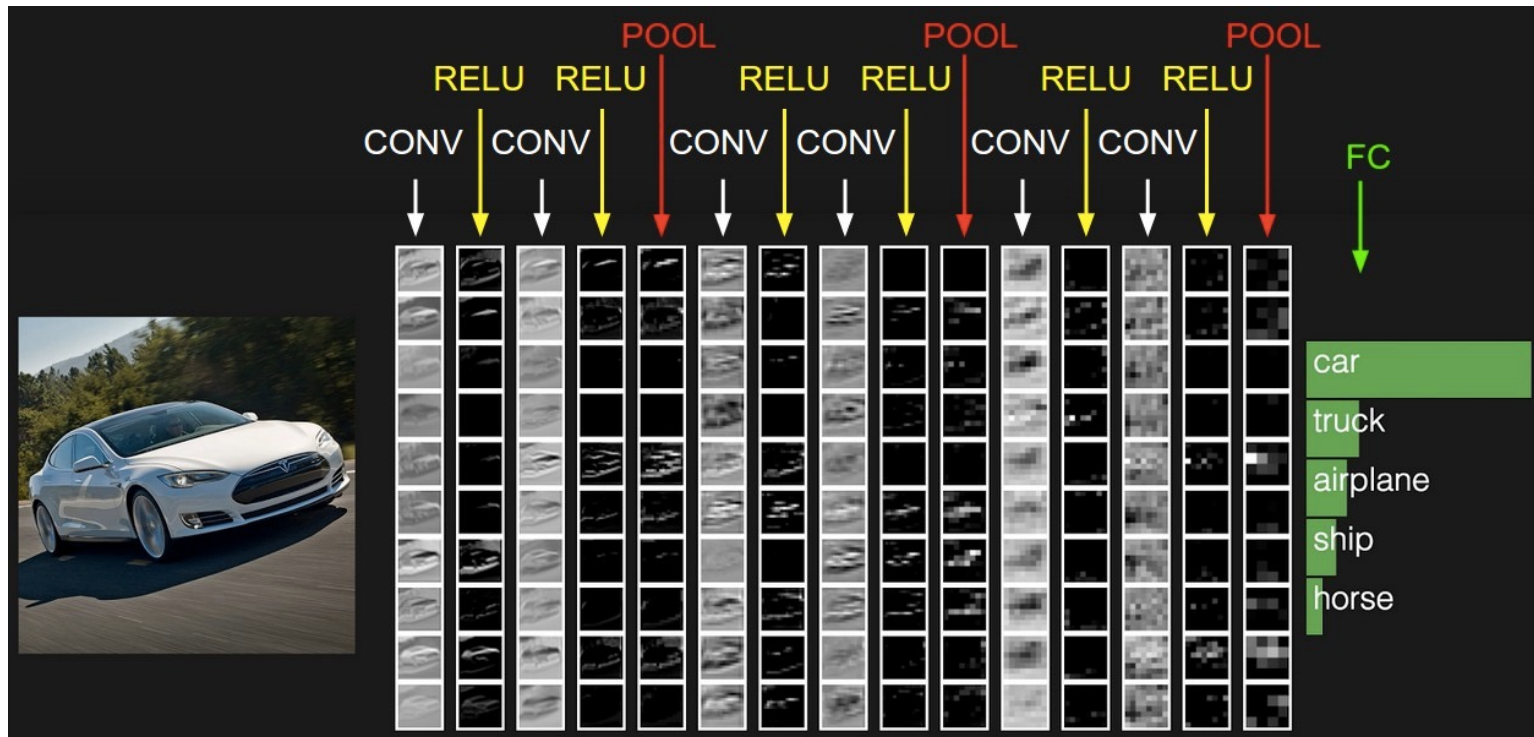


Convolutional neural networks

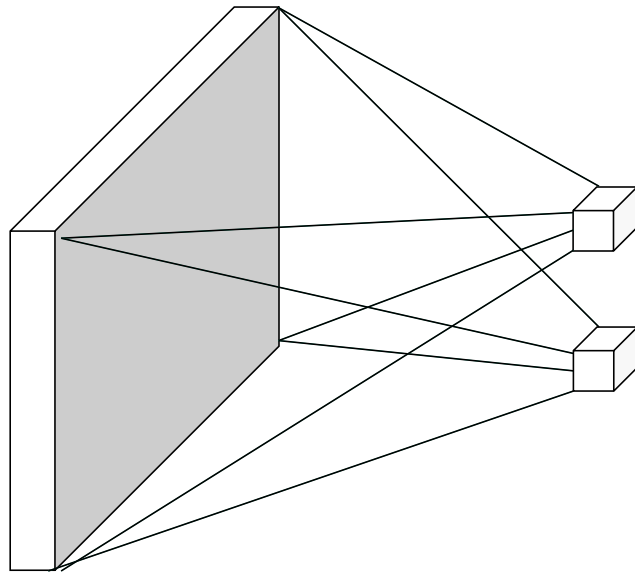


Many slides from Rob Fergus, Andrej Karpathy

Outline

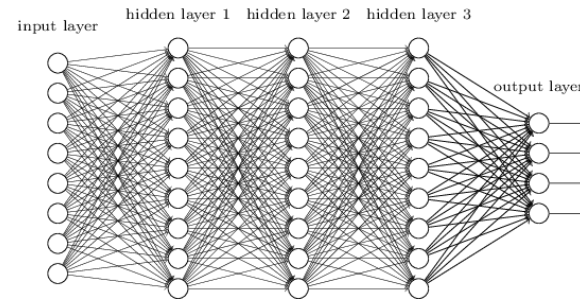
- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions
- Max pooling layer
- Backward pass

Let's design a neural network for images



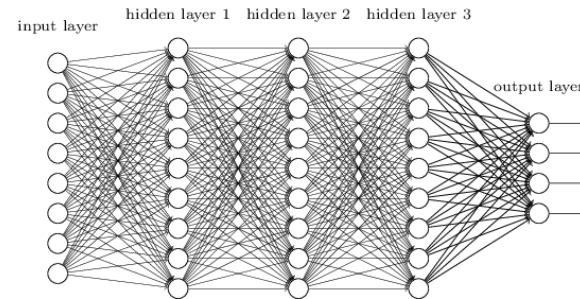
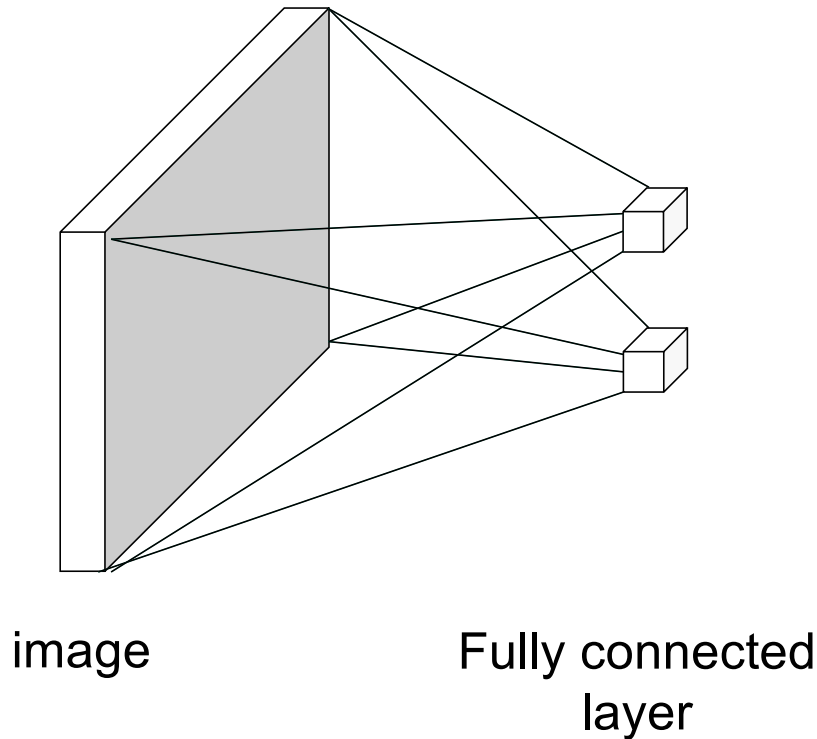
image

Fully connected
layer

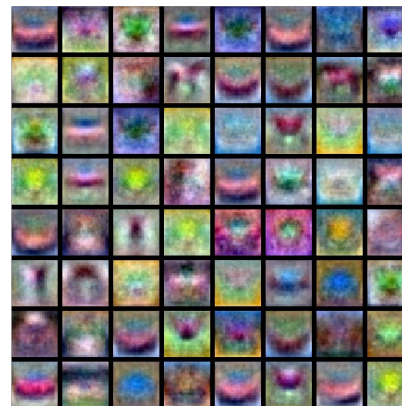


- This kind of design is known as *multi-layer perceptron (MLP)*

Let's design a neural network for images

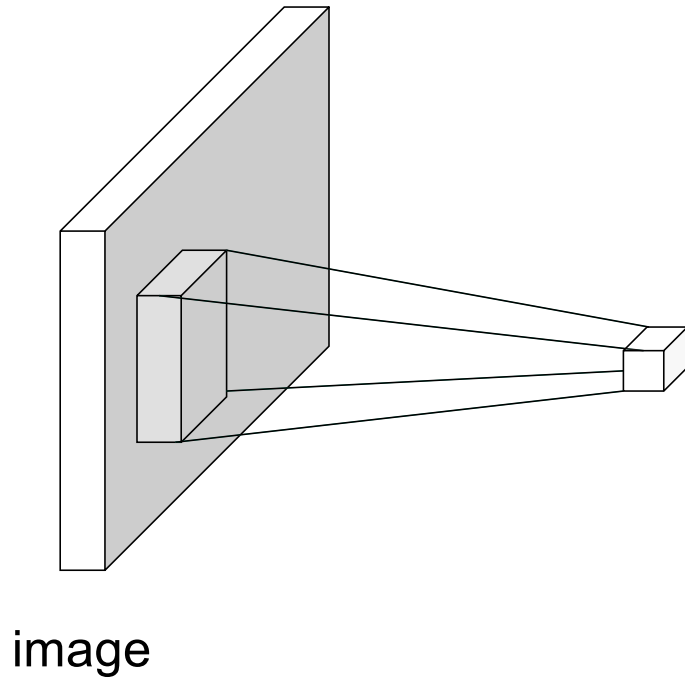


- This kind of design is known as *multi-layer perceptron* (MLP)
- What is wrong with this?



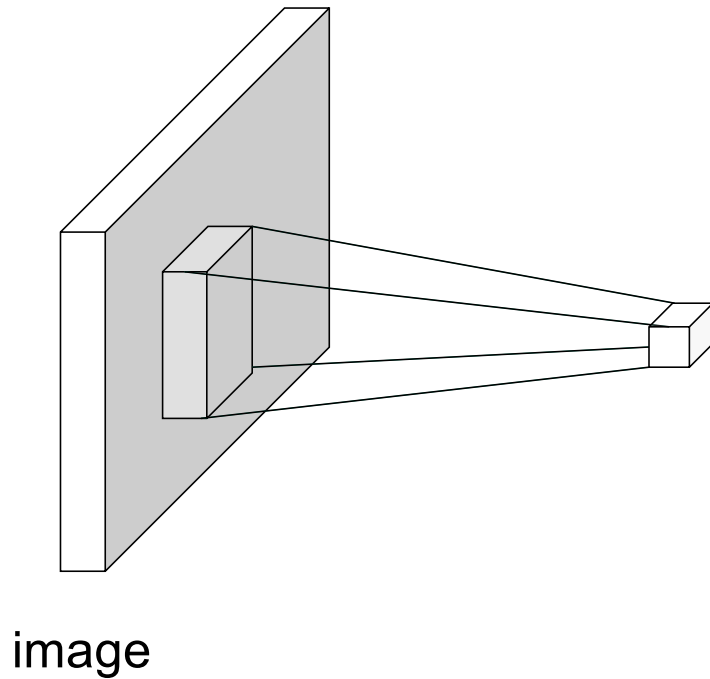
Recall: MLP as bank of whole-image templates

Convolutional architecture



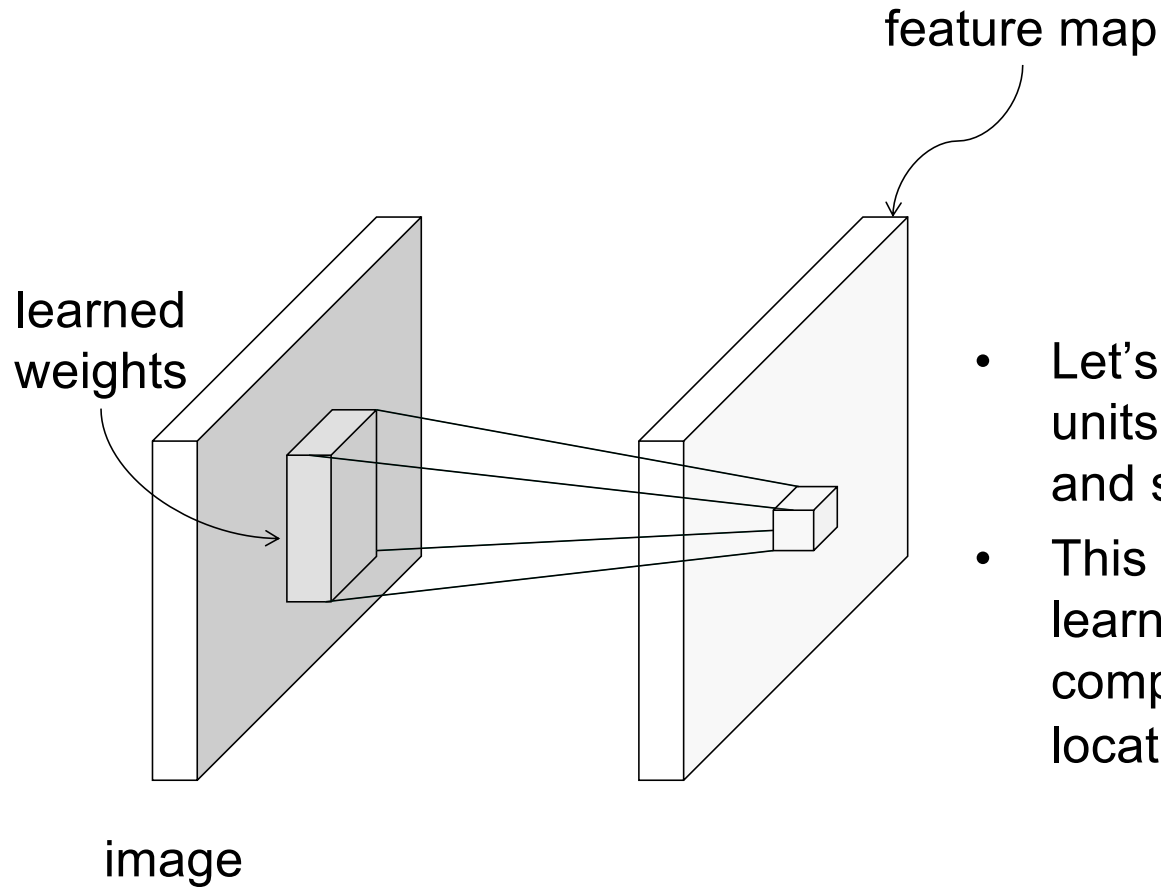
- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights

Convolutional architecture



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

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

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}

*

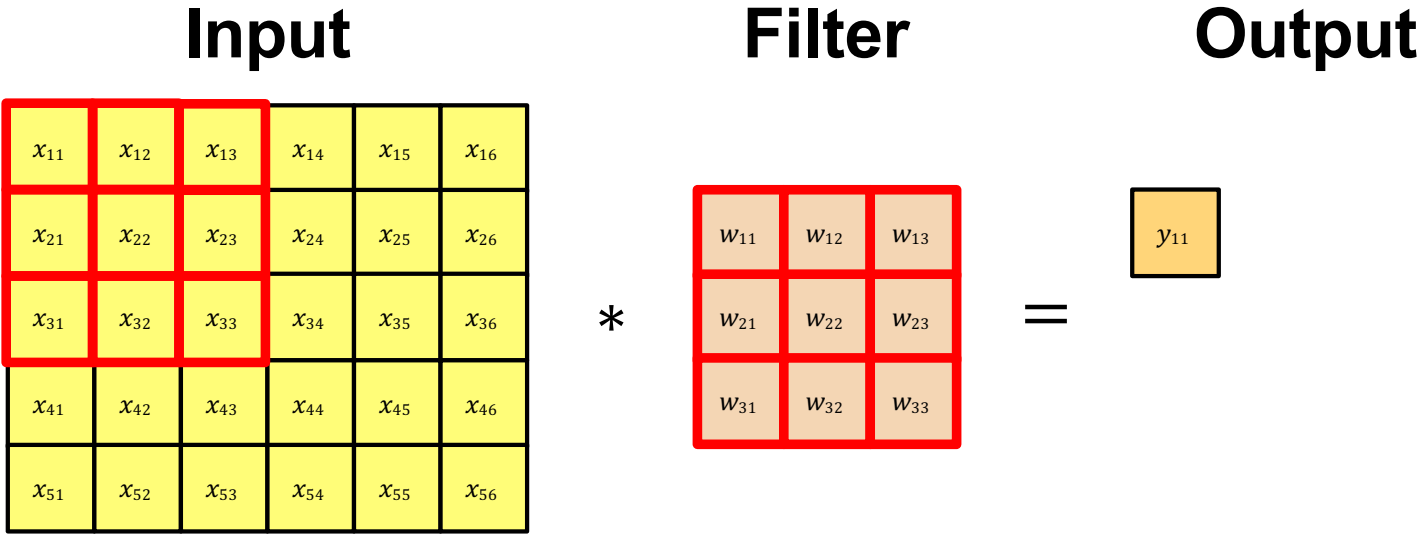
Filter

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

=

Output

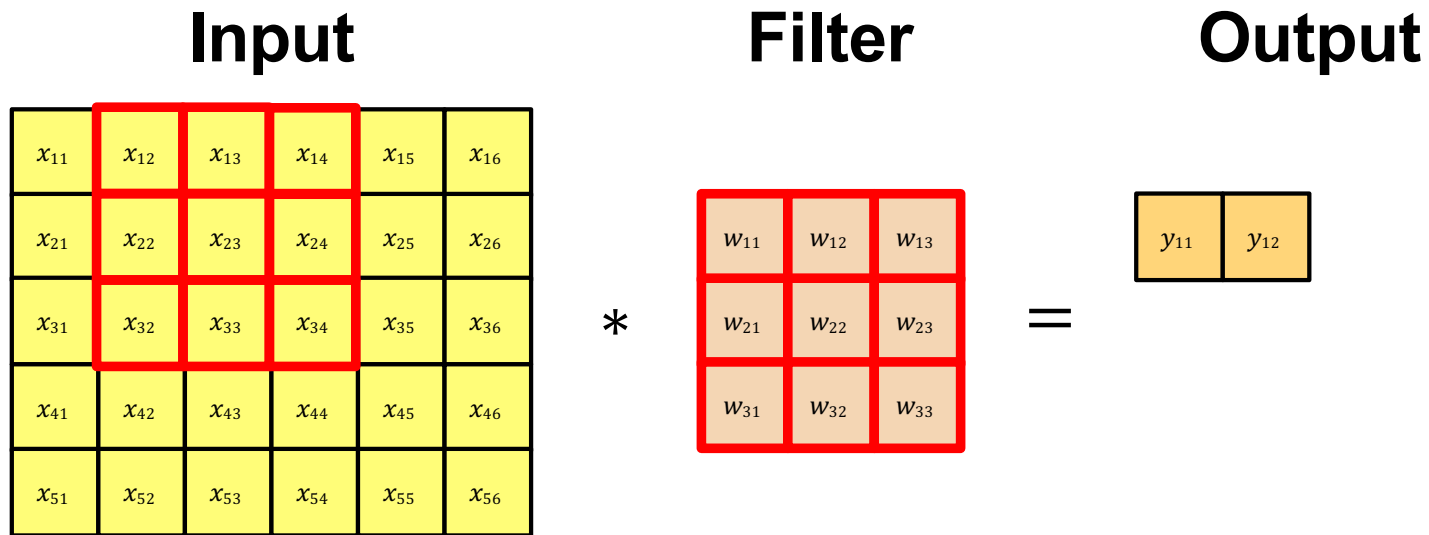
Convolution example



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

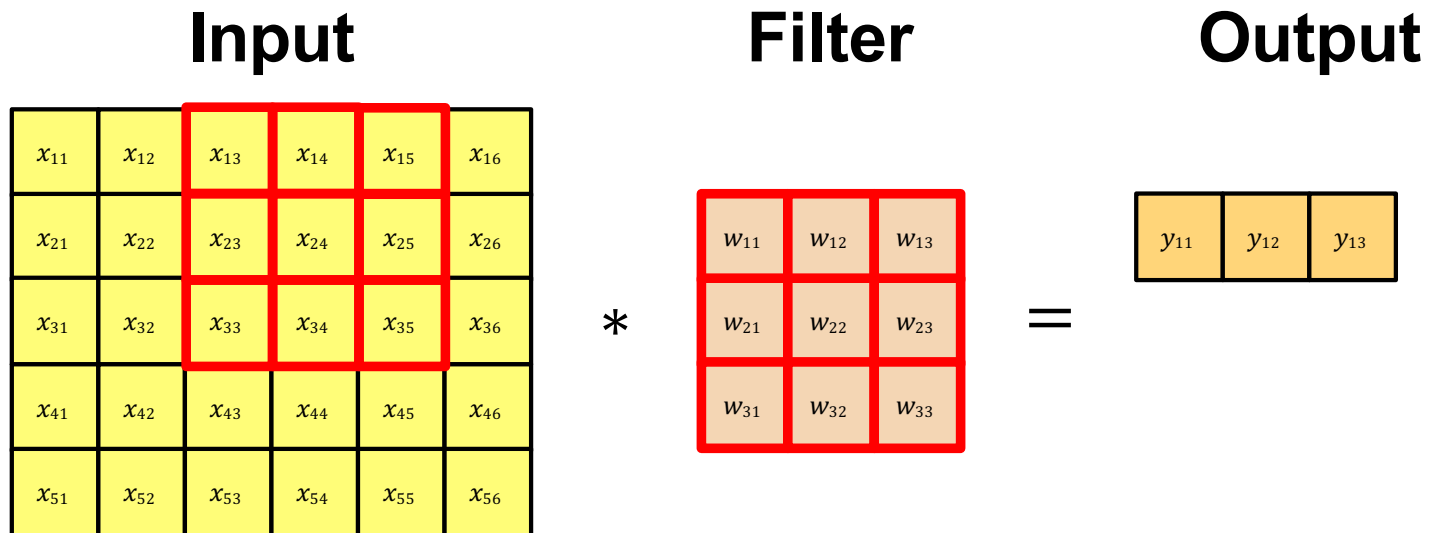
Adapted from [D. Fouhey and J. Johnson](#)

Convolution example



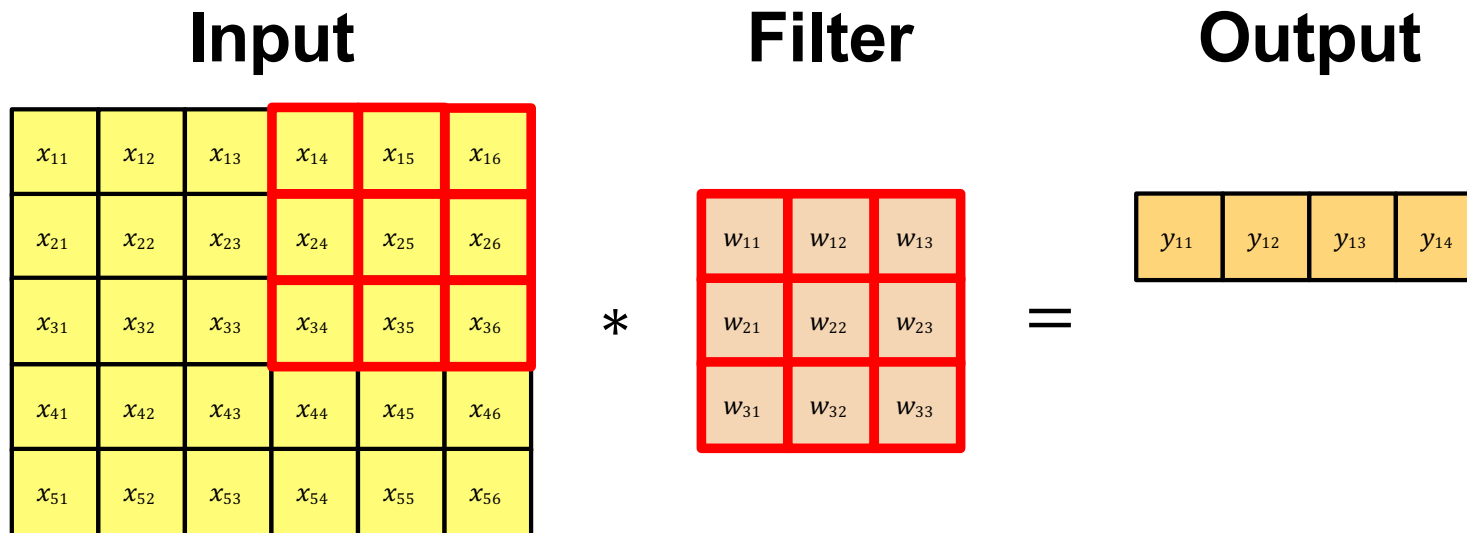
$$y_{12} = x_{12} \cdot w_{11} + x_{13} \cdot w_{12} + x_{14} \cdot w_{13} + \dots + x_{34} \cdot w_{33}$$

Convolution example



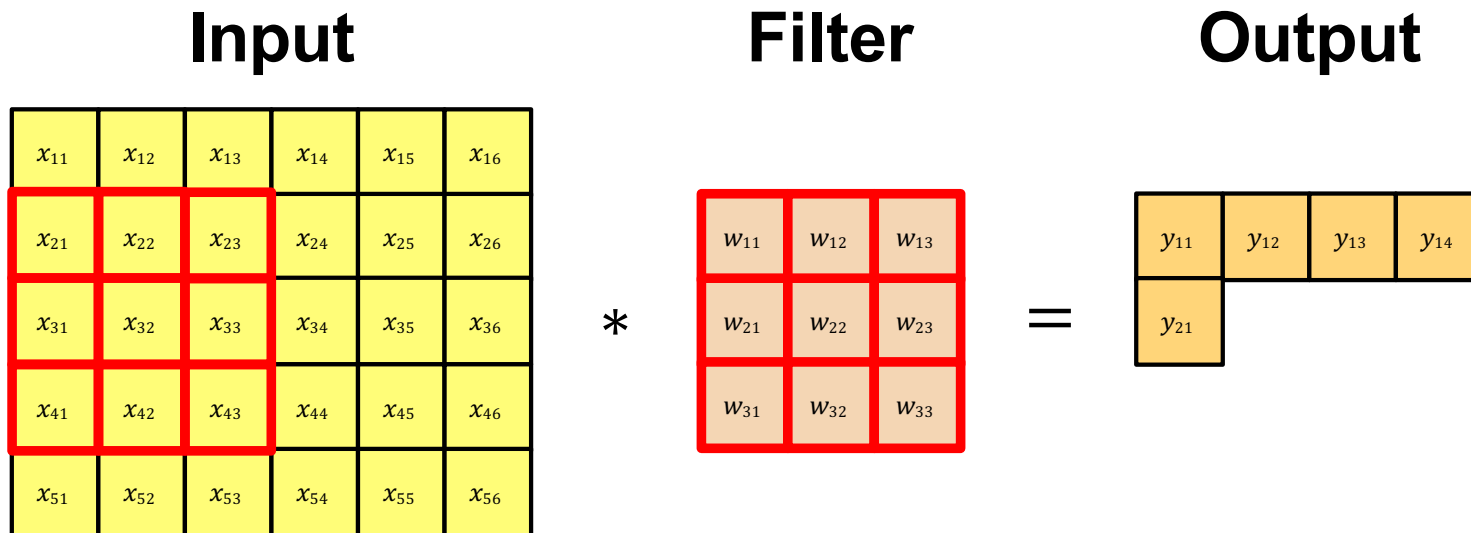
$$y_{13} = x_{13} \cdot w_{11} + x_{14} \cdot w_{12} + x_{15} \cdot w_{13} + \dots + x_{35} \cdot w_{33}$$

Convolution example



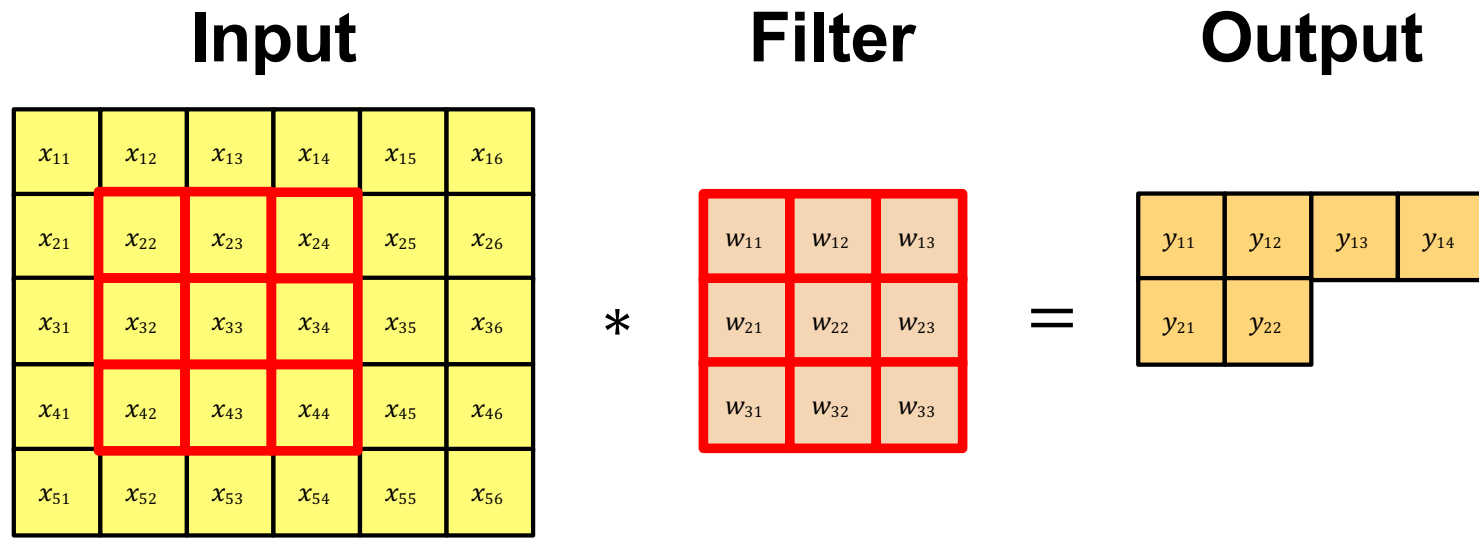
$$y_{14} = x_{14} \cdot w_{11} + x_{15} \cdot w_{12} + x_{16} \cdot w_{13} + \dots + x_{36} \cdot w_{33}$$

Convolution example



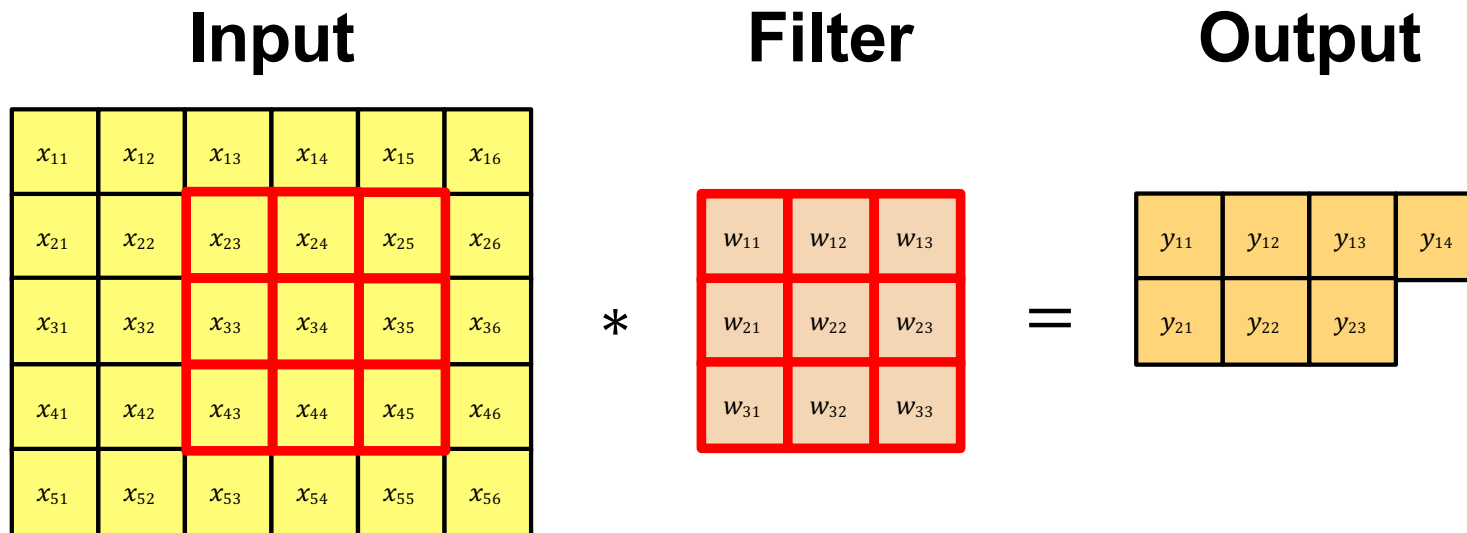
$$y_{21} = x_{21} \cdot w_{11} + x_{22} \cdot w_{12} + x_{23} \cdot w_{13} + \dots + x_{43} \cdot w_{33}$$

Convolution example



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

Convolution example



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

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}

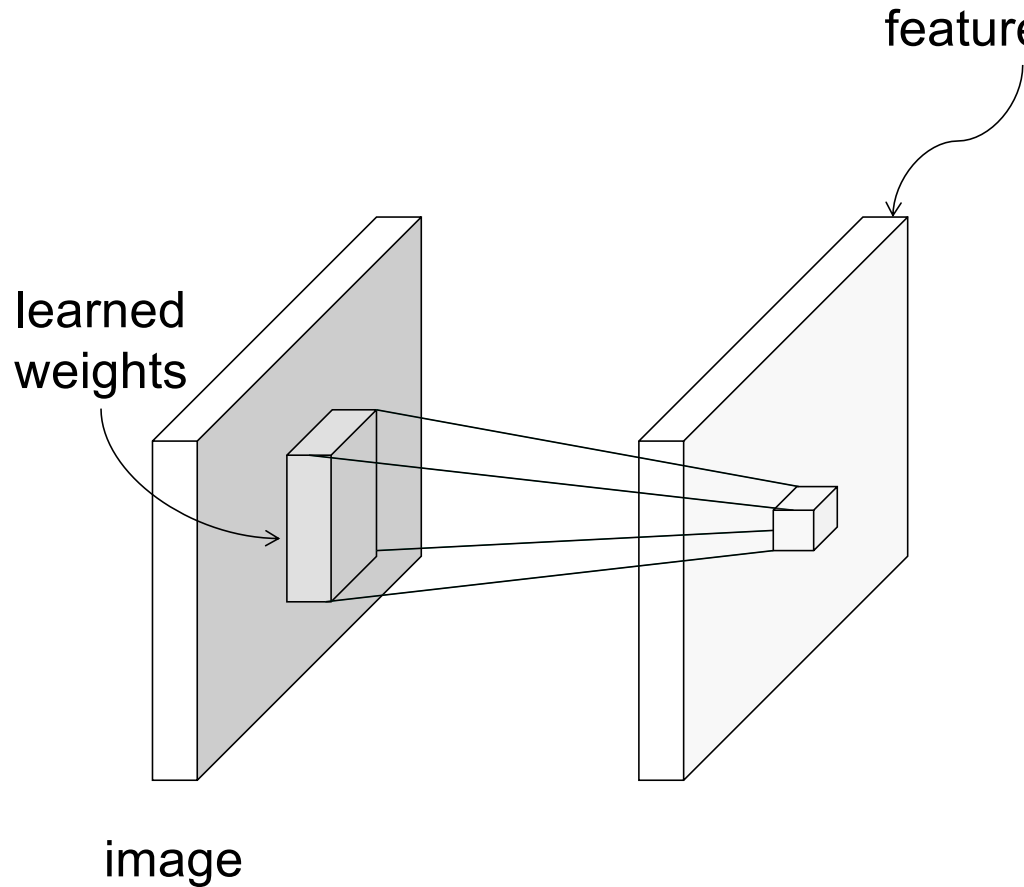
*

=

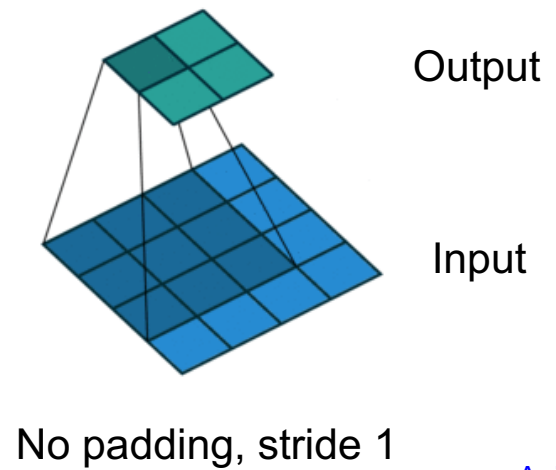
Output

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

Convolutional architecture

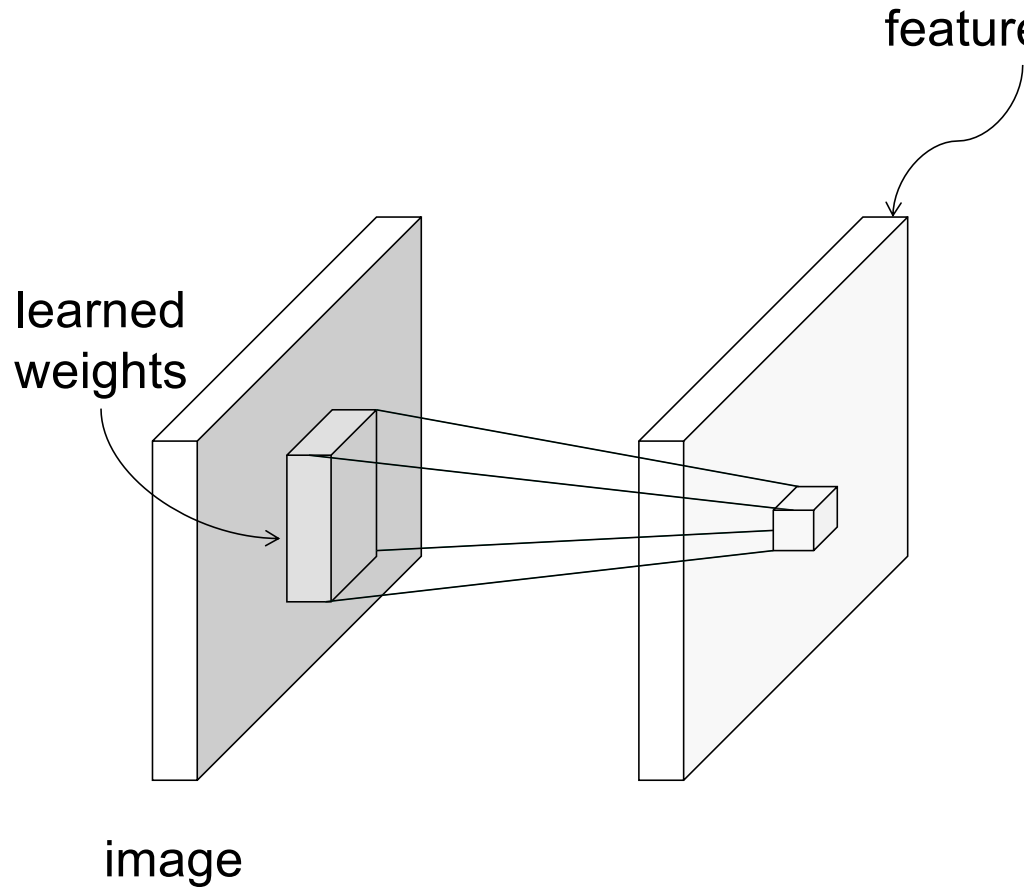


Output feature map resolution depends on *padding* and *stride*

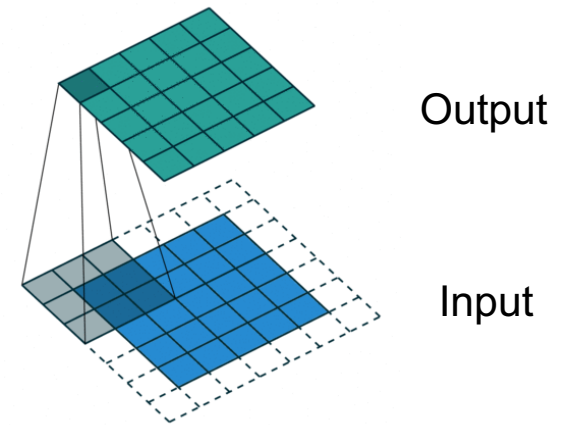


[Animation source](#)

Convolutional architecture



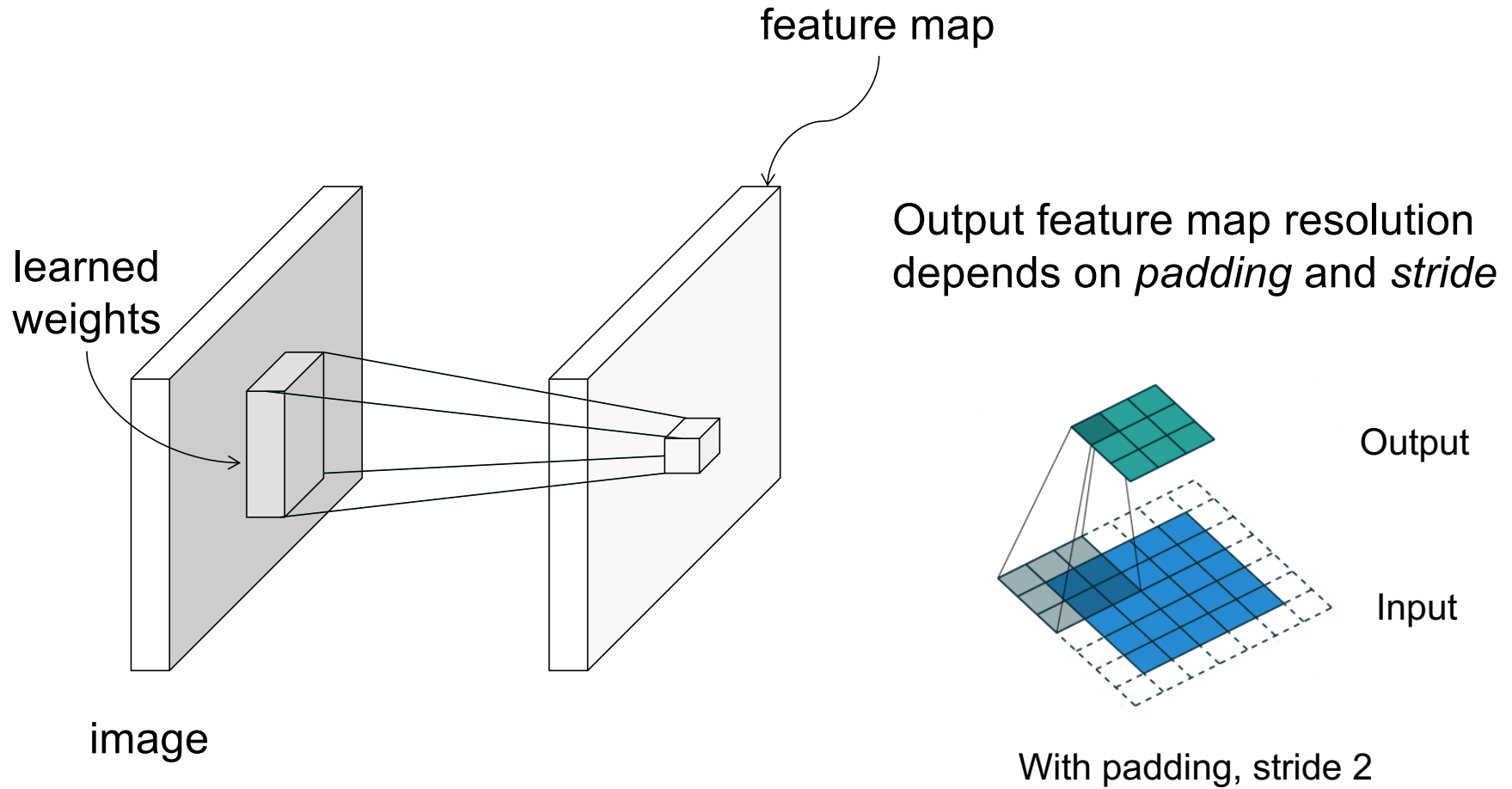
Output feature map resolution depends on *padding* and *stride*



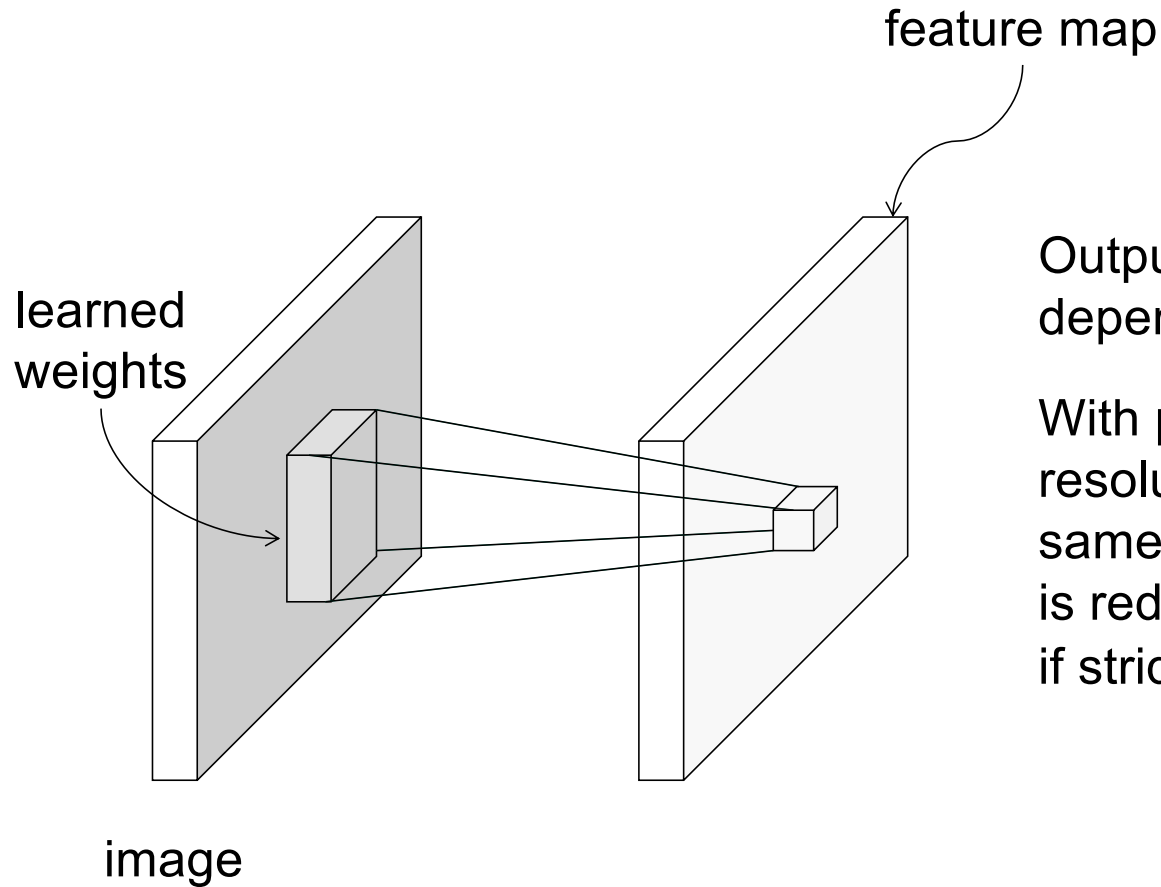
With padding, stride 1

[Animation source](#)

Convolutional architecture



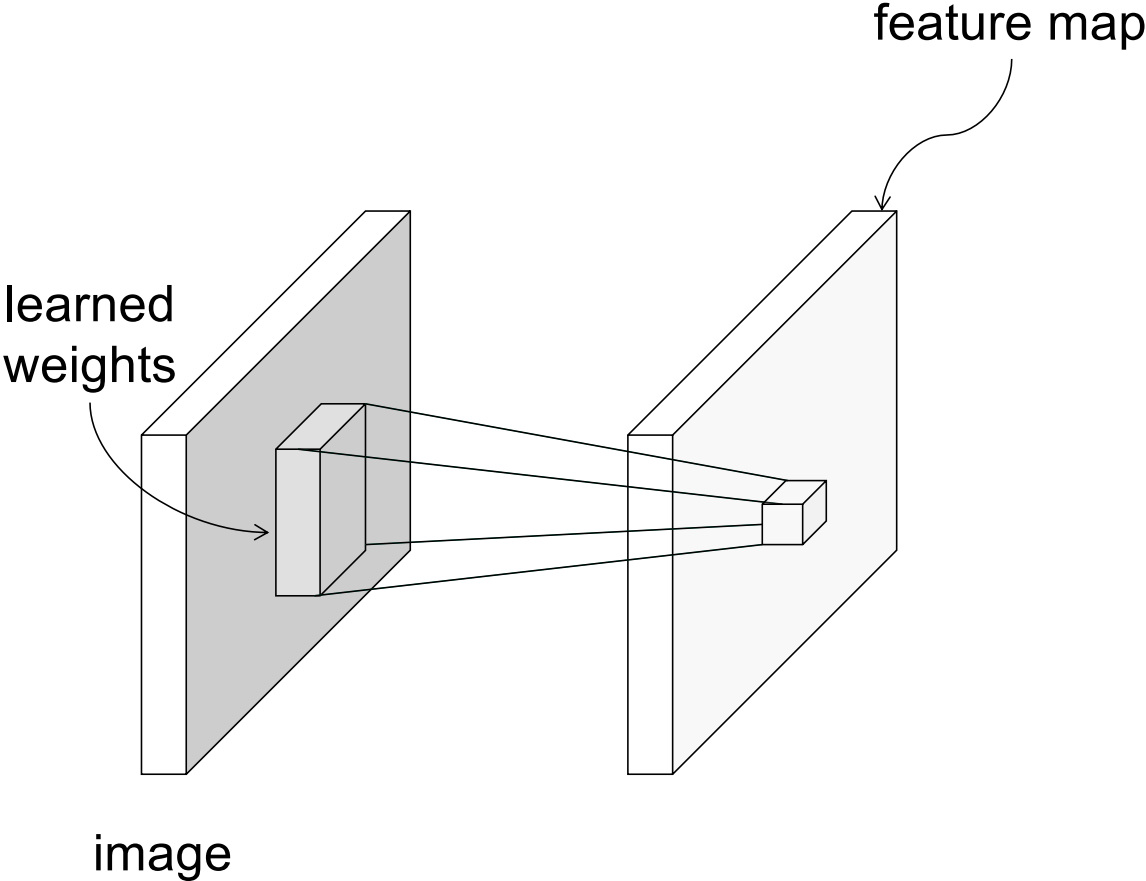
Convolutional architecture



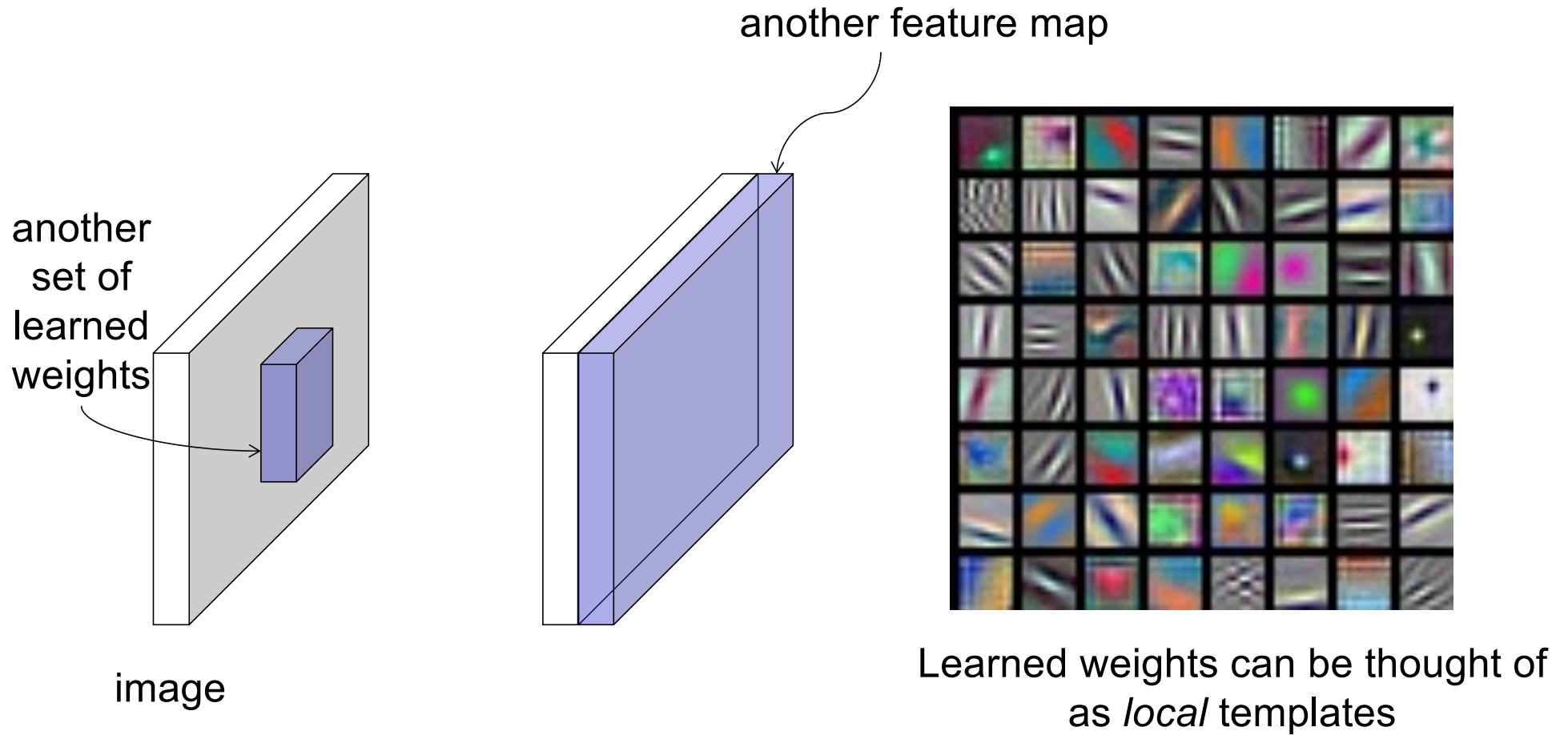
Output feature map resolution depends on *padding* and *stride*

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

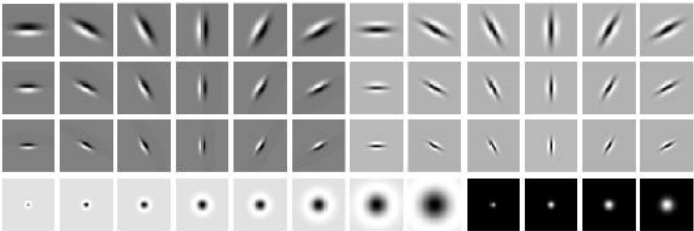


Convolutional architecture



Convolution and traditional feature extraction

bank of K filters



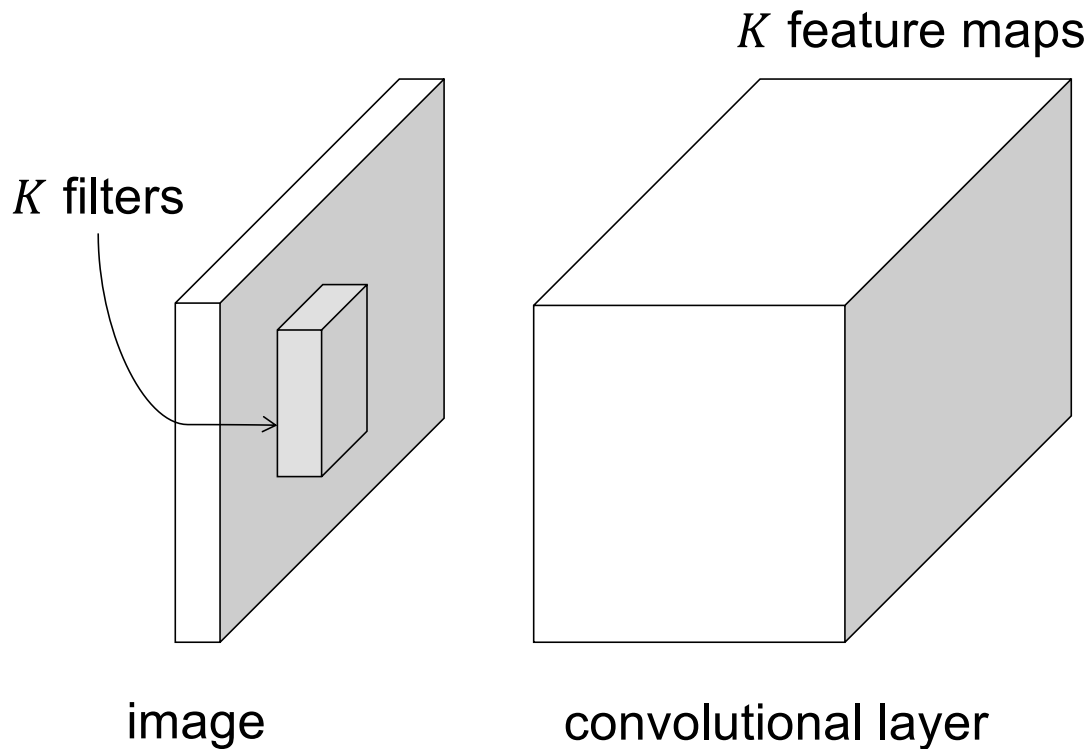
K feature maps



image

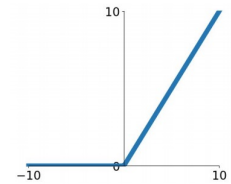
feature map

Elementwise nonlinearity



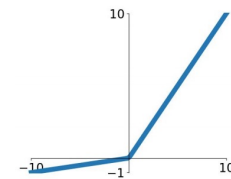
Almost always directly followed by a ReLU:

$$\max(0, x)$$

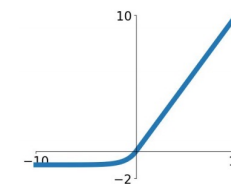


Some alternatives to ReLU:

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



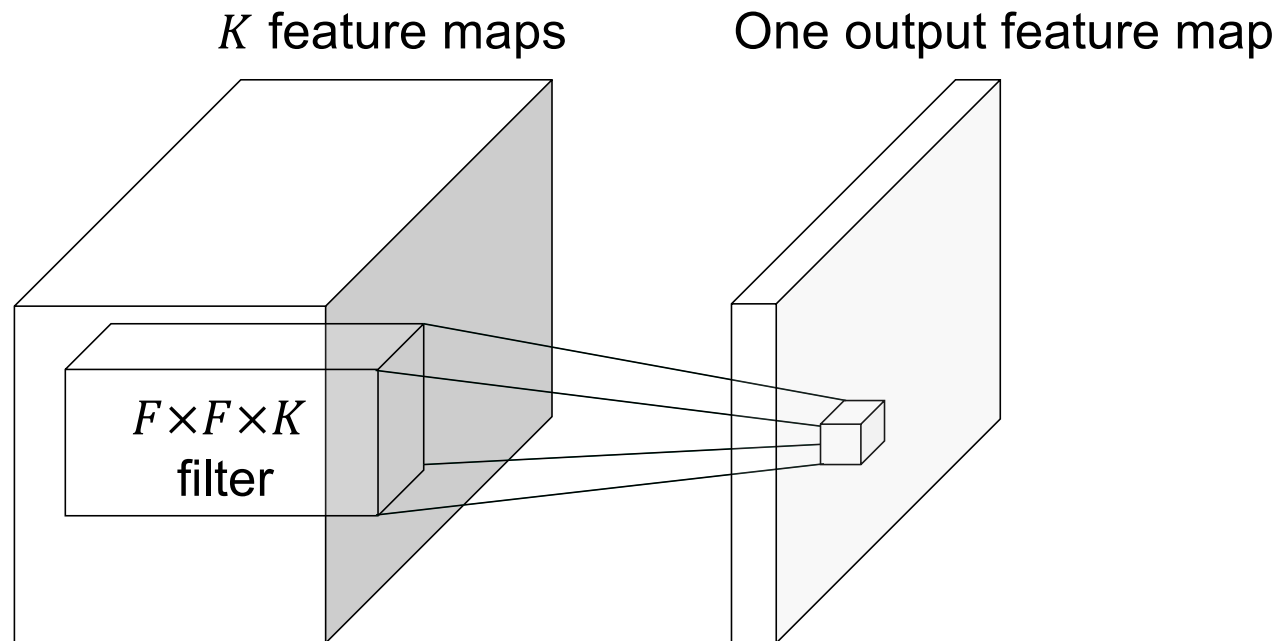
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Source: [Stanford 231n](#)

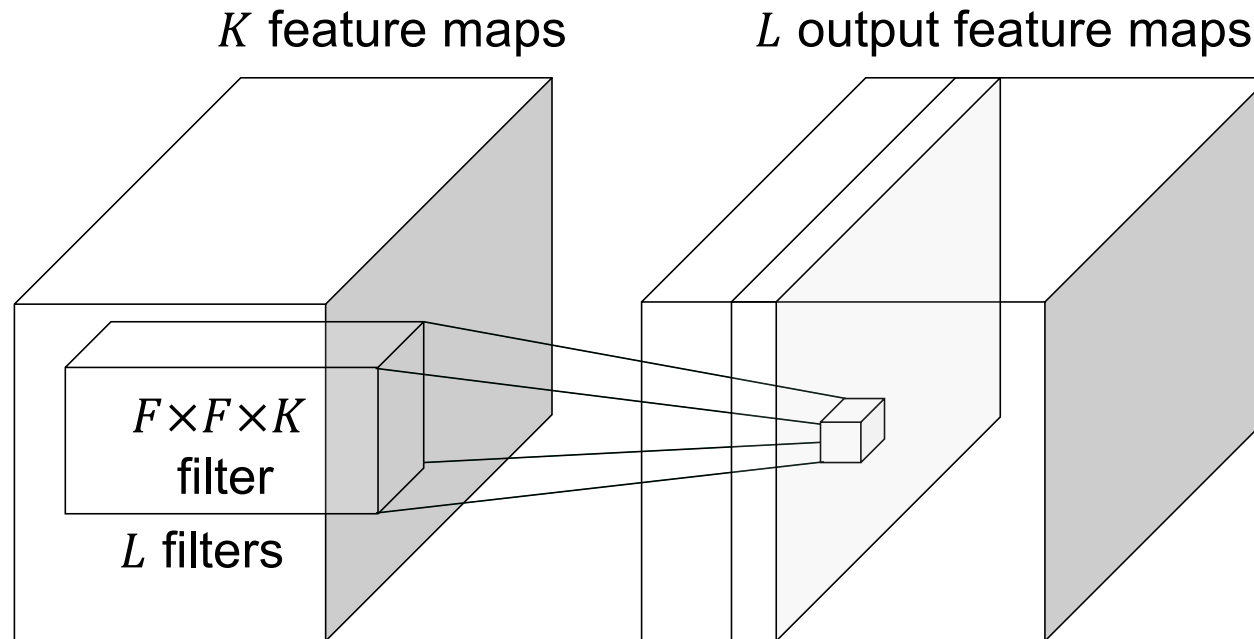
Three-dimensional convolutions

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

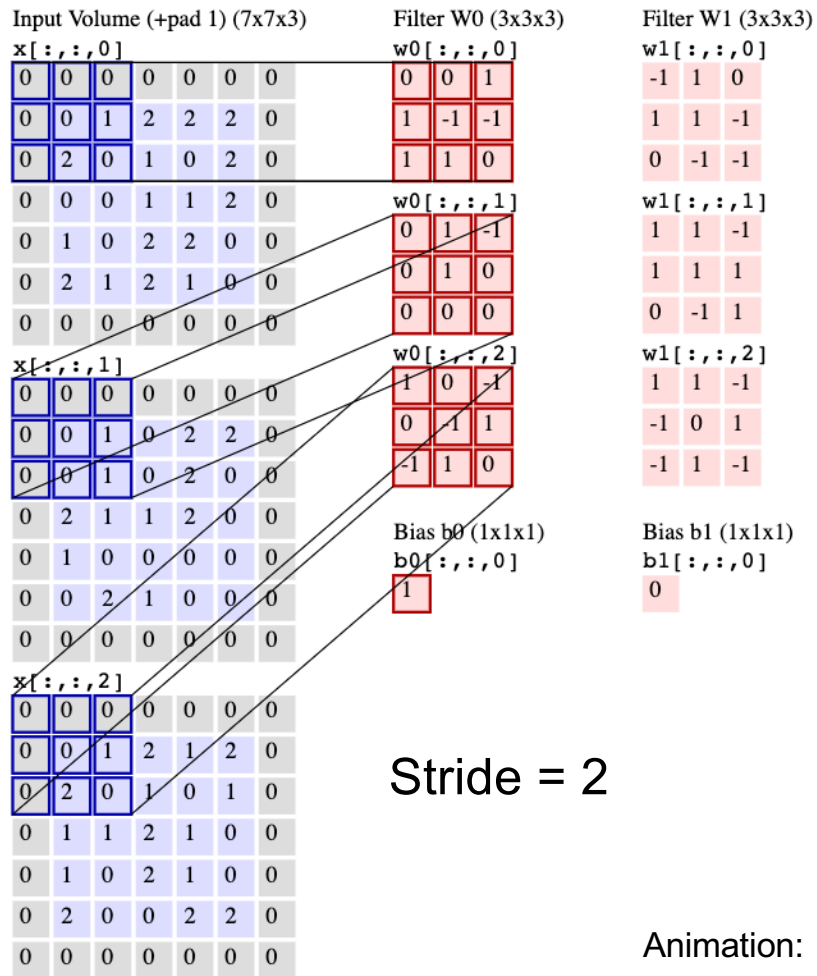


Three-dimensional convolutions

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

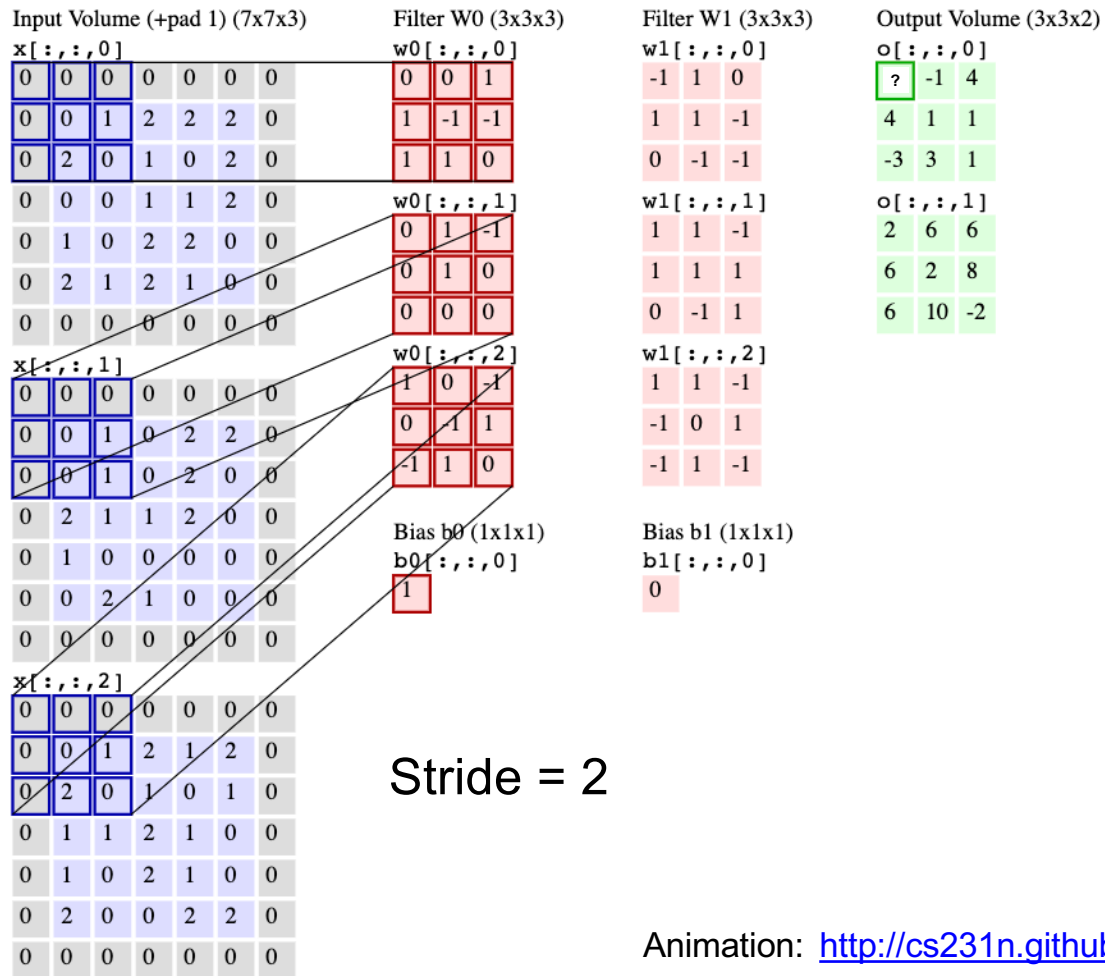


Convolutional layer example



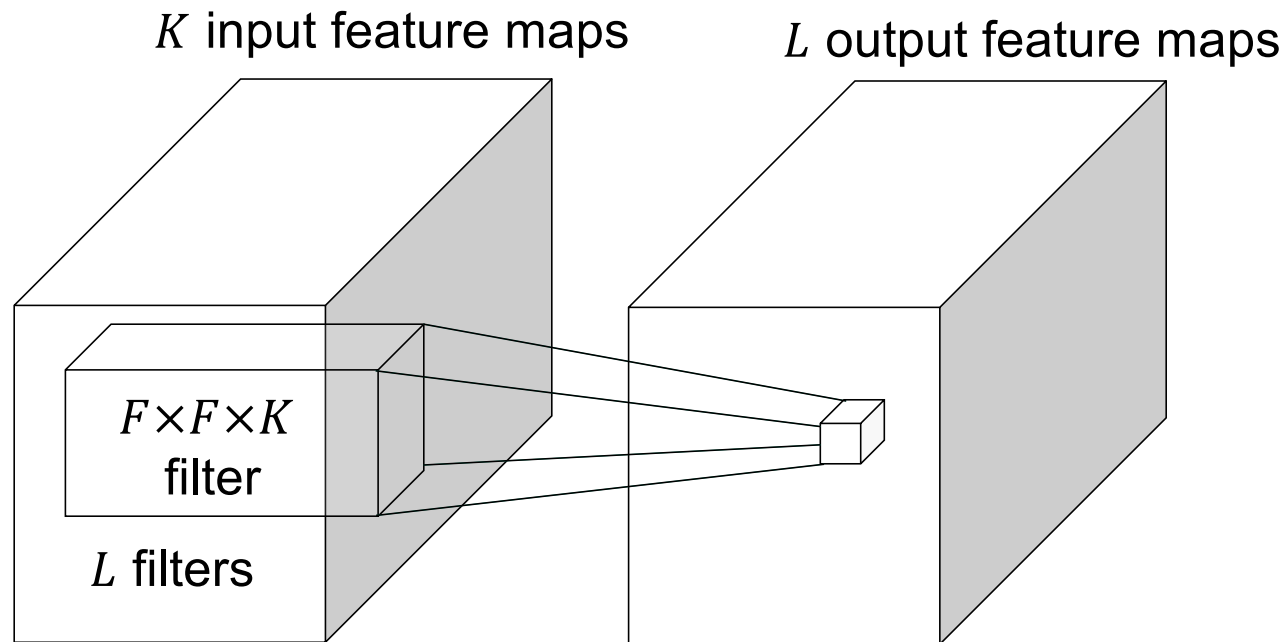
Animation: <http://cs231n.github.io/convolutional-networks/#conv>

Convolutional layer example



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

Convolutional layer: Computational cost

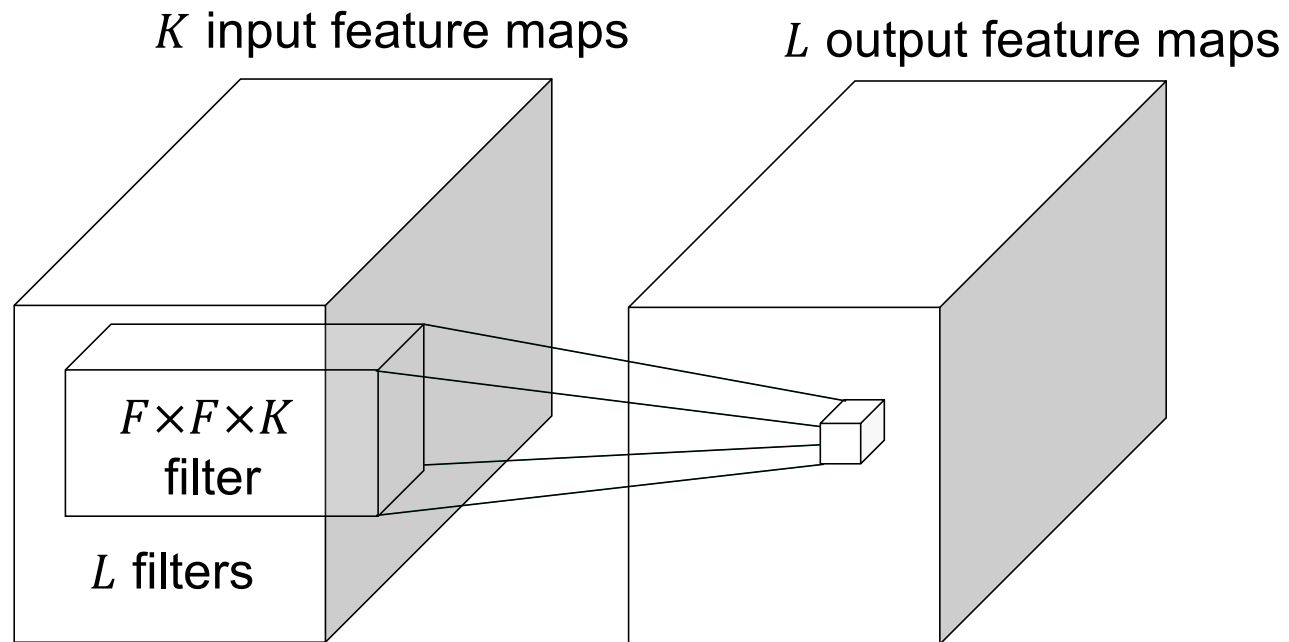


- Assuming the input feature maps have spatial resolution $H \times W$, how many operations are needed to compute the output feature volume?
 - F^2KLHW

Outline

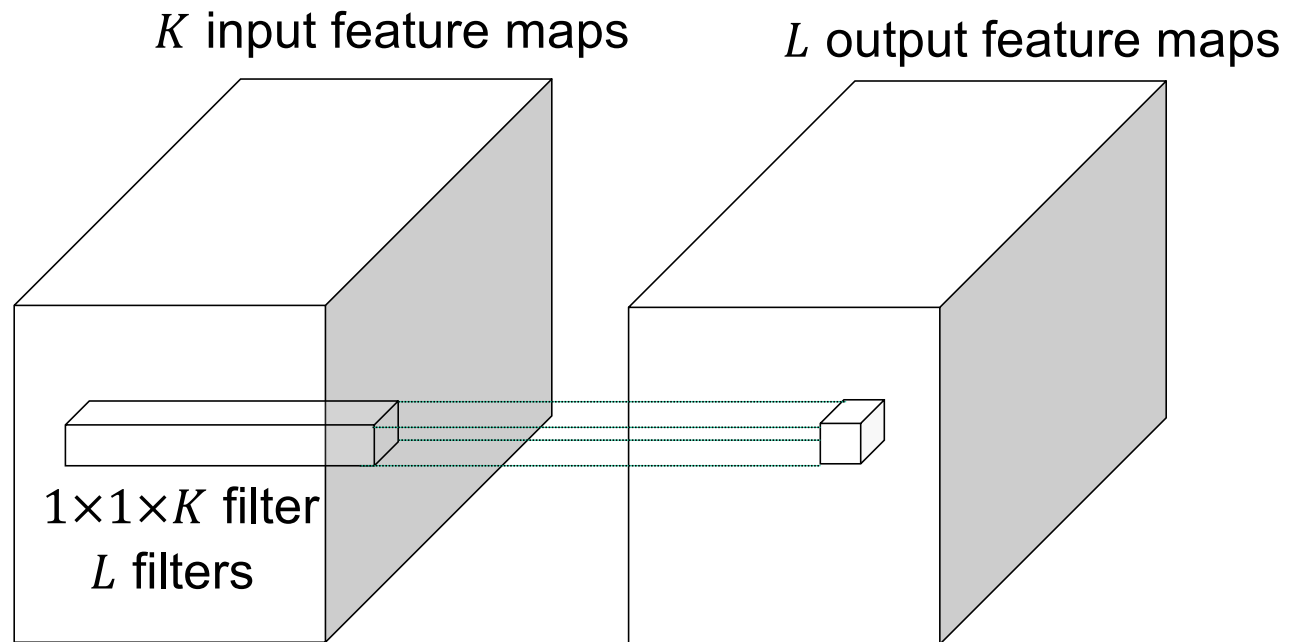
- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions

1x1 convolutional layer



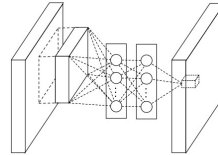
What if we make $F = 1$?

1x1 convolutional layer

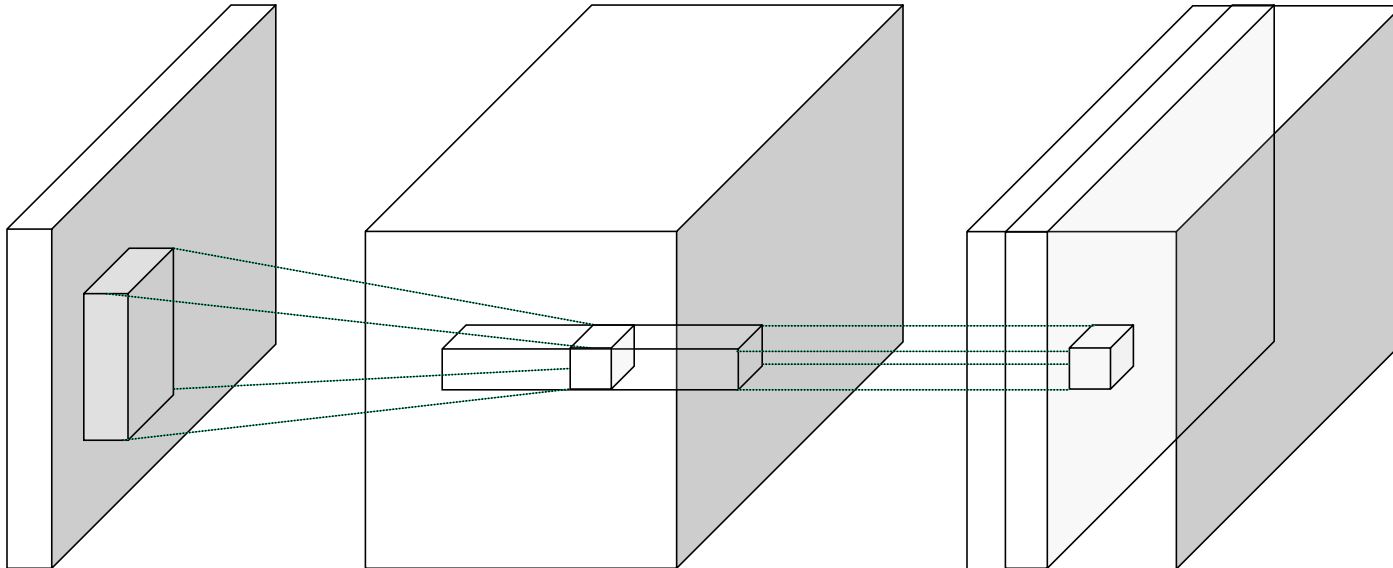


What if we make $F = 1$?

Why 1x1 convolutions?



“network in network”

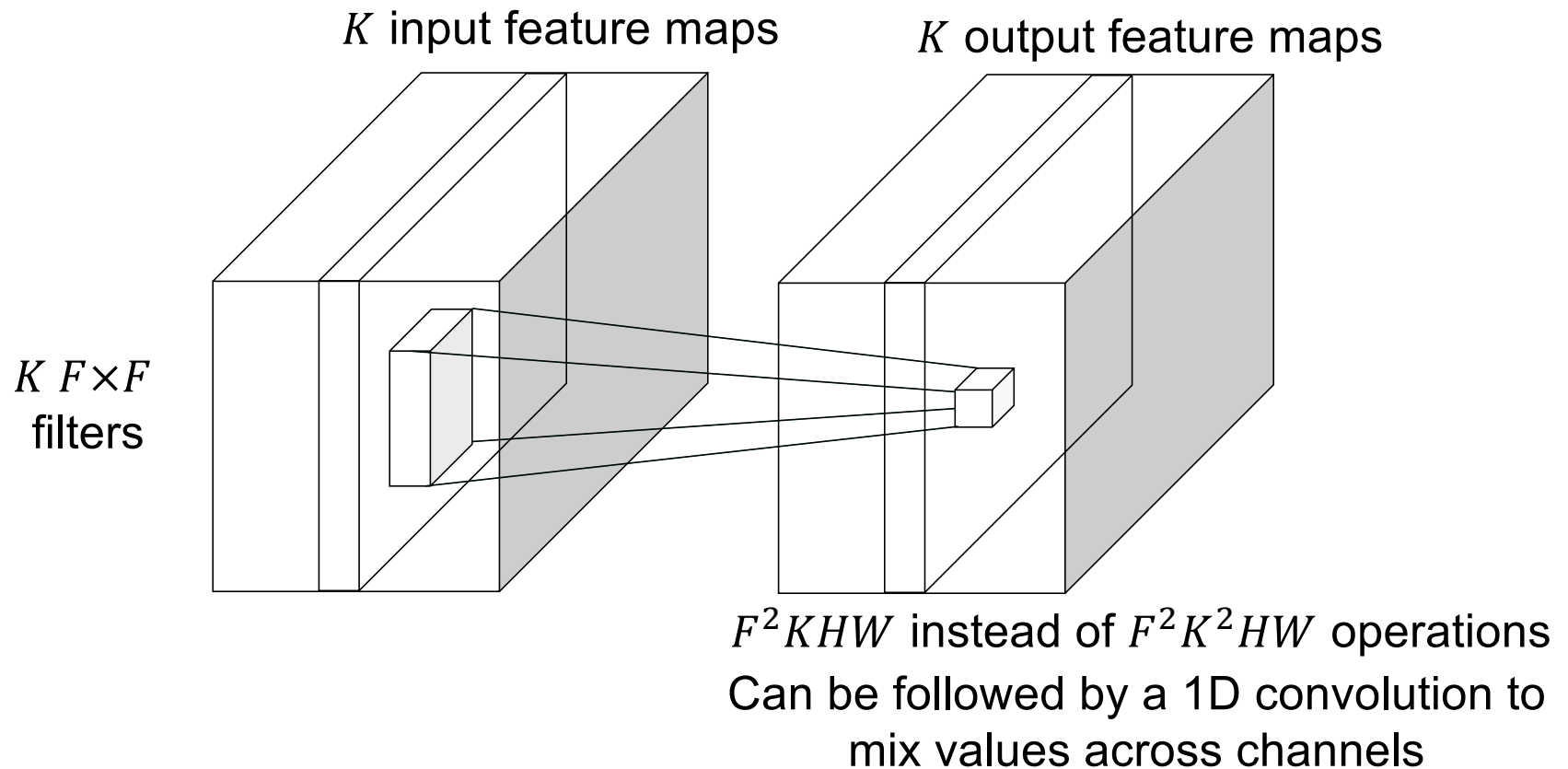


M. Lin, Q. Chen, and S. Yan, [Network in network](#), ICLR 2014

Why 1x1 convolutions?

- **Option 1:** 3×3 conv layer with 256 channels at input and output
 - $256 \times 256 \times 3 \times 3 \approx 600,000$ weights and operations (per location)
- **Option 2: “bottleneck module”**
 - 1×1 conv layer, 256 → 64 channels
 - 3×3 conv layer, 64 → 64 channels
 - 1×1 conv layer, 64 → 256 channels
 - $256 \times 64 \times 1 \times 1 \approx 16,000$
 - $64 \times 64 \times 3 \times 3 \approx 36,000$
 - $64 \times 256 \times 1 \times 1 \approx 16,000$
 - Total $\approx 70,000$ weights and operations

Alternative to 3D convolutions: Depthwise convolutions



MobileNets: Depthwise separable convolutions

- **Depthwise separable convolution block:**

- 3×3 conv layer, 256 → 256 channels (for example)

- 1×1 conv layer, $K \rightarrow K$ channels

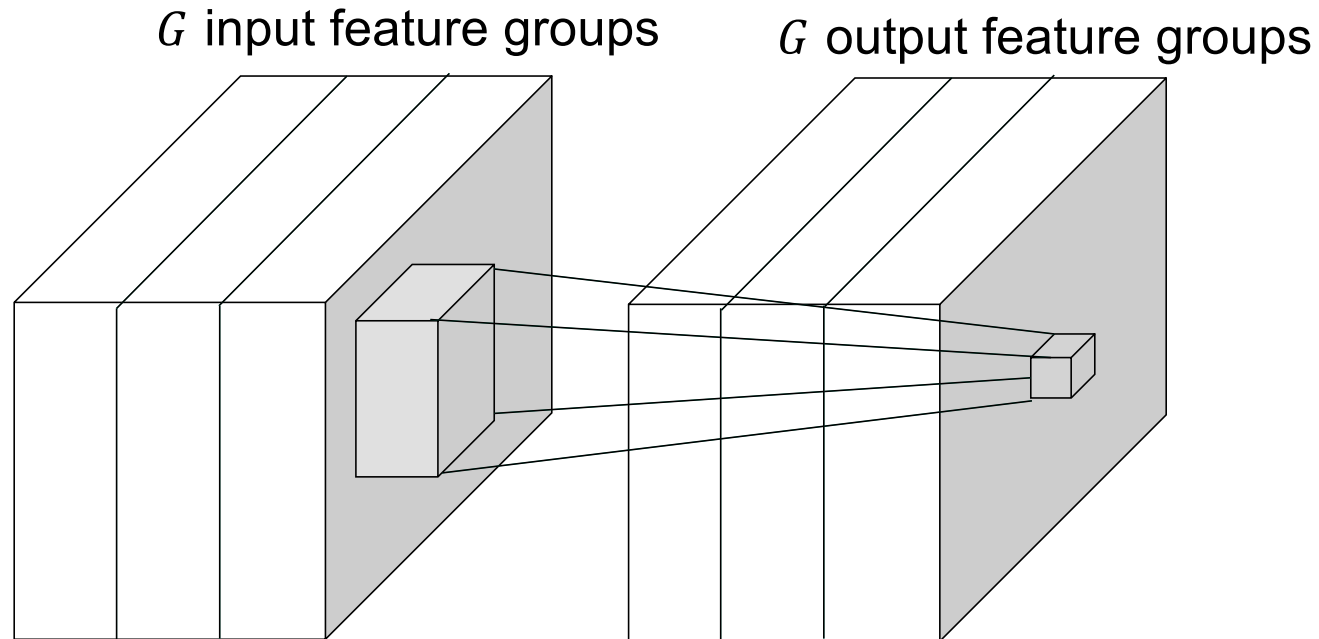
- $9 \times 256 \approx 2,300$ operations (per location)
 - $256 \times 256 \approx 65,500$ operations
 - Total $\approx 67,800$ operations

- Regular 3×3 conv layer, 256 → 256 channels

- $9 \times 256 \times 256 \approx 590,000$ operations

- Speedup factor: 8.7

More generally: Groupwise convolutions

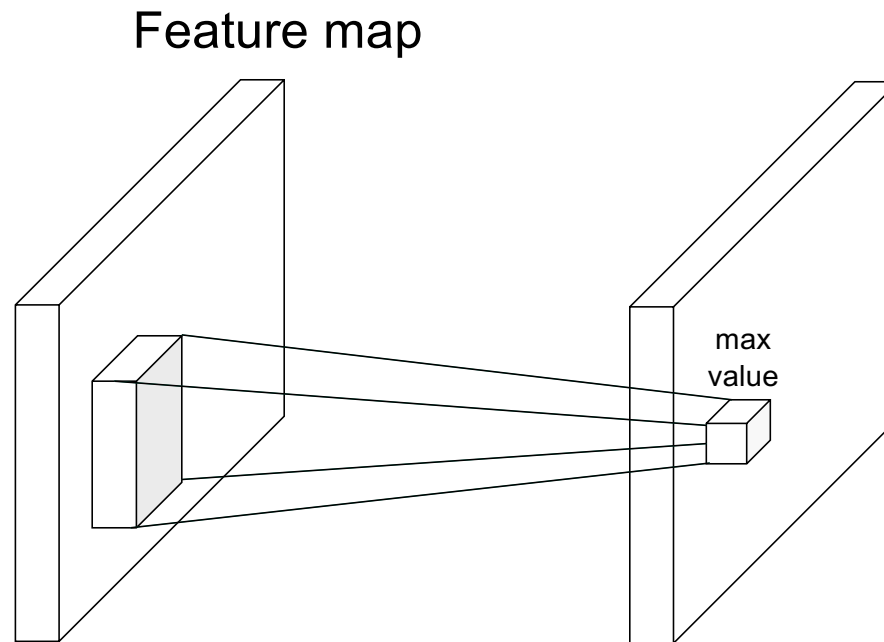


- Split up the K feature maps into G groups, perform convolutions within each group separately, concatenate the results

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

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



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



5	7
3	3

Max pooling: Example

Single channel

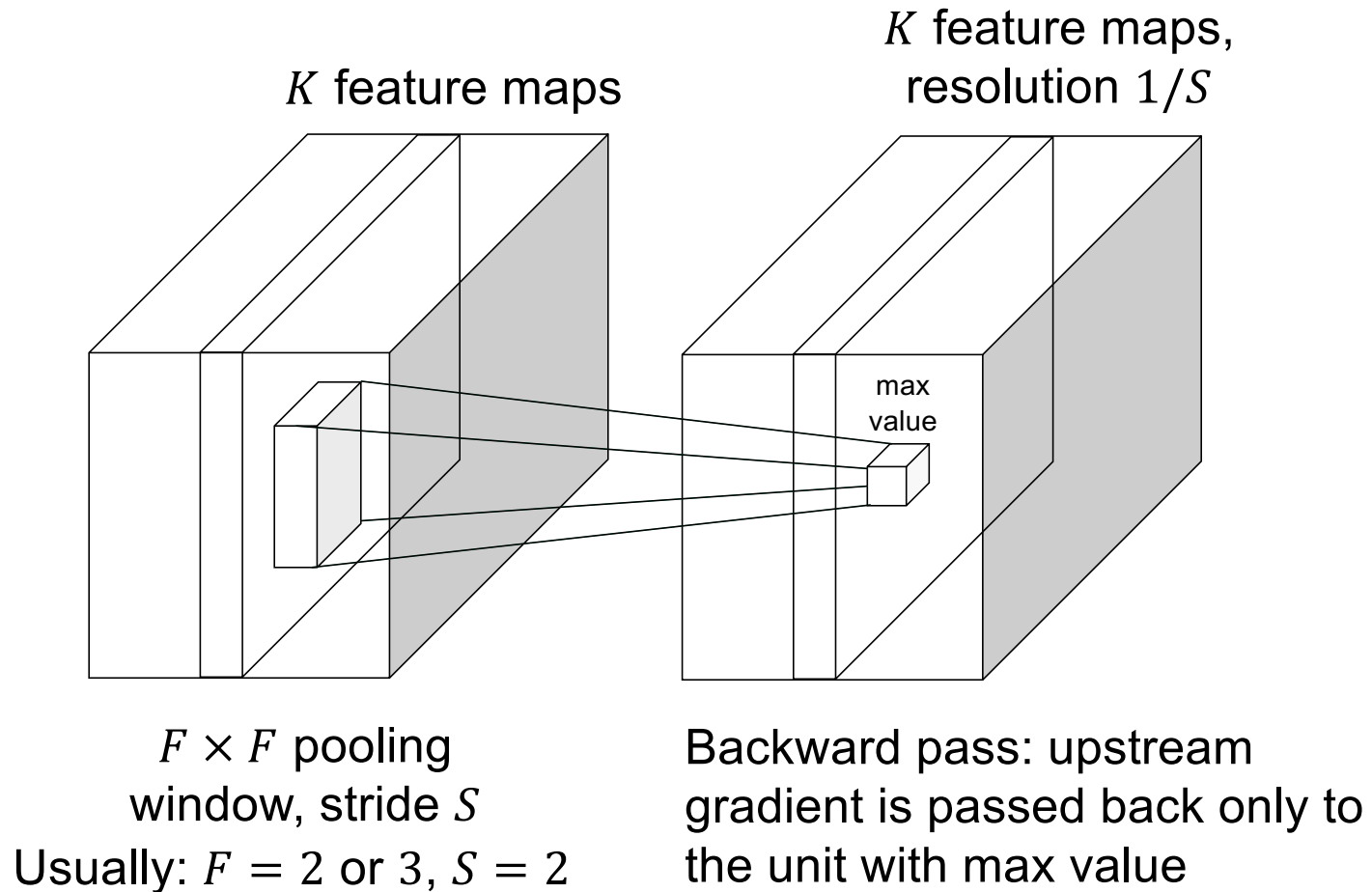
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2×2
kernel size and stride 2

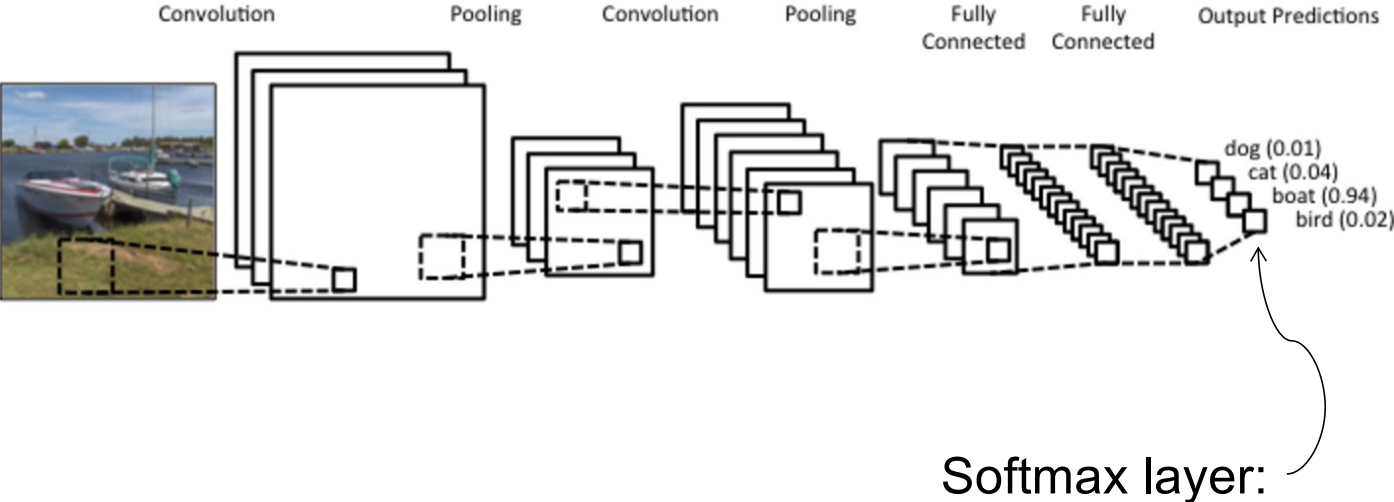


6	8
3	4

Max pooling layer



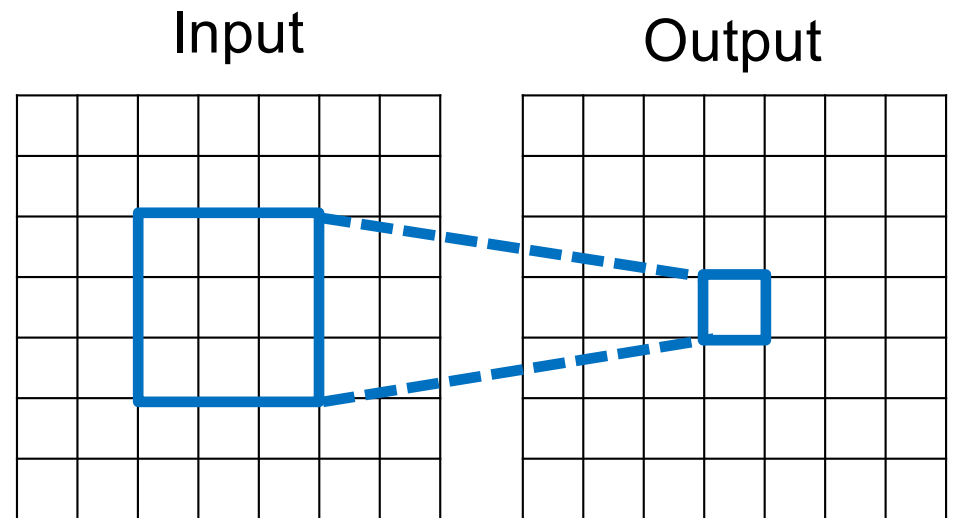
Simplified CNN pipeline



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)

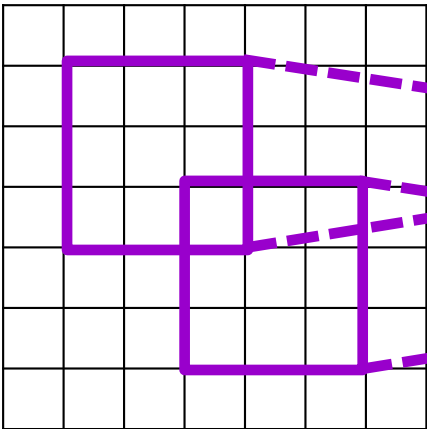


Receptive field size: 3

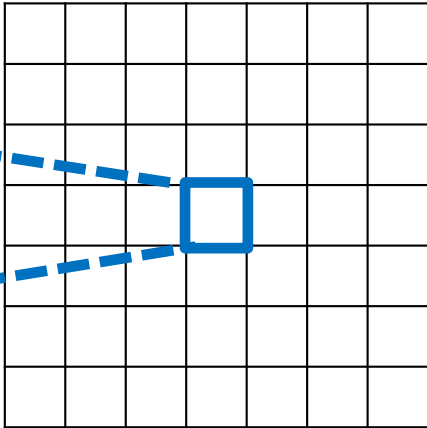
Receptive field

3x3 convolutions, stride 1

Input



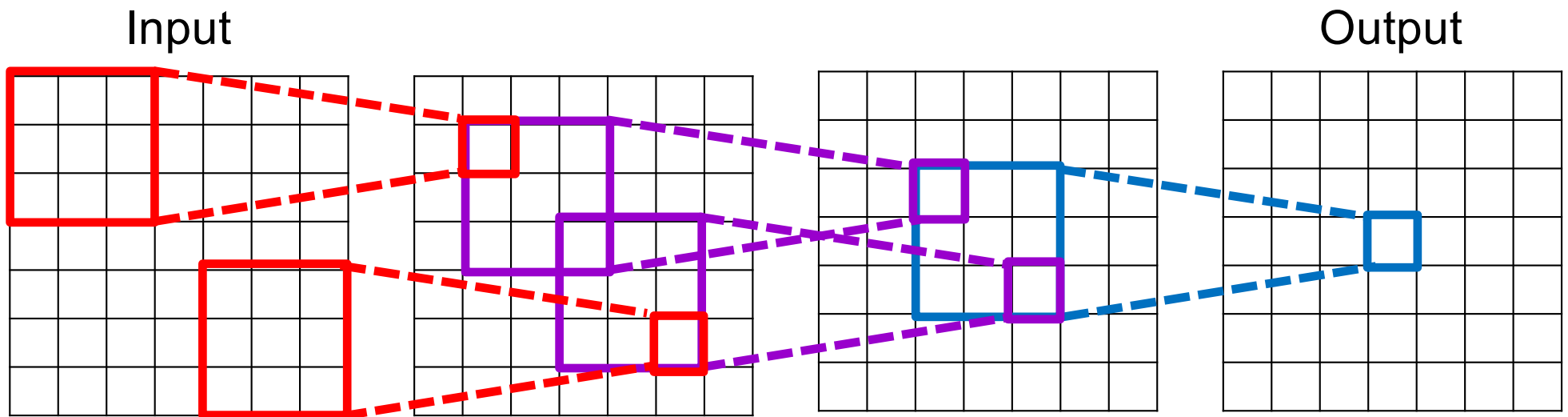
Output



Receptive field size: 5

Receptive field

3x3 convolutions, stride 1

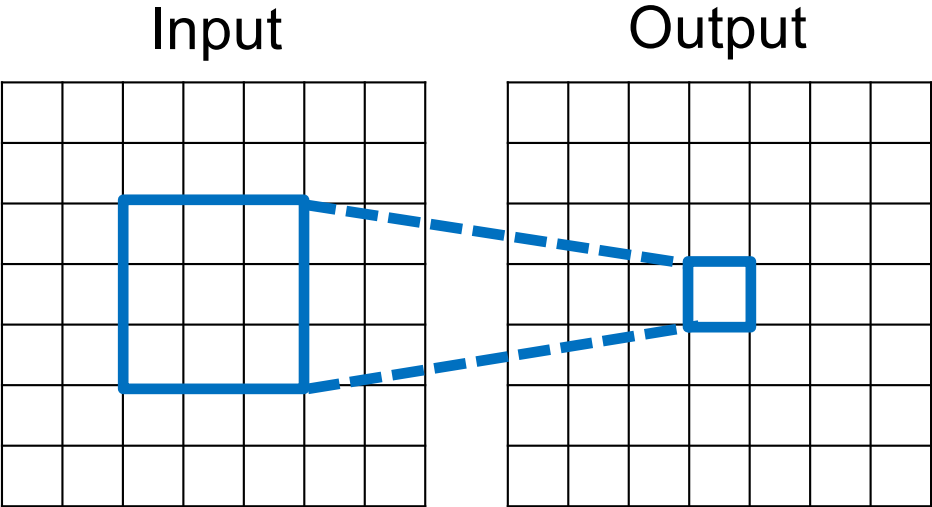


Receptive field size: 7

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

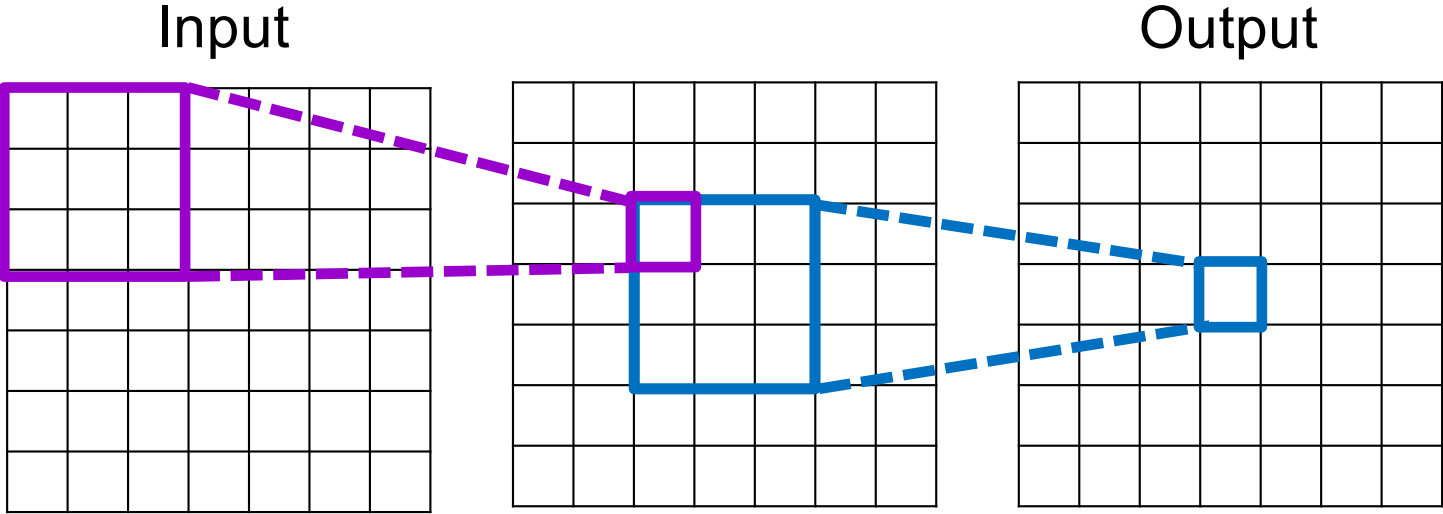
3x3 convolutions, stride 2



Receptive field size: 3

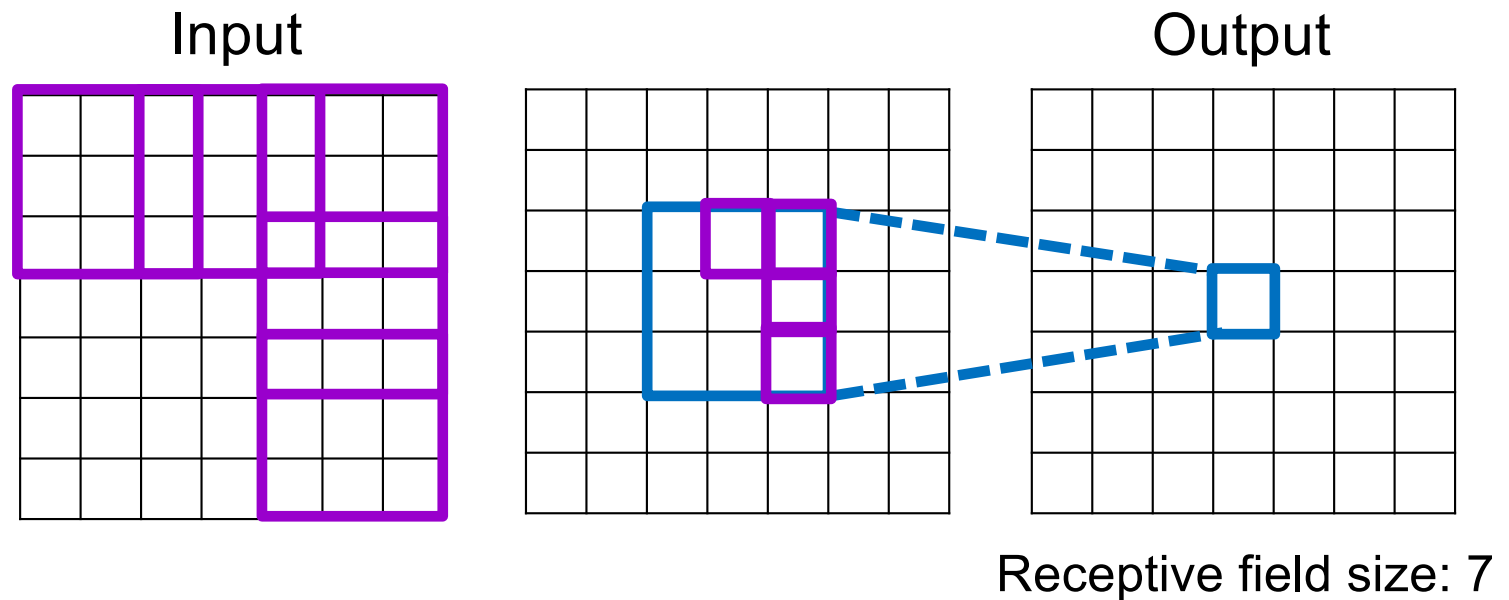
Receptive field

3x3 convolutions, stride 2



Receptive field

3x3 convolutions, stride 2



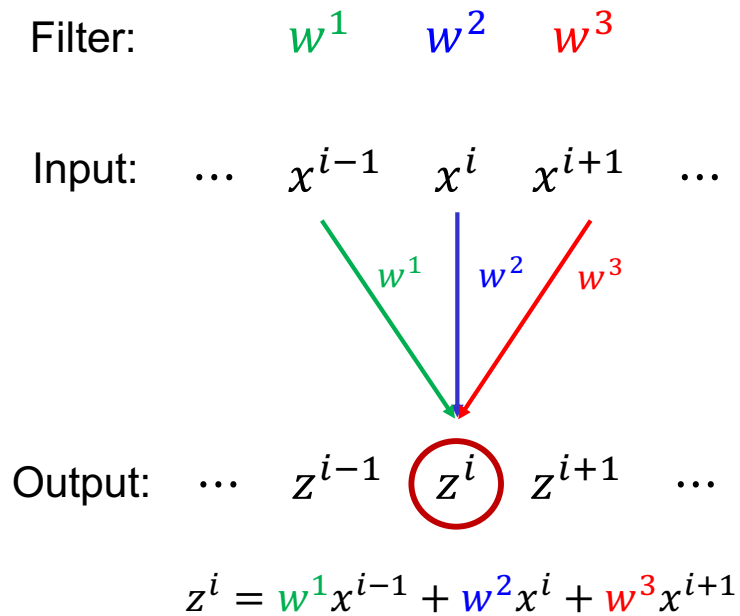
With a stride of 2, receptive field size is given by $2^{L+1} - 1$, i.e., it grows exponentially (though spatial resolution decreases exponentially)

Outline

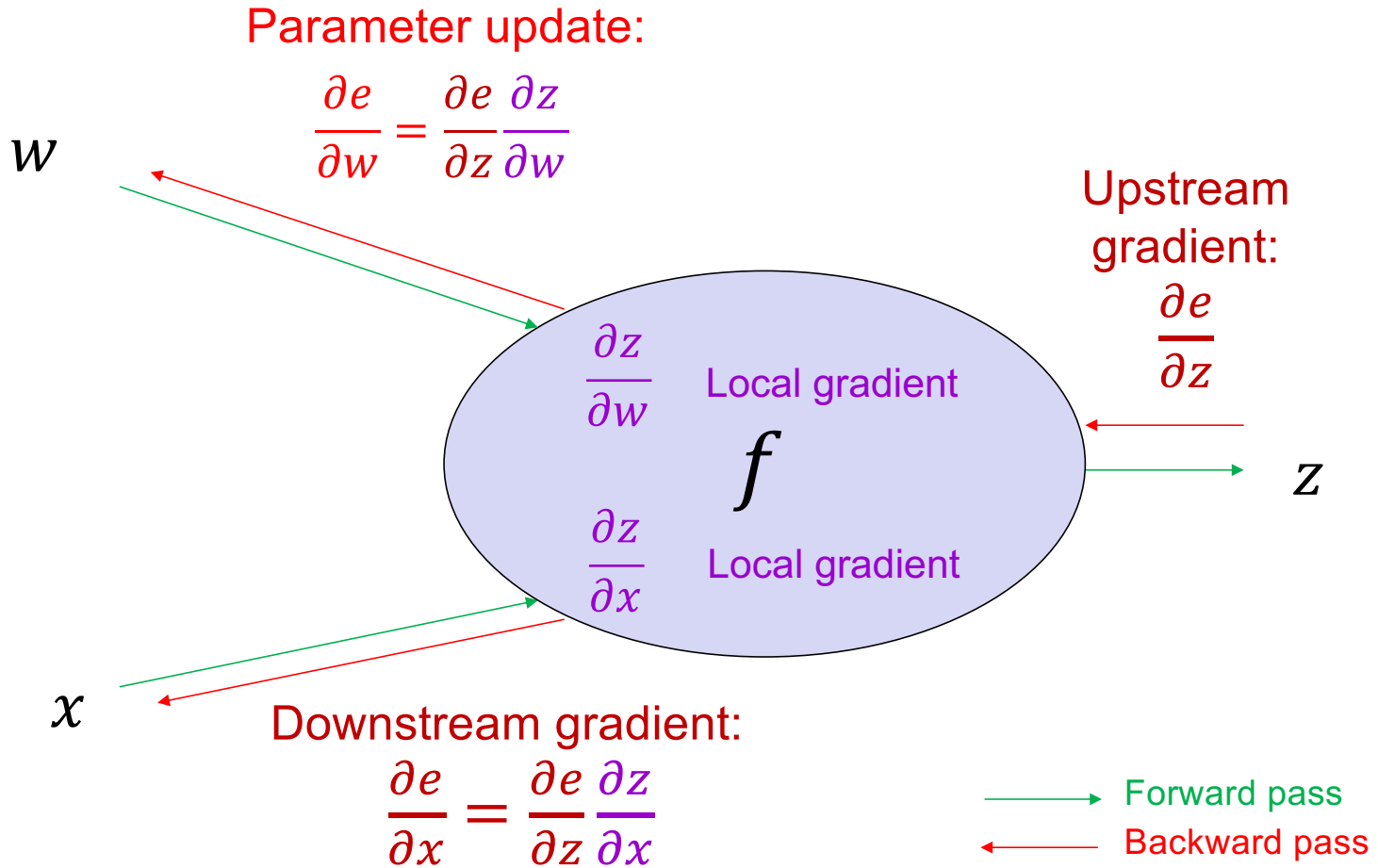
- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions
- Max pooling layer
- **Backward pass**

Convolutional layer: Backward pass

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



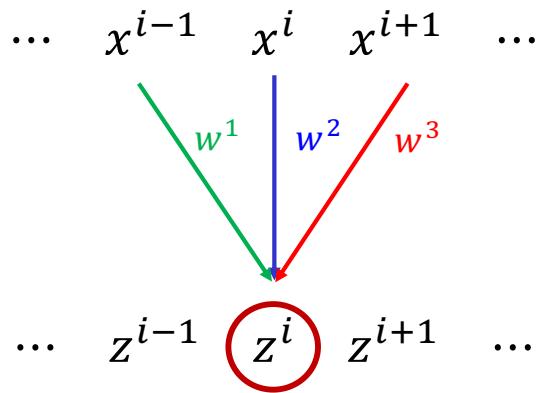
Review: Backward pass



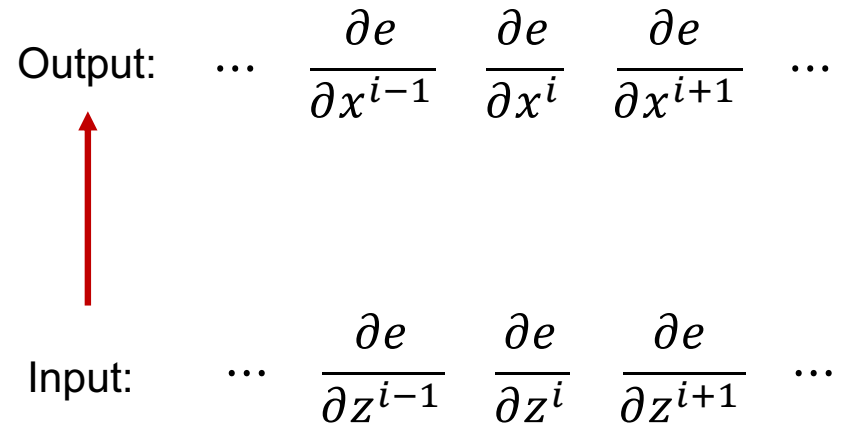
Convolutional layer: Backward pass

Backward pass (w.r.t. x)

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



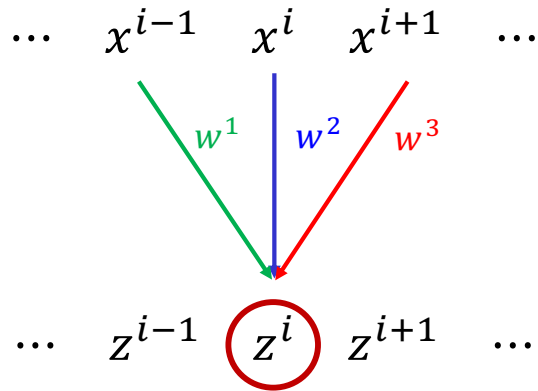
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



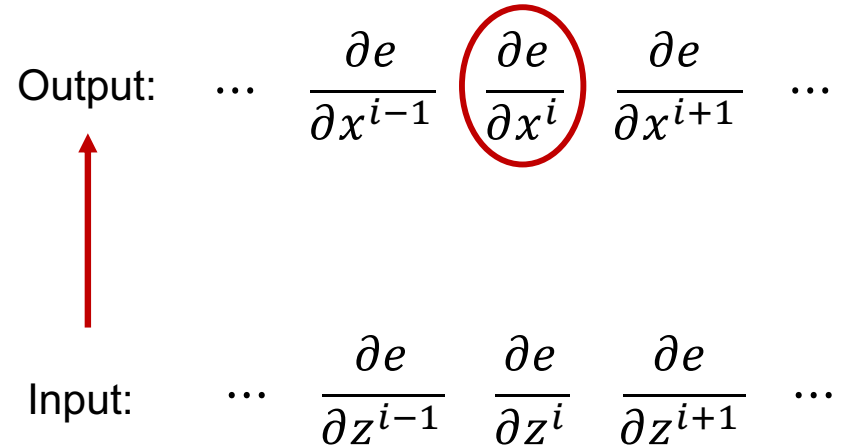
Convolutional layer: Backward pass

Backward pass (w.r.t. x)

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



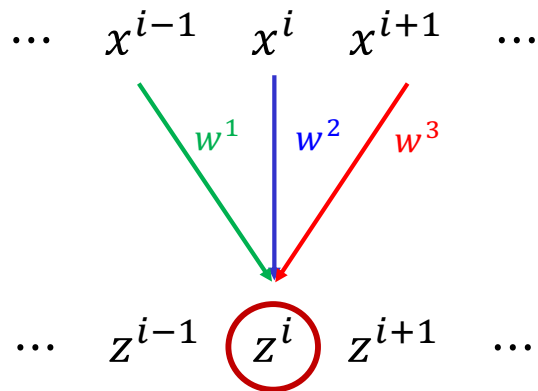
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



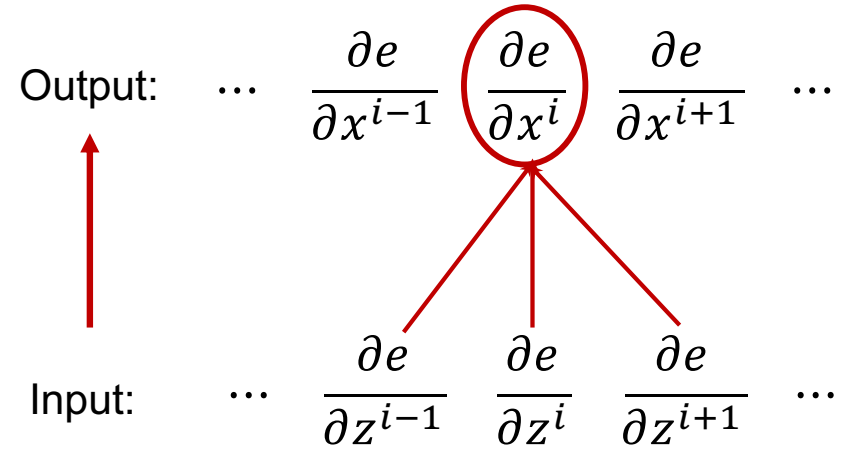
Convolutional layer: Backward pass

Backward pass (w.r.t. x)

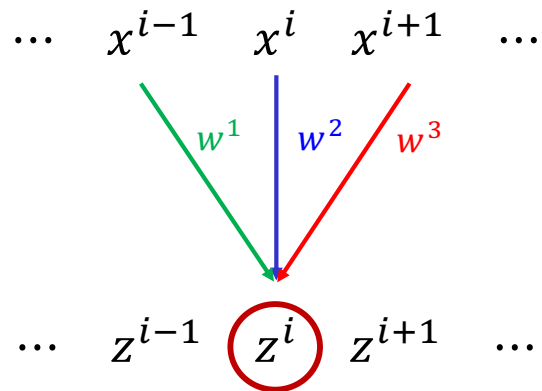
$$\begin{aligned}\frac{\partial e}{\partial x^i} &= \sum_{j=1}^N \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i} \\ &= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i}\end{aligned}$$



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



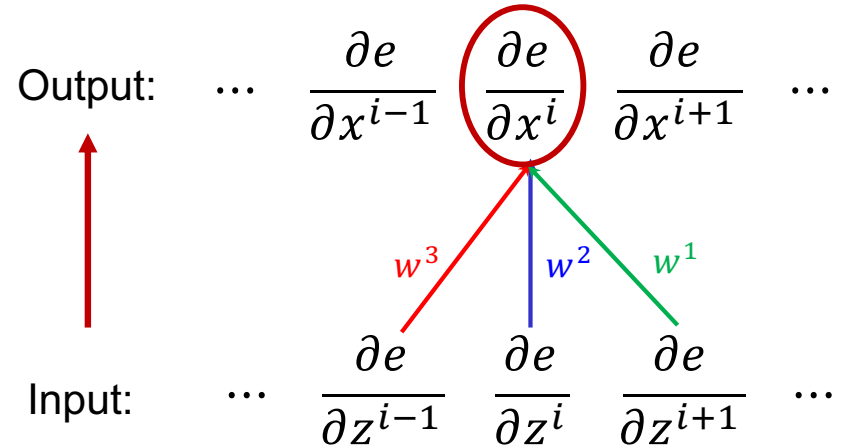
Convolutional layer: Backward pass



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

Backward pass (w.r.t. x)

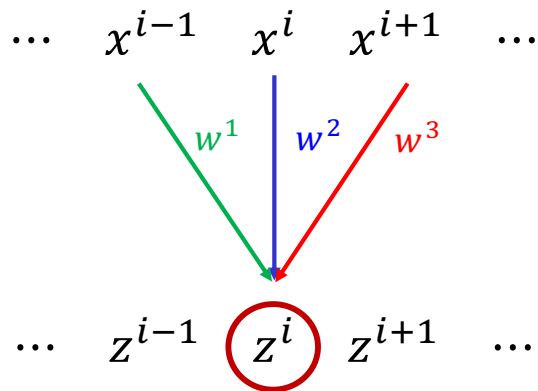
$$\begin{aligned} \frac{\partial e}{\partial x^i} &= \sum_{j=1}^N \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i} \\ &= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i} \\ &= w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^i} + w^1 \frac{\partial e}{\partial z^{i+1}} \end{aligned}$$



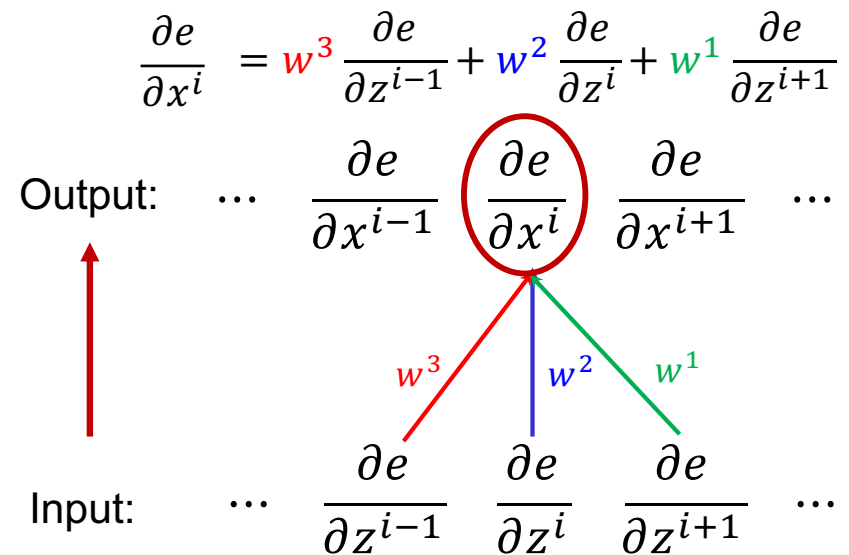
Convolutional layer: Backward pass

Backward pass (w.r.t. x)

This is called a *transposed convolution*



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

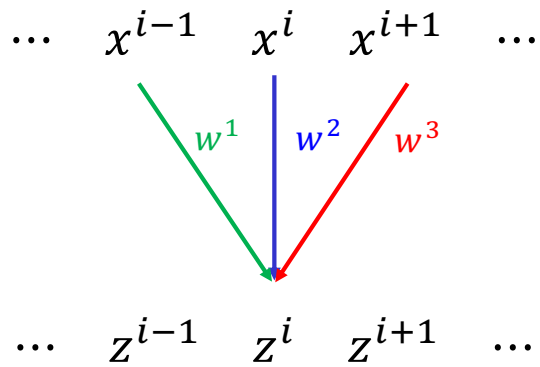


$$\frac{\partial e}{\partial x^i} = w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^i} + w^1 \frac{\partial e}{\partial z^{i+1}}$$

Backward pass

Backward pass (w.r.t. w)

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



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

Output:

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



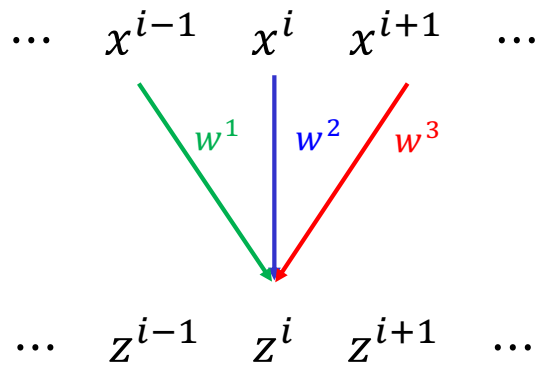
Input:

$$\dots \quad \frac{\partial e}{\partial z^{i-1}} \quad \frac{\partial e}{\partial z^i} \quad \frac{\partial e}{\partial z^{i+1}} \quad \dots$$

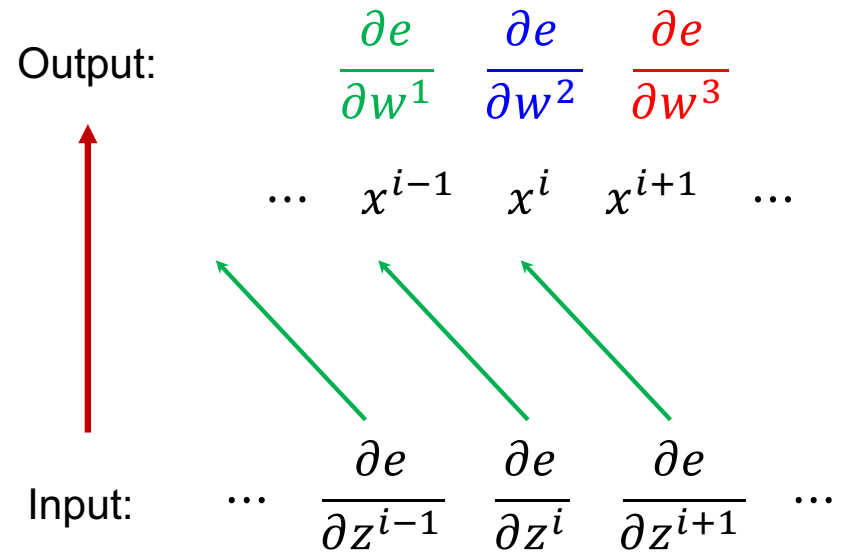
Backward pass

Backward pass (w.r.t. w)

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



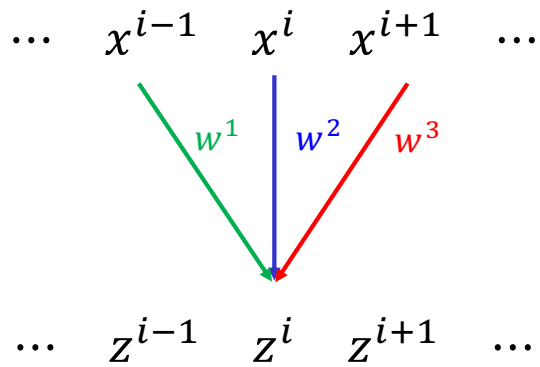
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



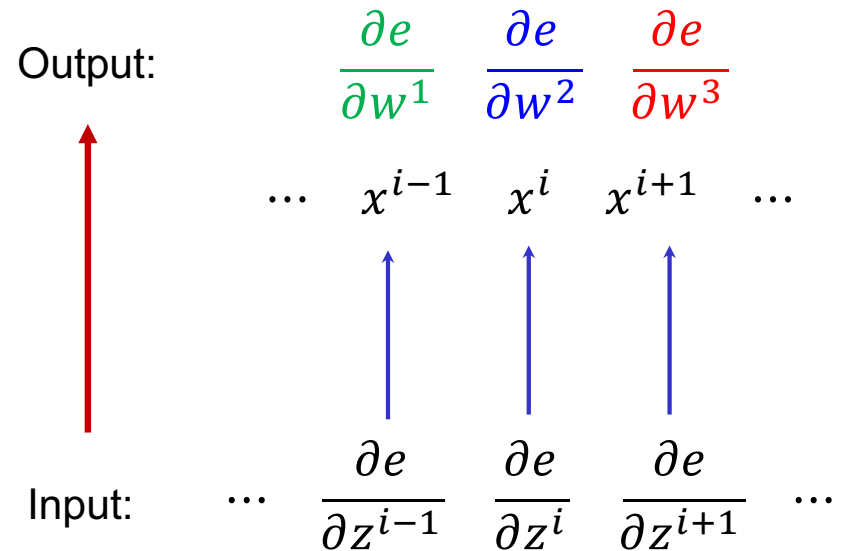
Backward pass

Backward pass (w.r.t. w)

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



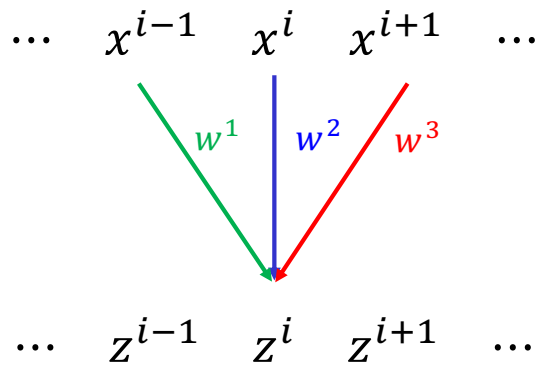
$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$



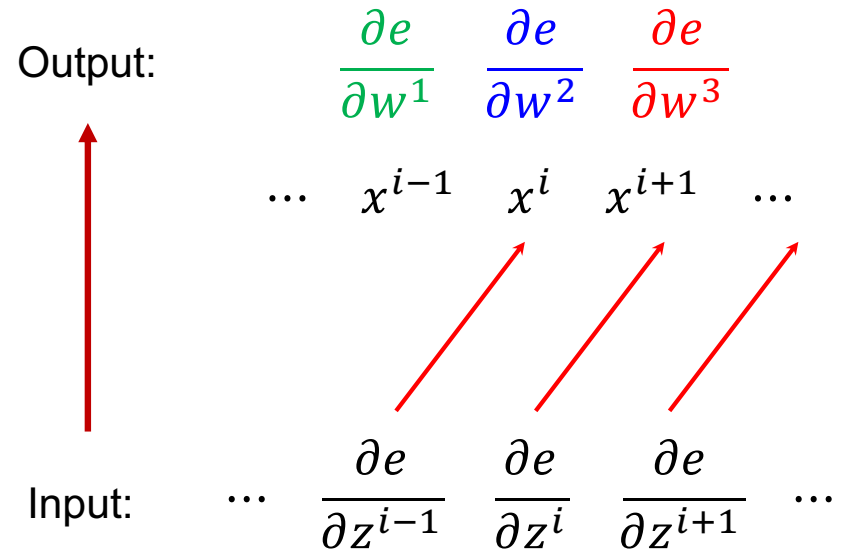
Backward pass

Backward pass (w.r.t. w)

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



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



Max pooling layer: Backward pass

