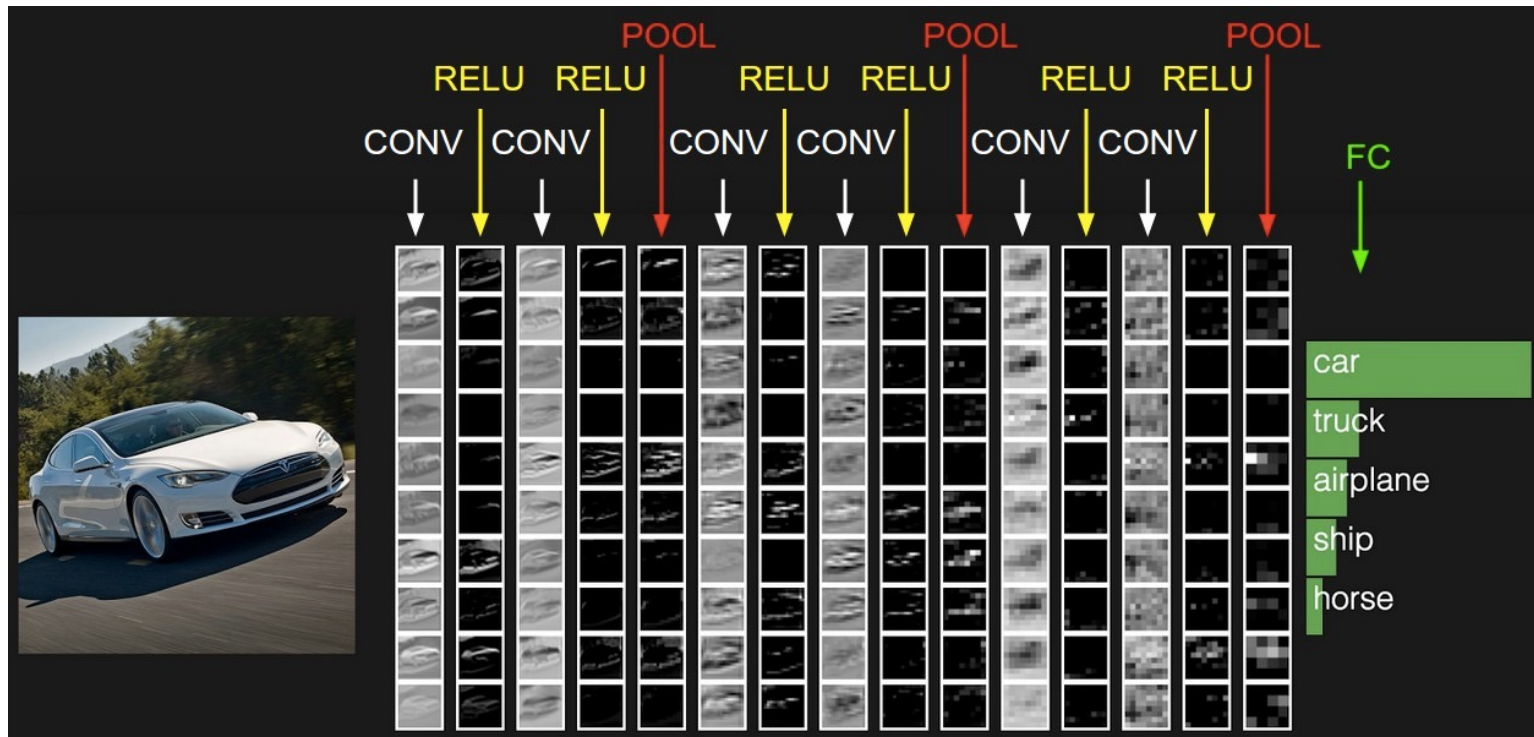


Convolutional neural networks



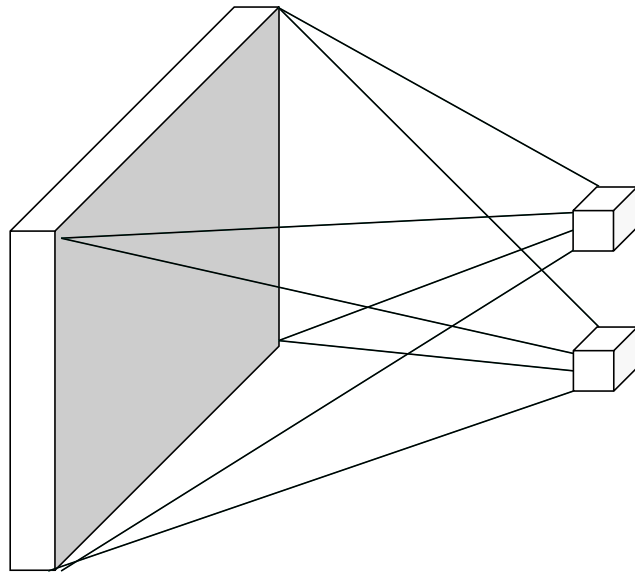
Many slides from Rob Fergus, Andrej Karpathy

Convolutional networks: Outline

- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions
- Max pooling
- Receptive fields
- Output layers

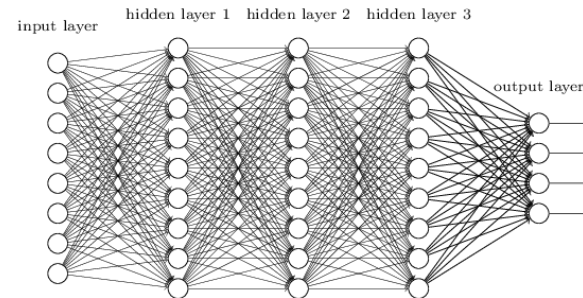
Let's design a neural network for images

Multi-layer perceptron (MLP)



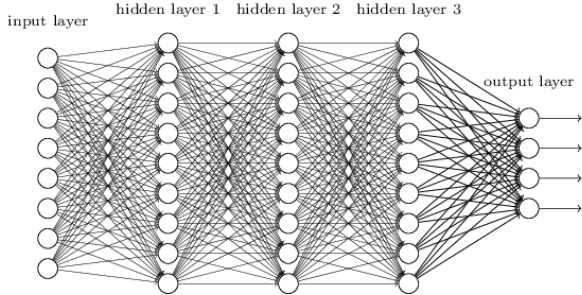
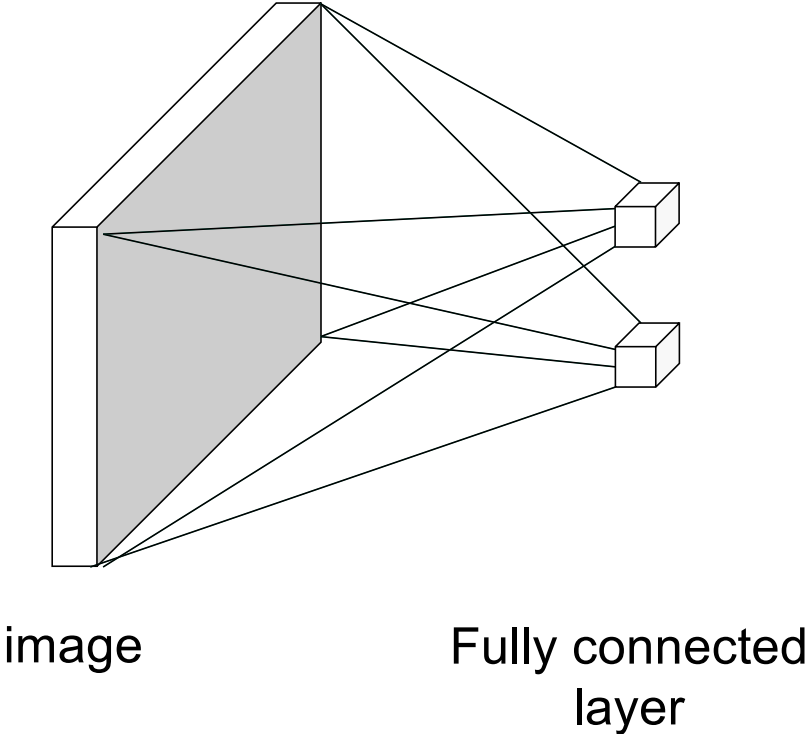
image

Fully connected
layer



Let's design a neural network for images

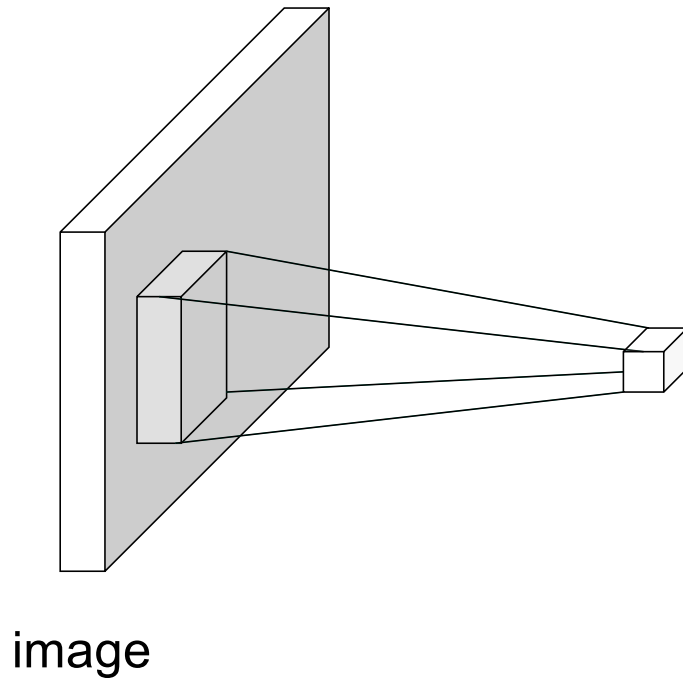
Multi-layer perceptron (MLP)



Recall: MLP as bank of whole-image templates

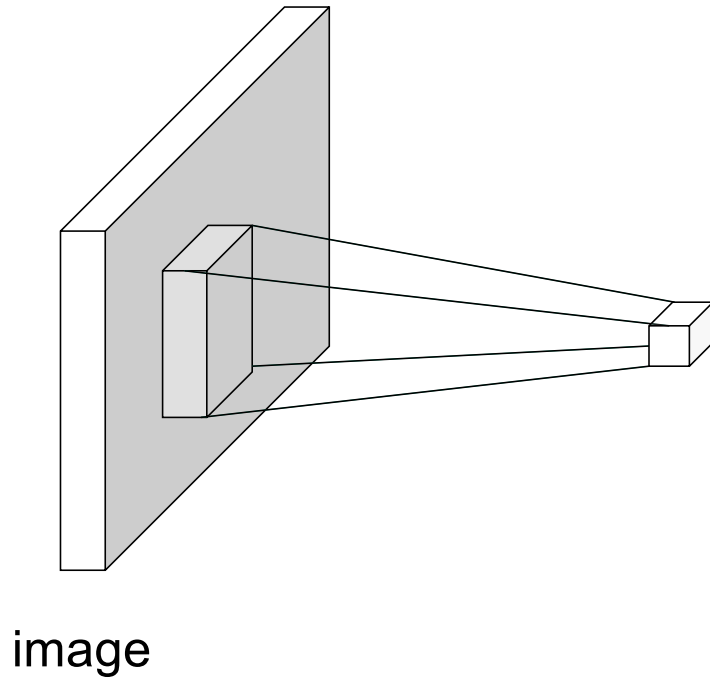
Anything wrong with this?

Convolutional architecture



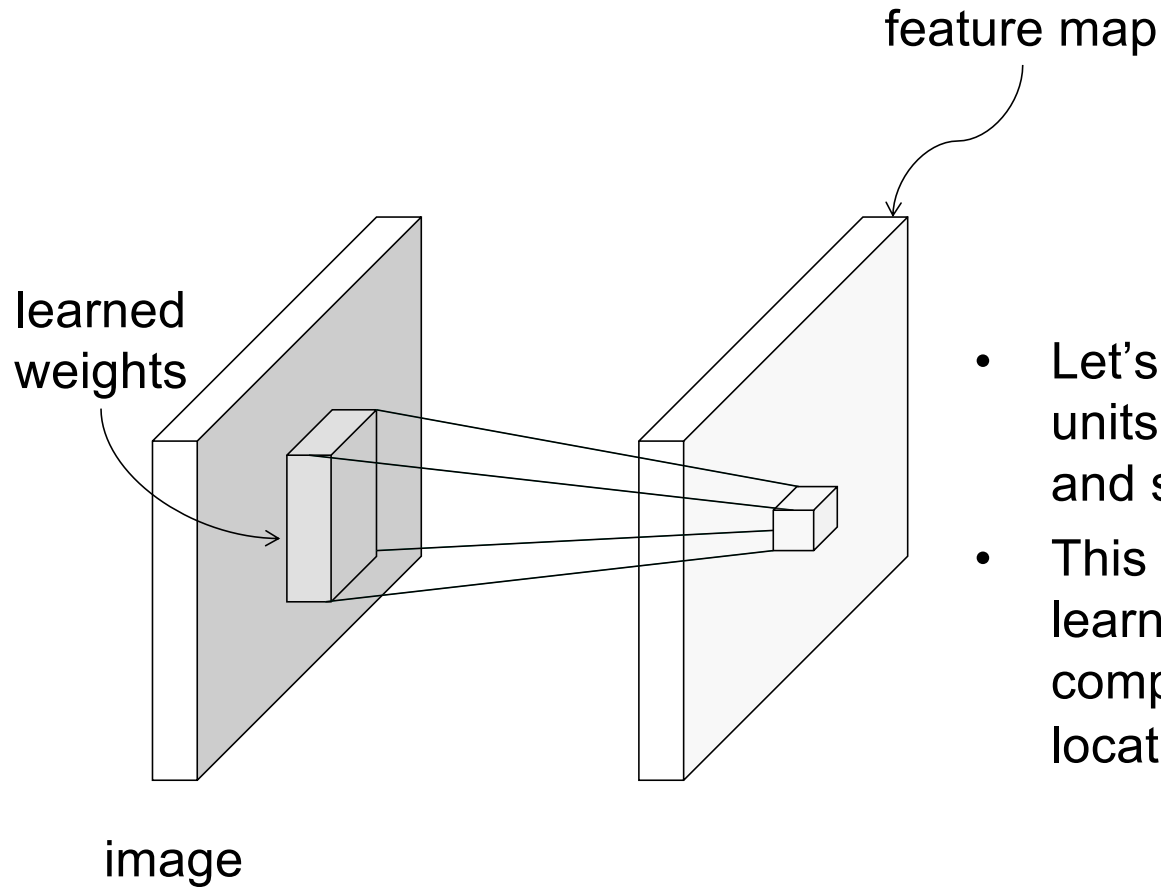
- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights

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}

*

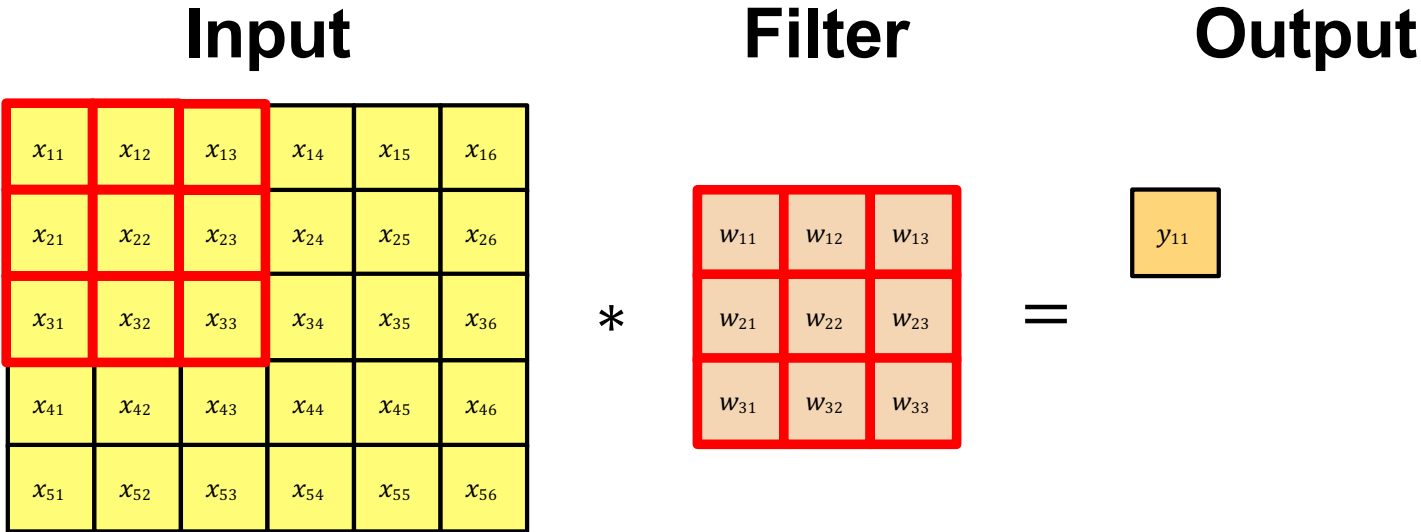
Filter

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

=

Output

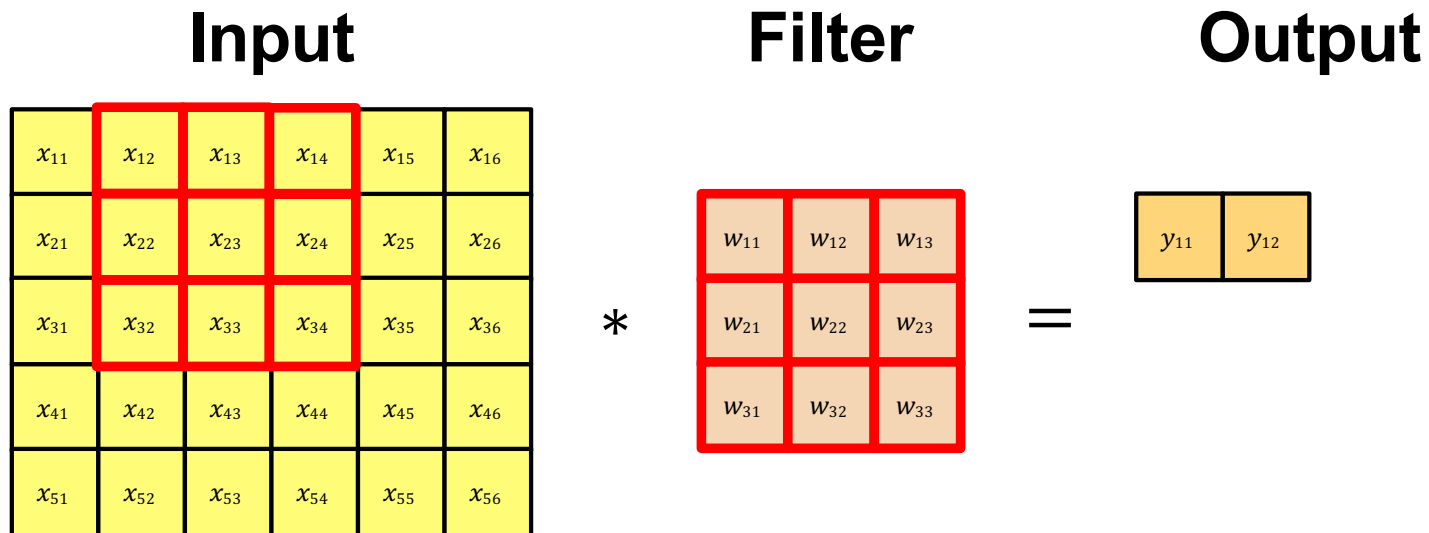
Convolution example



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

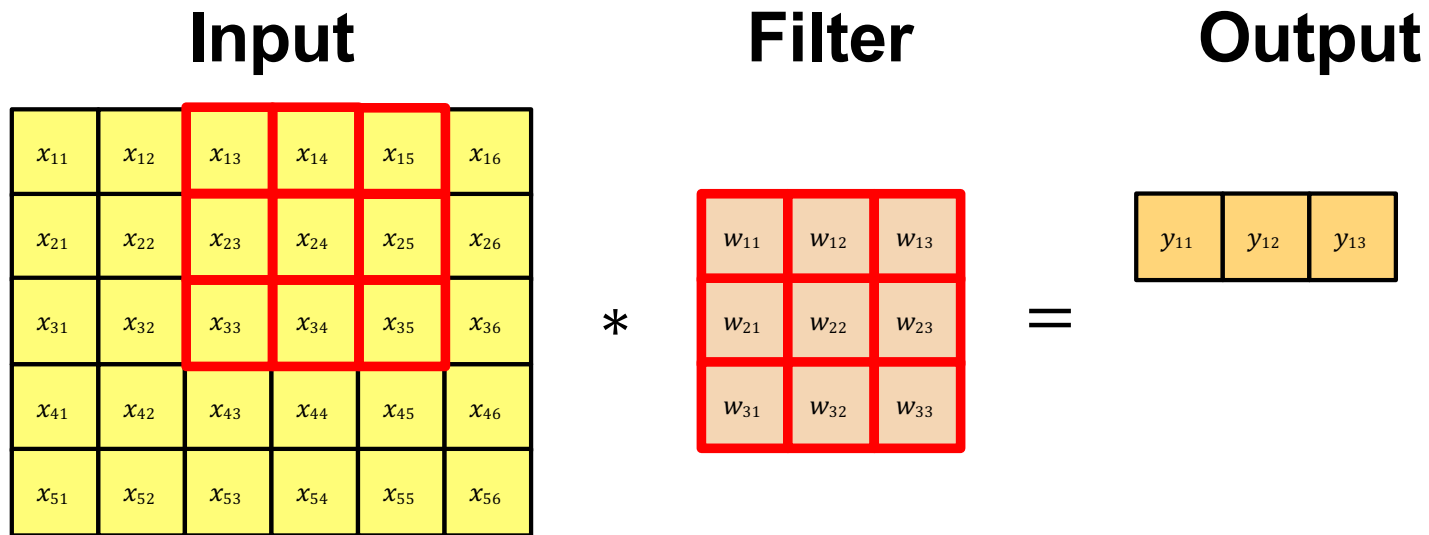
Adapted from [D. Fouhey and J. Johnson](#)

Convolution example



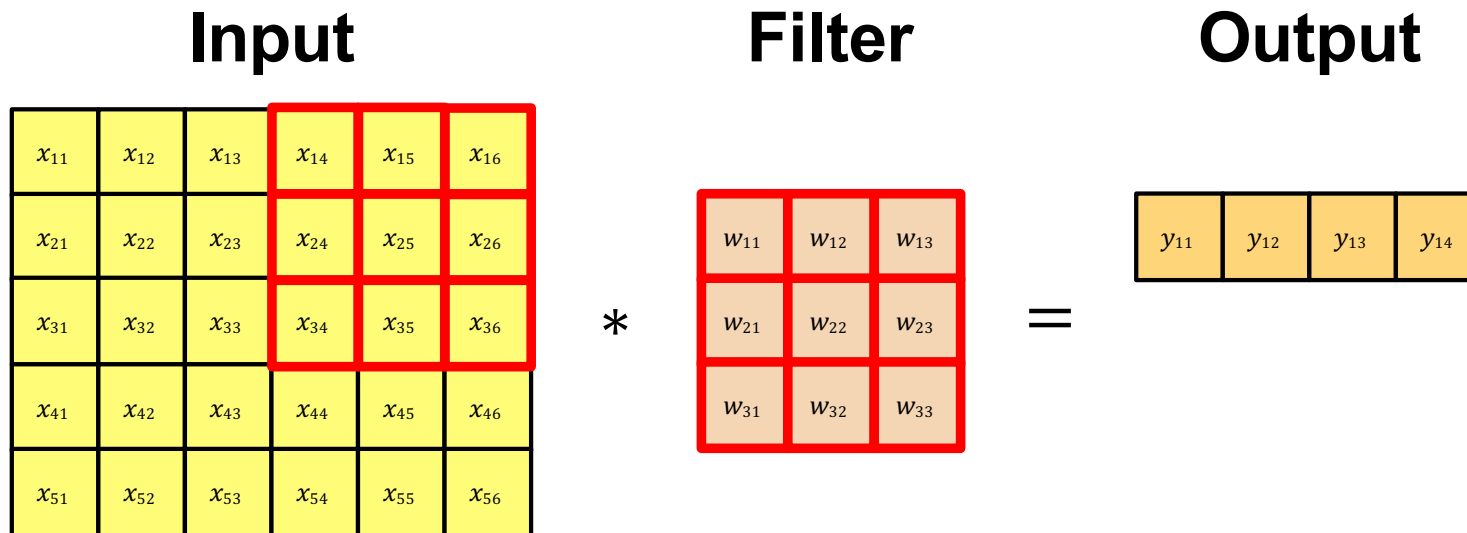
$$y_{12} = x_{12} \cdot w_{11} + x_{13} \cdot w_{12} + x_{14} \cdot w_{13} + \dots + x_{34} \cdot w_{33}$$

Convolution example



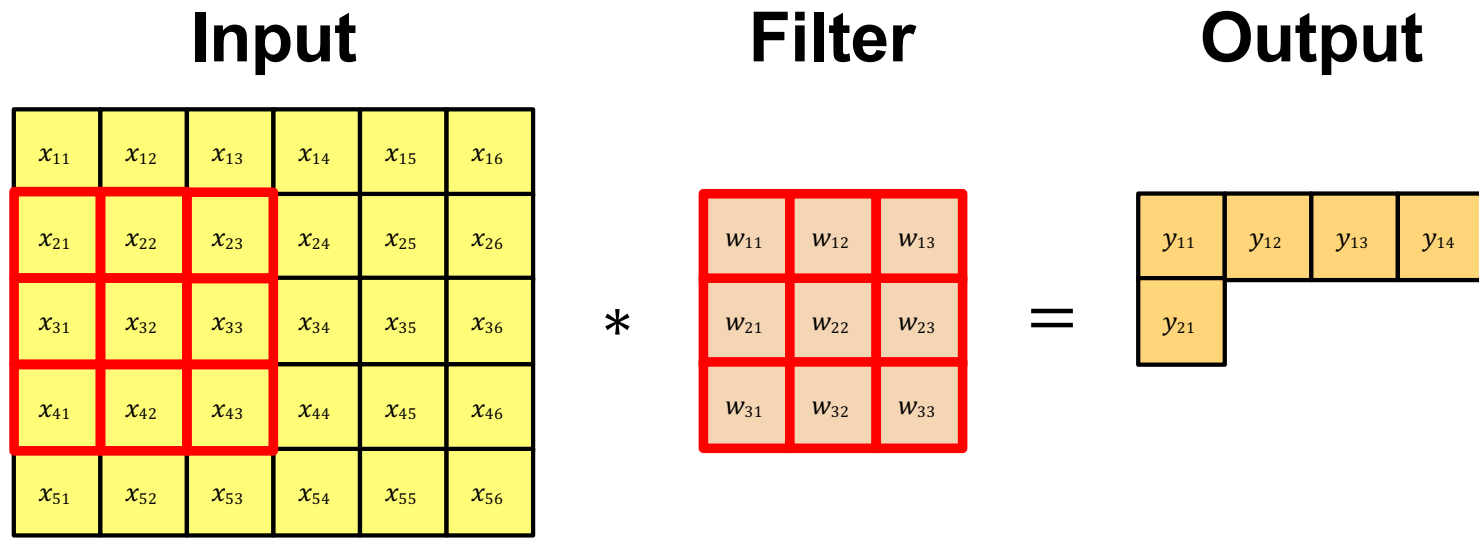
$$y_{13} = x_{13} \cdot w_{11} + x_{14} \cdot w_{12} + x_{15} \cdot w_{13} + \dots + x_{35} \cdot w_{33}$$

Convolution example



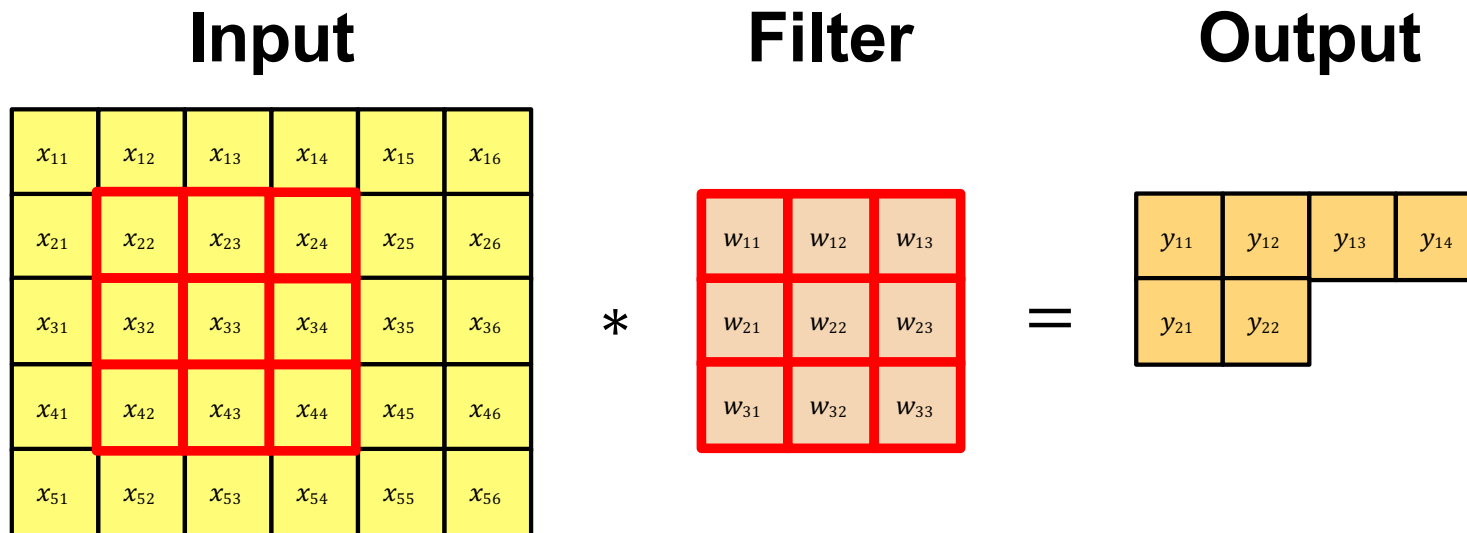
$$y_{14} = x_{14} \cdot w_{11} + x_{15} \cdot w_{12} + x_{16} \cdot w_{13} + \dots + x_{36} \cdot w_{33}$$

Convolution example



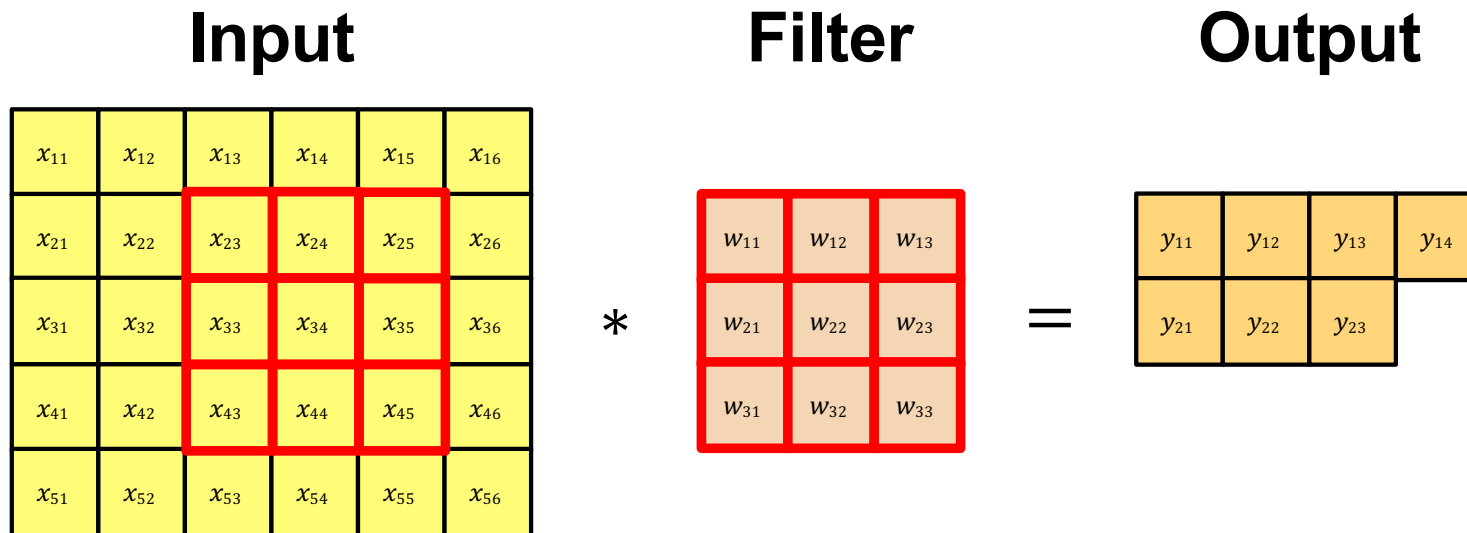
$$y_{21} = x_{21} \cdot w_{11} + x_{22} \cdot w_{12} + x_{23} \cdot w_{13} + \dots + x_{43} \cdot w_{33}$$

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}

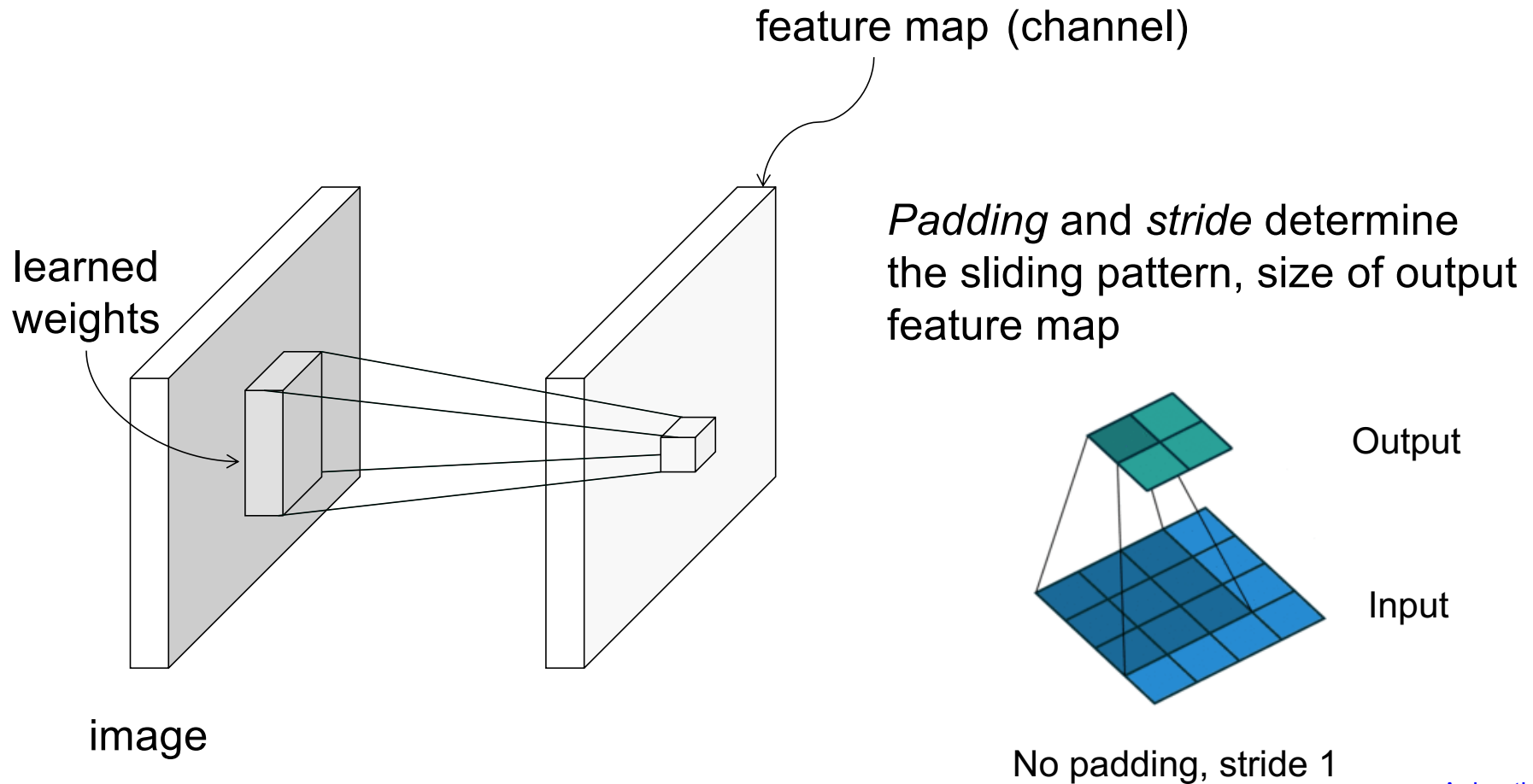
*

=

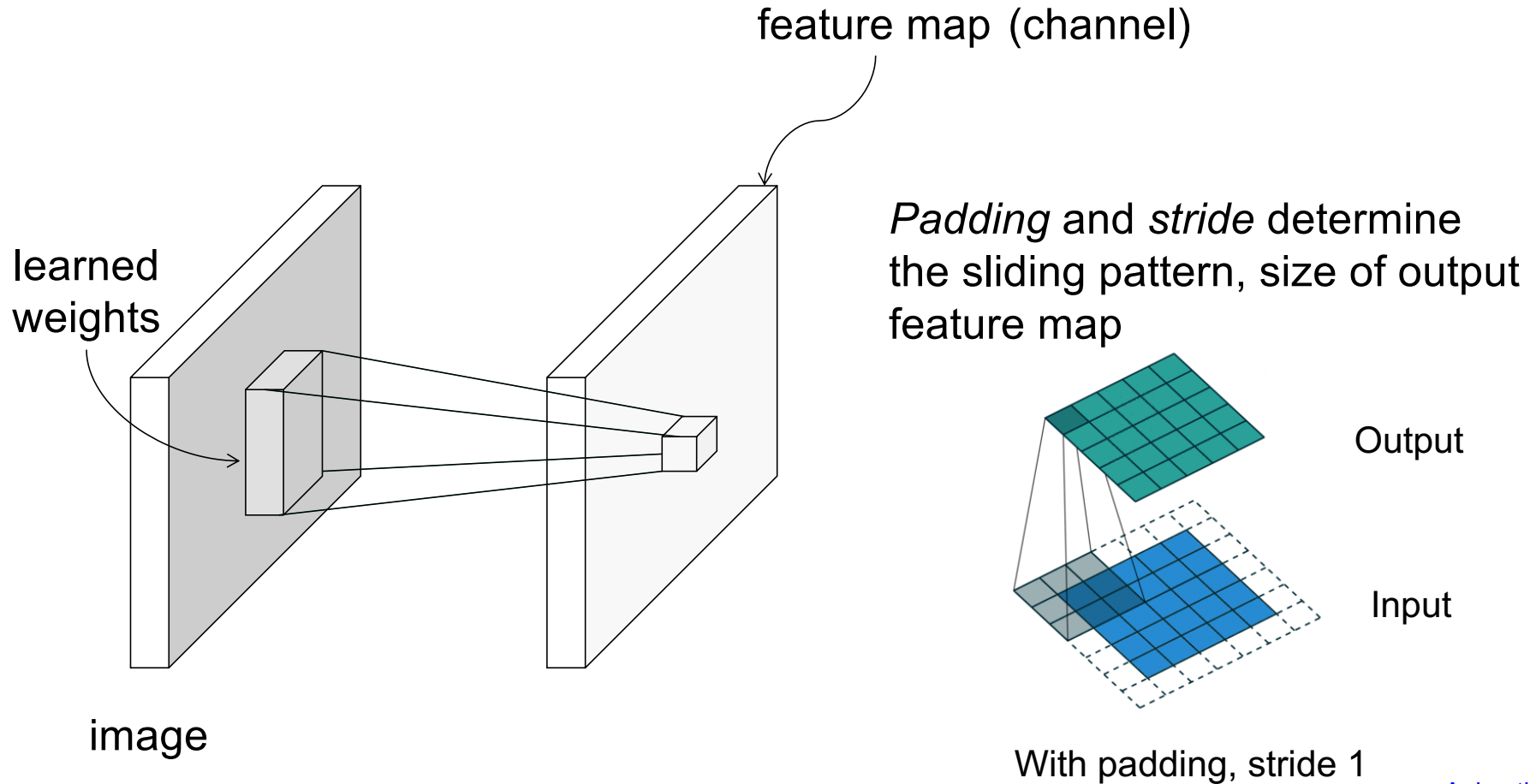
Output

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

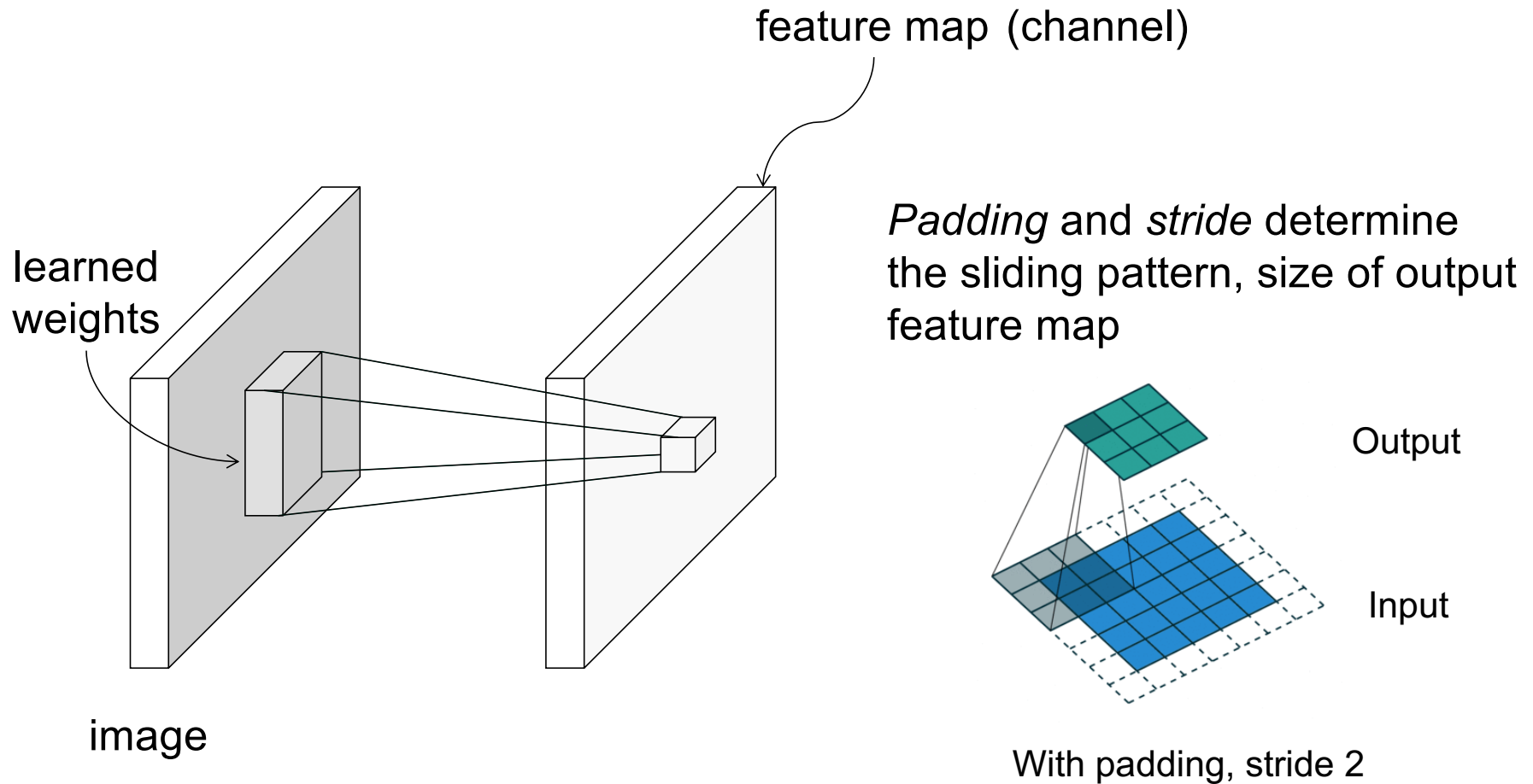
Convolutional architecture



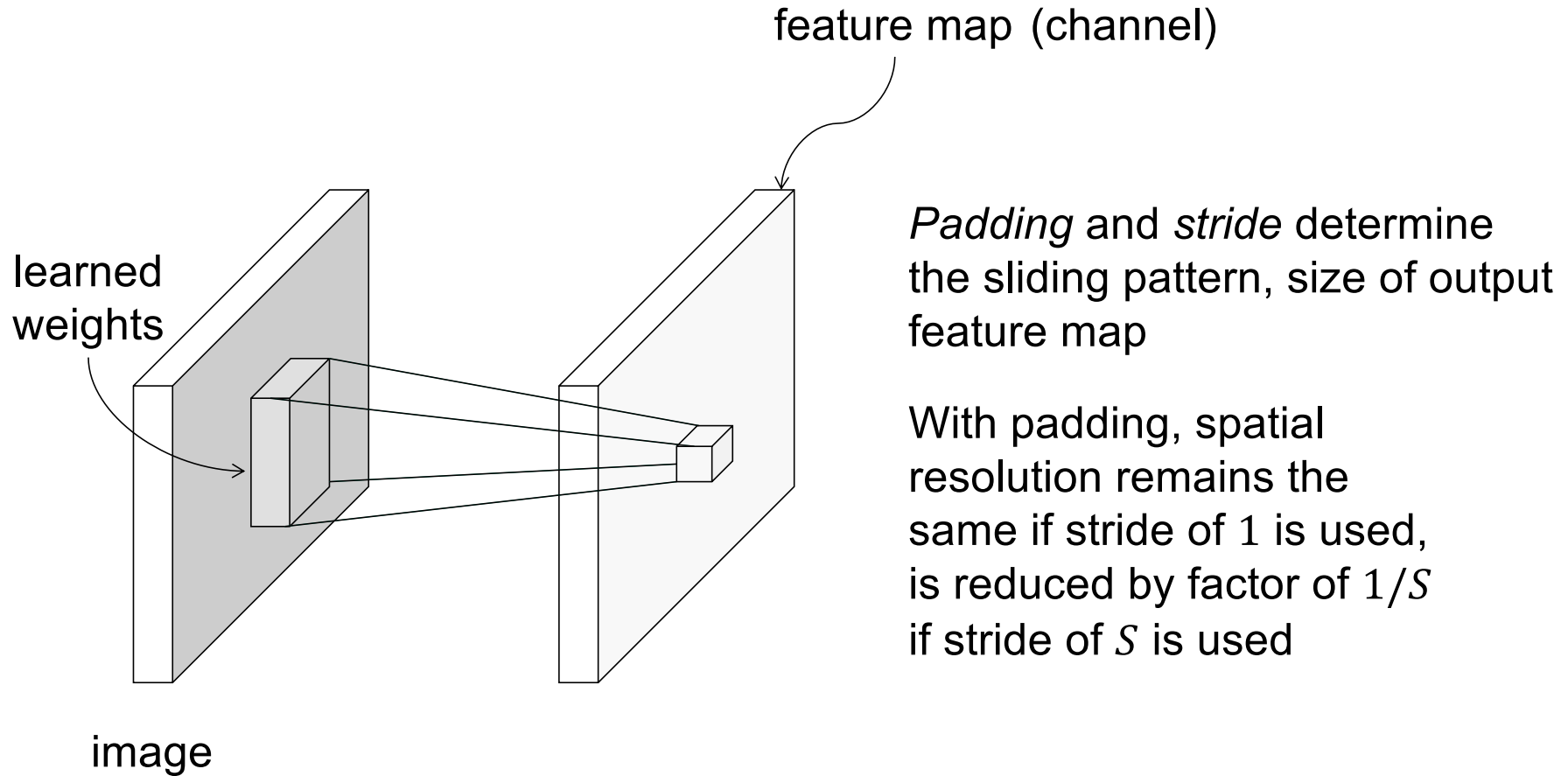
Convolutional architecture



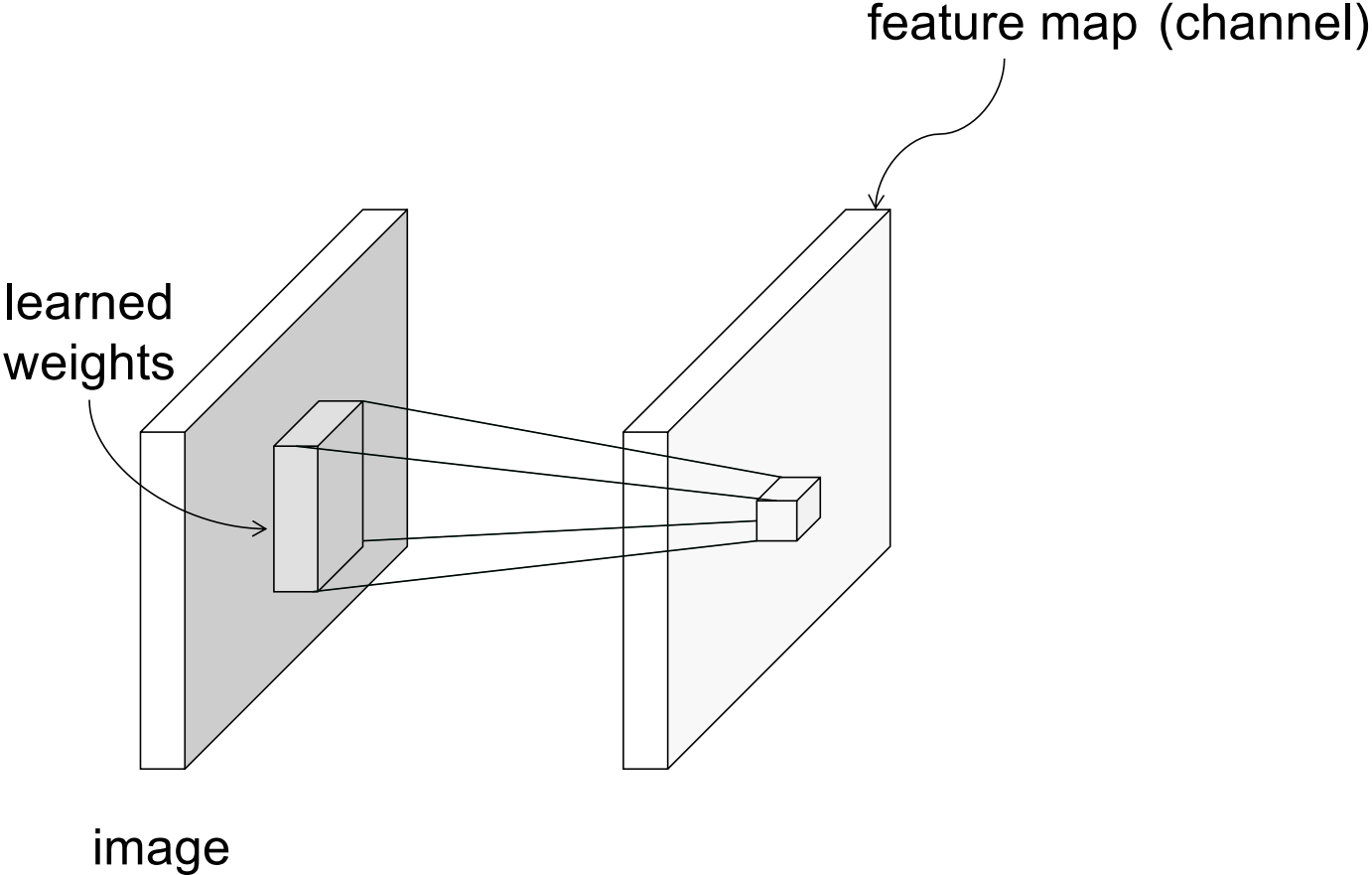
Convolutional architecture



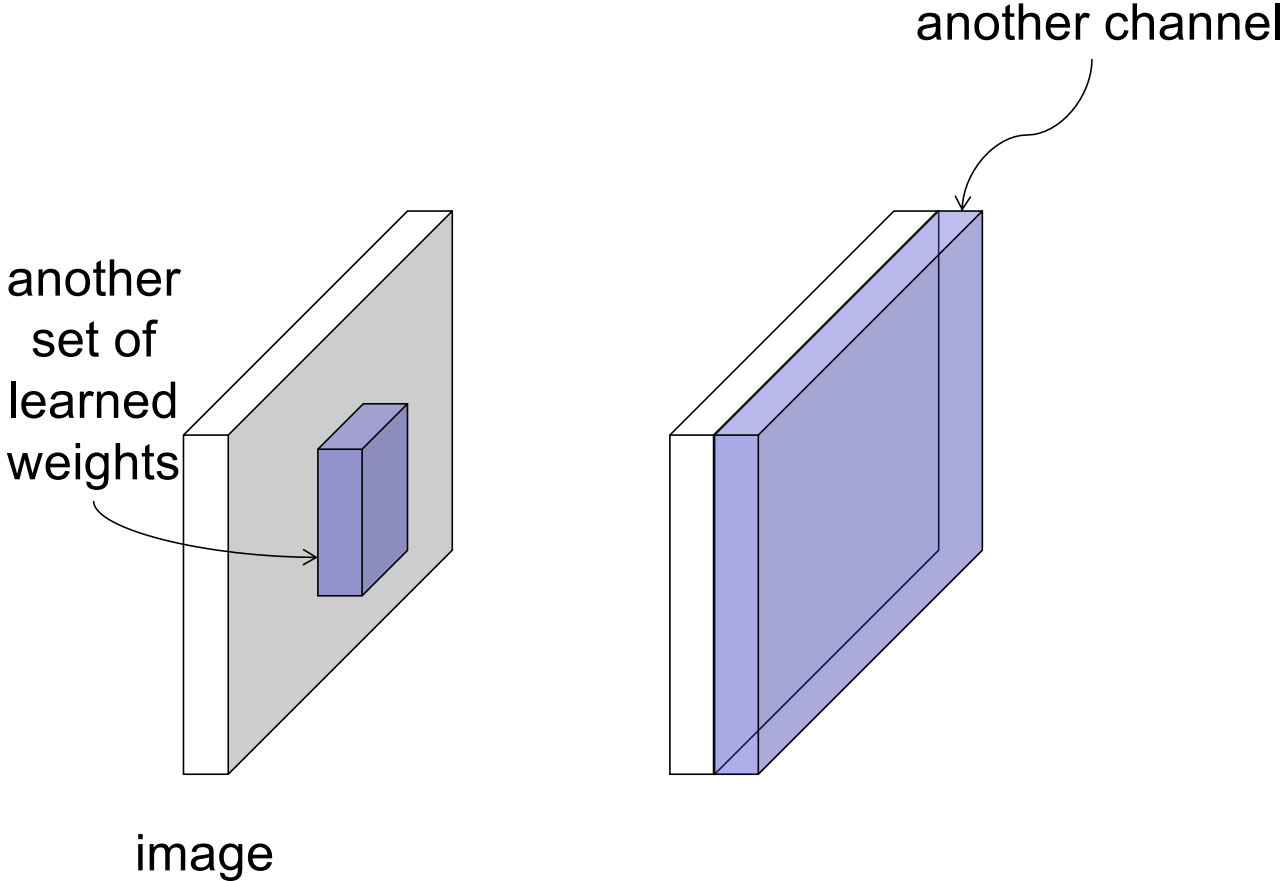
Convolutional architecture



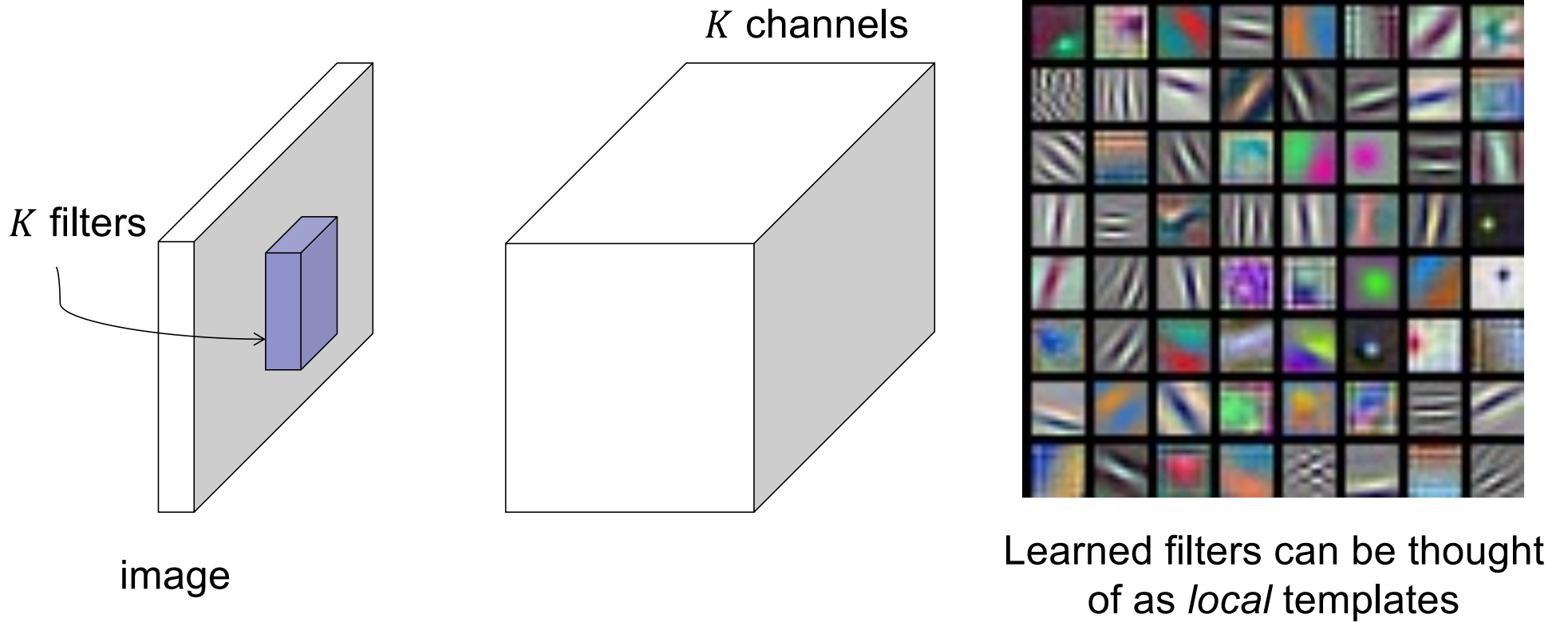
Convolutional architecture



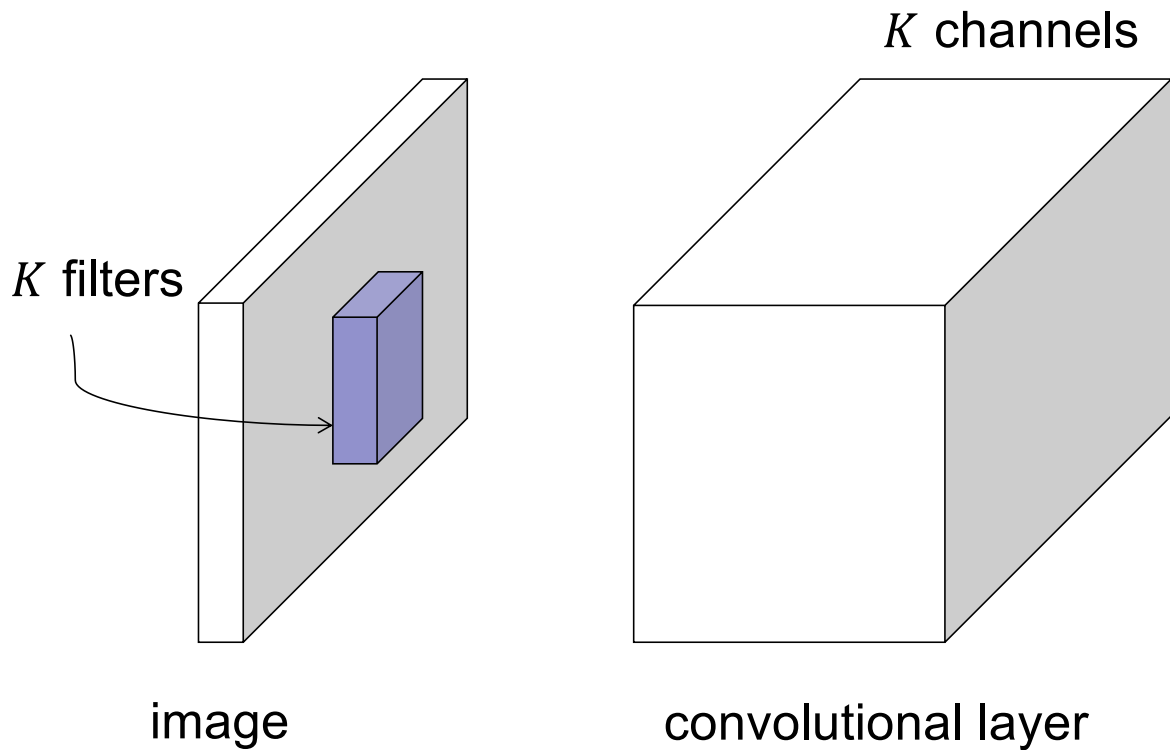
Convolutional architecture



Convolutional architecture

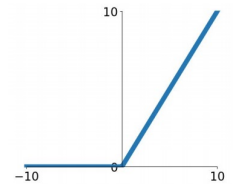


Convolutional architecture



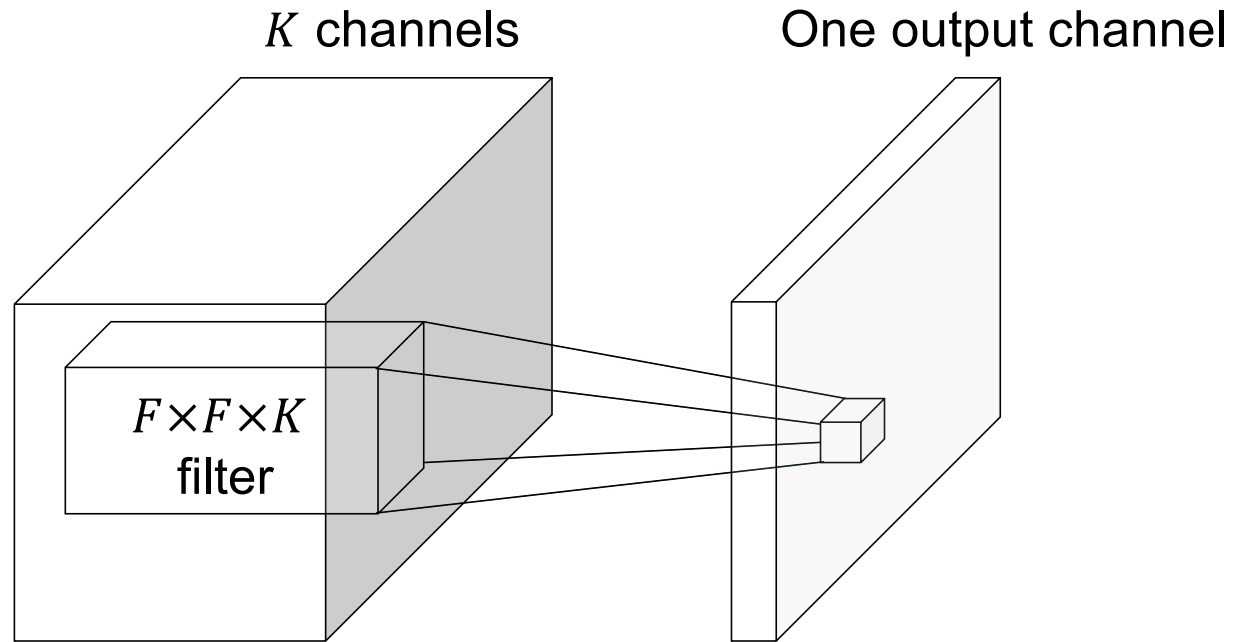
Almost always directly followed by a ReLU (or similar activation function)

$$\max(0, x)$$



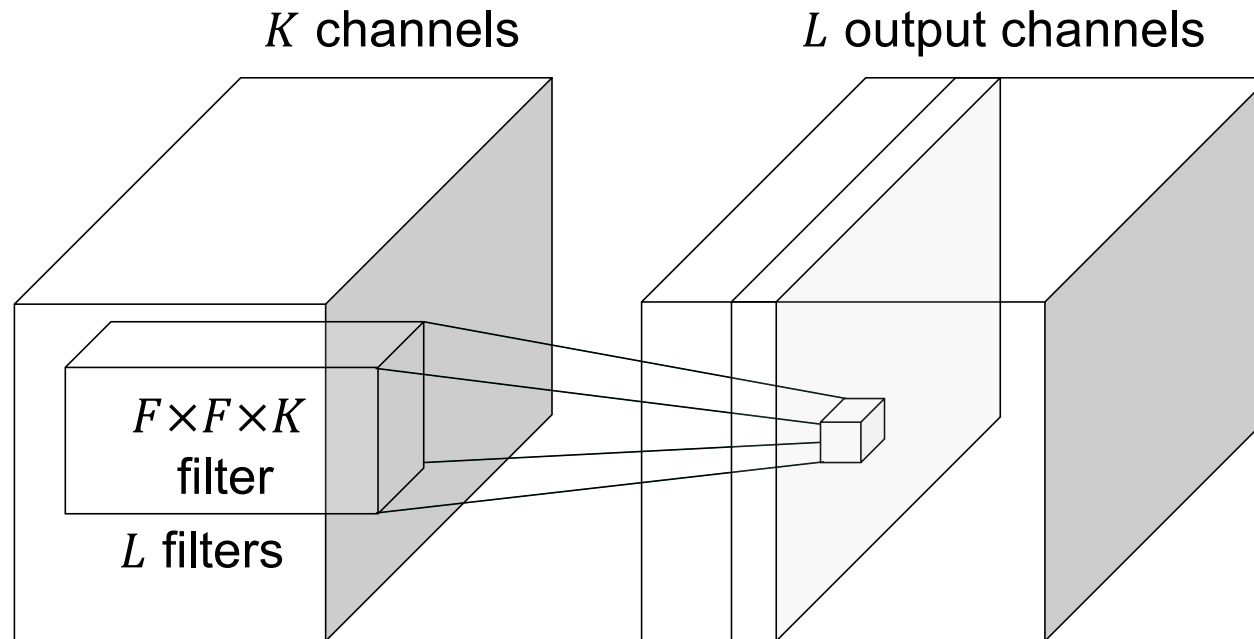
Three-dimensional convolutions

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

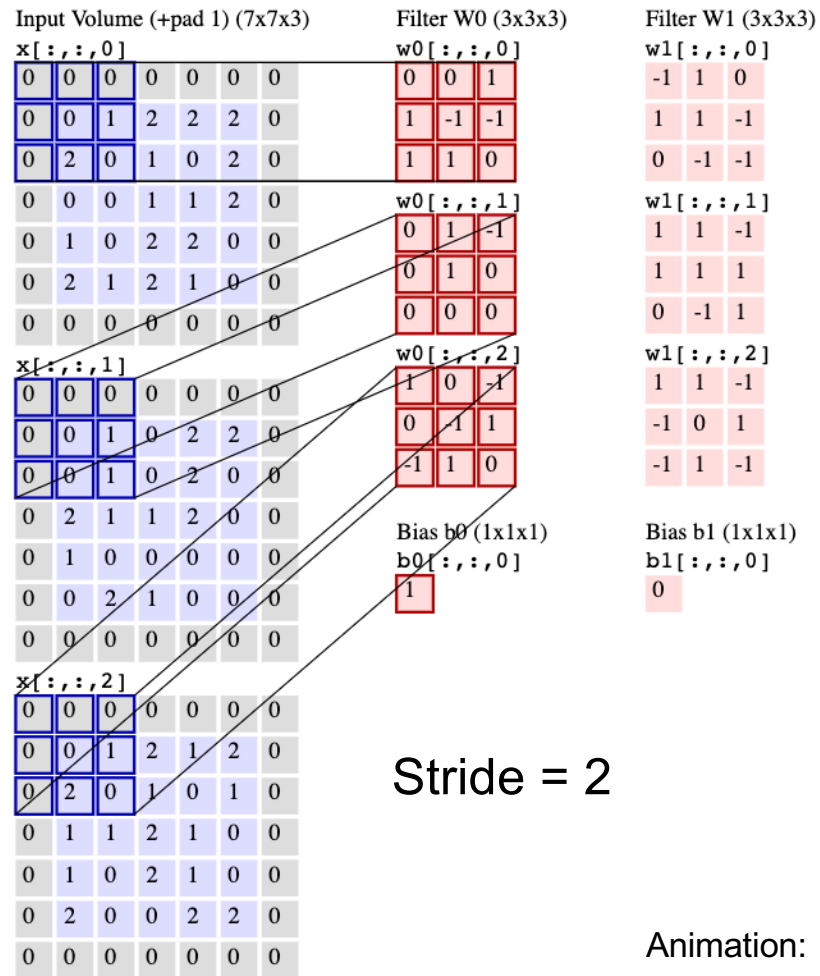


Three-dimensional convolutions

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

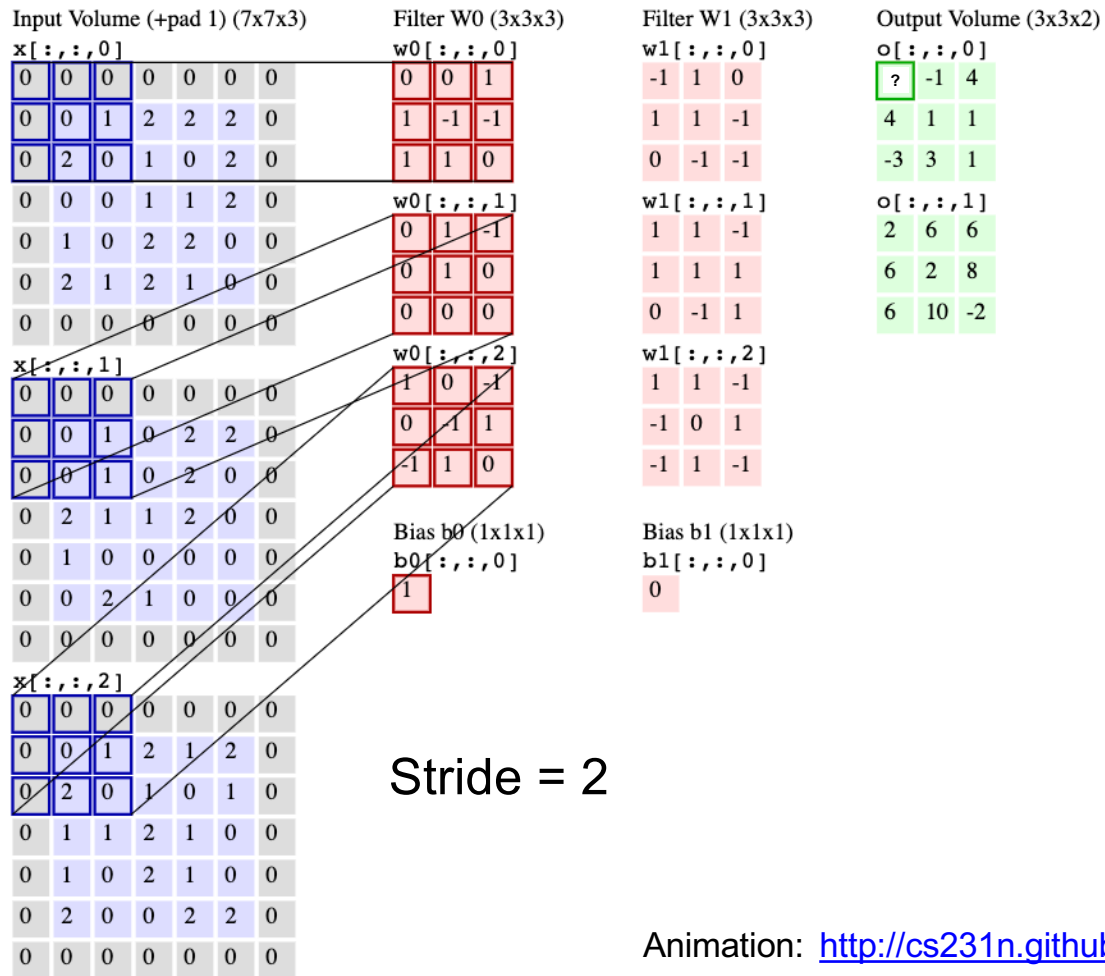


Convolutional layer example



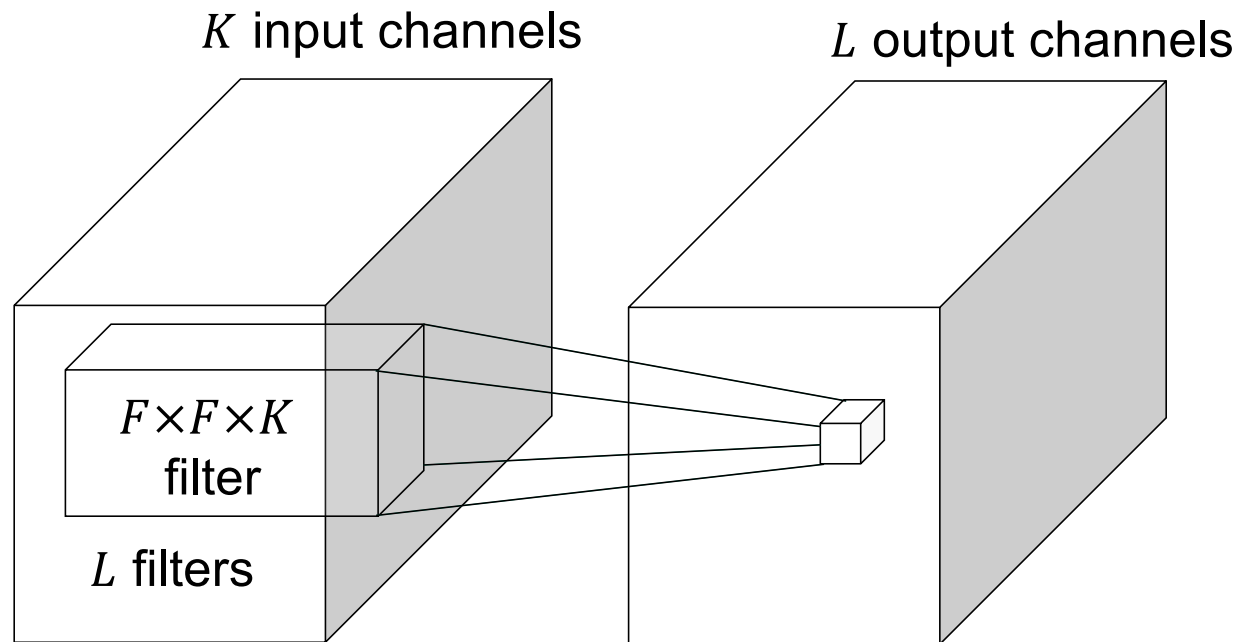
Animation: <http://cs231n.github.io/convolutional-networks/#conv>

Convolutional layer example



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

Convolutional layer: Computational cost

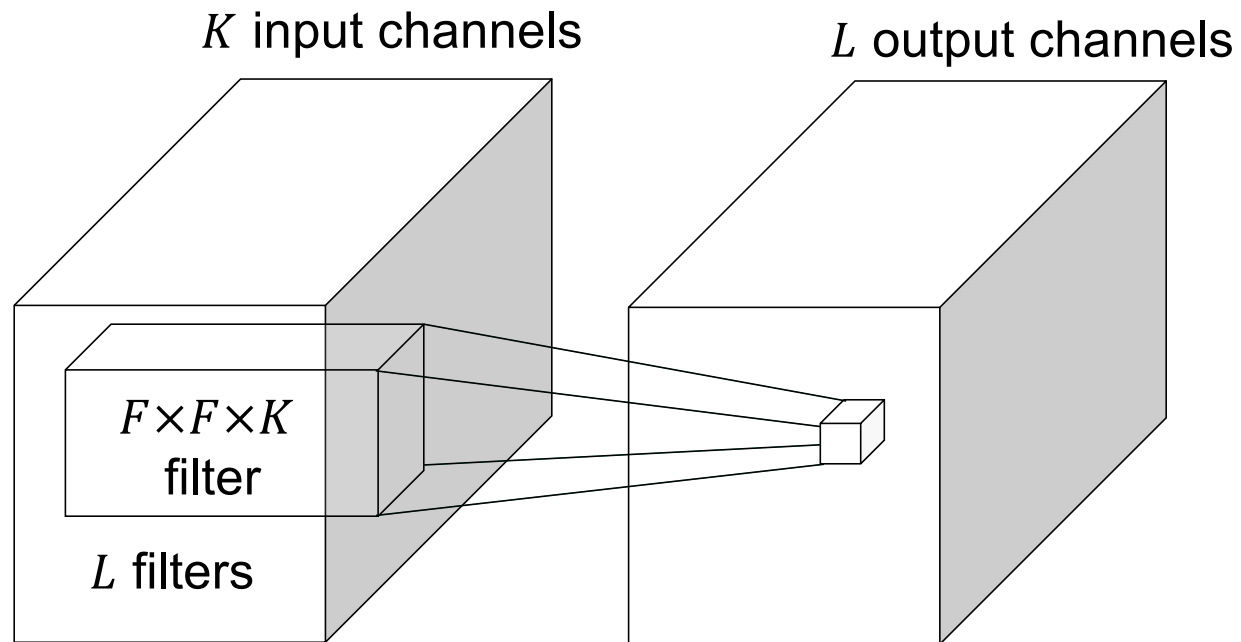


- Assuming the input feature maps have spatial resolution $H \times W$, how many operations are needed to compute the output feature volume?
 - F^2KLHW

Convolutional networks: Outline

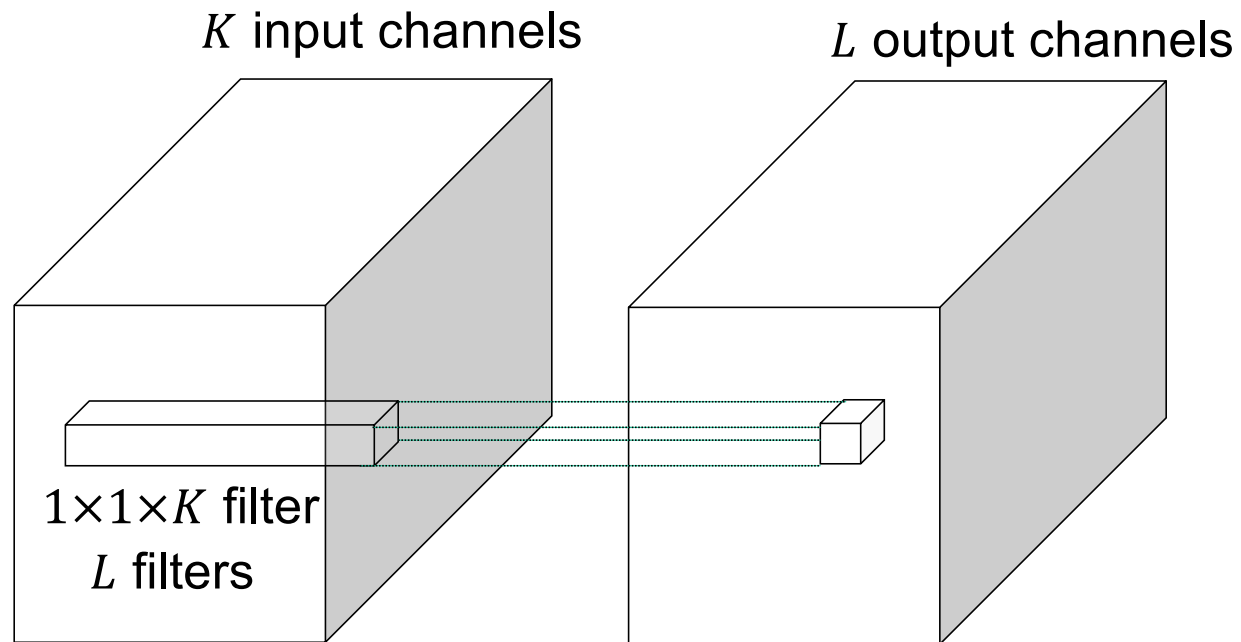
- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions

1x1 convolutional layer



What if we make $F = 1$?

1x1 convolutional layer

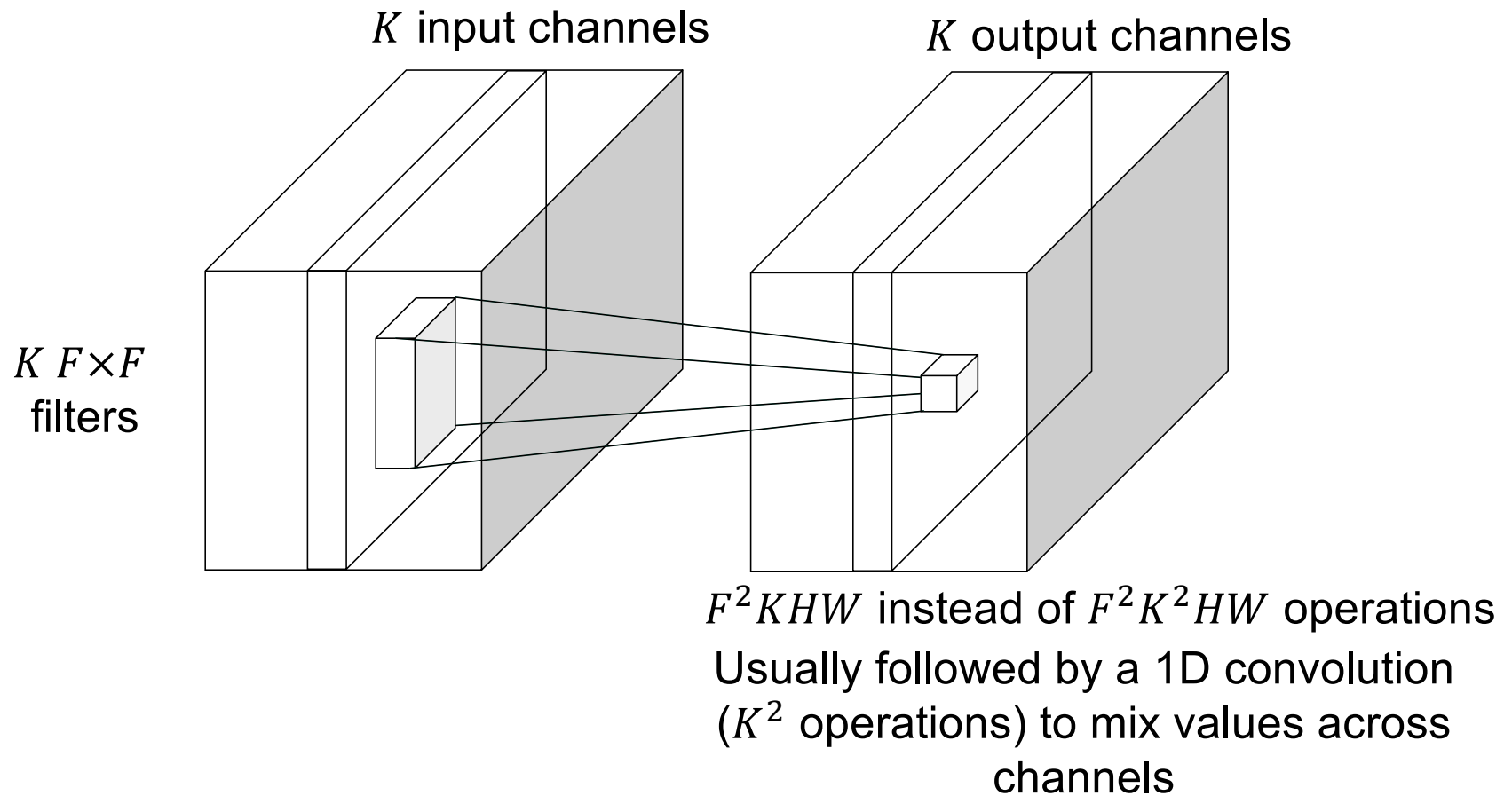


What if we make $F = 1$?

Why 1x1 convolutions?

- **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

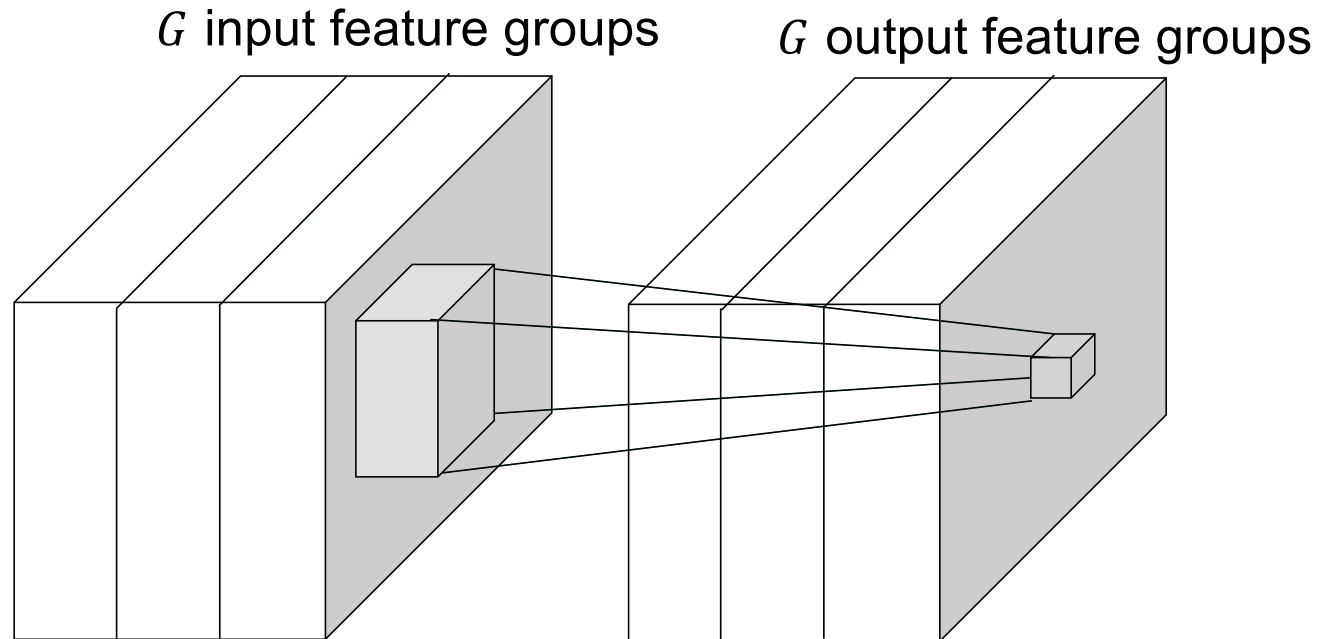
Alternative to 3D convolutions: Depthwise convolutions



Example: MobileNets

- **Depthwise separable convolution block:**
 - 3×3 conv layer, 256 → 256 channels
 - 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 (approaches 9 as K increases)

More generally: Groupwise convolutions

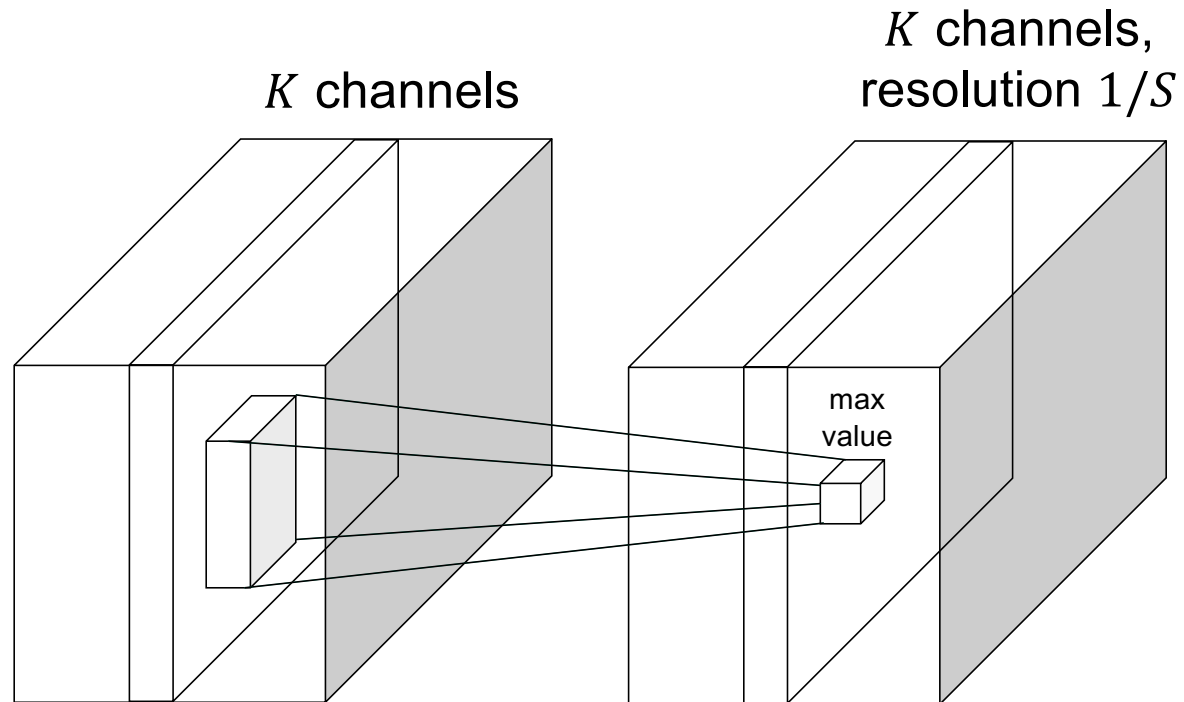


- Split up the K feature maps into G groups, perform convolutions within each group separately, concatenate the results

Convolutional networks: Outline

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

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

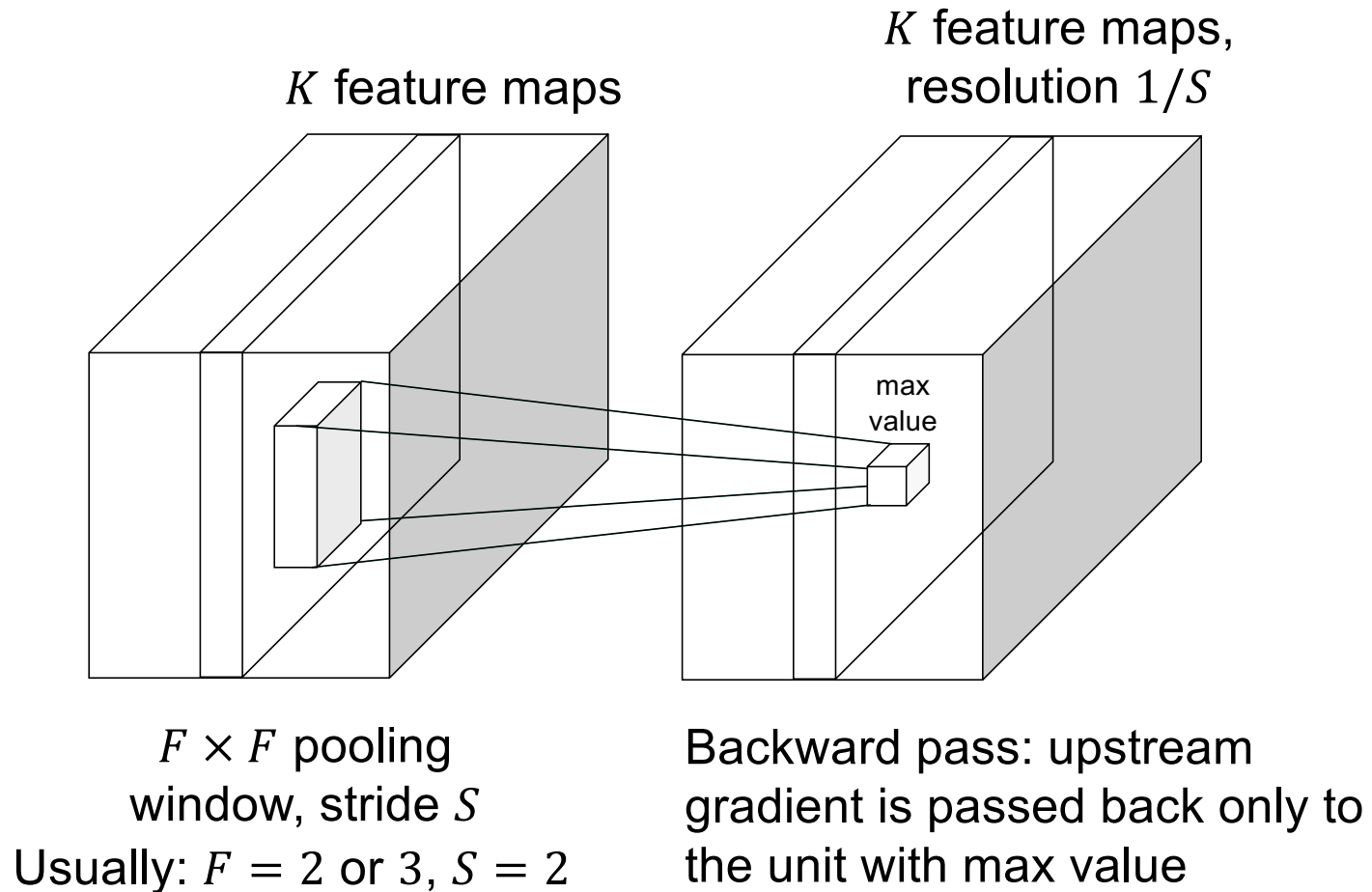
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2×2
kernel size and stride 2



6	8
3	4

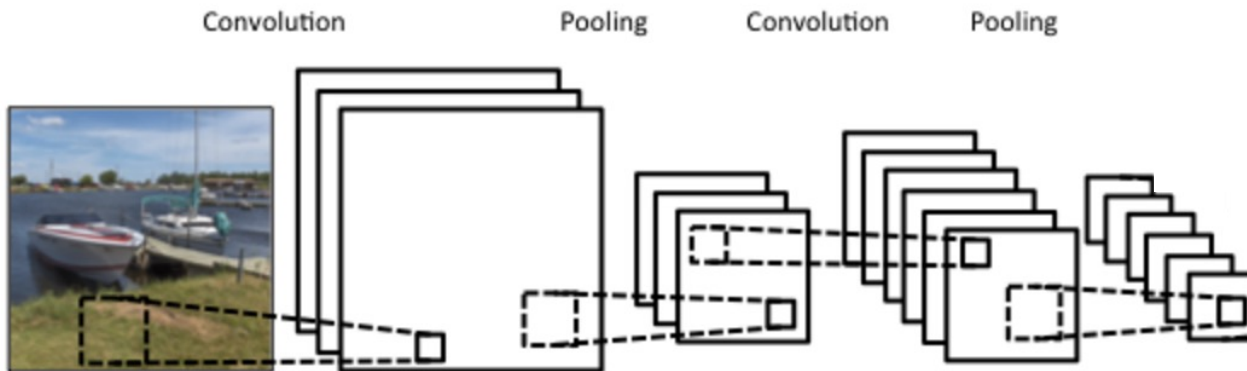
Max pooling layer



Convolutional networks: Outline

- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions
- Max pooling
- **Receptive fields**

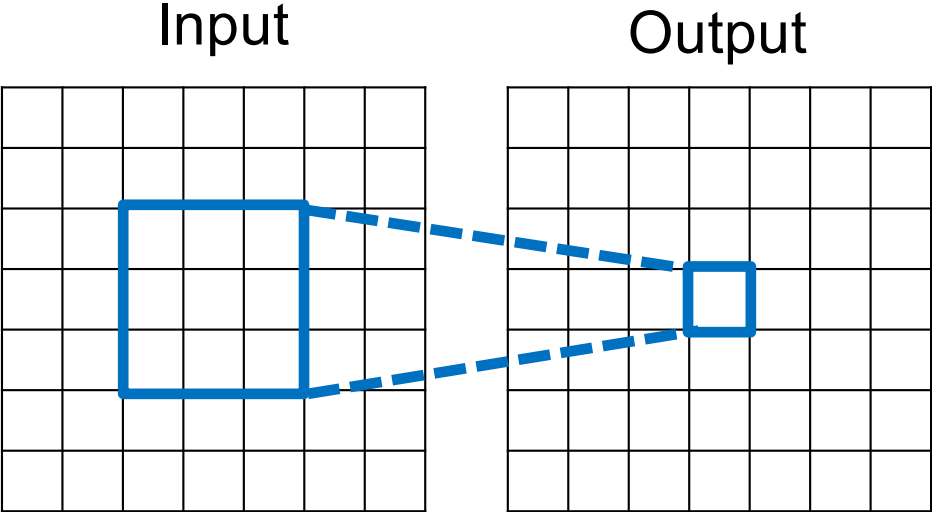
Receptive fields



- The *receptive field* of a unit is the region of an earlier input feature map whose values contribute to the response of that unit

Receptive fields

3x3 convolutions, stride 1

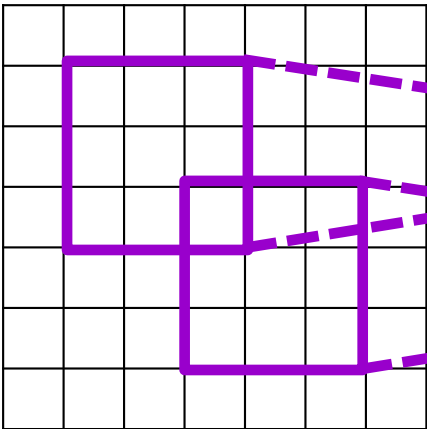


Receptive field size: 3

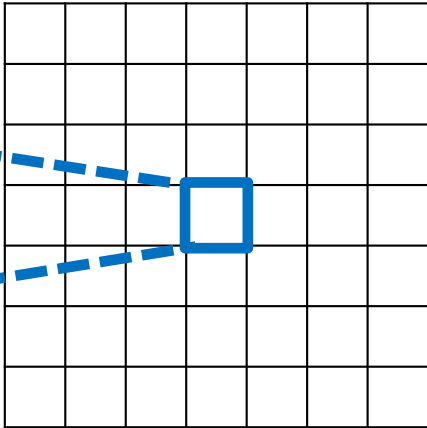
Receptive fields

3x3 convolutions, stride 1

Input



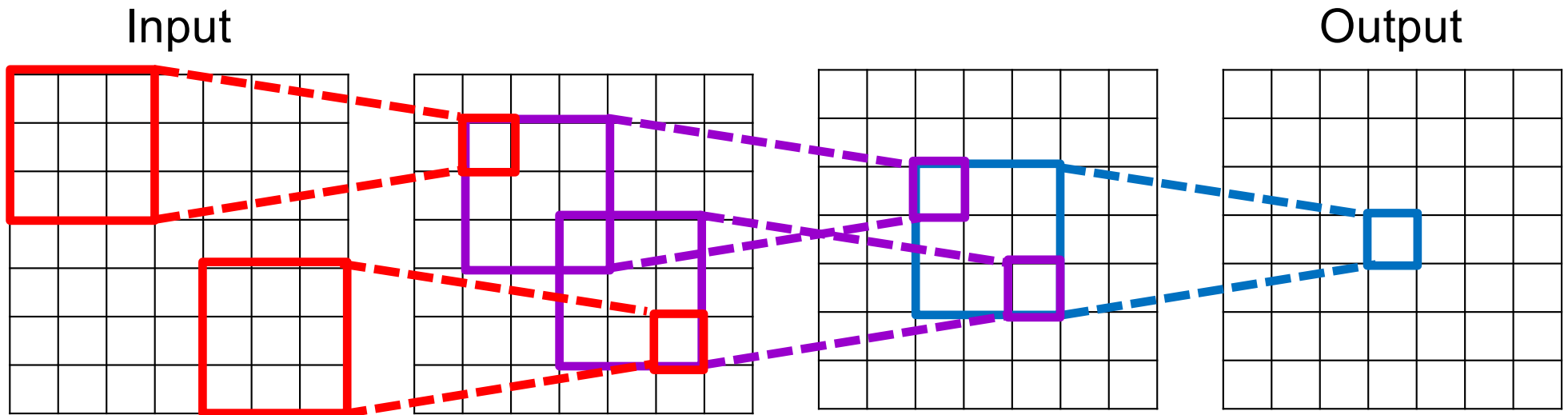
Output



Receptive field size: 5

Receptive fields

3x3 convolutions, stride 1

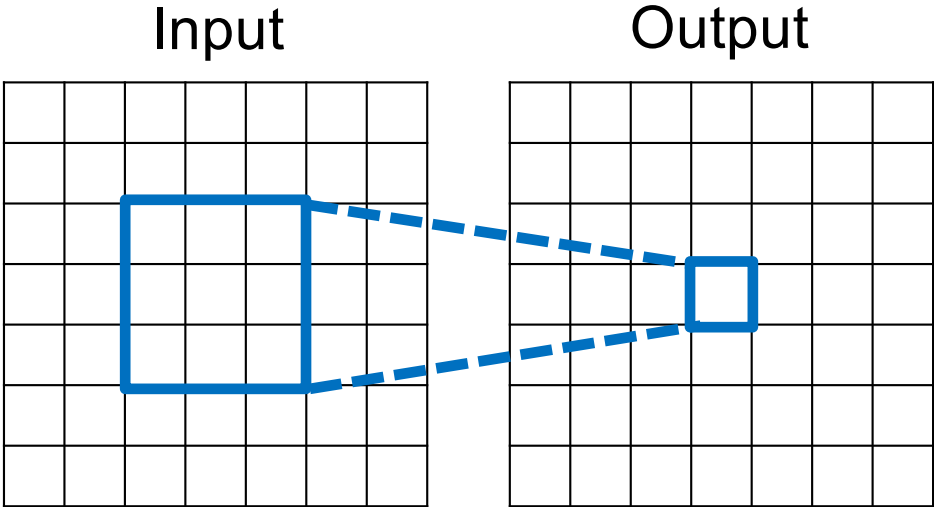


Receptive field size: 7

Each successive convolution adds $F - 1$ to the receptive field size
With L layers the receptive field size is $1 + L * (F - 1)$

Receptive fields

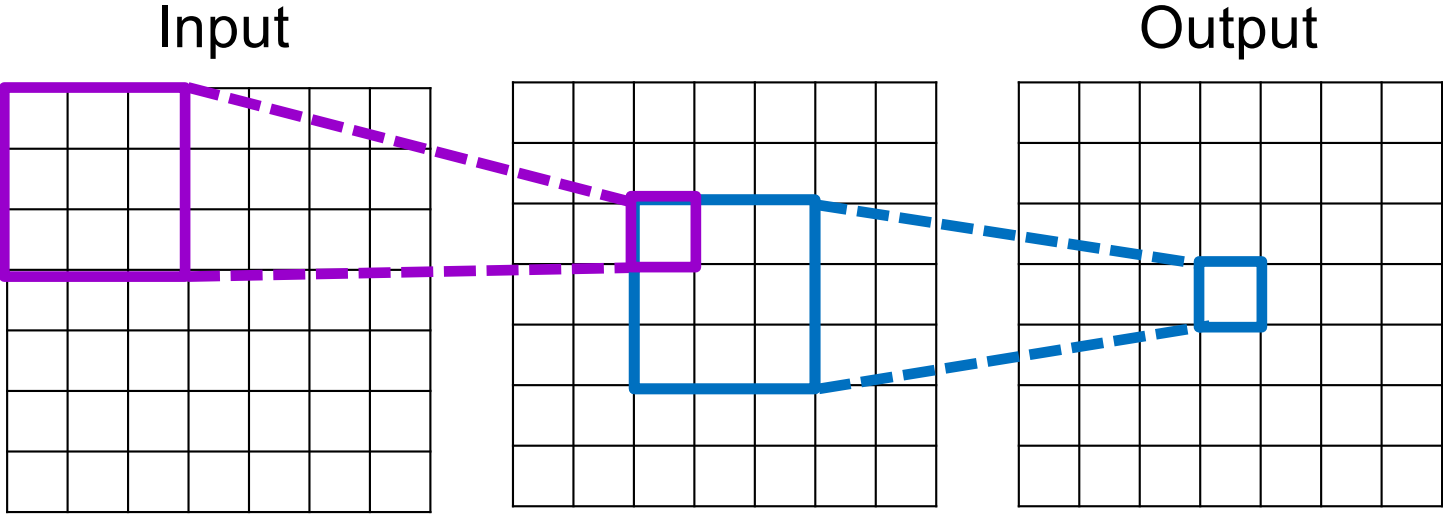
3x3 convolutions, stride 2



Receptive field size: 3

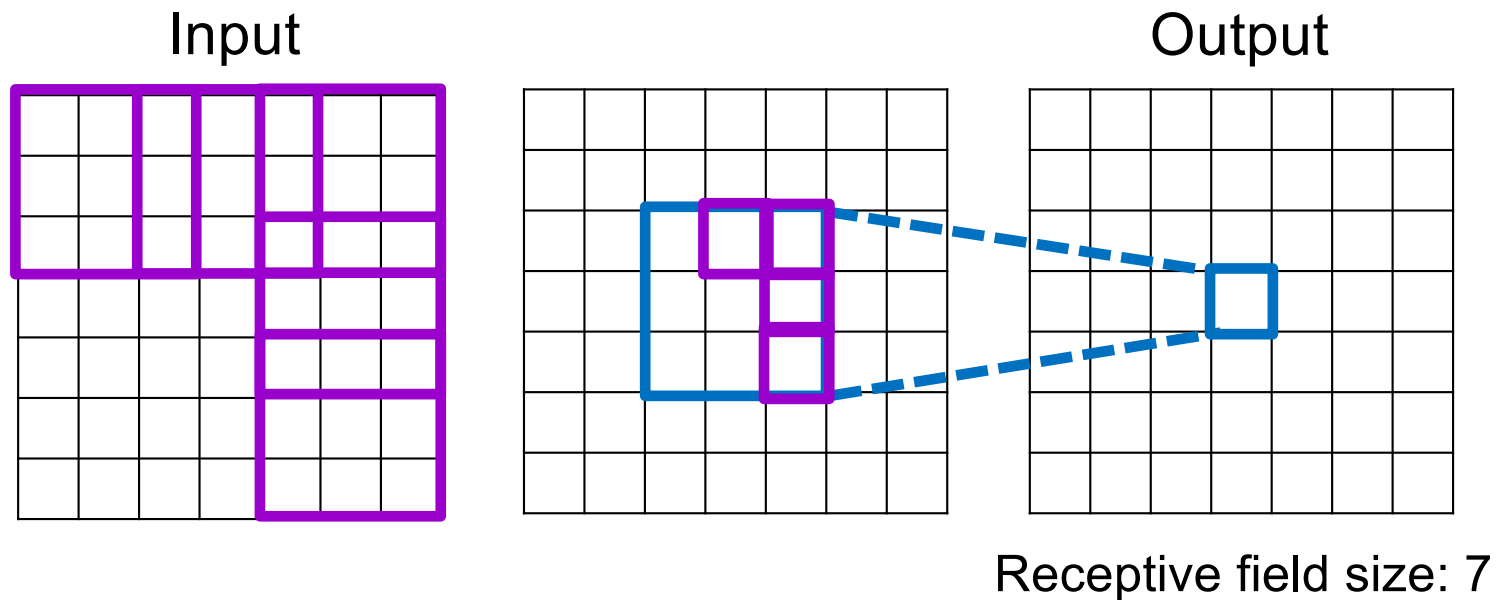
Receptive fields

3x3 convolutions, stride 2



Receptive fields

3x3 convolutions, stride 2

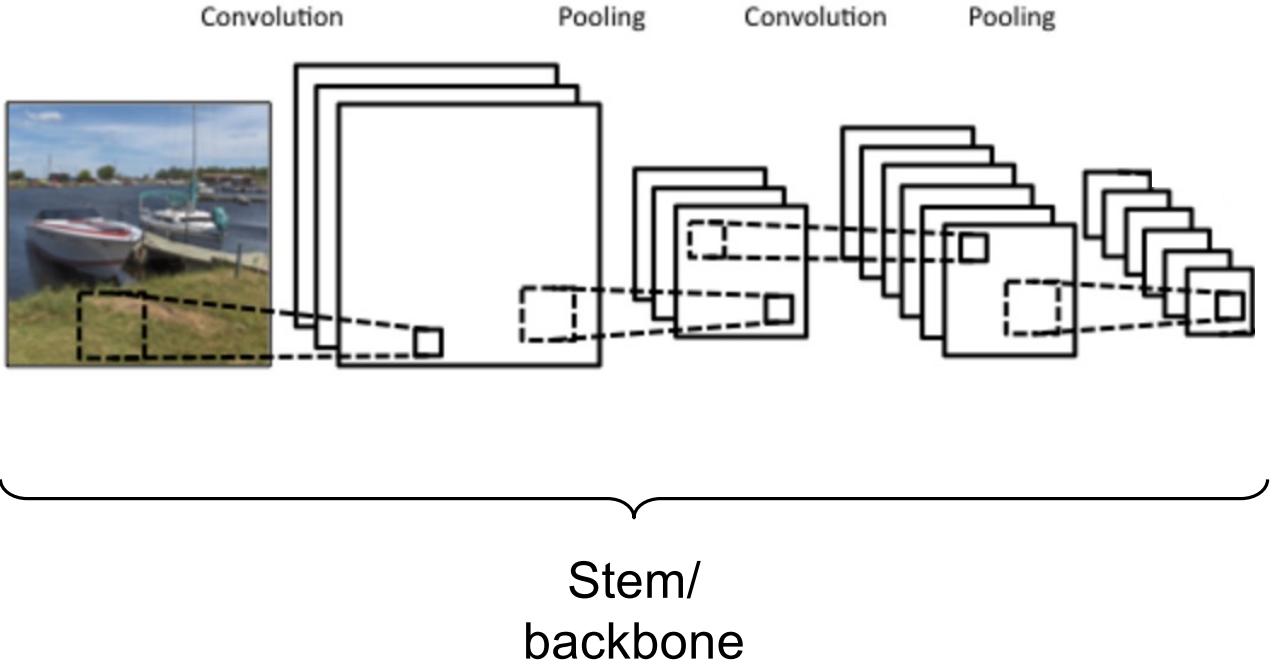


With a stride of 2, receptive field size is given by $2^{L+1} - 1$, i.e., it grows exponentially (though spatial resolution decreases exponentially)

Convolutional networks: Outline

- Basic convolutional layer
- Variants: 1x1 convolutions, depthwise convolutions
- Max pooling
- Receptive fields
- **Output layers**

Classification head: Fully connected layers, softmax



Classification head: Fully connected layers, softmax

