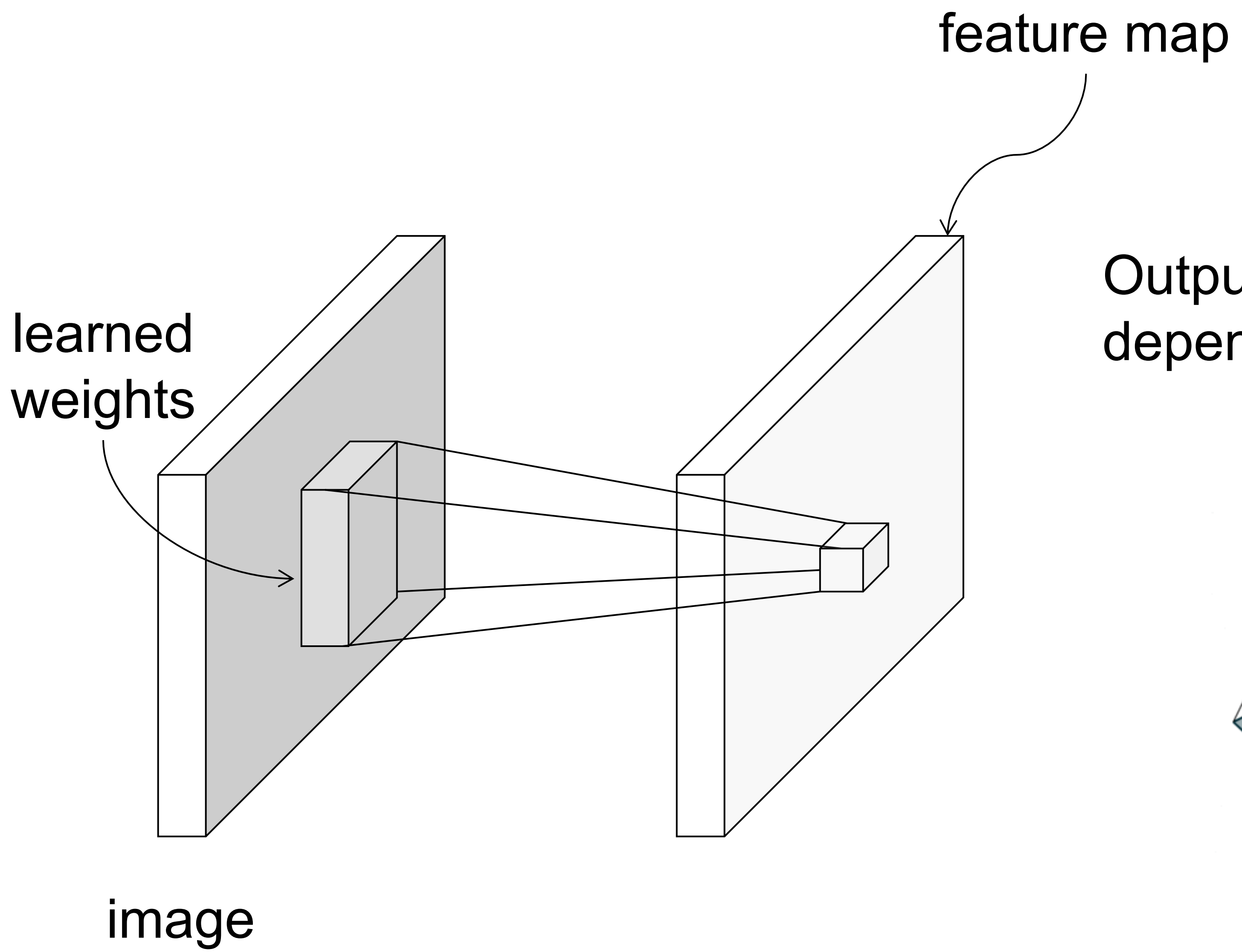
Output feature map resolution depends on *padding* and *stride*



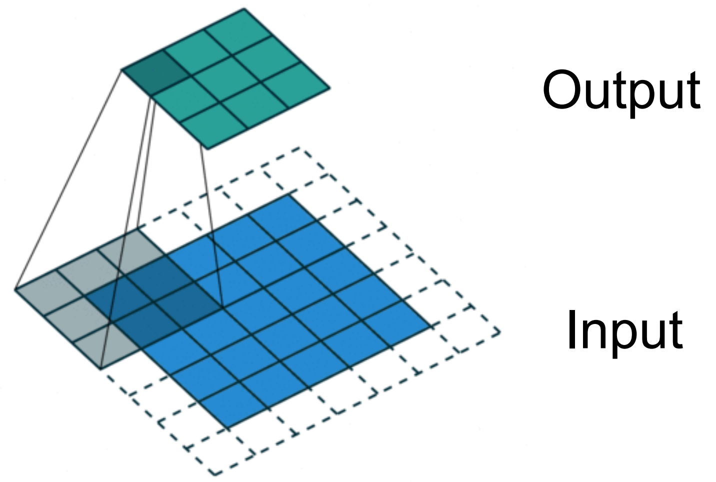
With padding, stride 1

[Animation source]

Convolutional architecture

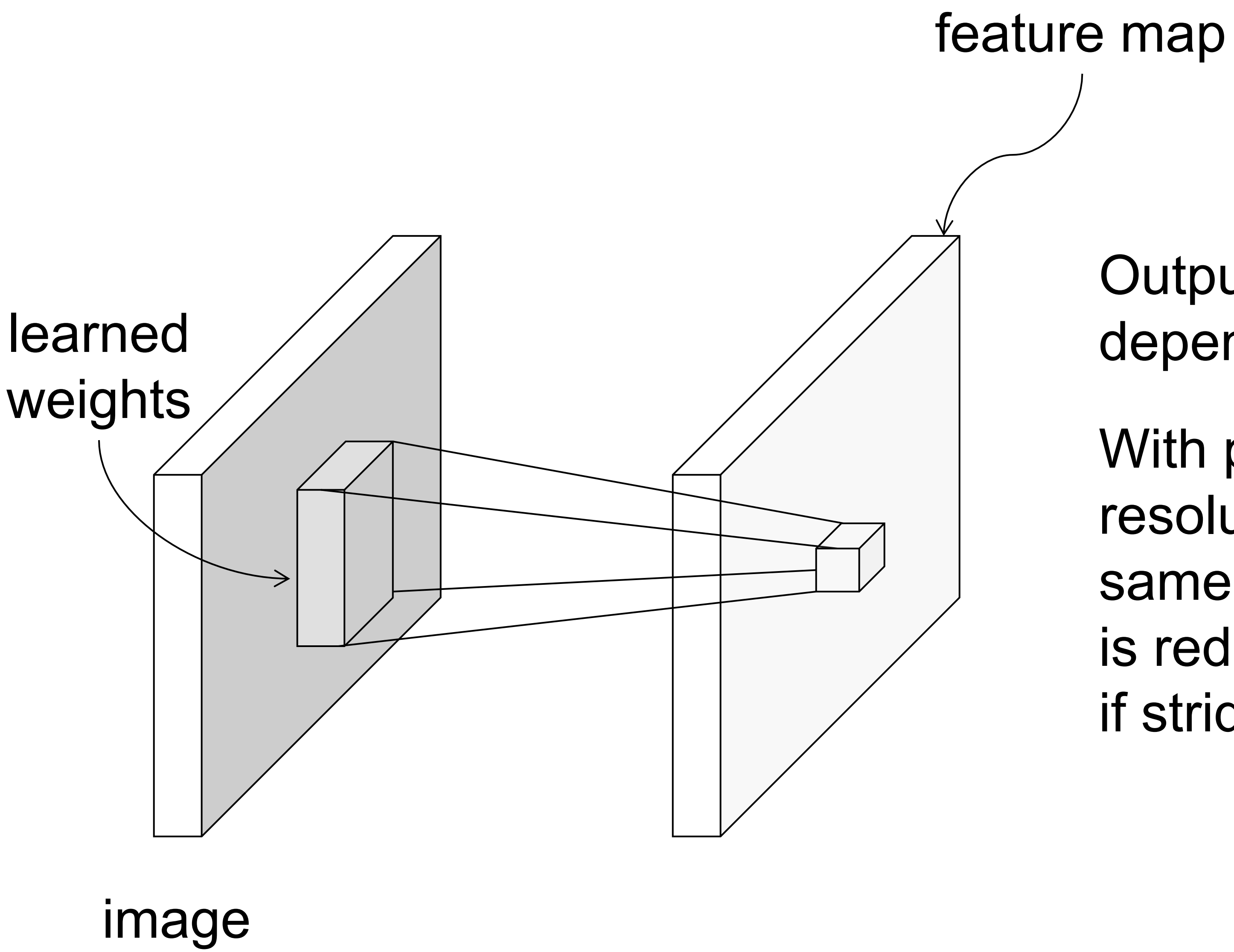


Output feature map resolution depends on *padding* and *stride*



With padding, stride 2

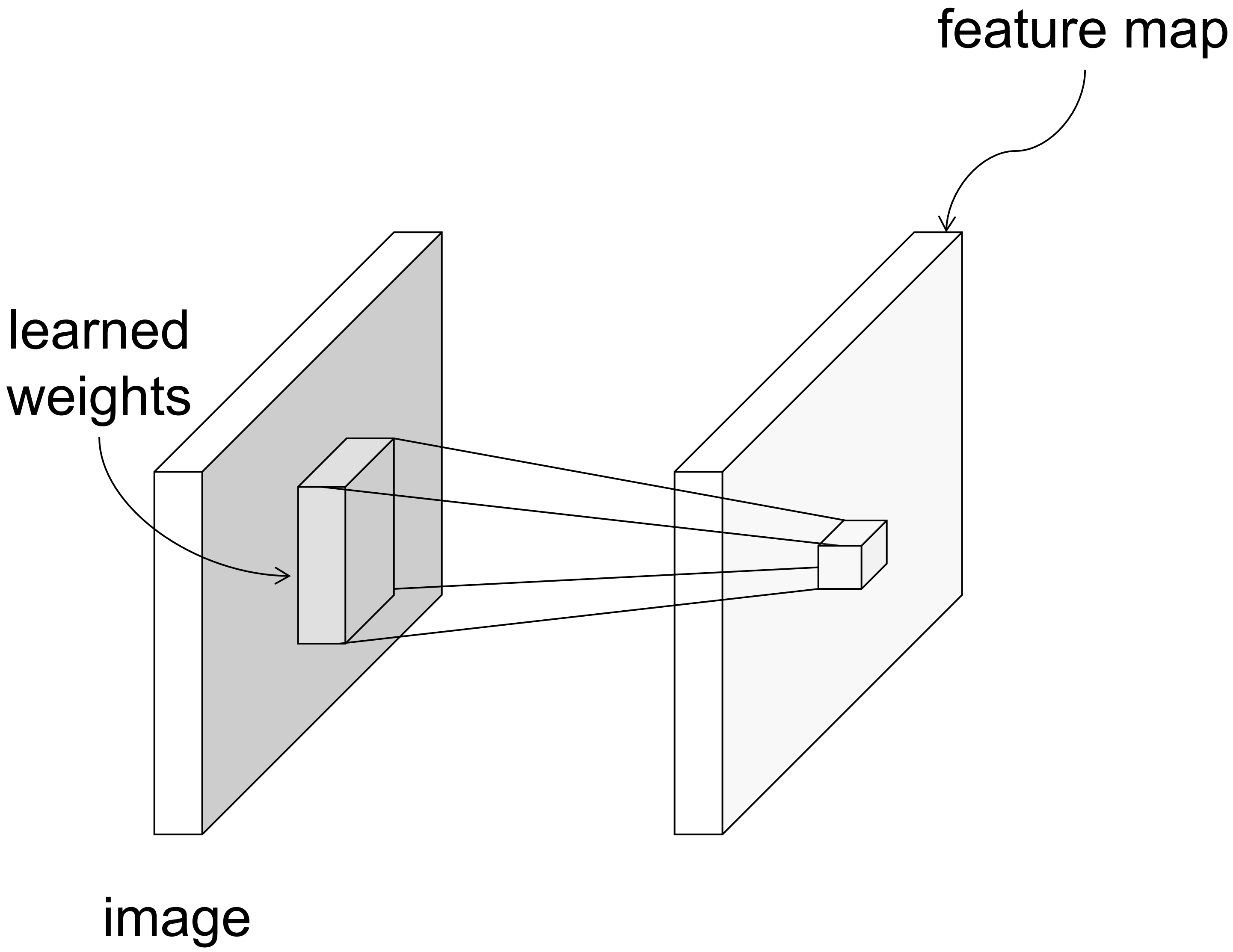
Convolutional architecture



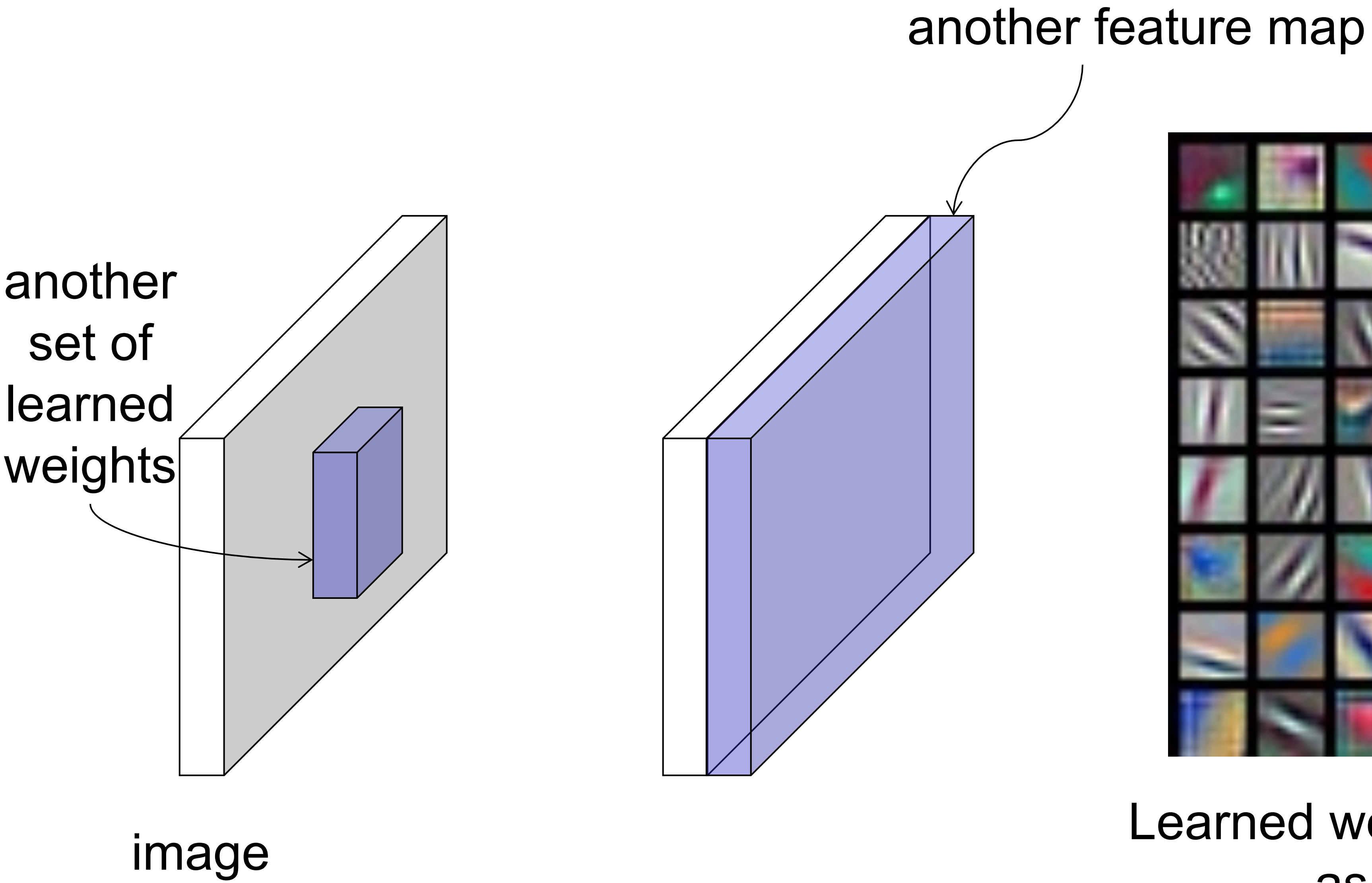
Output feature map resolution depends on *padding* and *stride*

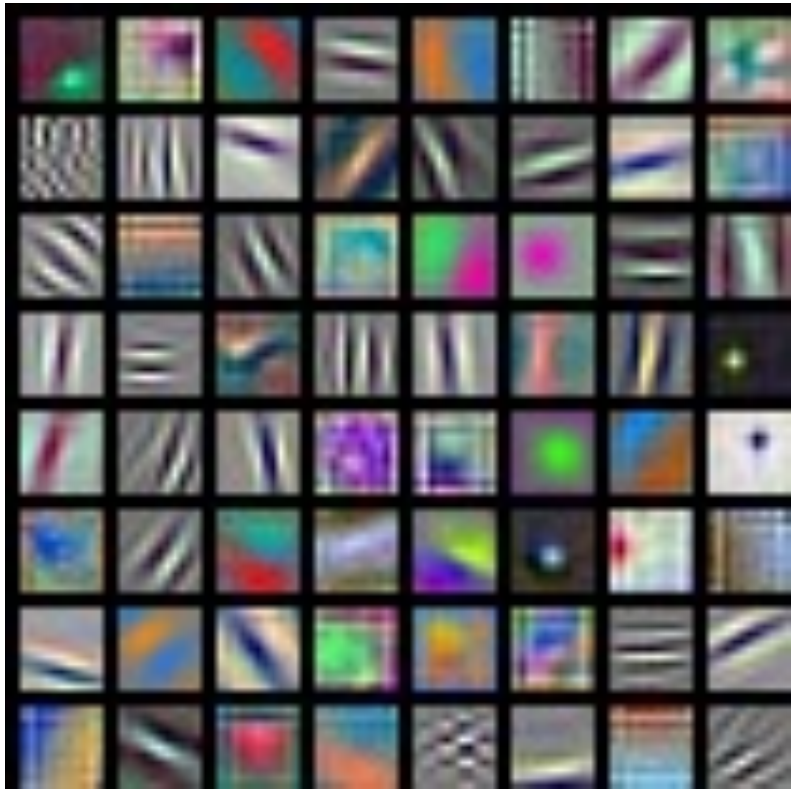
With padding, spatial resolution remains the same if stride of 1 is used, is reduced by factor of $1/S$ if stride of S is used

Convolutional architecture



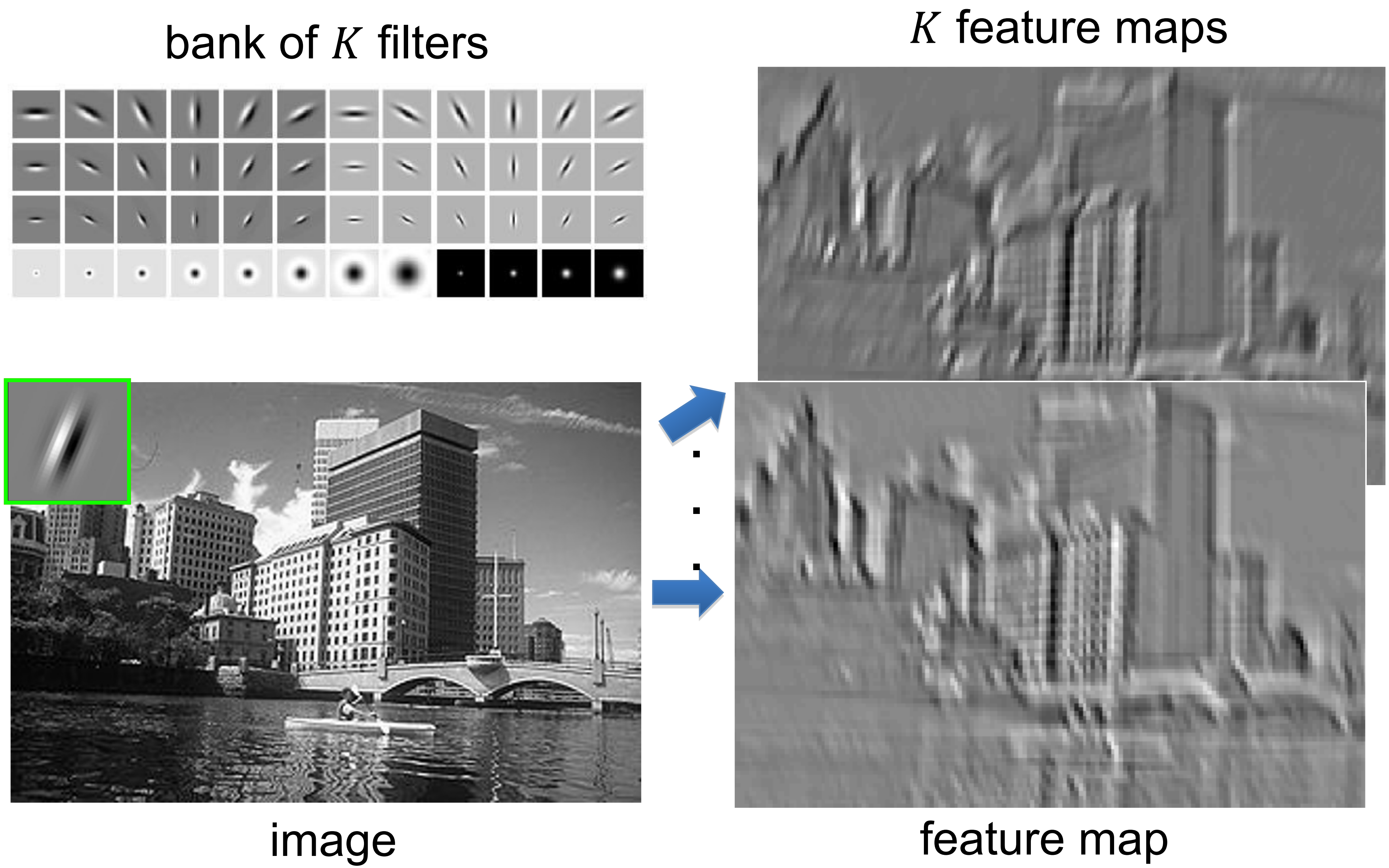
Convolutional architecture



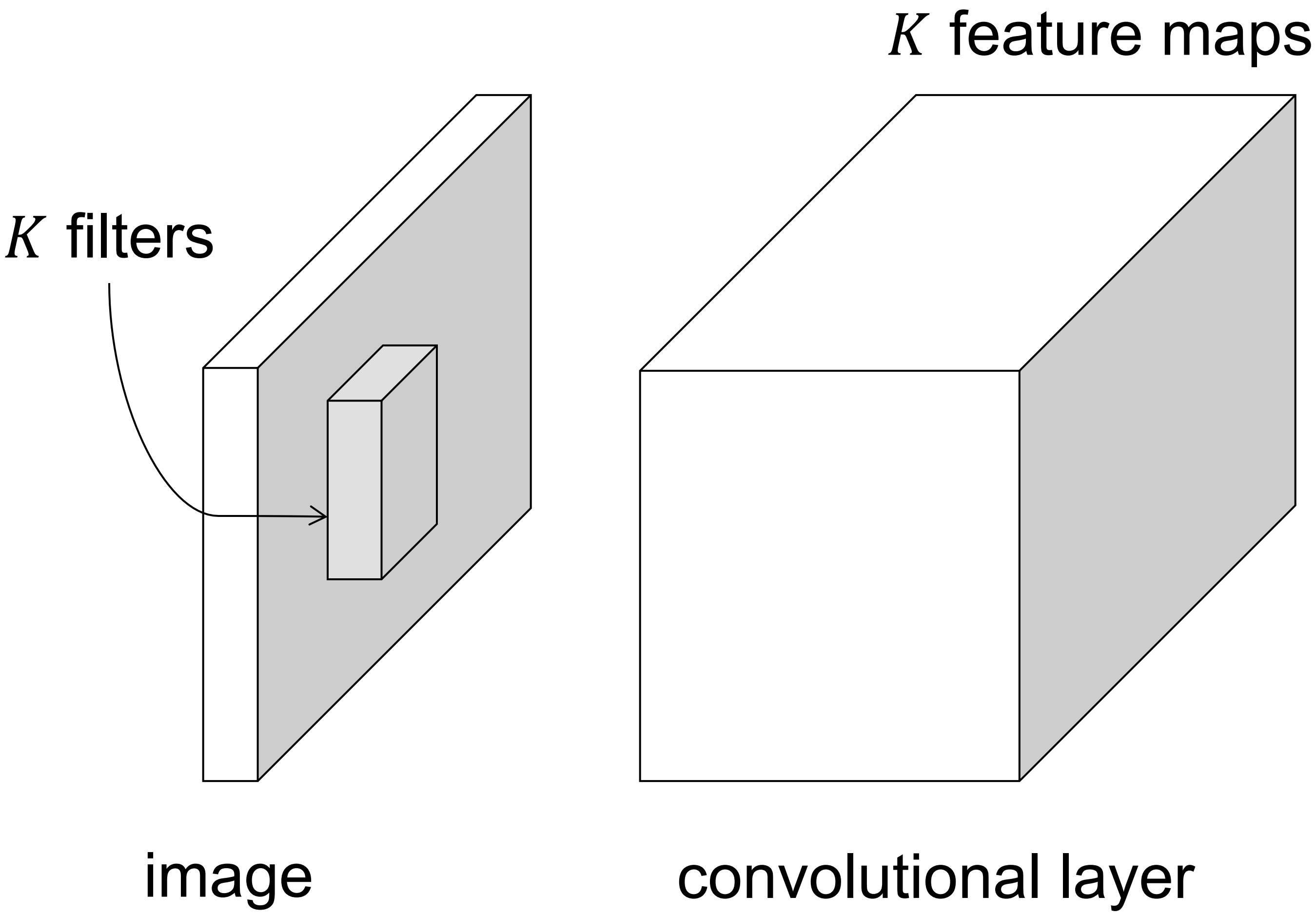


Learned weights can be thought of as *local* templates

Convolution and traditional feature extraction

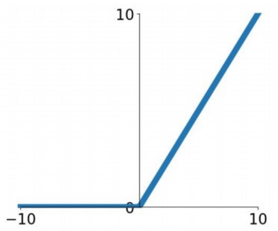


Elementwise nonlinearity



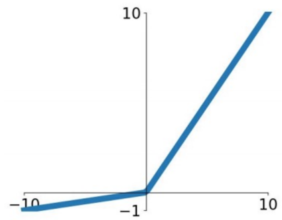
Almost always directly followed by a ReLU:

$\max(0, x)$



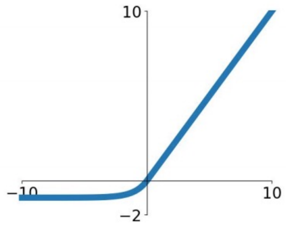
Some alternatives to ReLU:

Leaky ReLU
$\max(0.1x, x)$



ELU

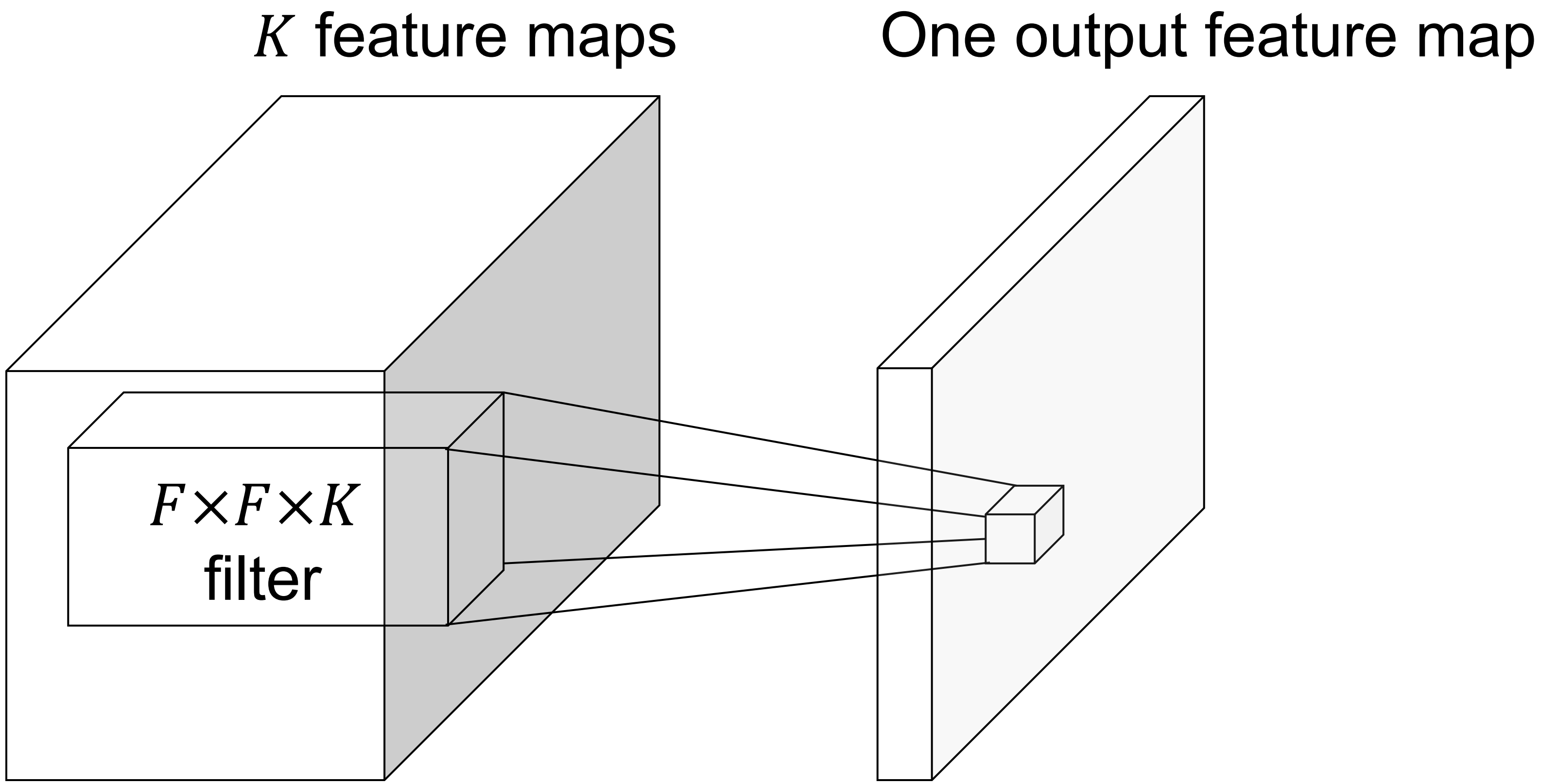
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Source: Stanford 231n

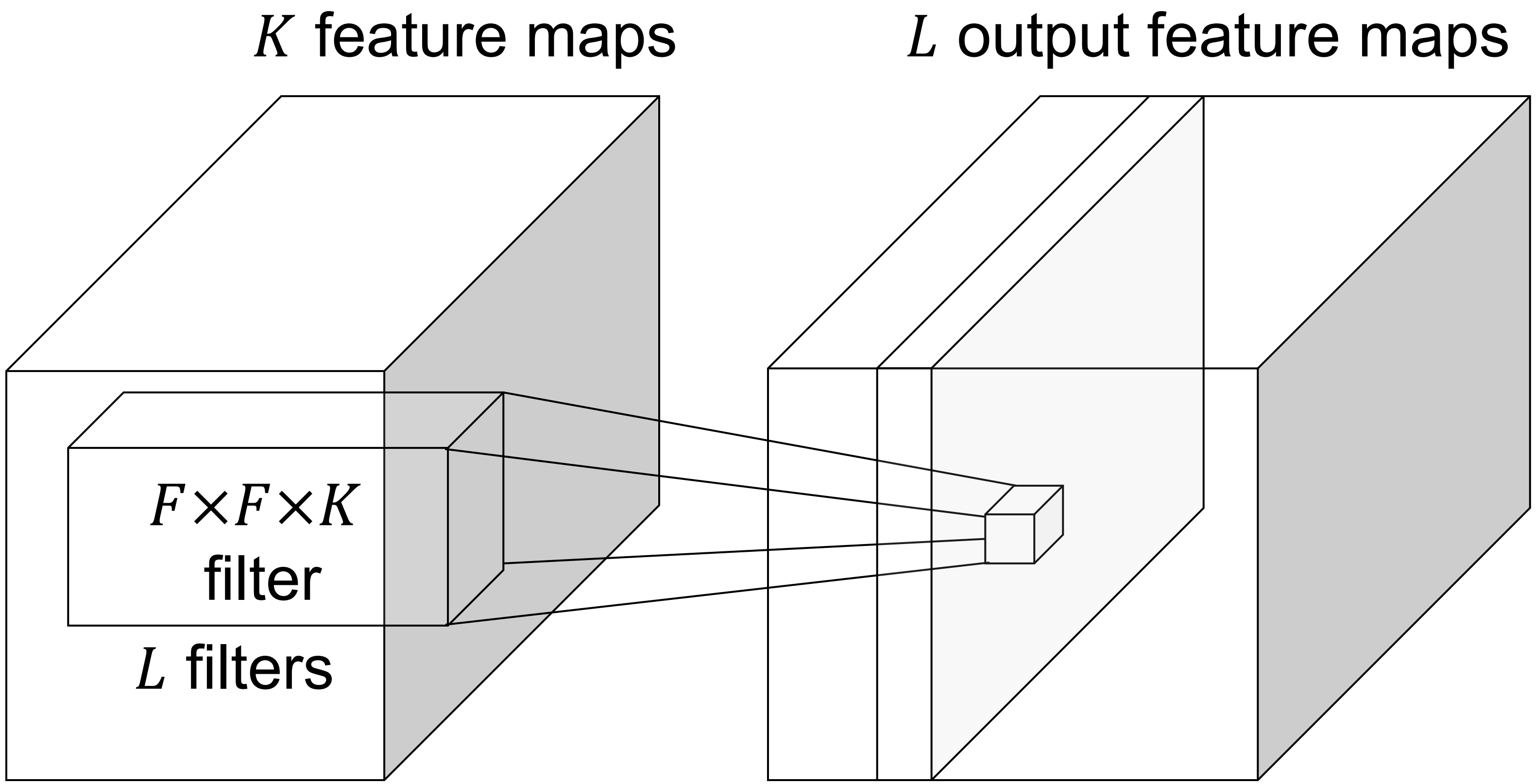
Three-dimensional convolutions

- What if the *input* to a convolutional layer is a stack of K feature maps?

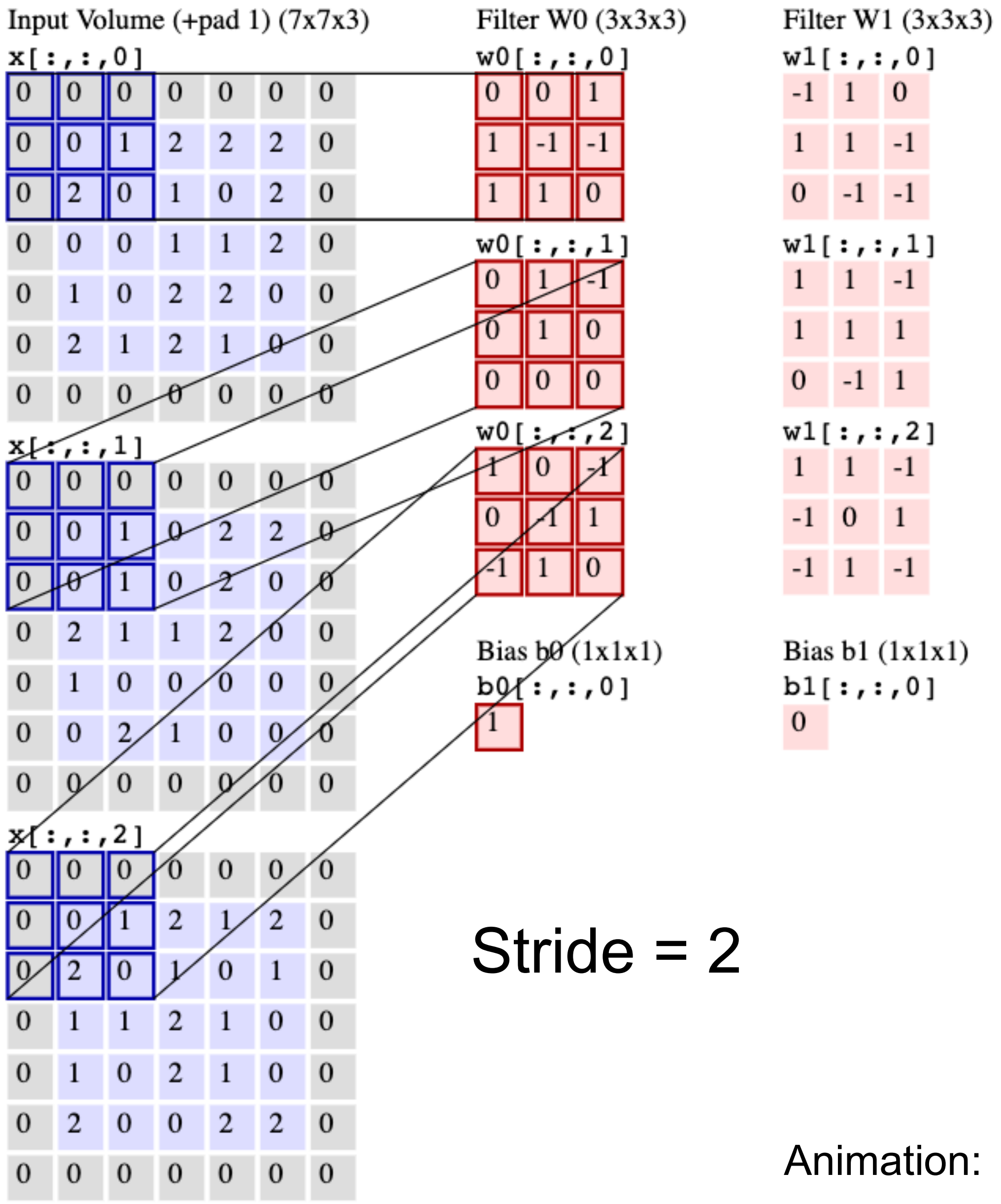


Three-dimensional convolutions

- What if the *input* to a convolutional layer is a stack of K feature maps?



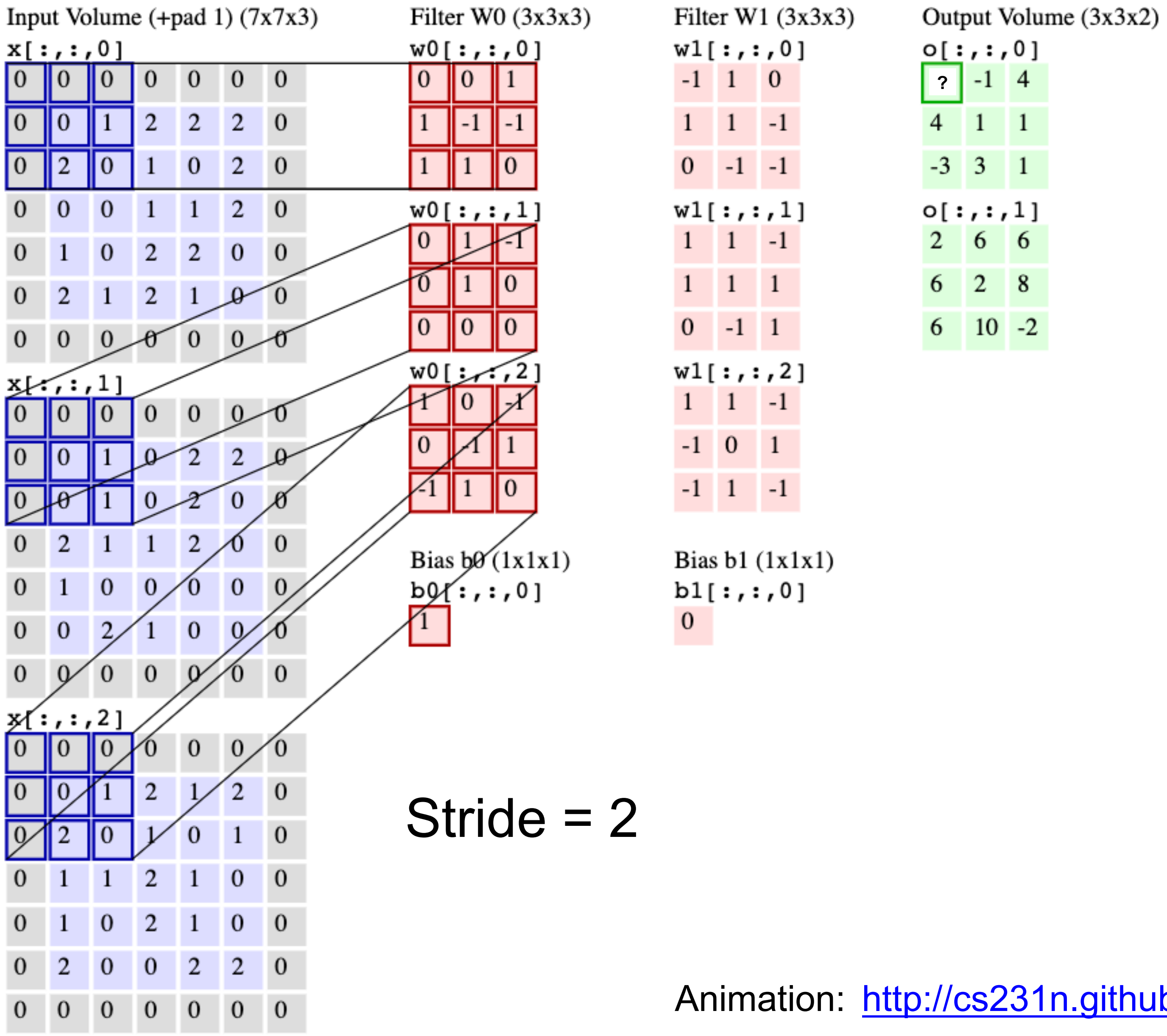
Convolutional layer example



Stride = 2

Animation: http://cs231n.github.io/convolutional-networks/#conv

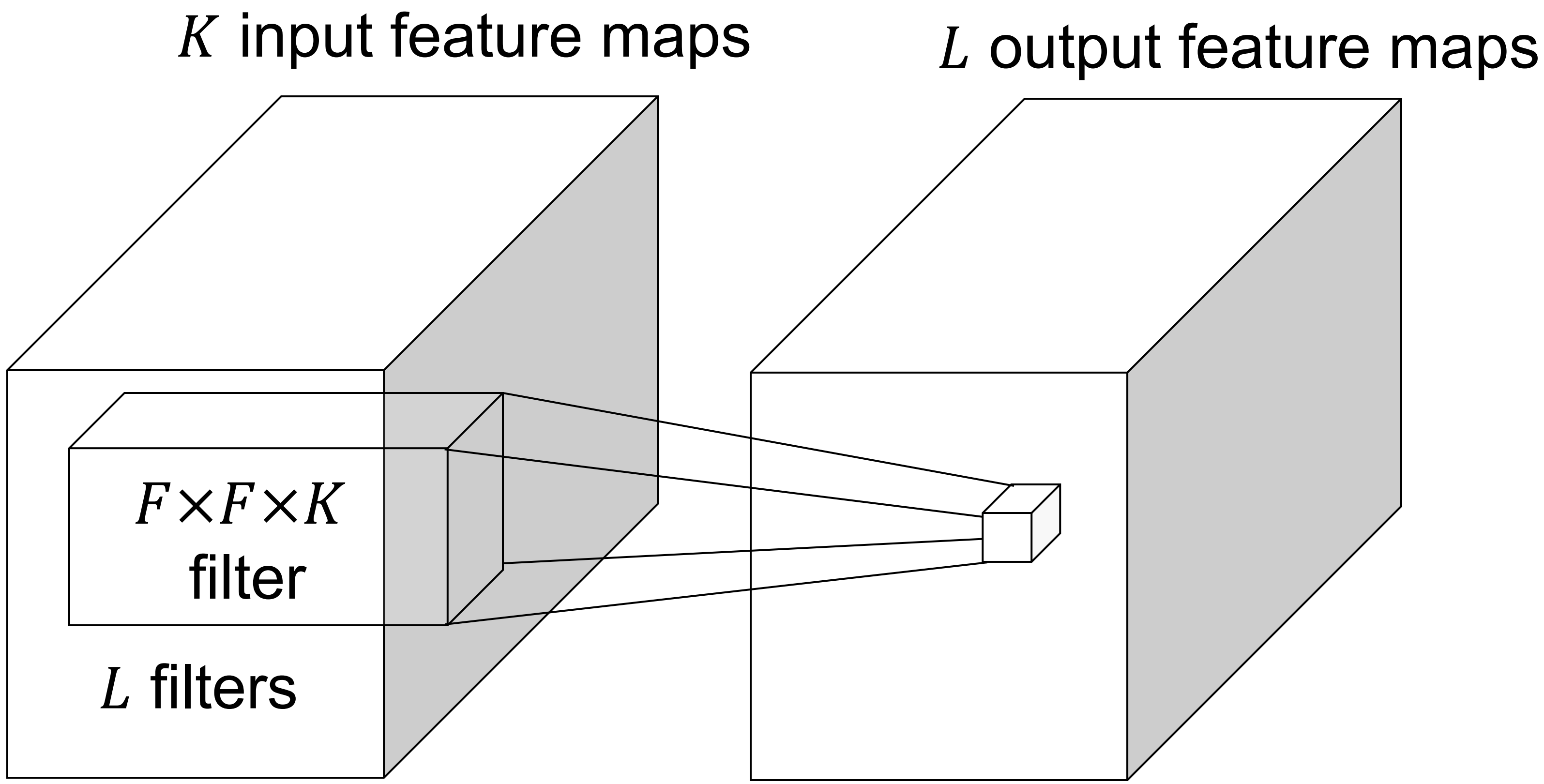
Convolutional layer example



Stride = 2

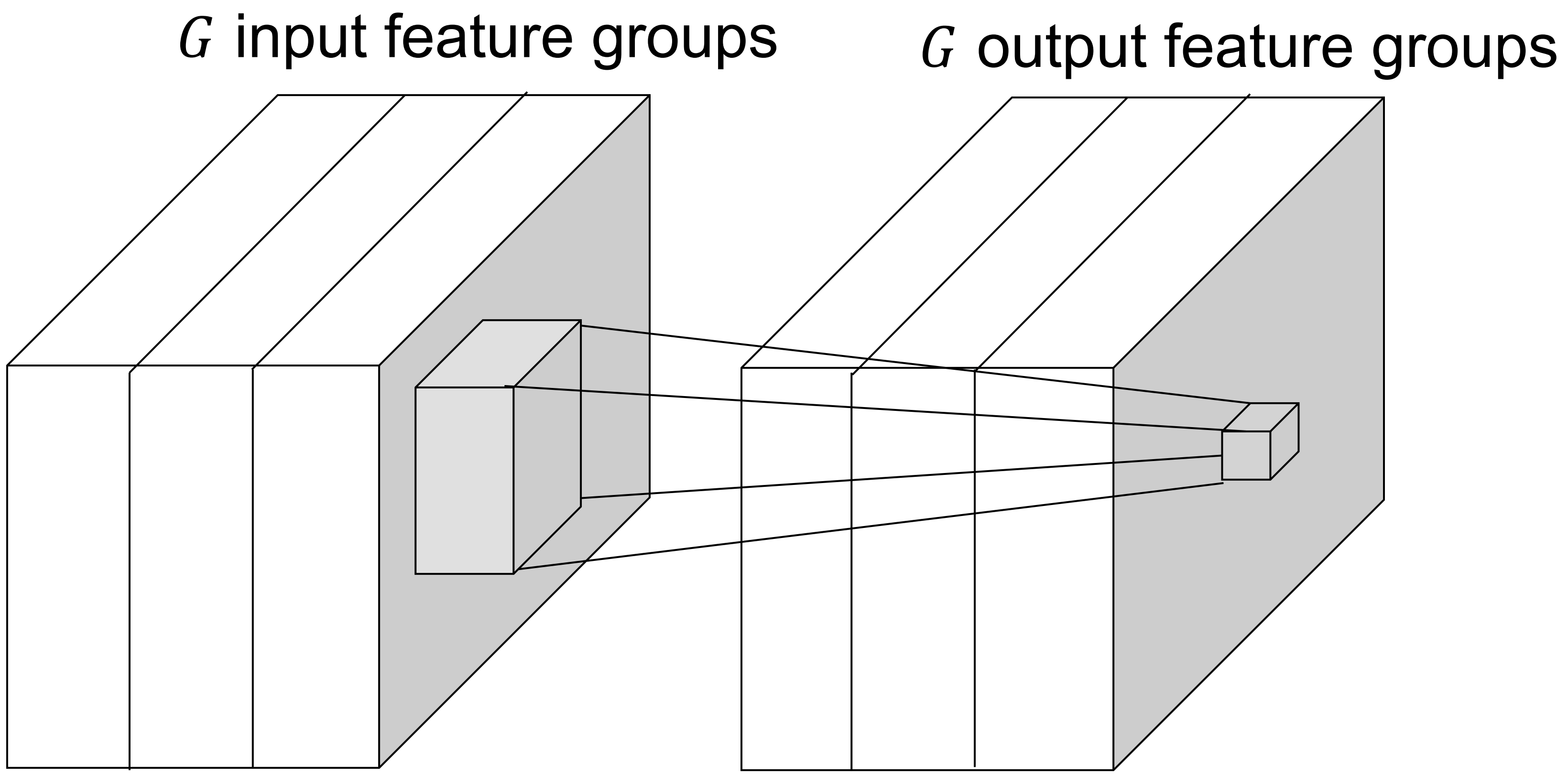
Animation: http://cs231n.github.io/convolutional-networks/#conv

Convolutional layer: Computational cost



- Assuming the input feature maps have spatial resolution $H \times W$, how many operations are needed to compute the output feature volume?
  - $F^2 KLHW$

More generally: Groupwise convolutions



- Split up the K feature maps into G groups, perform convolutions within each group separately, concatenate the results

Convolutional layer: Details

- Efficient implementation: reshape all image neighborhoods into columns (im2col operation), do matrix-vector multiplication
- Backward pass: special case of linear layer, operations also turn out to be convolutions
  - Downstream gradient (of error w.r.t. input) is a *transposed convolution*, or convolution of output with filter flipped both horizontally and vertically

Outline

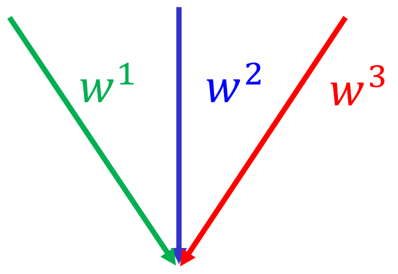
- Basic convolutional layer
- Backward pass

Convolutional layer: Backward pass

- Let’s take a 1D example with a filter of width 3:

Filter: w^1 w^2 w^3

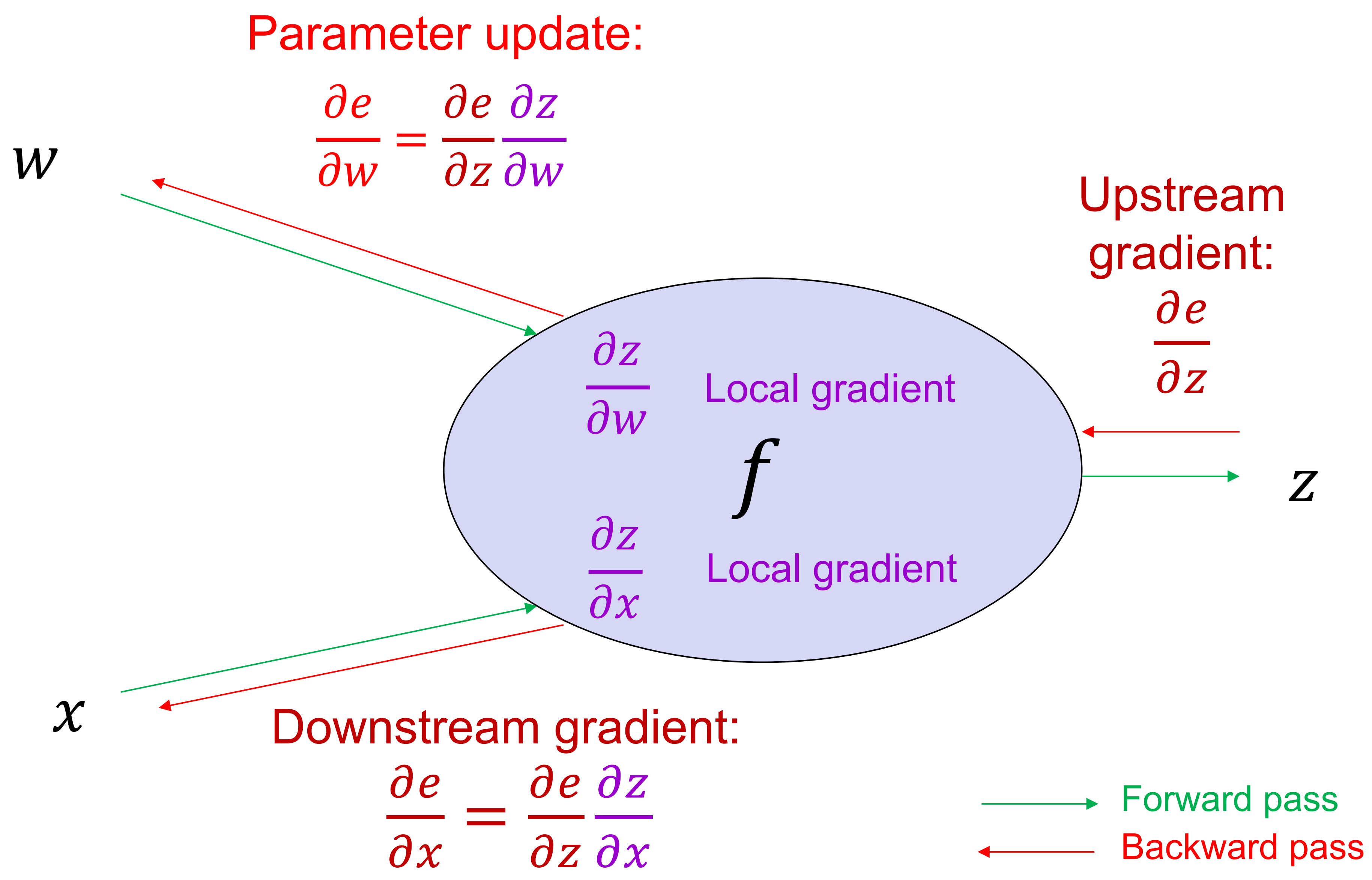
Input: … x^{i-1} x^i x^{i+1} …



Output: … z^{i-1} z^i z^{i+1} …

$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

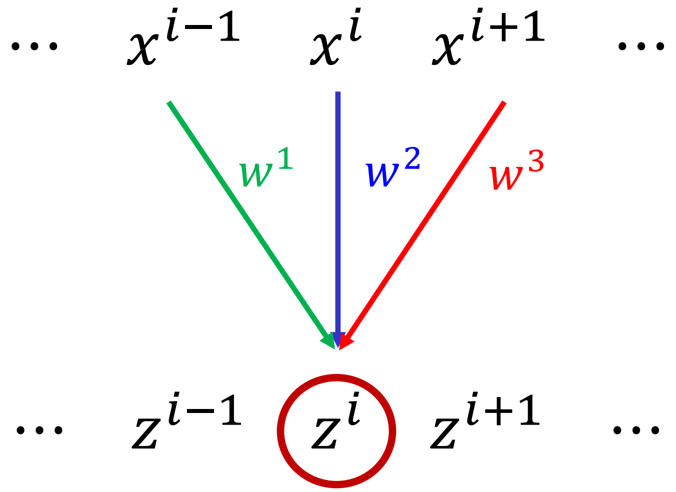
Review: Backward pass



Convolutional layer: Backward pass

Backward pass (w.r.t. x)

Vector-matrix form: $$\underset{1\times N}{\frac{\partial e}{\partial x}} = \underset{1\times N}{\frac{\partial e}{\partial z}} \underset{N\times N}{\frac{\partial z}{\partial x}}$$

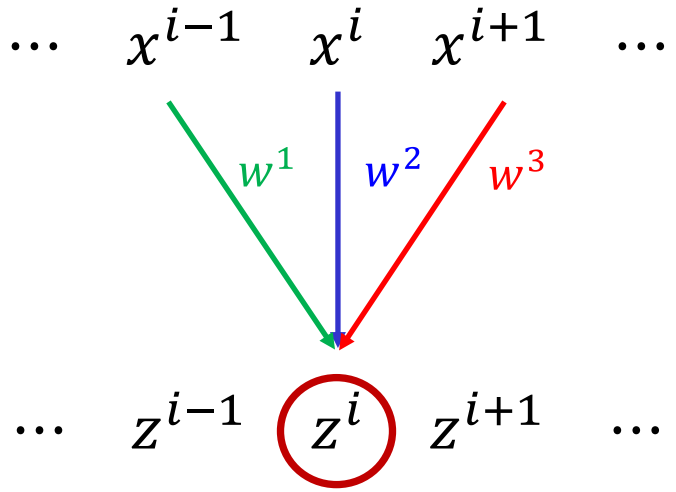


$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

Output: $\cdots \quad \frac{\partial e}{\partial x^{i-1}} \quad \frac{\partial e}{\partial x^{i}} \quad \frac{\partial e}{\partial x^{i+1}} \quad \cdots$

Input: $\cdots \quad \frac{\partial e}{\partial z^{i-1}} \quad \frac{\partial e}{\partial z^{i}} \quad \frac{\partial e}{\partial z^{i+1}} \quad \cdots$

Convolutional layer: Backward pass



$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

Backward pass (w.r.t. x)

Vector-matrix form: $\underset{1\times N}{\dfrac{\partial e}{\partial x}} = \underset{1\times N}{\dfrac{\partial e}{\partial z}} \underset{N\times N}{\dfrac{\partial z}{\partial x}}$

Output: $\cdots \quad \dfrac{\partial e}{\partial x^{i-1}} \quad \dfrac{\partial e}{\partial x^{i}} \quad \dfrac{\partial e}{\partial x^{i+1}} \quad \cdots$

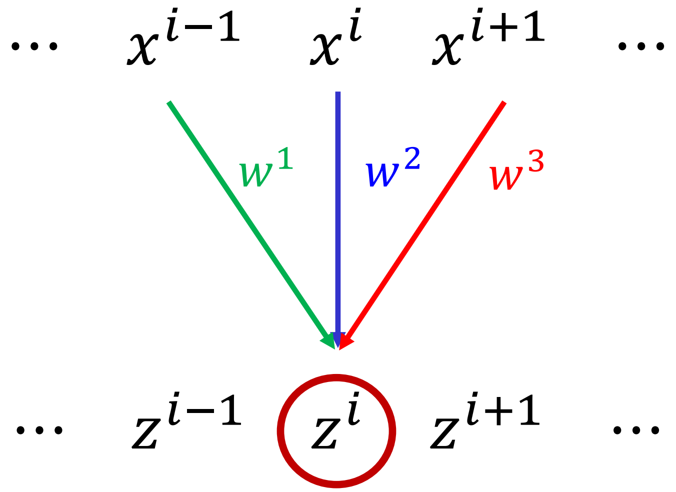
Input: $\cdots \quad \dfrac{\partial e}{\partial z^{i-1}} \quad \dfrac{\partial e}{\partial z^{i}} \quad \dfrac{\partial e}{\partial z^{i+1}} \quad \cdots$

Convolutional layer: Backward pass

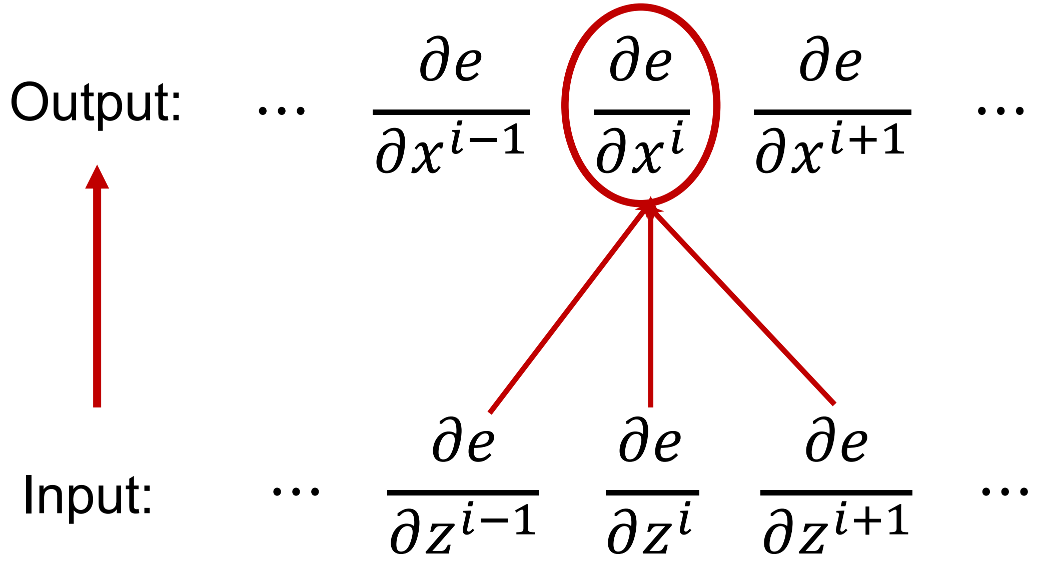
Backward pass (w.r.t. x)

$$\frac{\partial e}{\partial x^i} = \sum_{j=1}^{N} \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i}$$

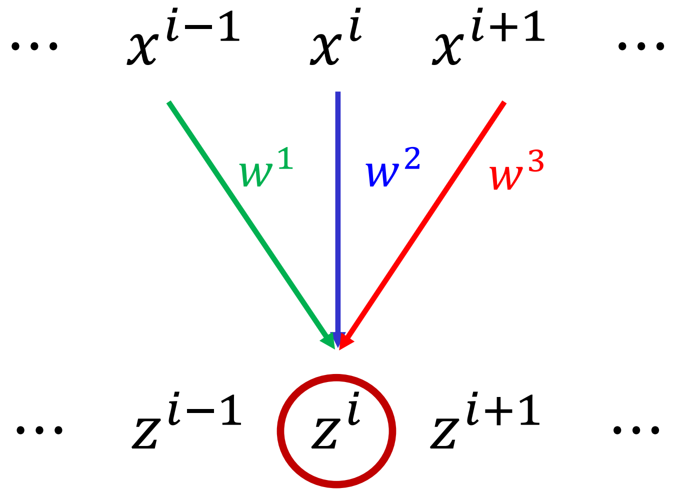
$$= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i}$$



$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



Convolutional layer: Backward pass



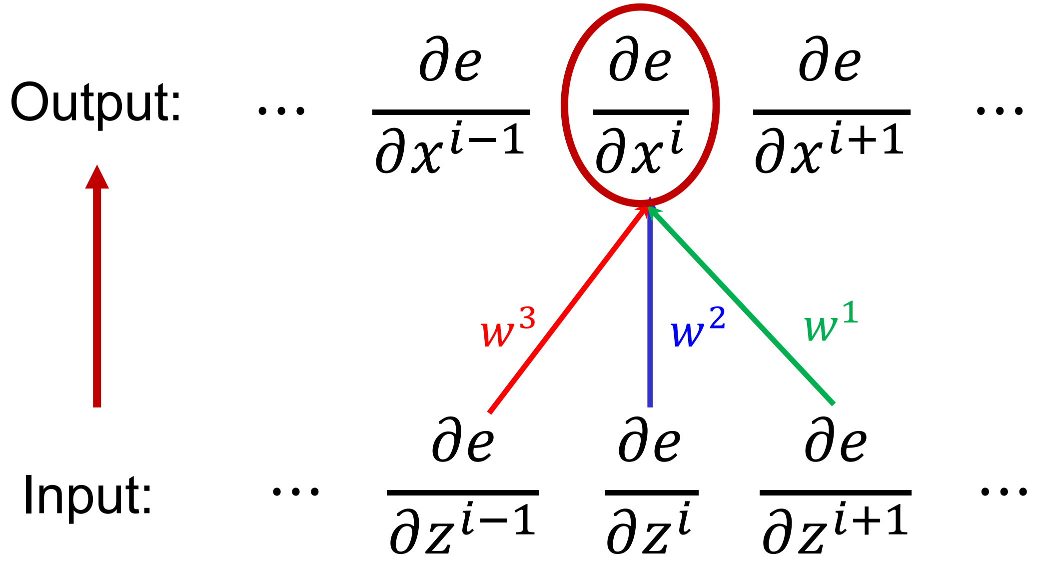
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

Backward pass (w.r.t. x)

$$\frac{\partial e}{\partial x^i} = \sum_{j=1}^{N} \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i}$$

$$= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i}$$

$$= w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^i} + w^1 \frac{\partial e}{\partial z^{i+1}}$$

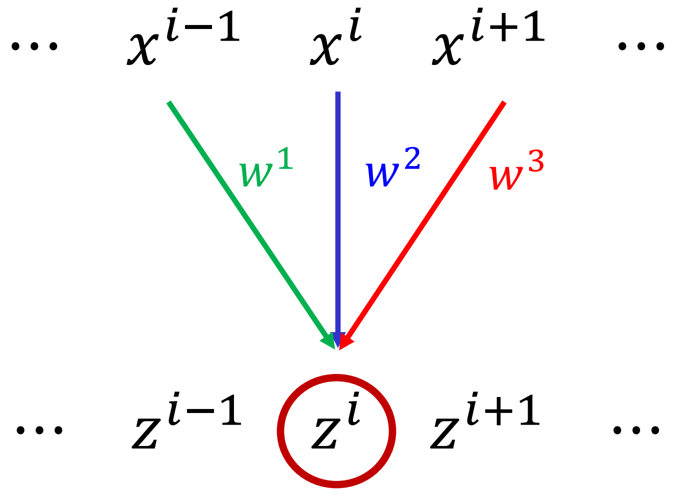


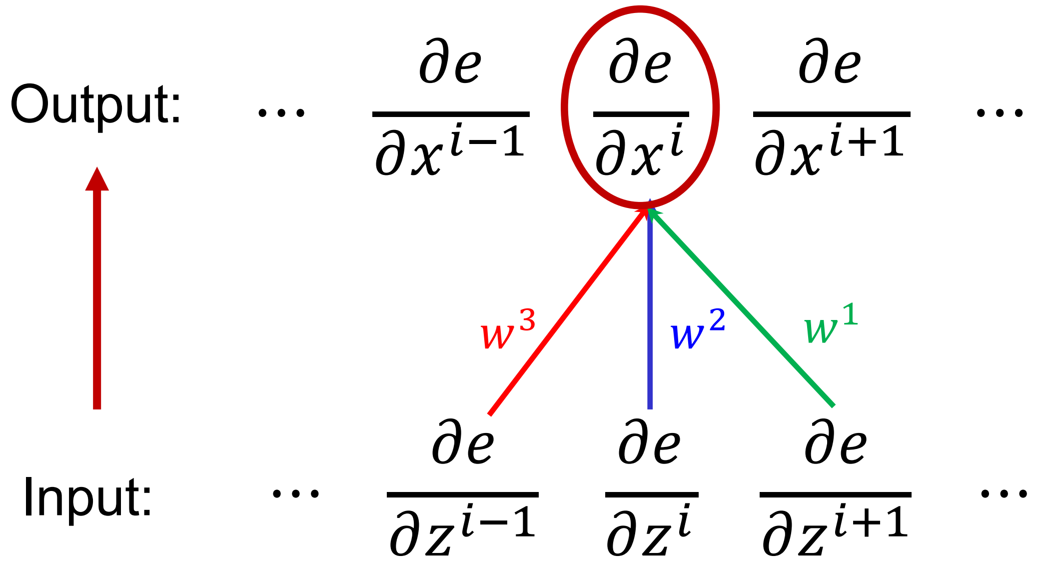
Convolutional layer: Backward pass

Backward pass (w.r.t. x)

This is called a *transposed convolution*

$$\frac{\partial e}{\partial x^i} = w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^{i}} + w^1 \frac{\partial e}{\partial z^{i+1}}$$





$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

Backward pass

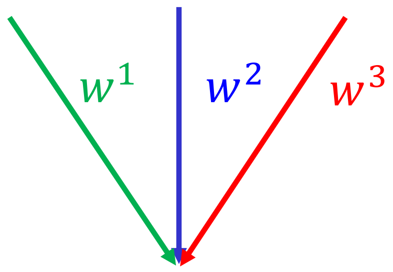
Backward pass (w.r.t. w)

$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial z}\frac{\partial z}{\partial w}$$

Output: $\frac{\partial e}{\partial w^1}$ $\frac{\partial e}{\partial w^2}$ $\frac{\partial e}{\partial w^3}$

Input: $\cdots$ $\frac{\partial e}{\partial z^{i-1}}$ $\frac{\partial e}{\partial z^{i}}$ $\frac{\partial e}{\partial z^{i+1}}$ $\cdots$

$$\cdots \quad x^{i-1} \quad x^{i} \quad x^{i+1} \quad \cdots$$



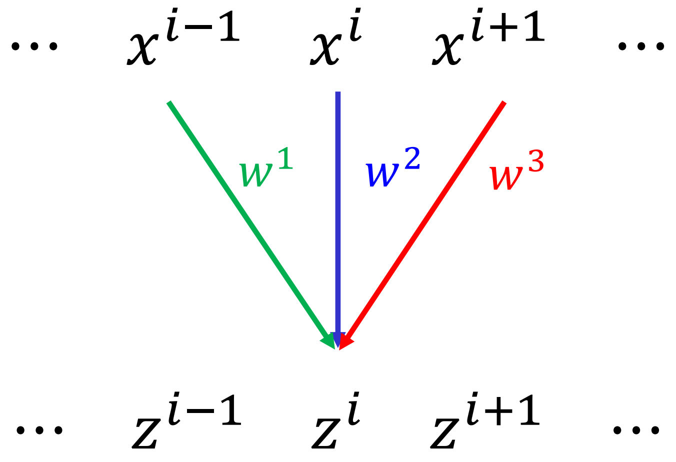
$$\cdots \quad z^{i-1} \quad z^{i} \quad z^{i+1} \quad \cdots$$

$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$

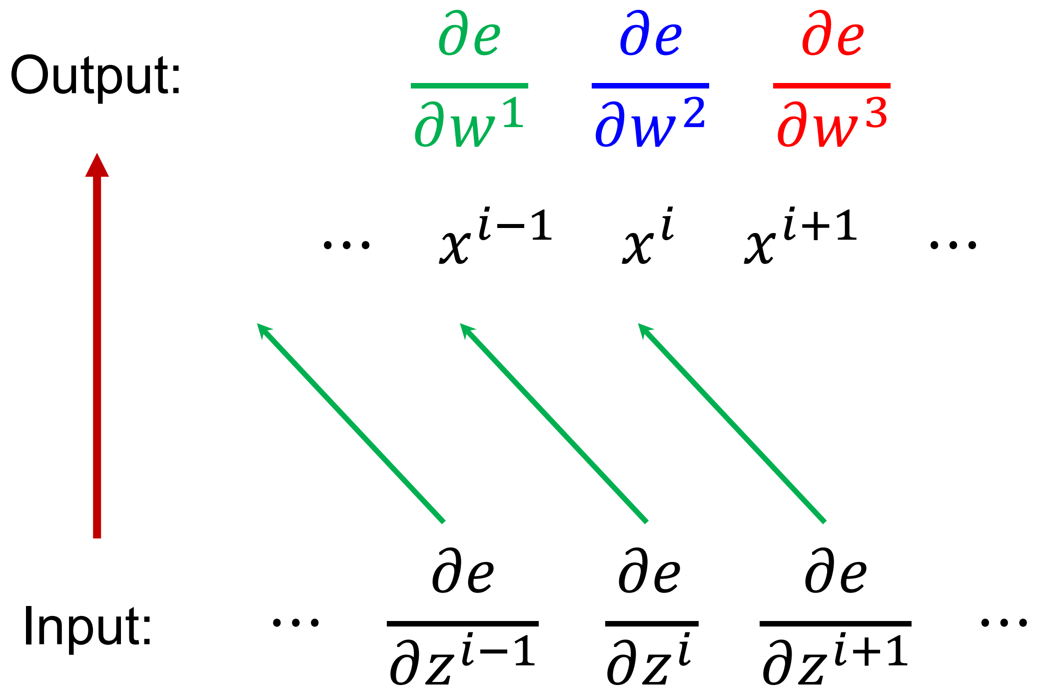
Backward pass

Backward pass (w.r.t. w)

$$\frac{\partial e}{\partial w^1} = \sum_i \frac{\partial e}{\partial z^i}\frac{\partial z^i}{\partial w^1} = \sum_i \frac{\partial e}{\partial z^i} x^{i-1}$$



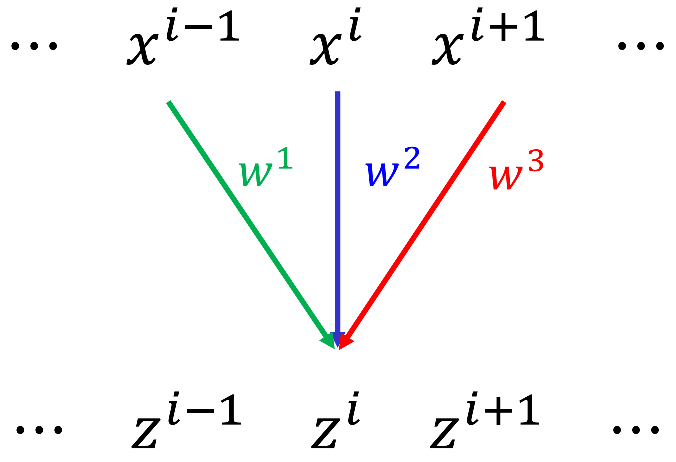
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



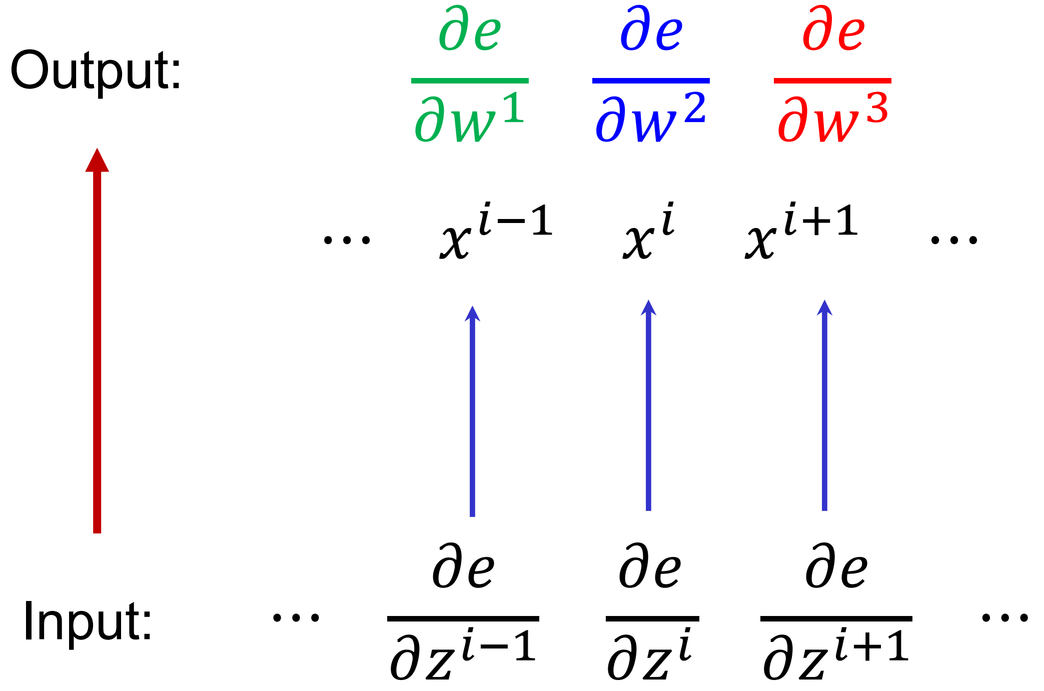
Backward pass

Backward pass (w.r.t. w)

$$\frac{\partial e}{\partial w^2} = \sum_i \frac{\partial e}{\partial z^i}\frac{\partial z^i}{\partial w^2} = \sum_i \frac{\partial e}{\partial z^i} x^i$$



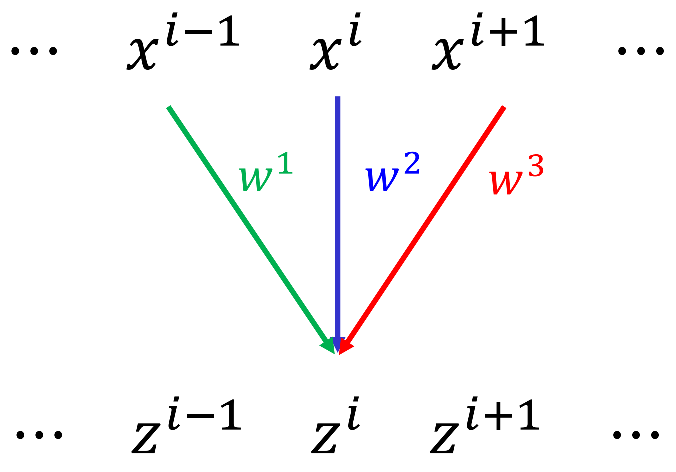
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



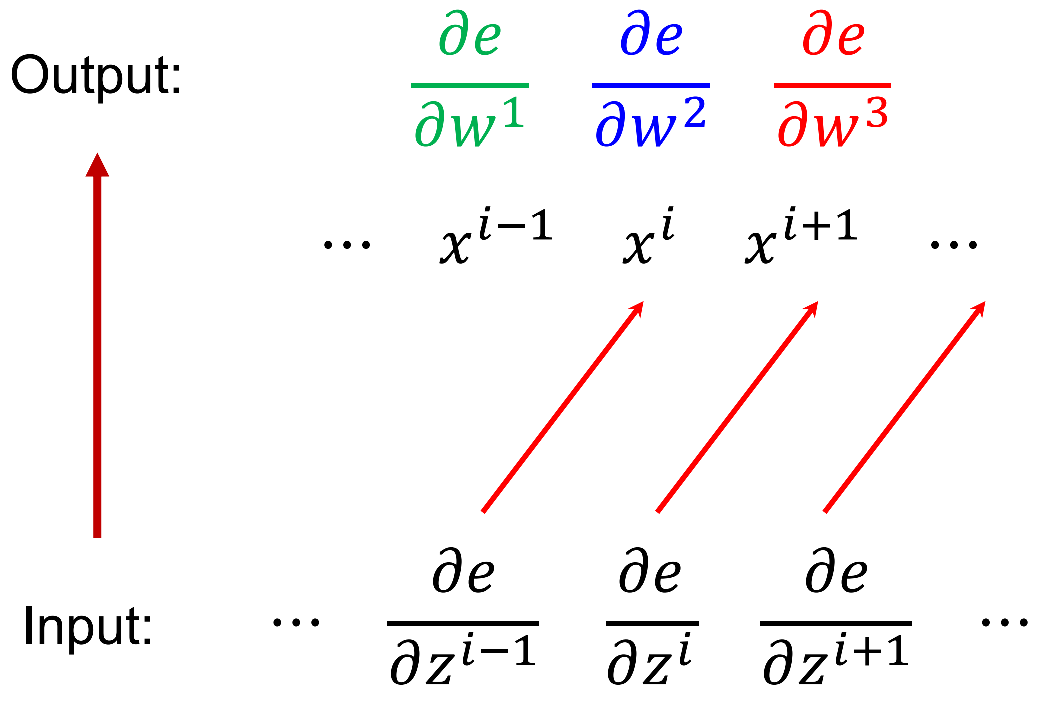
Backward pass

Backward pass (w.r.t. w)

$$\frac{\partial e}{\partial w^3} = \sum_i \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial w^3} = \sum_i \frac{\partial e}{\partial z^i} x^{i+1}$$



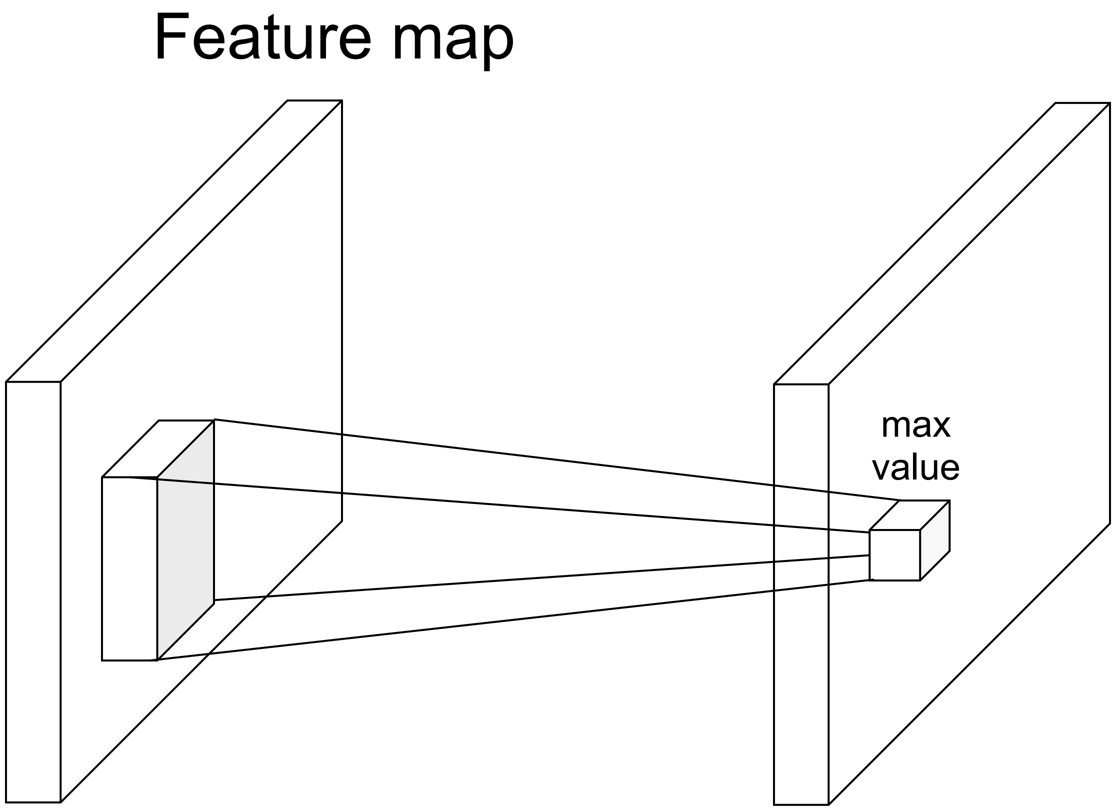
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



Outline

- Basic convolutional layer
  - Backward pass
- Max pooling layer

Max pooling layer



$F \times F$ pooling
window, stride S
Usually: $F = 2$ or 3, $S = 2$

Max pooling: Example

Single channel

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

Max pooling with 2×2 kernel size and stride 2

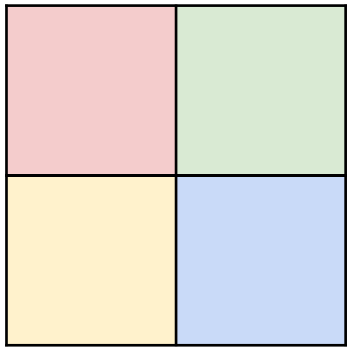
Source: [J. Johnson]

Max pooling: Example

Single channel

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

Max pooling with 2×2 kernel size and stride 2



Source: [J. Johnson]

Max pooling: Example

Single channel

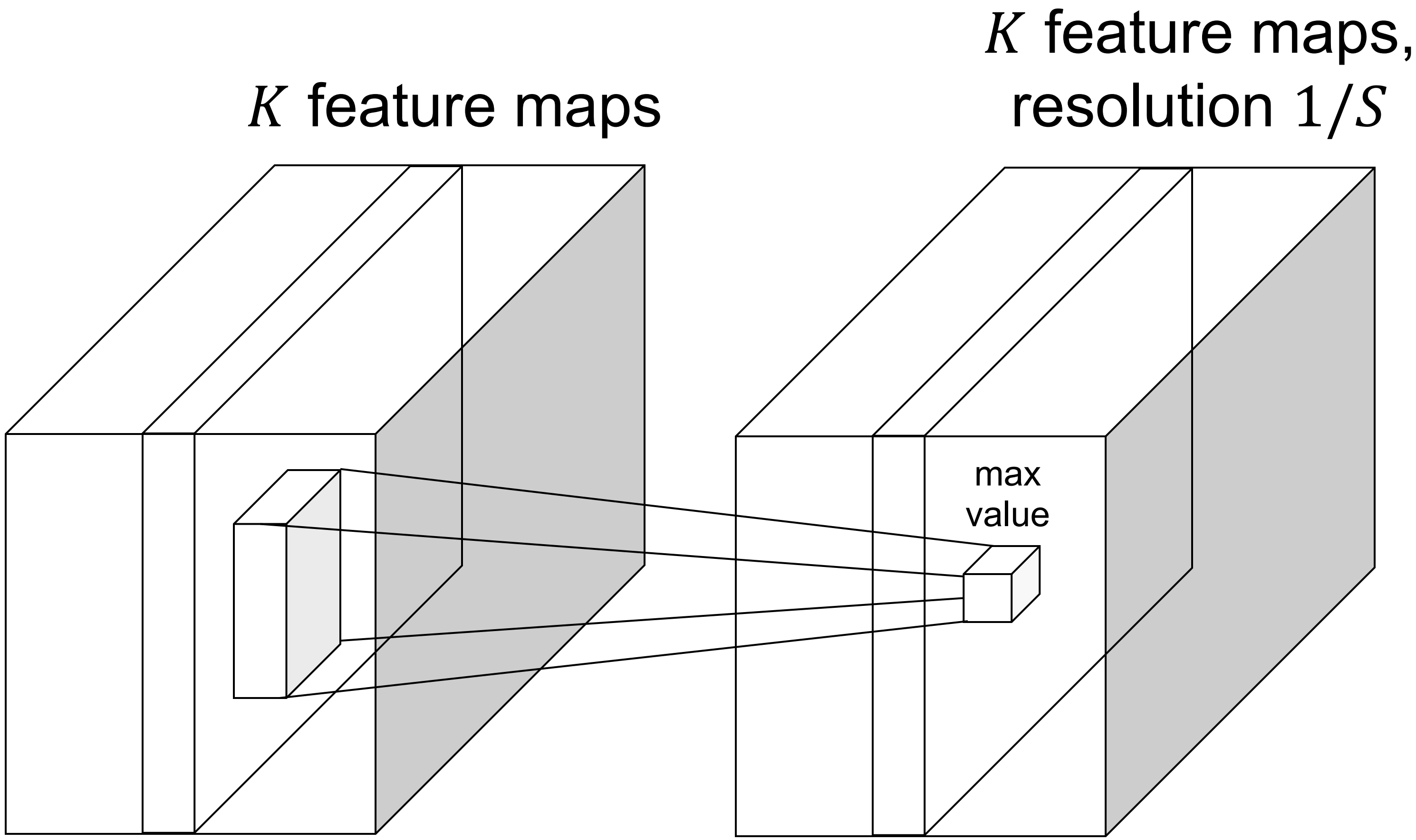
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2×2 kernel size and stride 2 →

6	8
3	4

Source: J. Johnson

Max pooling layer

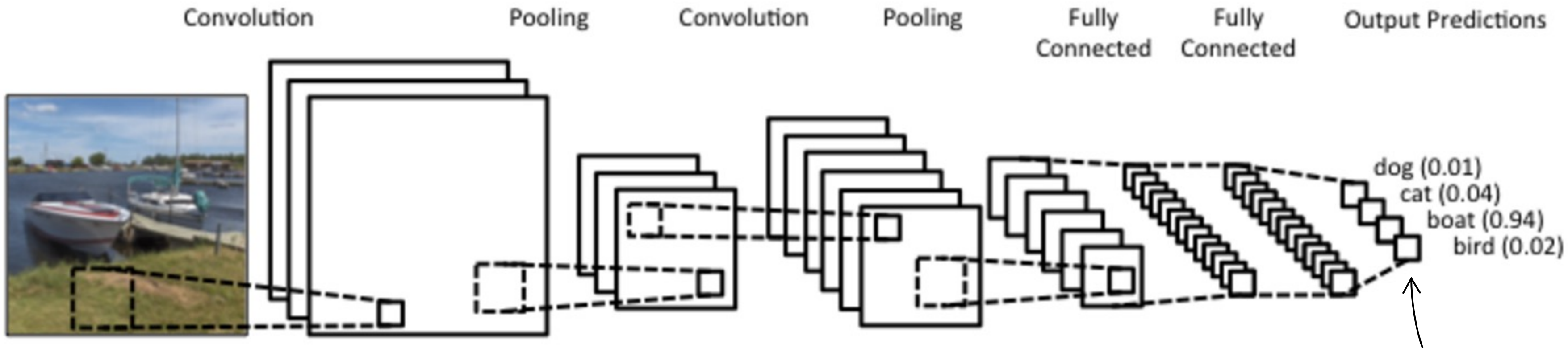


$F \times F$ pooling window, stride S

Usually: $F = 2$ or 3, $S = 2$

Backward pass: upstream gradient is passed back only to the unit with max value

Simplified CNN pipeline

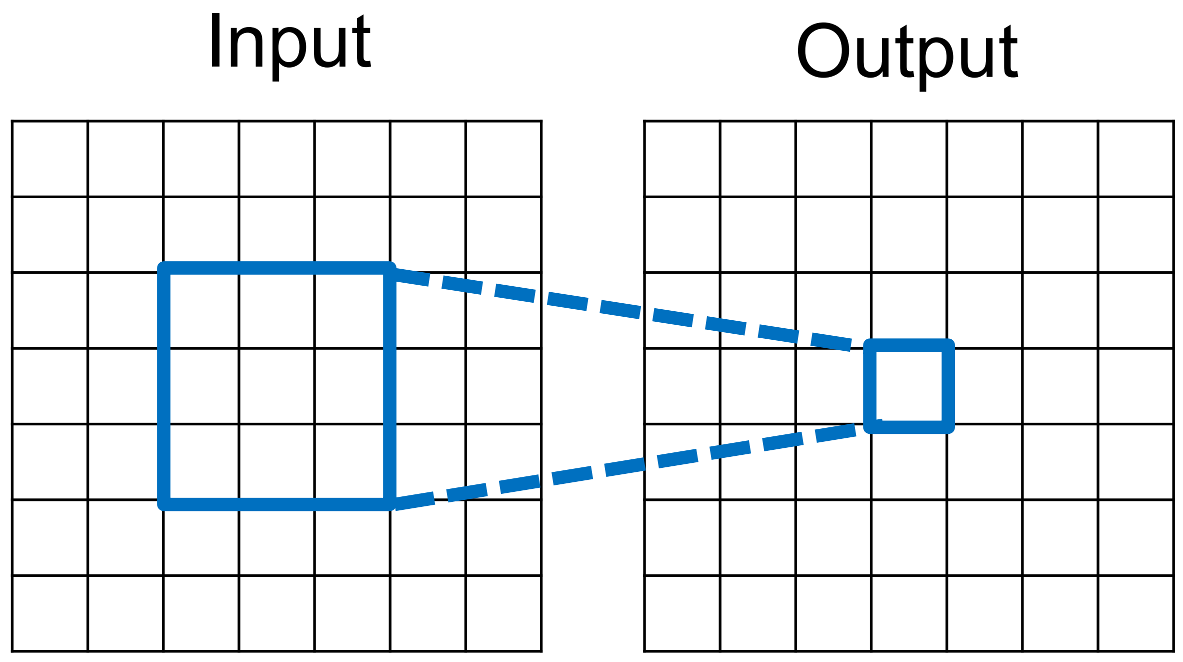


Softmax layer:

Receptive field

3x3 convolutions, stride 1

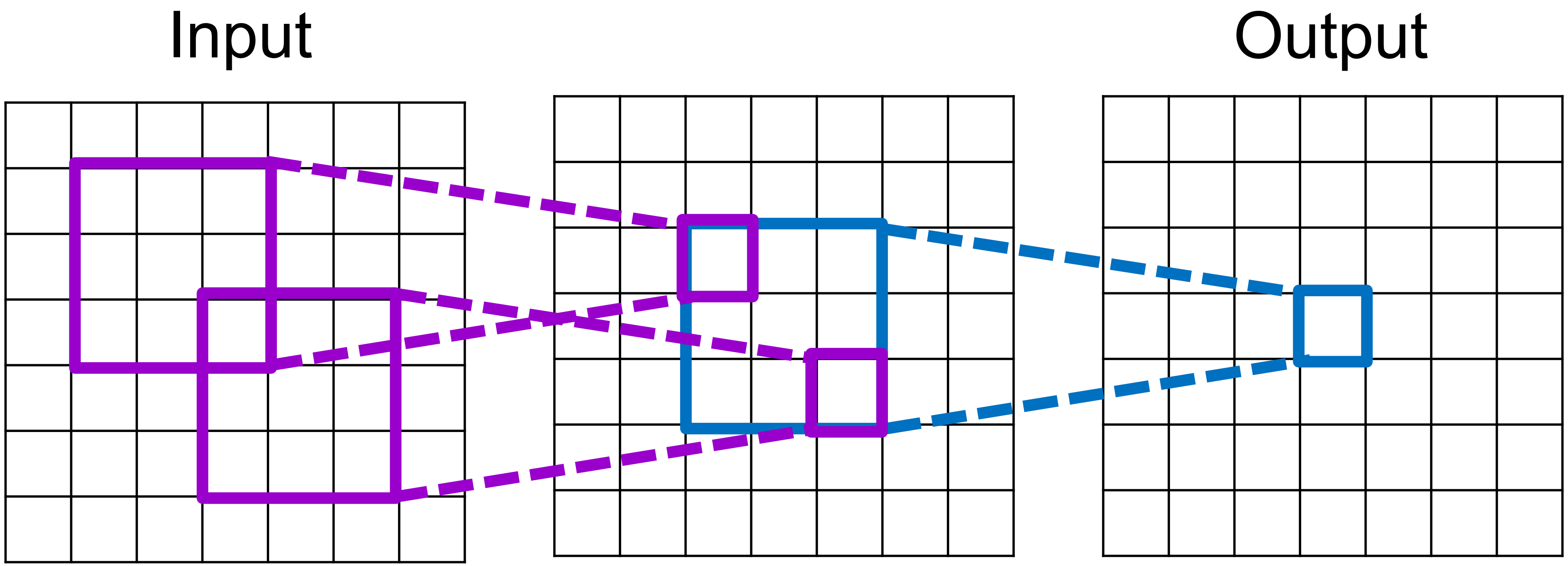
The *receptive field* of a unit is the region of the input feature map whose values contribute to the response of that unit (either in the previous layer or in the initial image)



Receptive field size: 3

Receptive field

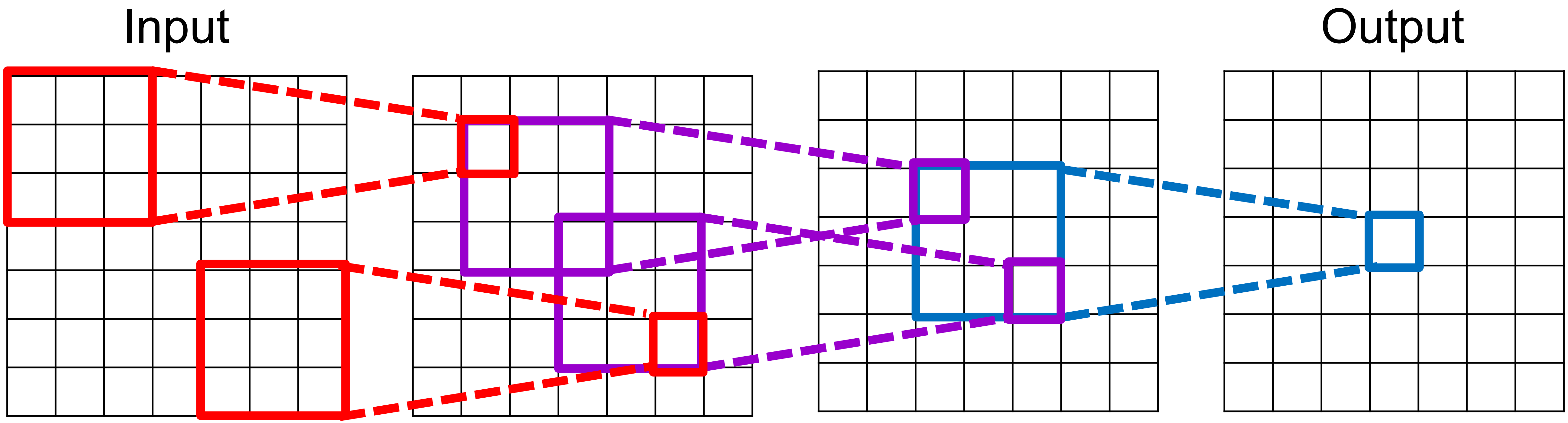
3x3 convolutions, stride 1



Receptive field size: 5

Receptive field

3x3 convolutions, stride 1



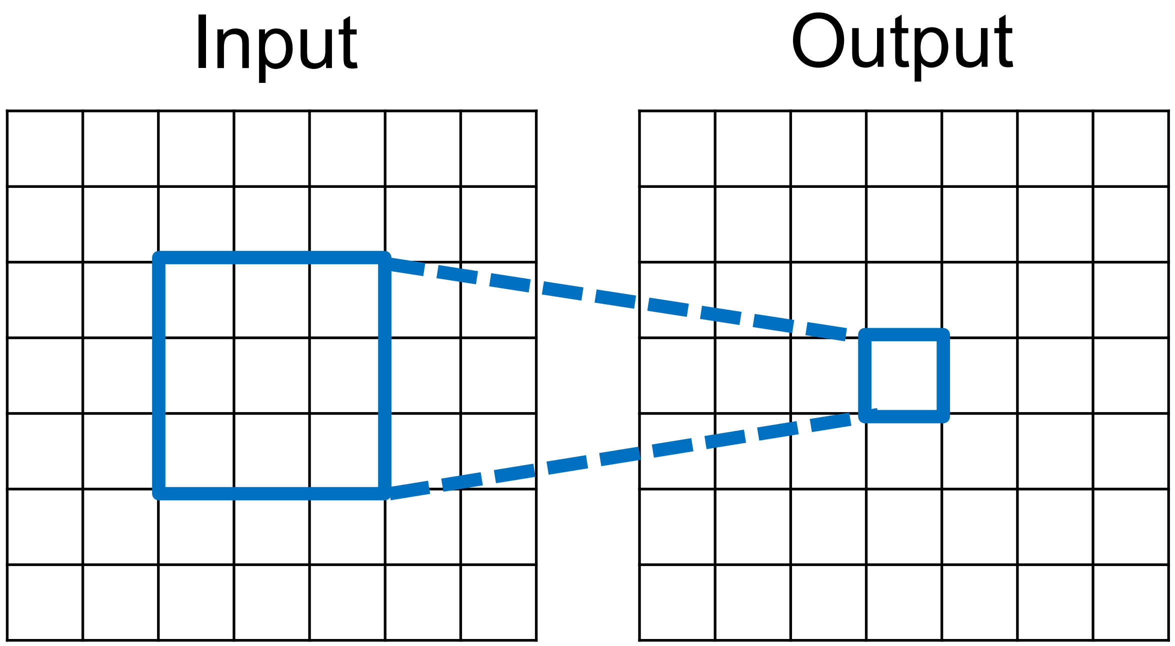
Receptive field size: 7

Each successive convolution adds $F - 1$ to the receptive field size

With L layers the receptive field size is $1 + L * (F - 1)$

Receptive field

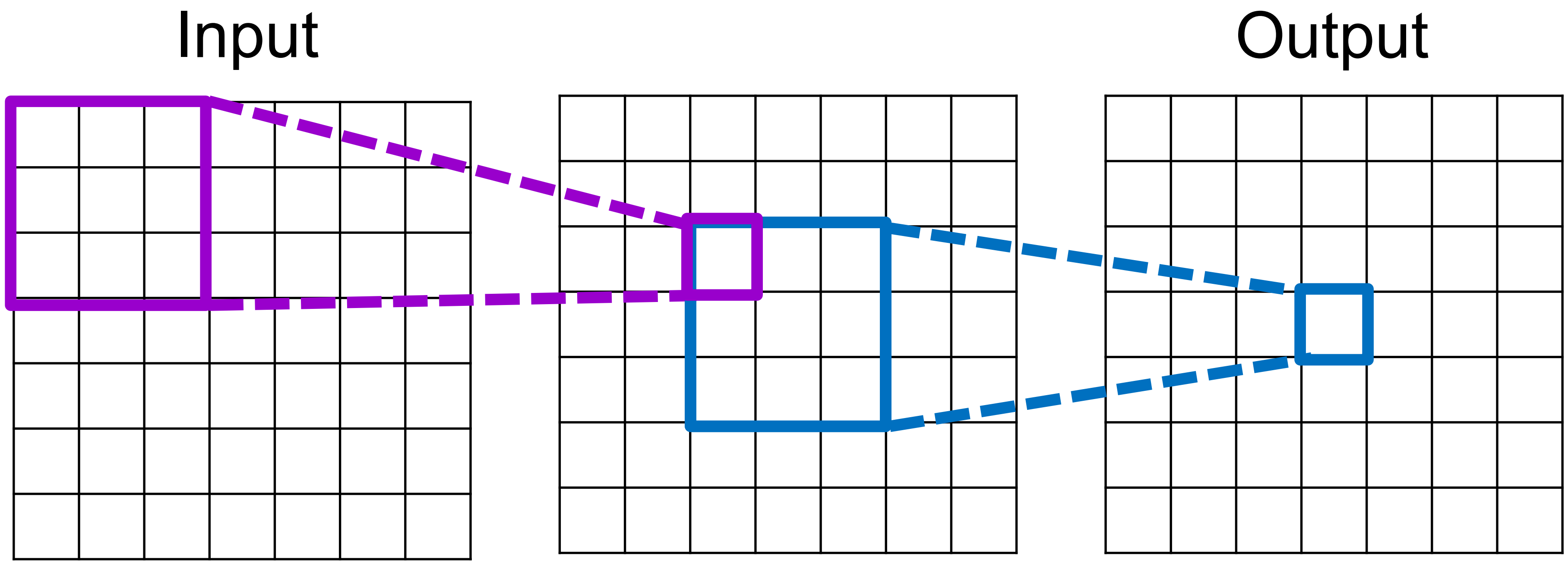
3x3 convolutions, stride 2



Receptive field size: 3

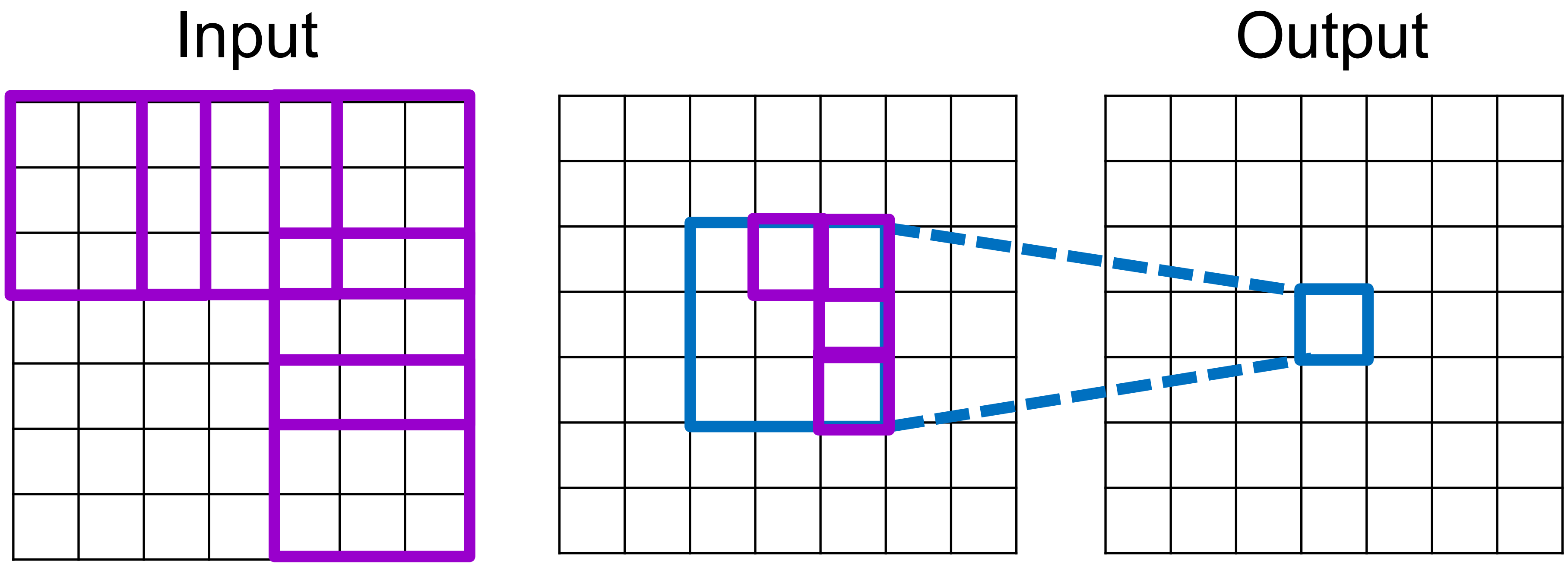
Receptive field

3x3 convolutions, stride 2



Receptive field

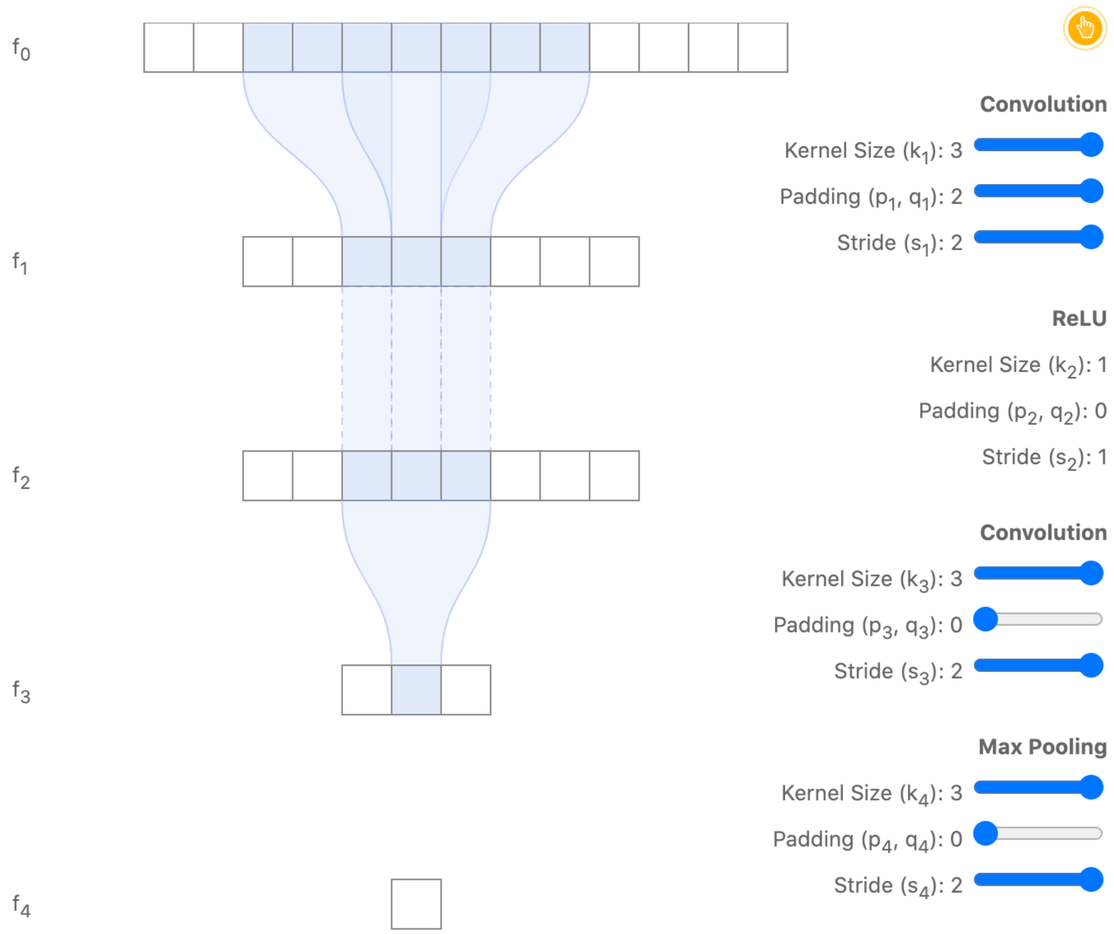
3x3 convolutions, stride 2



Receptive field size: 7

Receptive Field

Deep Nets with striding have large receptive fields



Source: https://distill.pub/2019/computing-receptive-fields/