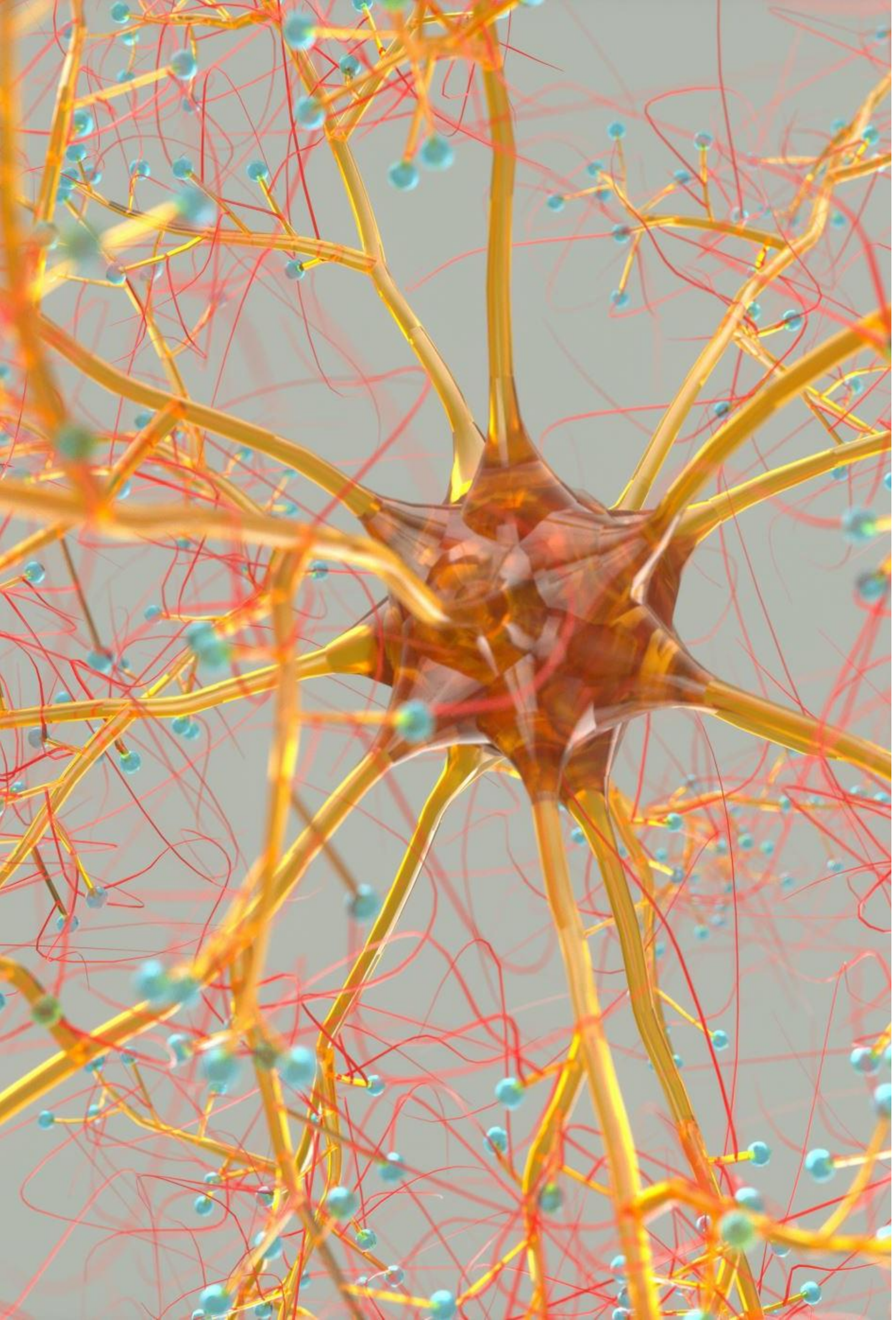


Advanced Topics in Neural Networks

Course 10

CONVOLUTIONAL NEURAL NETWORKS

MIHAELA BREABĂN

Agenda

Basic image processing: about filters and convolutions

Convolutional layers in NNs. Filter updates through backpropagation

Pooling operations

A simple convolutional neural network (CNN)

Problems solved with CNNs

Filters and convolutions

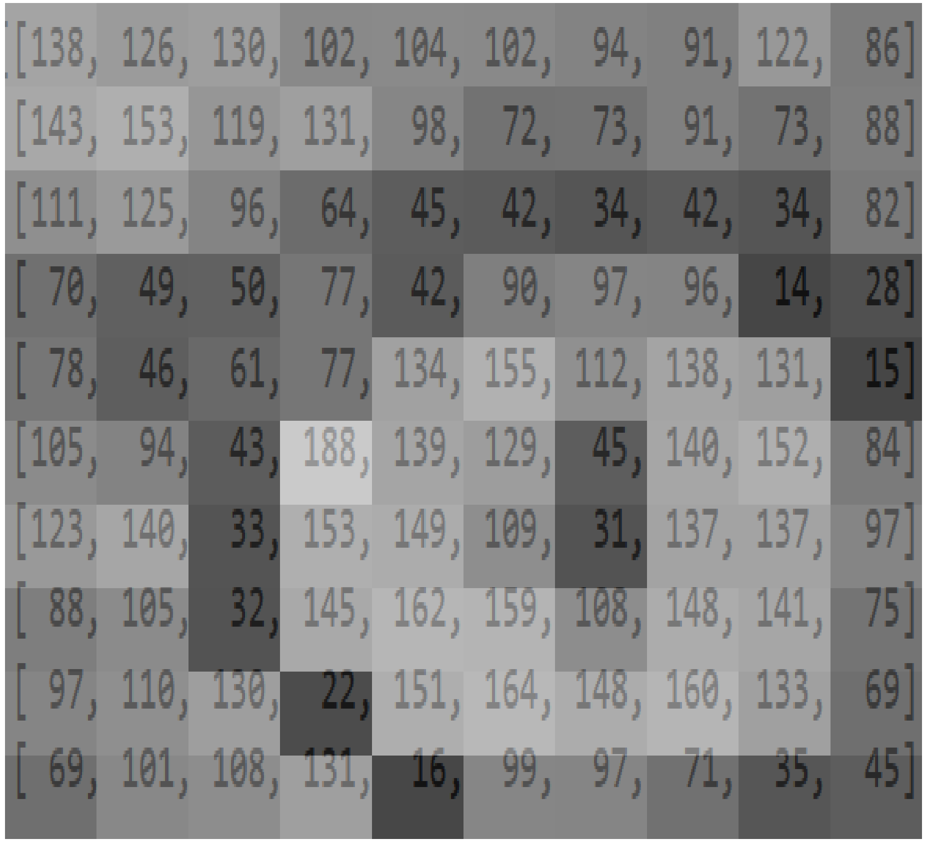
Everyone is applying image filters
Basic examples: blurring/smoothing

Everyone is applying image filters
Basic examples: sharpening

Everyone is applying image filters
Basic examples: identifying edges

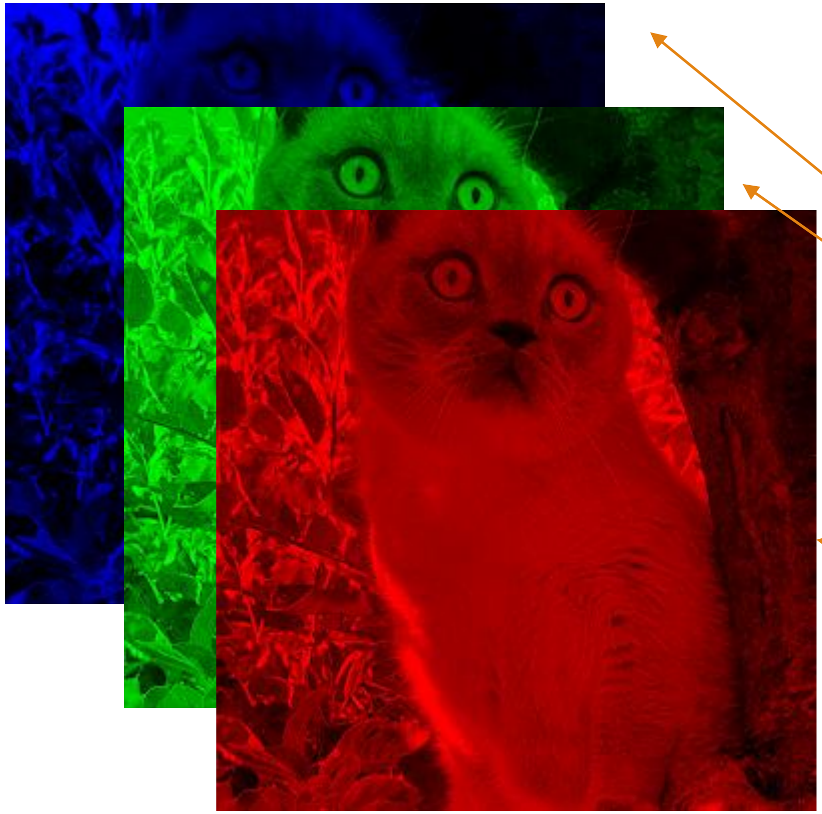


Images as functions



$F: [y_1, y_2] \times [x_1, x_2] \rightarrow [0, 255]$

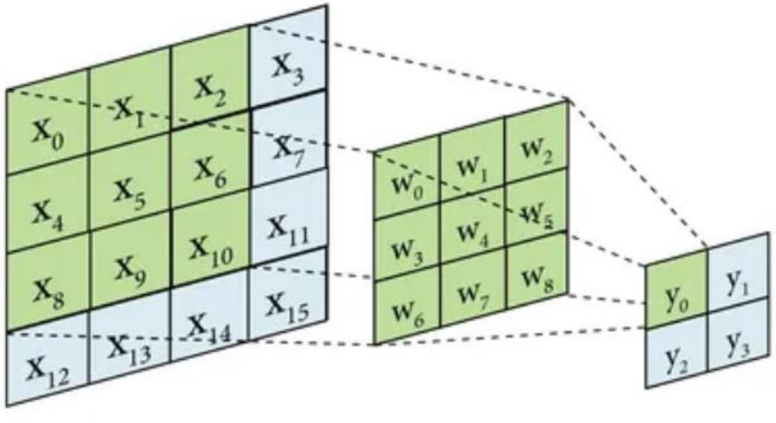
Images as functions



$$F(y,x) = \begin{cases} R(y,x) \\ G(y,x) \\ B(y,x) \end{cases}$$

CHANELS

The CONVOLUTION operation



$$y_0 = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \\ + x_4 \cdot w_3 + x_5 \cdot w_4 + x_6 \cdot w_5 + \\ + x_8 \cdot w_6 + x_9 \cdot w_7 + x_{10} \cdot w_8$$

1	**1**	**1**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**

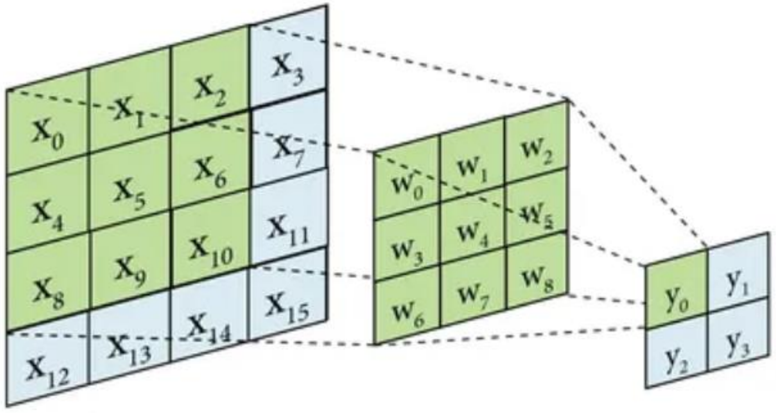
*****

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

filter/kernel

=

The CONVOLUTION operation



$$
\begin{aligned}
y_0 = {} & x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \\
& + x_4 \cdot w_3 + x_5 \cdot w_4 + x_6 \cdot w_5 + \\
& + x_8 \cdot w_6 + x_9 \cdot w_7 + x_{10} \cdot w_8
\end{aligned}
$$

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

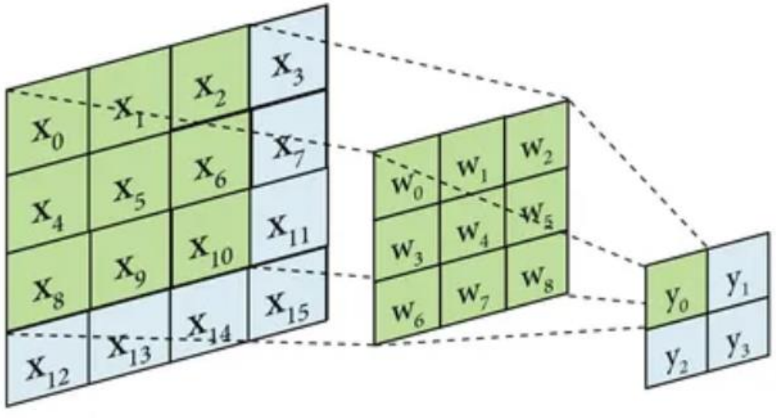
*

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

=

0			

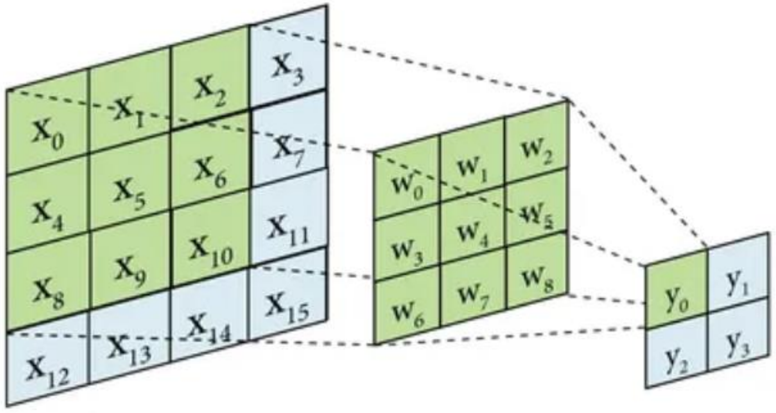
The CONVOLUTION operation



$$
\begin{aligned}
y_0 = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \\
+ x_4 \cdot w_3 + x_5 \cdot w_4 + x_6 \cdot w_5 + \\
+ x_8 \cdot w_6 + x_9 \cdot w_7 + x_{10} \cdot w_8
\end{aligned}
$$

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 & 0 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
0 & 3 & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

The CONVOLUTION operation



$$
\begin{aligned}
y_0 = {} & x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \\
& + x_4 \cdot w_3 + x_5 \cdot w_4 + x_6 \cdot w_5 + \\
& + x_8 \cdot w_6 + x_9 \cdot w_7 + x_{10} \cdot w_8
\end{aligned}
$$

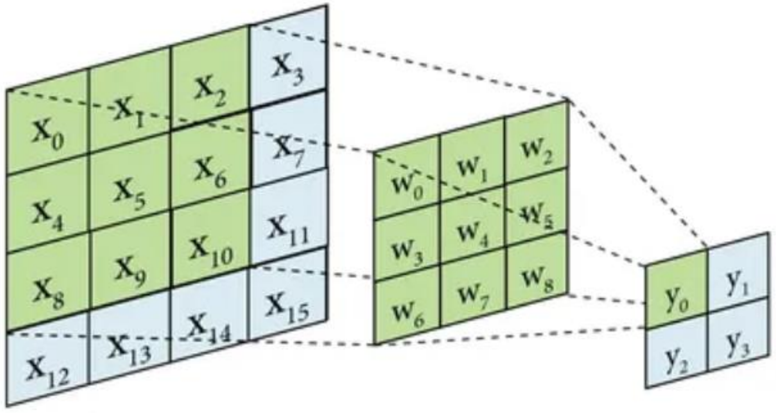
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

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

=

0	3	3	

The CONVOLUTION operation



$$y_0 = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \\ + x_4 \cdot w_3 + x_5 \cdot w_4 + x_6 \cdot w_5 + \\ + x_8 \cdot w_6 + x_9 \cdot w_7 + x_{10} \cdot w_8$$

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

*

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

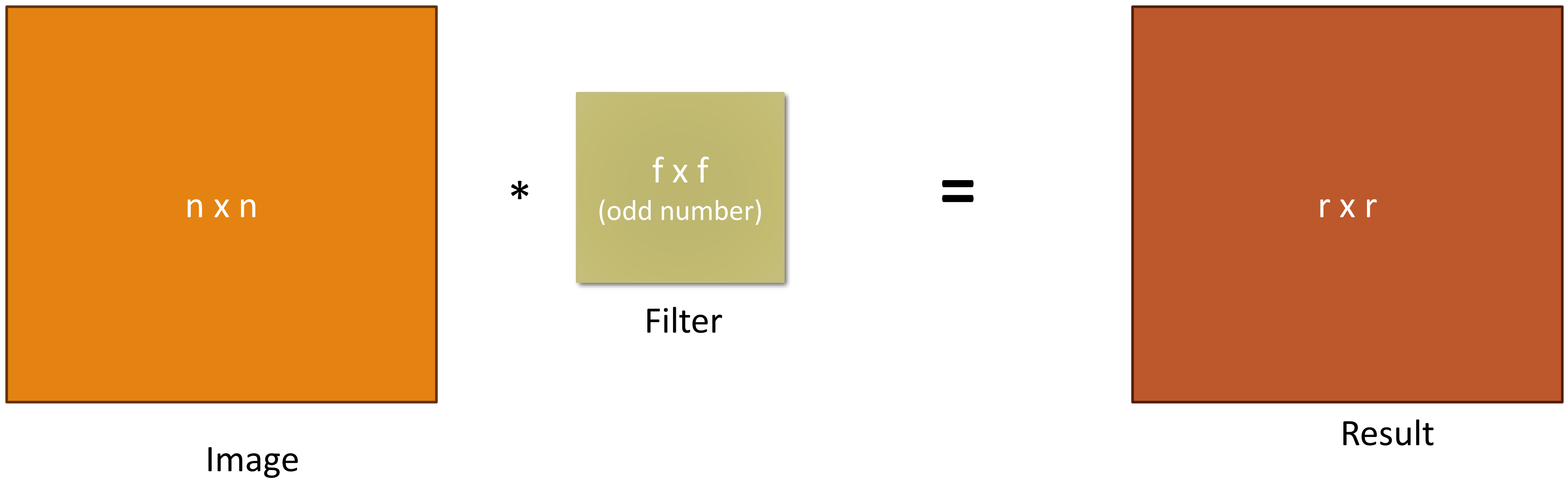
=

0	3	3	0

The CONVOLUTION operation
More examples

https://drive.google.com/file/d/1u0xEEzhCx2uPjrEUx1Krk6xas3tjNWkH/view?usp=sharing

The CONVOLUTION operation
Size of the result



$$\boxed{r = n - f + 1}$$

Padding

- Avoids shrinking the output
- Exploits better edge information

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & 6 \\ 6 & 6 \end{bmatrix}$$

Padding

- Avoids shrinking the output
- Exploits better edge information

p=1

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

*

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

=

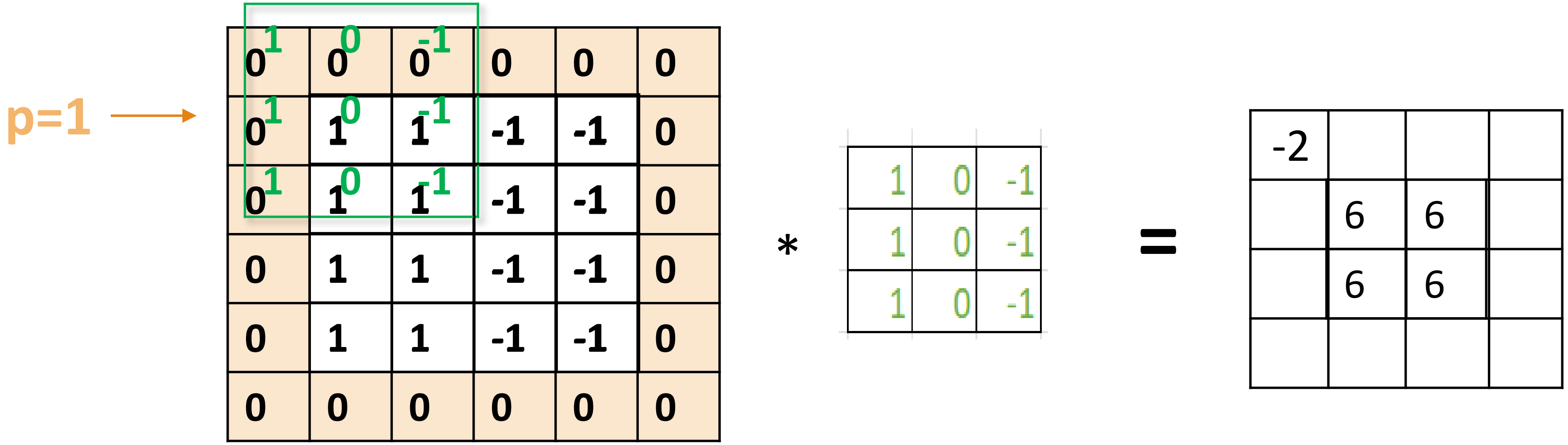
	6	6	
	6	6	

$$\boxed{r = n+2p-f+1}$$

- ***"same"* convolution**: r=n -> p=(f-1)/2
- ***"valid"* convolution**: r<n (no padding)

Padding

- Avoids shrinking the output
- Exploits better edge information



$$\boxed{r = n+2p-f+1}$$

- ***"same"* convolution**: r=n -> p=(f-1)/2
- ***"valid"* convolution**: r<n (no padding)

Padding

- Avoids shrinking the output
- Exploits better edge information

p=1

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

$*$

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

$=$

-2	0	0	-2
	6	6	
	6	6	

$\boxed{r = n+2p-f+1}$

- ***"same"* convolution**: r=n -> p=(f-1)/2
- ***"valid"* convolution**: r<n (no padding)

Strided convolutions

Slide the filter over a number of steps/strides **s**

s=1:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \end{bmatrix}$$

s=2:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 3 \\ 0 & 3 \end{bmatrix}$$

$$\boxed{r = (n+2p-f)/s+1}$$

Dilated/atrous convolutions

- Expend the kernel by inserting empty cells
- Cover larger areas of the input -> increase the receptive field of the filter without increasing the number of parameters

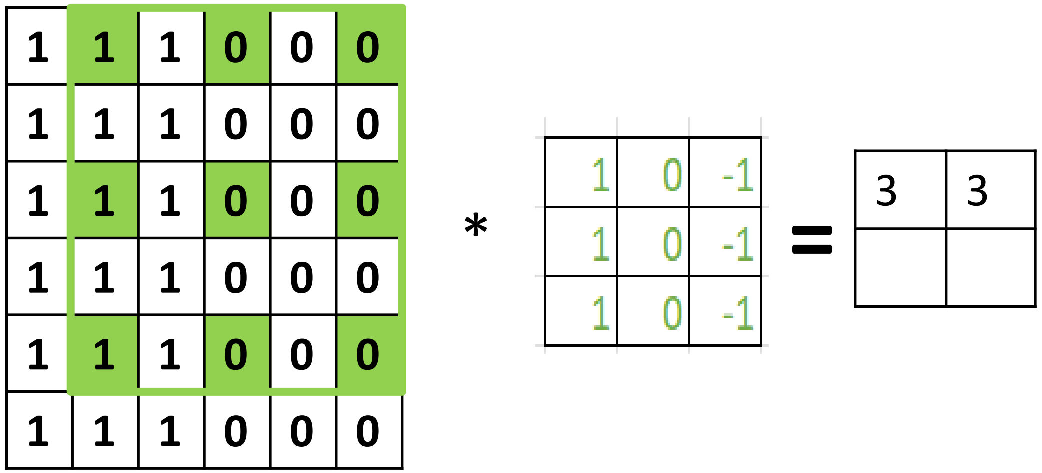
l=2: skip every 1 (l-1) cells in the input

$$
\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}
*
\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}
=
\begin{bmatrix} 3 & \\ & \end{bmatrix}
$$

Dilated/atrous convolutions

- Expend the kernel by inserting empty cells
- Cover larger areas of the input -> increase the receptive field of the filter without increasing the number of parameters

l=2: skip every 1 (l-1) cells in the input



Dilated/atrous convolutions

- Expend the kernel by inserting empty cells
- Cover larger areas of the input -> increase the receptive field of the filter without increasing the number of parameters

l=2: skip every 1 (l-1) cells in the input

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

*

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

=

3	3
3	

Dilated/atrous convolutions

- Expend the kernel by inserting empty cells
- Cover larger areas of the input -> increase the receptive field of the filter without increasing the number of parameters

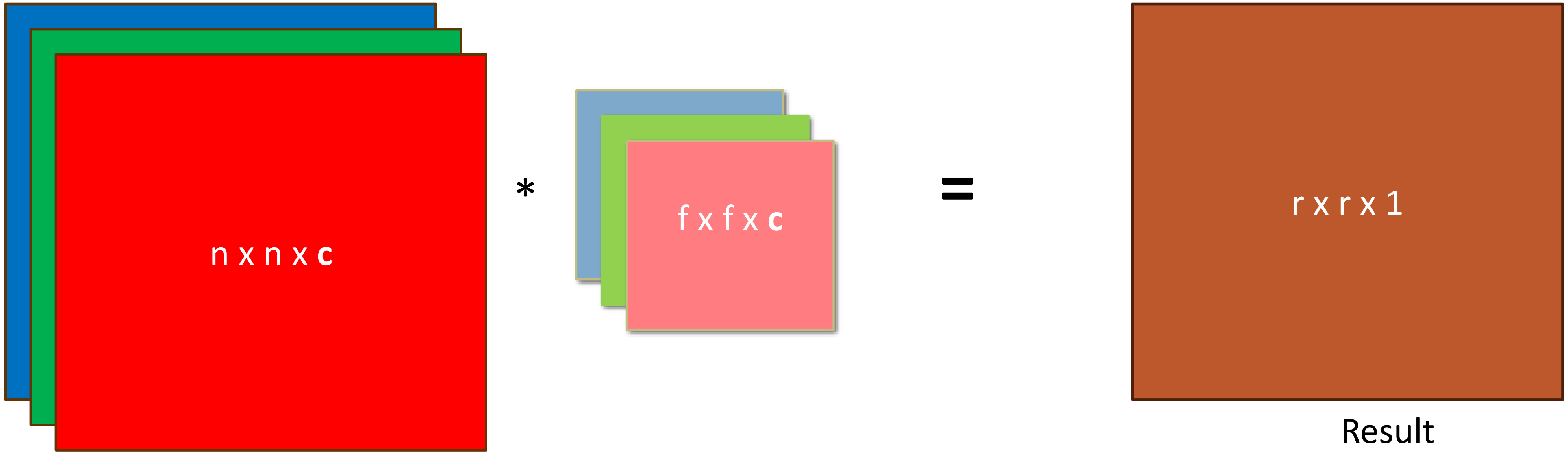
l=2: skip every 1 (l-1) cells in the input

$$\begin{array}{|c|c|c|c|c|c|}\hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 3 & 3 \\ \hline 3 & 3 \\ \hline \end{array}$$

this is equivalent to using a new kernel of size (f-1)*l+1

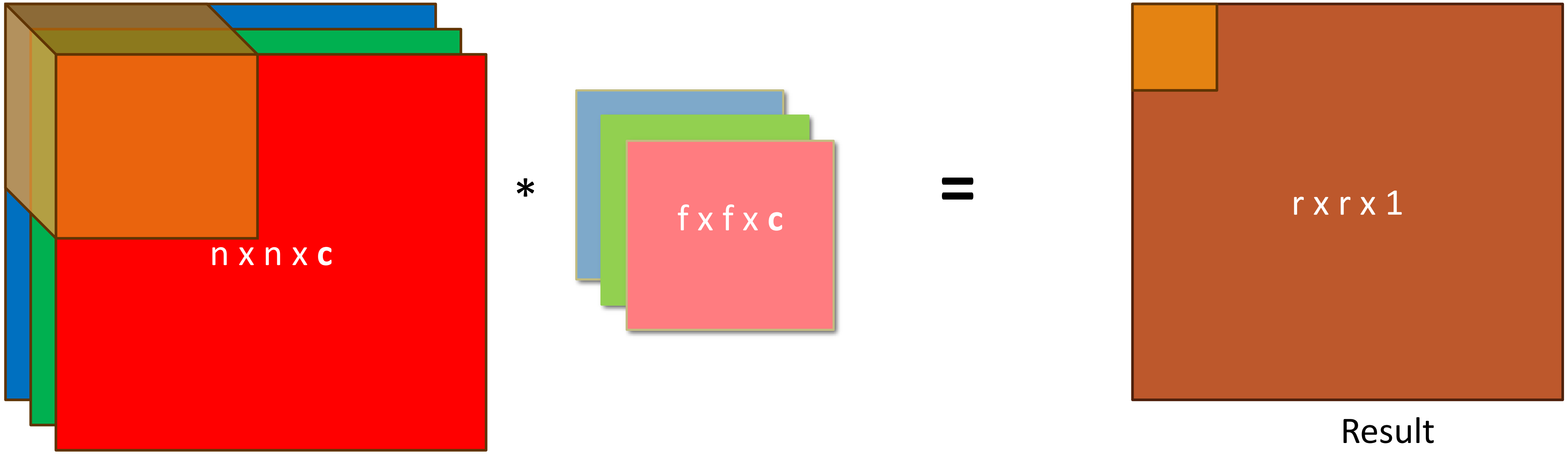
$$\boxed{r = (n+2p-(f-1)*l-1)/s+1}$$

Convolutions for RGB images



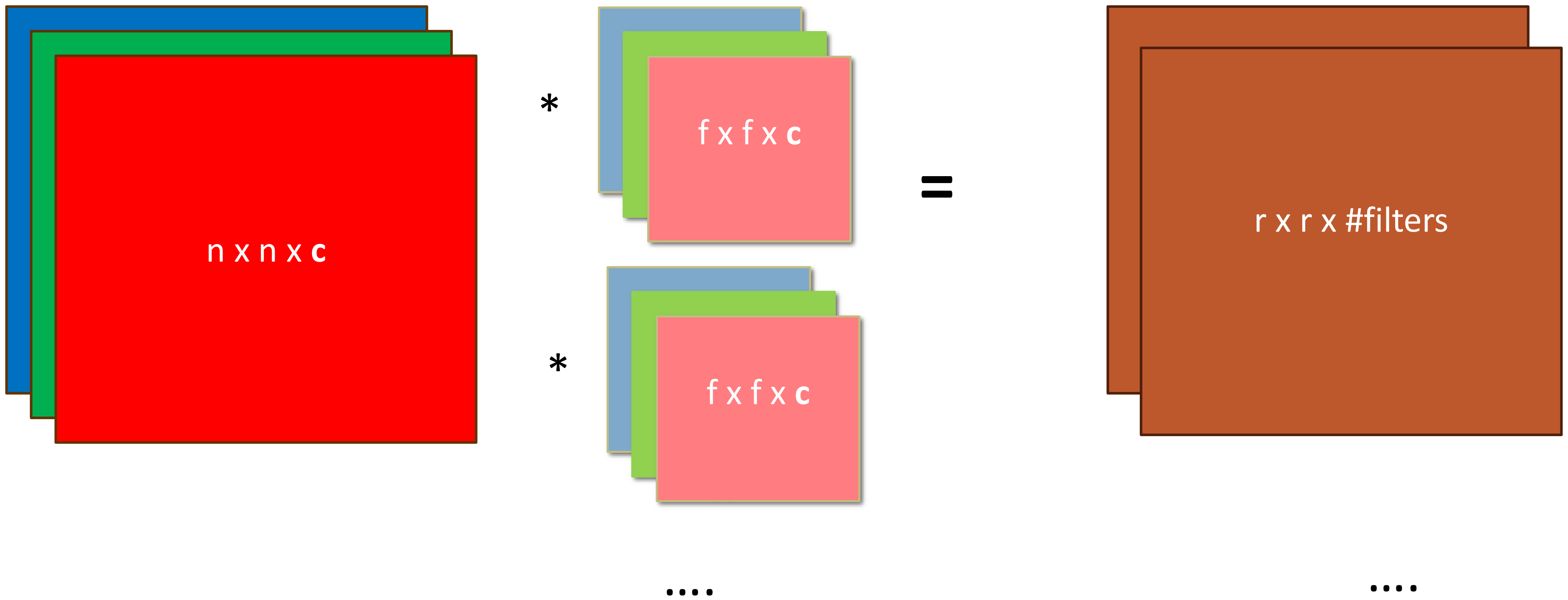
Result

Convolutions for RGB images



f x f x c factors in the sum

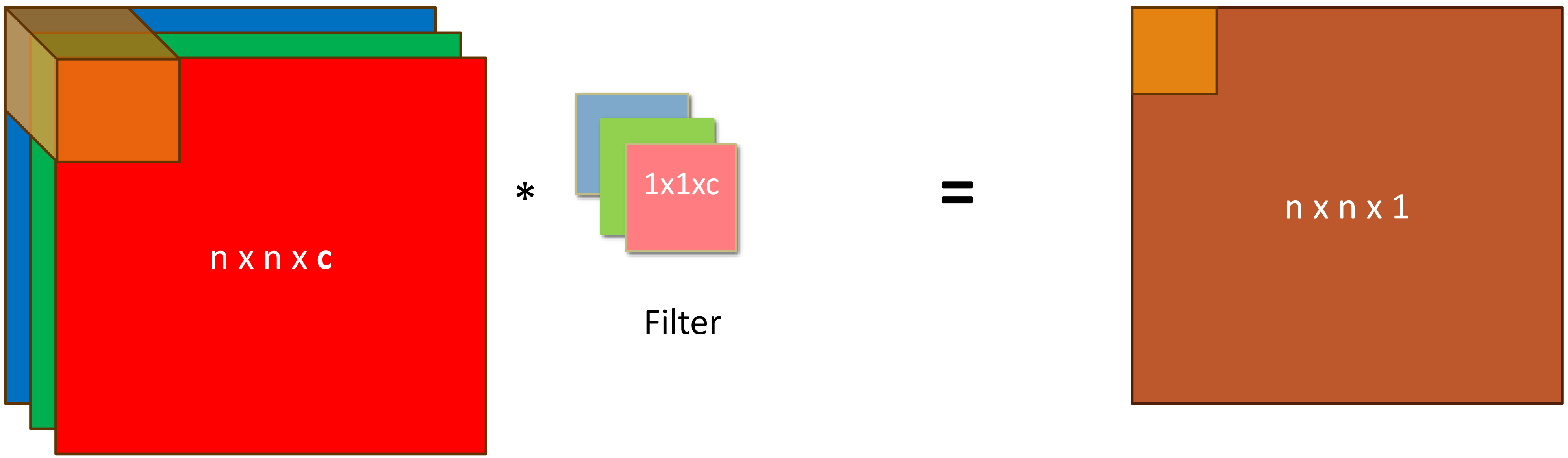
Use multiple filters to extract several features



Number of filters = number of channels in the result

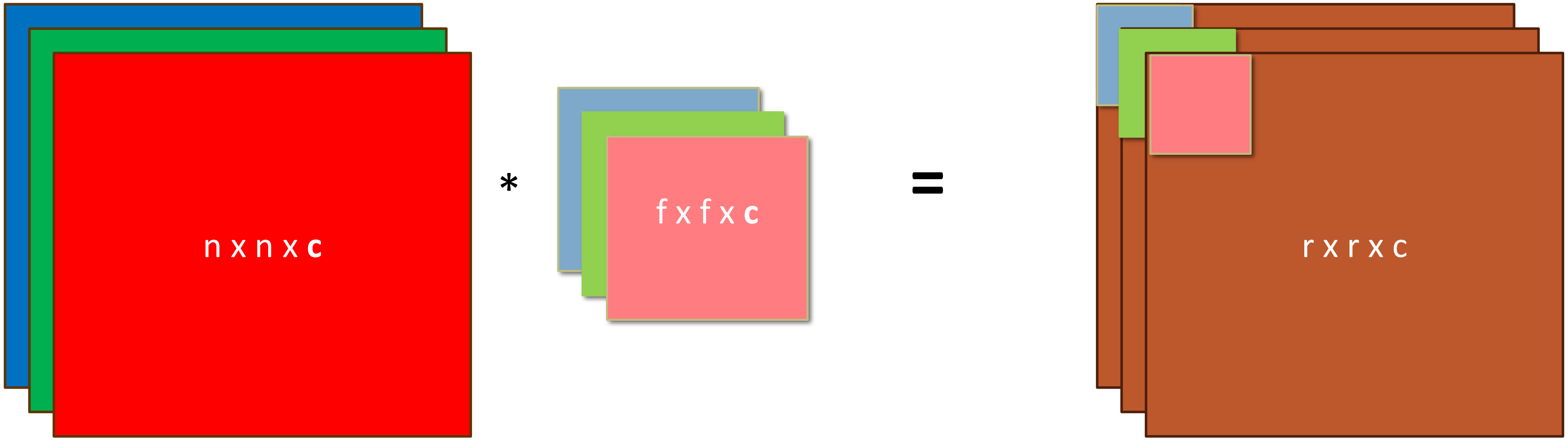
1x1 convolutions

- Combines the values on the 3rd dimension of the input matrix
- Used to reduce the number of channels



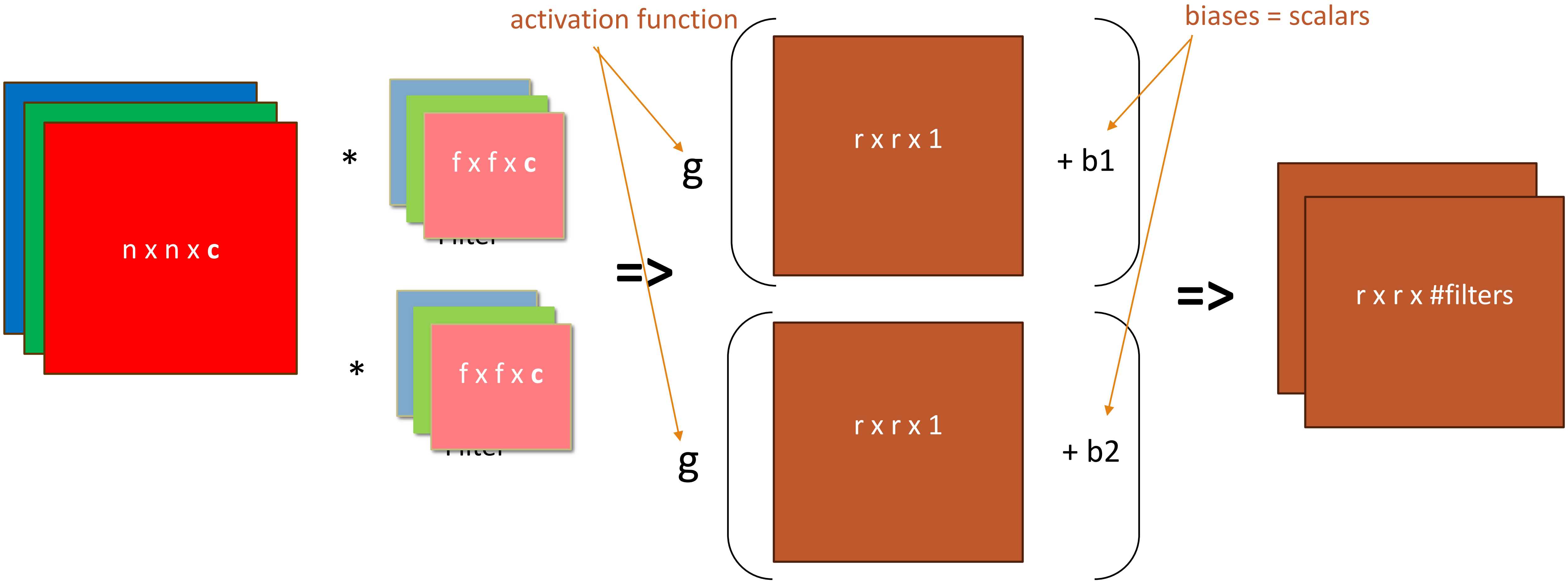
Depthwise-separable convolution

- Every channel in the input is processed independently with one channel in the filter



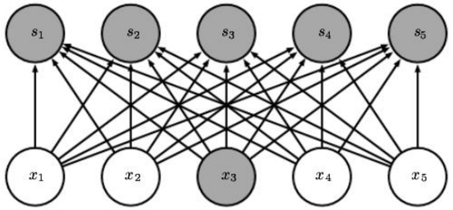
A convolutional layer

Local linear combinations (=convolutions) + pointwise non-linearities



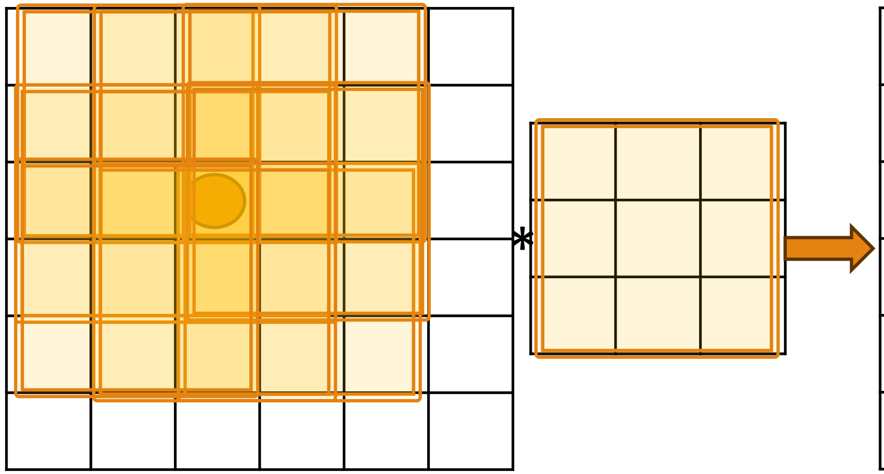
Benefits of convolution in NNs
Sparse interactions

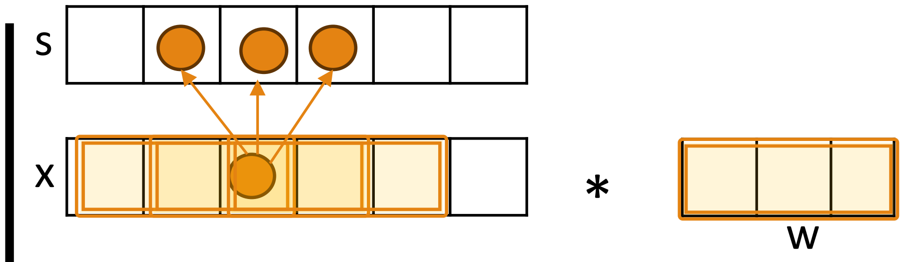
- Traditional NNs: each input unit interacts with each output unit through a parameter – O(m x n)

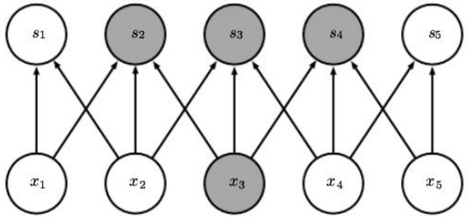


fully connected layers

- CNNs: an entire input layer interacts with an output layer through a kernel of reduced size -> reduces the memory requirements and the number of operations – O(k x n)



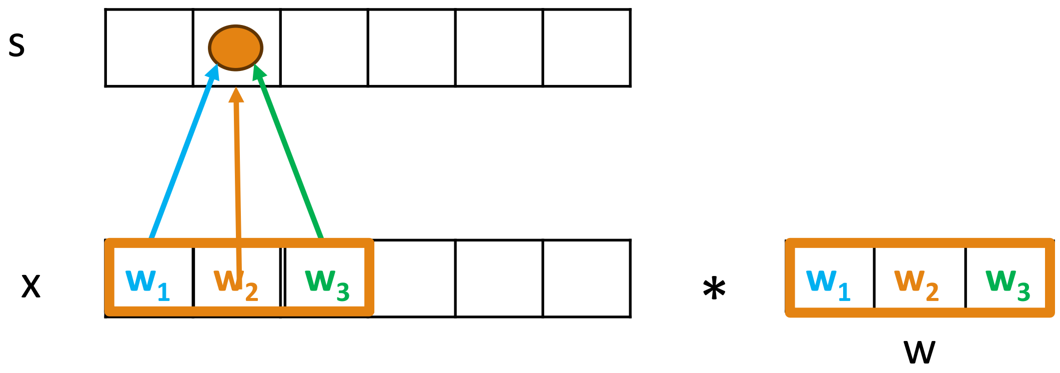


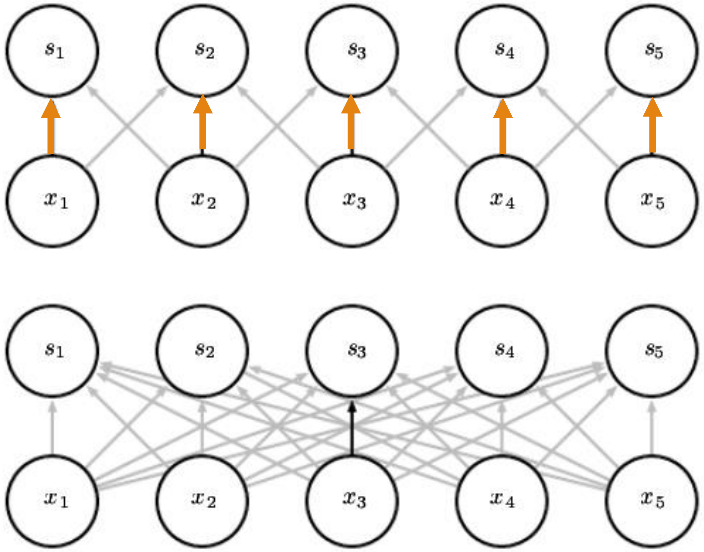


convolutional layers

Benefits of convolution in NNs
Parameter sharing

Convolution shares the same parameters across all spatial locations



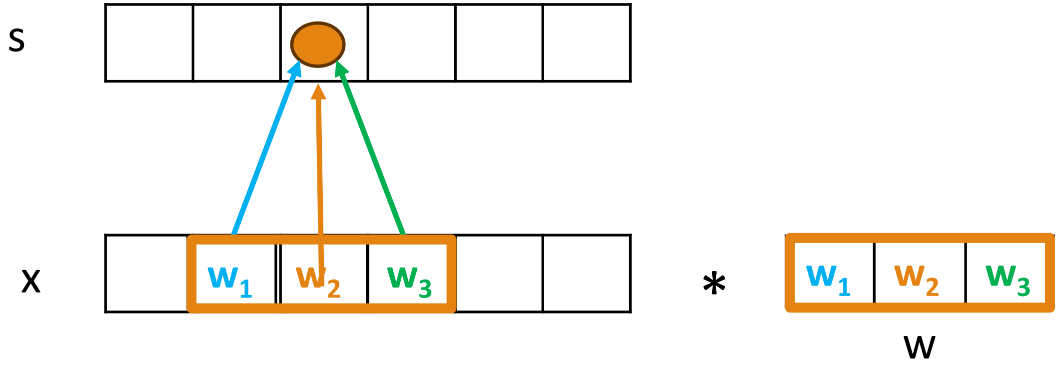


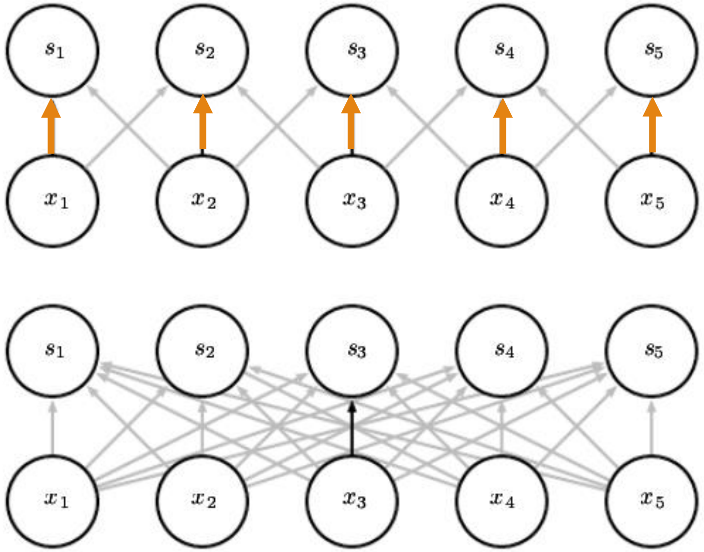
Benefits of convolution in NNs
Parameter sharing

Convolution shares the same parameters across all spatial locations

=> Rather than learning a separate set of parameters for every location, we learn only one set

=> Equivariance to translation





A convolutional layer in PyTorch

torch.nn.Conv2d (*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None*)
torch.nn.Conv3d (*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)*

- **in_channels (int)** – Number of channels in the input image.
- **out_channels (int)** – Number of channels produced by the convolution.
- **kernel_size (int or tuple)** – Size of the convolving kernel.
- **bias (bool, optional)** – If True, adds a learnable bias to the output. Default: True.
- **stride :** controls the stride for the cross-correlation, a single number or a tuple.
- **padding :** controls the amount of padding applied to the input. It can be either a string {'valid', 'same'} or a tuple of ints giving the amount of implicit padding applied on both sides.
- **dilation :** controls the spacing between the kernel points; also known as the à trous algorithm.
- **groups :** controls the connections between inputs and outputs. in_channels and out_channels must both be divisible by groups.

https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

Backprop through a convolutional layer

Use case:
f=3, s=2, p=0

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \Rightarrow \ldots\ldots \Rightarrow \hat{y} \quad \Rightarrow L(\hat{y}, y)$$

Forward pass

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

$*$

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

=>

z_{11}	z_{12}
z_{21}	z_{22}

=>…… => $\hat{y}$ => $L(\hat{y}, y)$

Backward pass

$\frac{\delta L}{\delta w_{11}}$	$\frac{\delta L}{\delta w_{12}}$	$\frac{\delta L}{\delta w_{13}}$
$\frac{\delta L}{\delta w_{21}}$	$\frac{\delta L}{\delta w_{22}}$	$\frac{\delta L}{\delta w_{23}}$
$\frac{\delta L}{\delta w_{31}}$	$\frac{\delta L}{\delta w_{32}}$	$\frac{\delta L}{\delta w_{33}}$

$$w_{ij} = w_{ij} - \alpha \frac{\delta L}{\delta w_{ij}}$$

Use case:
f=3, s=2, p=0

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

=>

z_{11}	z_{12}
z_{21}	z_{22}

backpropagated

$$\frac{\delta L}{\delta w_{11}} = \frac{\delta z_{11}}{\delta w_{11}}\frac{\delta L}{\delta z_{11}} + \frac{\delta z_{12}}{\delta w_{11}}\frac{\delta L}{\delta z_{12}} + \frac{\delta z_{21}}{\delta w_{11}}\frac{\delta L}{\delta z_{21}} + \frac{\delta z_{22}}{\delta w_{11}}\frac{\delta L}{\delta z_{22}}$$

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}$$

backpropagated

$$\frac{\delta L}{\delta w_{11}} = \boxed{\frac{\delta z_{11}}{\delta w_{11}}}\frac{\delta L}{\delta z_{11}} + \boxed{\frac{\delta z_{12}}{\delta w_{11}}}\frac{\delta L}{\delta z_{12}} + \boxed{\frac{\delta z_{21}}{\delta w_{11}}}\frac{\delta L}{\delta z_{21}} + \boxed{\frac{\delta z_{22}}{\delta w_{11}}}\frac{\delta L}{\delta z_{22}}$$

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\frac{\delta L}{\delta z_{12}} + a_{31}\frac{\delta L}{\delta z_{21}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\frac{\delta L}{\delta z_{12}} + a_{32}\frac{\delta L}{\delta z_{21}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\frac{\delta L}{\delta z_{12}} + a_{31}\frac{\delta L}{\delta z_{21}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\frac{\delta L}{\delta z_{12}} + a_{32}\frac{\delta L}{\delta z_{21}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{13}} = a_{13}\frac{\delta L}{\delta z_{11}} + a_{15}\frac{\delta L}{\delta z_{12}} + a_{33}\frac{\delta L}{\delta z_{21}} + a_{35}\frac{\delta L}{\delta z_{22}}$$

...

$$\frac{\delta L}{\delta w_{33}} = a_{33}\frac{\delta L}{\delta z_{11}} + a_{35}\frac{\delta L}{\delta z_{12}} + a_{53}\frac{\delta L}{\delta z_{21}} + a_{55}\frac{\delta L}{\delta z_{22}}$$

$$z_{11} = w_{11}a_{11} + w_{12}a_{12} + w_{13}a_{13} + w_{21}a_{21} + w_{22}a_{22} + w_{23}a_{23} + w_{31}a_{31} + w_{32}a_{32} + w_{33}a_{33}$$

$$z_{12} = w_{11}a_{13} + w_{12}a_{14} + w_{13}a_{15} + w_{21}a_{23} + w_{22}a_{24} + w_{23}a_{25} + w_{31}a_{33} + w_{32}a_{34} + w_{33}a_{35}$$

$$z_{21} = w_{11}a_{31} + w_{12}a_{32} + w_{13}a_{33} + w_{21}a_{41} + w_{22}a_{42} + w_{23}a_{43} + w_{31}a_{51} + w_{32}a_{52} + w_{33}a_{53}$$

$$z_{22} = w_{11}a_{33} + w_{12}a_{34} + w_{13}a_{35} + w_{21}a_{43} + w_{22}a_{44} + w_{23}a_{45} + w_{31}a_{53} + w_{32}a_{54} + w_{33}a_{55}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

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

$$\frac{\delta L}{\delta z_{11}} +$$

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\frac{\delta L}{\delta z_{12}} + a_{31}\frac{\delta L}{\delta z_{21}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\frac{\delta L}{\delta z_{12}} + a_{32}\frac{\delta L}{\delta z_{21}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{13}} = a_{13}\frac{\delta L}{\delta z_{11}} + a_{15}\frac{\delta L}{\delta z_{12}} + a_{33}\frac{\delta L}{\delta z_{21}} + a_{35}\frac{\delta L}{\delta z_{22}}$$

$$\ldots$$

$$\frac{\delta L}{\delta w_{33}} = a_{33}\frac{\delta L}{\delta z_{11}} + a_{35}\frac{\delta L}{\delta z_{12}} + a_{53}\frac{\delta L}{\delta z_{21}} + a_{55}\frac{\delta L}{\delta z_{22}}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

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

$\dfrac{\delta L}{\delta z_{11}}$ +

a_{13}	a_{14}	a_{15}
a_{23}	a_{24}	a_{25}
a_{33}	a_{34}	a_{35}

$\dfrac{\delta L}{\delta z_{12}}$ +

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\boxed{\frac{\delta L}{\delta z_{12}}} + a_{31}\frac{\delta L}{\delta z_{21}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\boxed{\frac{\delta L}{\delta z_{12}}} + a_{32}\frac{\delta L}{\delta z_{21}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{13}} = a_{13}\frac{\delta L}{\delta z_{11}} + a_{15}\boxed{\frac{\delta L}{\delta z_{12}}} + a_{33}\frac{\delta L}{\delta z_{21}} + a_{35}\frac{\delta L}{\delta z_{22}}$$

$$\cdots$$

$$\frac{\delta L}{\delta w_{33}} = a_{33}\frac{\delta L}{\delta z_{11}} + a_{35}\boxed{\frac{\delta L}{\delta z_{12}}} + a_{53}\frac{\delta L}{\delta z_{21}} + a_{55}\frac{\delta L}{\delta z_{22}}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

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

$\frac{\delta L}{\delta z_{11}}$ +

a_{13}	a_{14}	a_{15}
a_{23}	a_{24}	a_{25}
a_{33}	a_{34}	a_{35}

$\frac{\delta L}{\delta z_{12}}$ +

a_{31}	a_{32}	a_{33}
a_{41}	a_{42}	a_{43}
a_{51}	a_{52}	a_{53}

$\frac{\delta L}{\delta z_{21}}$ +

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\frac{\delta L}{\delta z_{12}} + a_{31}\boxed{\frac{\delta L}{\delta z_{21}}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\frac{\delta L}{\delta z_{12}} + a_{32}\boxed{\frac{\delta L}{\delta z_{21}}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{13}} = a_{13}\frac{\delta L}{\delta z_{11}} + a_{15}\frac{\delta L}{\delta z_{12}} + a_{33}\boxed{\frac{\delta L}{\delta z_{21}}} + a_{35}\frac{\delta L}{\delta z_{22}}$$

...

$$\frac{\delta L}{\delta w_{33}} = a_{33}\frac{\delta L}{\delta z_{11}} + a_{35}\frac{\delta L}{\delta z_{12}} + a_{53}\boxed{\frac{\delta L}{\delta z_{21}}} + a_{55}\frac{\delta L}{\delta z_{22}}$$

Use case:
f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \frac{\delta L}{\delta z_{11}} + \begin{bmatrix} a_{13} & a_{14} & a_{15} \\ a_{23} & a_{24} & a_{25} \\ a_{33} & a_{34} & a_{35} \end{bmatrix} \frac{\delta L}{\delta z_{12}} + \begin{bmatrix} a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} \frac{\delta L}{\delta z_{21}} + \begin{bmatrix} a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} \frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{11}} = a_{11}\frac{\delta L}{\delta z_{11}} + a_{13}\frac{\delta L}{\delta z_{12}} + a_{31}\frac{\delta L}{\delta z_{21}} + a_{33}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{12}} = a_{12}\frac{\delta L}{\delta z_{11}} + a_{14}\frac{\delta L}{\delta z_{12}} + a_{32}\frac{\delta L}{\delta z_{21}} + a_{34}\frac{\delta L}{\delta z_{22}}$$

$$\frac{\delta L}{\delta w_{13}} = a_{13}\frac{\delta L}{\delta z_{11}} + a_{15}\frac{\delta L}{\delta z_{12}} + a_{33}\frac{\delta L}{\delta z_{21}} + a_{35}\frac{\delta L}{\delta z_{22}}$$

...

$$\frac{\delta L}{\delta w_{33}} = a_{33}\frac{\delta L}{\delta z_{11}} + a_{35}\frac{\delta L}{\delta z_{12}} + a_{53}\frac{\delta L}{\delta z_{21}} + a_{55}\frac{\delta L}{\delta z_{22}}$$

Use case:
f=3, s=2, p=0

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \frac{\delta L}{\delta z_{11}} + \begin{bmatrix} a_{13} & a_{14} & a_{15} \\ a_{23} & a_{24} & a_{25} \\ a_{33} & a_{34} & a_{35} \end{bmatrix} \frac{\delta L}{\delta z_{12}} + \begin{bmatrix} a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} \frac{\delta L}{\delta z_{21}} + \begin{bmatrix} a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} \frac{\delta L}{\delta z_{22}} = \begin{bmatrix} \frac{\delta L}{\delta w_{11}} & \frac{\delta L}{\delta w_{12}} & \frac{\delta L}{\delta w_{13}} \\ \frac{\delta L}{\delta w_{21}} & \frac{\delta L}{\delta w_{22}} & \frac{\delta L}{\delta w_{23}} \\ \frac{\delta L}{\delta w_{31}} & \frac{\delta L}{\delta w_{32}} & \frac{\delta L}{\delta w_{33}} \end{bmatrix}$$

$$w_{ij} = w_{ij} - \alpha \frac{\delta L}{\delta w_{ij}}$$

Use case:
f=3, s=2, p=0

$$\cdots\cdots \; \Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \Rightarrow \cdots\cdots \Rightarrow \hat{y} \quad \Rightarrow L(\hat{y}, y)$$

Backward pass

We have computed previously $\frac{\delta L}{\delta w_{ij}}$, which is necessary to update the weights of the filter.

In order to further backpropagate the error towards the first layer we also need to compute $\frac{\delta L}{\delta a_{ij}}$. This is a simple exercise; for similar computations check backpropagation through the pooling layer – after a few slides…

The pooling operator

Pooling operations

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size
- With a proper stride, used to reduce size (width and height)
- Makes the representation approximately invariant to small translations of the input

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

$\xrightarrow{f=2,\ s=2}$

$$\boxed{r = (n-f)/s+1}$$

Pooling operations
Max pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Max pooling** with f=2, s=2

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

$\xrightarrow{\text{f=2, s=2}}$

5		

Pooling operations
Max pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Max pooling** with f=2, s=2

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

$\xrightarrow{\text{f=2, s=2}}$

5	6	

Pooling operations
Max pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Max pooling** with f=2, s=2

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

$\xrightarrow{\text{f=2, s=2}}$

5	6	5

Pooling operations
Max pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Max pooling** with f=2, s=2

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

$\xrightarrow{\text{f=2, s=2}}$

5	6	5
7		

Pooling operations
Max pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics: max pooling, average pooling
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Max pooling** with f=2, s=2

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

$\xrightarrow{\text{f=2, s=2}}$

5	6	5
7	8	3
7	8	4

Pooling operations
Average pooling

- No parameters needed: aggregate the values of the matrix (using windowing-filters) extracting simple statistics: max pooling, average pooling
- Same hyper-parameters:
  - f – filter/window size
  - s – stride/step size

Example: **Average pooling** with f=2, s=2

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

$\xrightarrow{f=2,\ s=2}$

4	4.25	3
3.5	4.75	2.75
3	6.5	2

Pooling in PyTorch

torch.nn.MaxPool2d(*kernel_size*, *stride=None*, *padding=0*, *dilation=1*, *return_indices=False*, *ceil_mode=False*)

- **kernel_size** (*Union[int, Tuple[int, int]]*) – the size of the window to take a max over
- **stride** (*Union[int, Tuple[int, int]]*) – the stride of the window. Default value is kernel_size
- **padding** (*Union[int, Tuple[int, int]]*) – Implicit negative infinity padding to be added on both sides
- **dilation** (*Union[int, Tuple[int, int]]*) – a parameter that controls the stride of elements in the window
- **return_indices** (*bool*) – if True, will return the max indices along with the outputs. Useful for torch.nn.MaxUnpool2d later
- **ceil_mode** (*bool*) – when True, will use *ceil* instead of *floor* to compute the output shape

torch.nn.AvgPool2d

Backprop through a pooling layer

Use case – average pooling: f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

Average pooling ===>

z_{11}	z_{12}
z_{21}	z_{22}

=>…… => $\hat{y}$ => $L(\hat{y}, y)$

Forward pass

$$z_{11} = (a_{11}+a_{12} + a_{13} + a_{21} + a_{22} + a_{23} + a_{31} + a_{32} + a_{33})/9$$

$$z_{12} = (a_{13}+a_{14} + a_{15} + a_{23} + a_{24} + a_{25} + a_{33} + a_{34} + a_{35})/9$$

$$z_{21} = (a_{31}+a_{32} + a_{33} + a_{41} + a_{42} + a_{43} + a_{51} + a_{52} + a_{53})/9$$

$$z_{22} = (a_{33}+a_{34} + a_{35} + a_{43} + a_{44} + a_{45} + a_{53} + a_{54} + a_{55})/9$$

Use case – average pooling: f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

Average pooling ===>

z_{11}	z_{12}
z_{21}	z_{22}

=>…… => $\hat{y}$ => $L(\hat{y}, y)$

Backward pass

$$\frac{\delta L}{\delta a_{ij}} = ?$$

Use case – average pooling: f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

=>

z_{11}	z_{12}
z_{21}	z_{22}

backpropagated

$$\frac{\delta L}{\delta a_{11}} = \frac{\delta z_{11}}{\delta a_{11}} \frac{\delta L}{\delta z_{11}} = \frac{1}{9} \frac{\delta L}{\delta z_{11}}$$

$$z_{11} = (a_{11} + a_{12} + a_{13} + a_{21} + a_{22} + a_{23} + a_{31} + a_{32} + a_{33})/9$$

$$z_{12} = (a_{13} + a_{14} + a_{15} + a_{23} + a_{24} + a_{25} + a_{33} + a_{34} + a_{35})/9$$

$$z_{21} = (a_{31} + a_{32} + a_{33} + a_{41} + a_{42} + a_{43} + a_{51} + a_{52} + a_{53})/9$$

$$z_{22} = (a_{33} + a_{34} + a_{35} + a_{43} + a_{44} + a_{45} + a_{53} + a_{54} + a_{55})/9$$

Use case – average pooling: f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

=>

z_{11}	z_{12}
z_{21}	z_{22}

backpropagated

$$\frac{\delta L}{\delta a_{23}} = \frac{\delta z_{11}}{\delta a_{23}} \frac{\delta L}{\delta z_{11}} + \frac{\delta z_{12}}{\delta a_{23}} \frac{\delta L}{\delta z_{12}} = $$

$$= \frac{1}{9}\left(\frac{\delta L}{\delta z_{11}} + \frac{\delta L}{\delta z_{12}}\right)$$

$$z_{11} = (a_{11}+a_{12}+a_{13}+a_{21}+a_{22}+a_{23}+a_{31}+a_{32}+a_{33})/9$$

$$z_{12} = (a_{13}+a_{14}+a_{15}+a_{23}+a_{24}+a_{25}+a_{33}+a_{34}+a_{35})/9$$

$$z_{21} = (a_{31}+a_{32}+a_{33}+a_{41}+a_{42}+a_{43}+a_{51}+a_{52}+a_{53})/9$$

$$z_{22} = (a_{33}+a_{34}+a_{35}+a_{43}+a_{44}+a_{45}+a_{53}+a_{54}+a_{55})/9$$

Use case – average pooling: f=3, s=2, p=0

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

=>

z_{11}	z_{12}
z_{21}	z_{22}

backpropagated

$$\frac{\delta L}{\delta a_{33}} = \frac{\delta z_{11}}{\delta a_{33}}\frac{\delta L}{\delta z_{11}} + \frac{\delta z_{12}}{\delta a_{33}}\frac{\delta L}{\delta z_{12}} + \frac{\delta z_{21}}{\delta a_{33}}\frac{\delta L}{\delta z_{21}} + \frac{\delta z_{22}}{\delta a_{33}}\frac{\delta L}{\delta z_{22}} =$$

$$= \frac{1}{9}\left(\frac{\delta L}{\delta z_{11}} + \frac{\delta L}{\delta z_{12}} + \frac{\delta L}{\delta z_{21}} + \frac{\delta L}{\delta z_{22}}\right)$$

$$z_{11} = (a_{11} + a_{12} + a_{13} + a_{21} + a_{22} + a_{23} + a_{31} + a_{32} + a_{33})/9$$

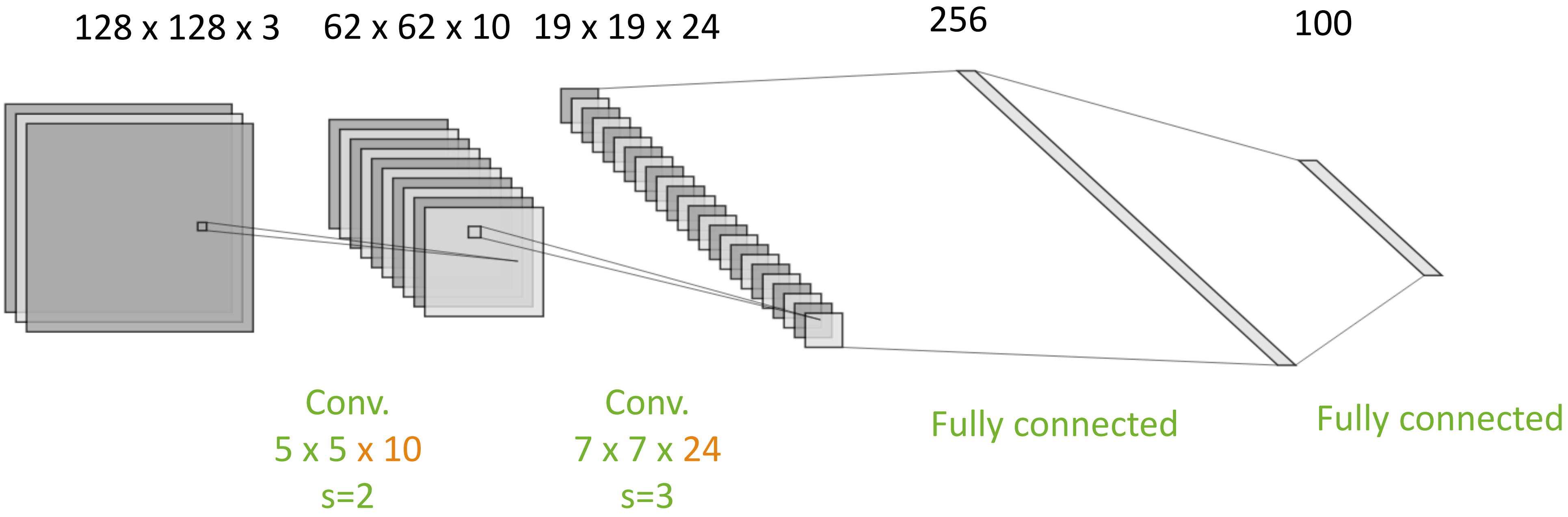
$$z_{12} = (a_{13} + a_{14} + a_{15} + a_{23} + a_{24} + a_{25} + a_{33} + a_{34} + a_{35})/9$$

$$z_{21} = (a_{31} + a_{32} + a_{33} + a_{41} + a_{42} + a_{43} + a_{51} + a_{52} + a_{53})/9$$

$$z_{22} = (a_{33} + a_{34} + a_{35} + a_{43} + a_{44} + a_{45} + a_{53} + a_{54} + a_{55})/9$$

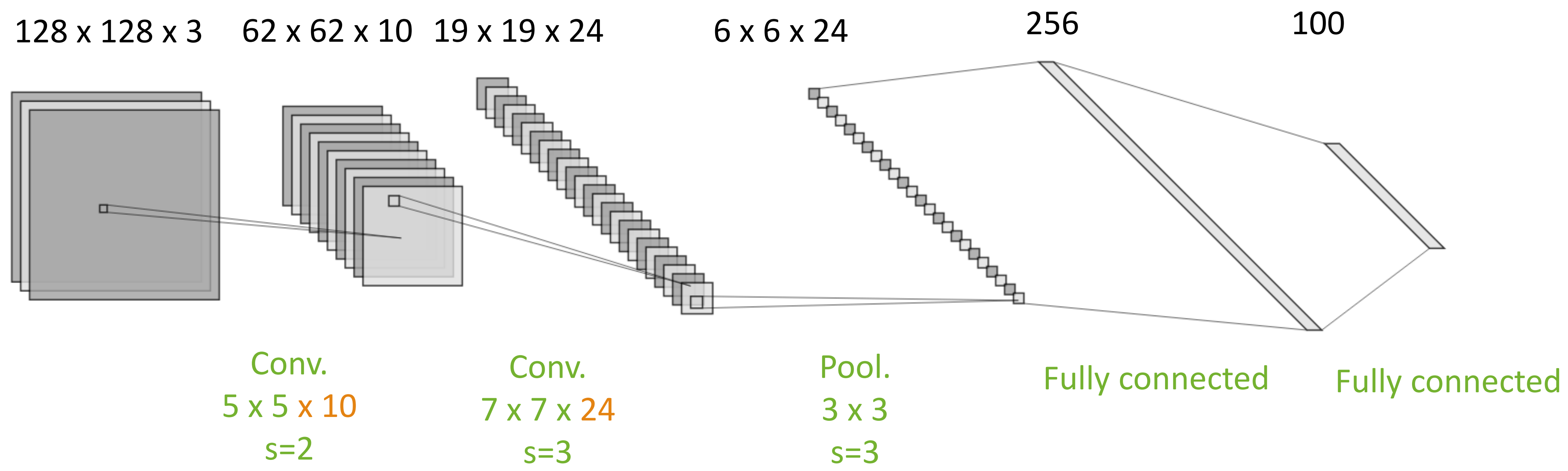
A simple CNN

Stack convolutional layers



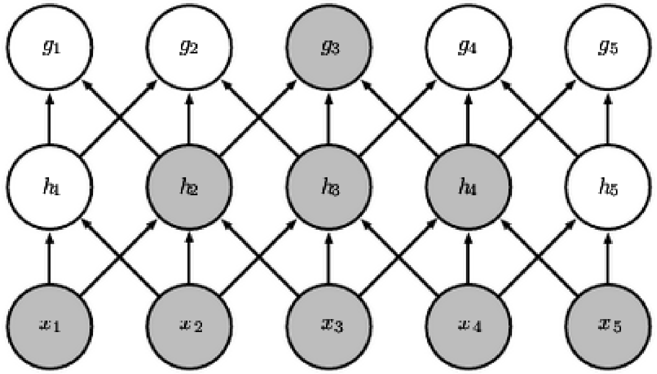
http://alexlenail.me/NN-SVG/

Stack convolutional and pooling layers



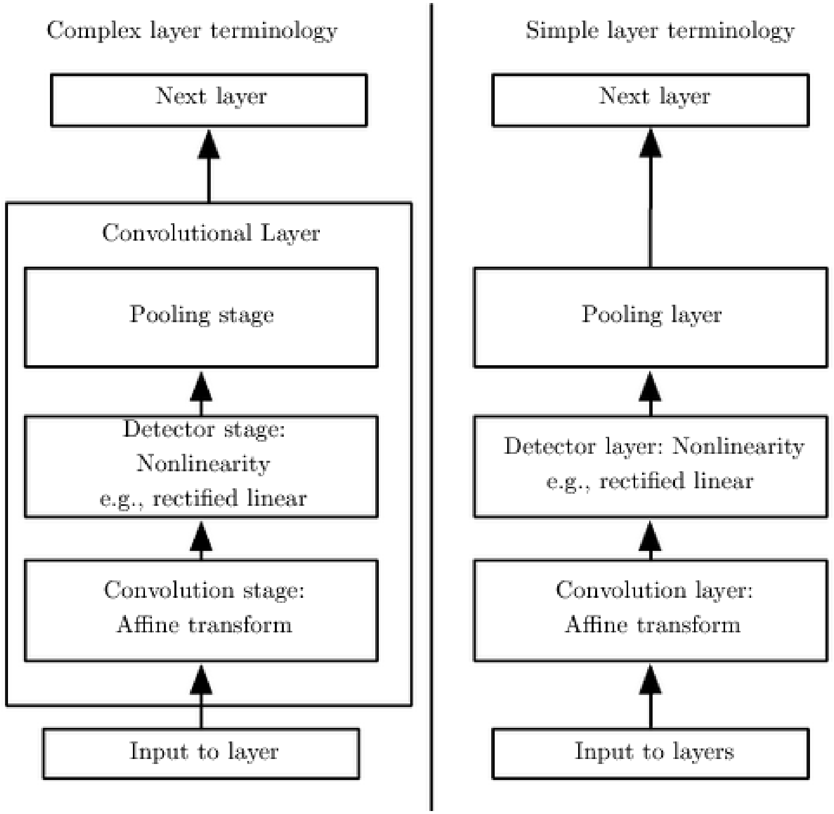
How many weights compared to the previous network?

Each layer extracts new features from the image



Units in the deeper layers may indirectly interact with a larger portion of the input => simple building blocks (from first layers) generate more complex features in superior layers

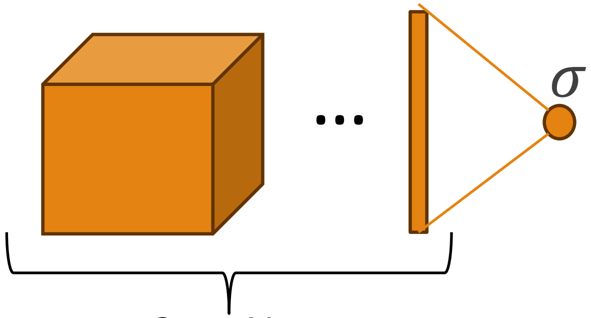
Two distinct conventions for layer count



Applications

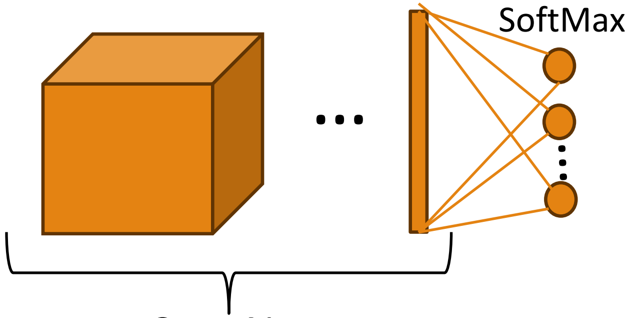
Classification

Binary classification



ConvNet

Multiclass classification



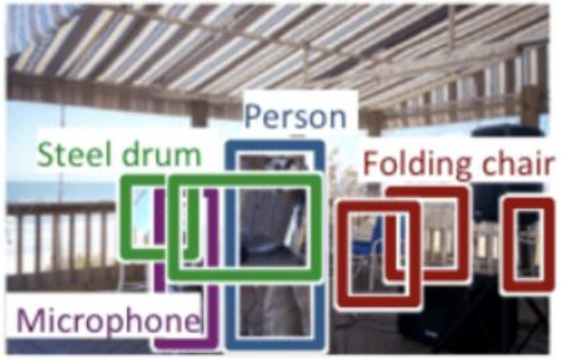
ConvNet

Popular loss function:

Cross Entropy $L(\hat{y}, y) = -\sum_{k=1..K} y_k \, log\hat{y}_k$

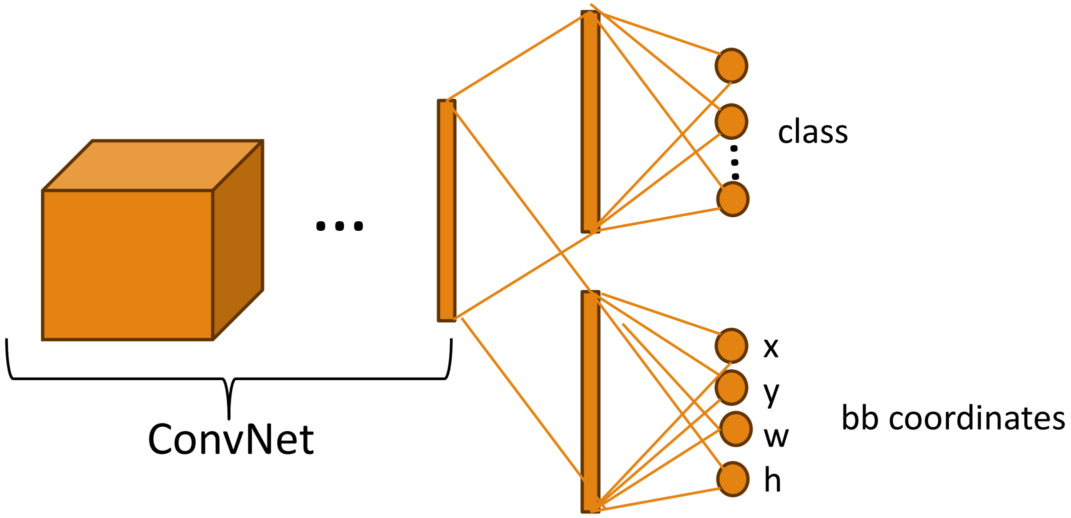
Focal loss $\mathrm{F}L(\hat{y}, y) = -\sum_{k=1..K} y_k \, (1-\hat{y}_k)^{\gamma} log\hat{y}_k$ (for imbalanced data)

Object detection



Two objectives:

- Object localization (bounding box defined by a point, width, and height) = bounding box regression
- Object classification



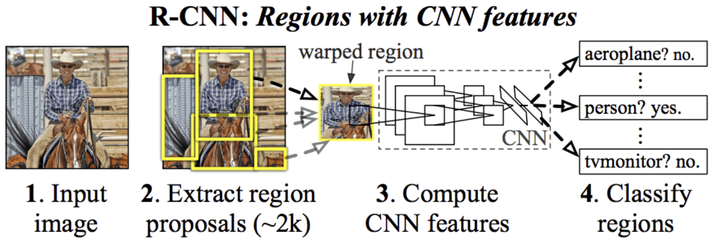
Popular architectures:

- R-CNN – several versions (Fast/Faster R-CNN)
- YOLO – several versions

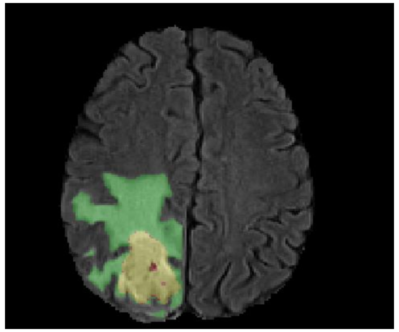
Popular loss functions:

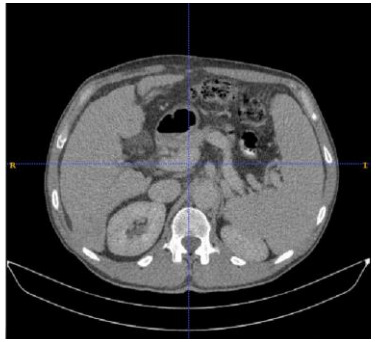
- RMSE
- 1- IOU (Intersection over Union = Jaccard metric)

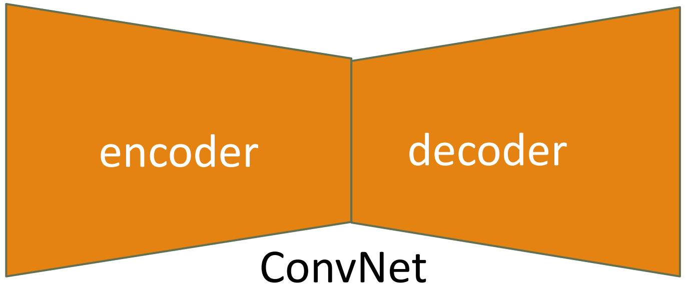
$$\text{IOU} = \frac{|A \cap B|}{|A \cup B|}$$

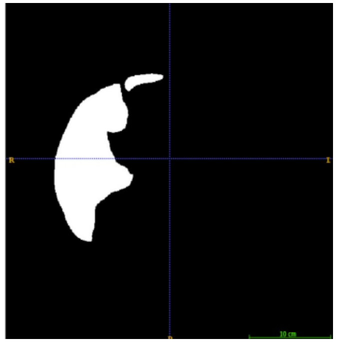


Semantic segmentation









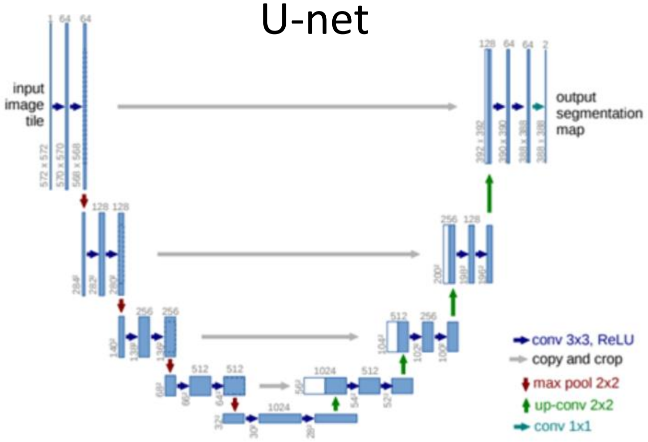
ConvNet

The most popular architecture: U-net

Popular loss functions:

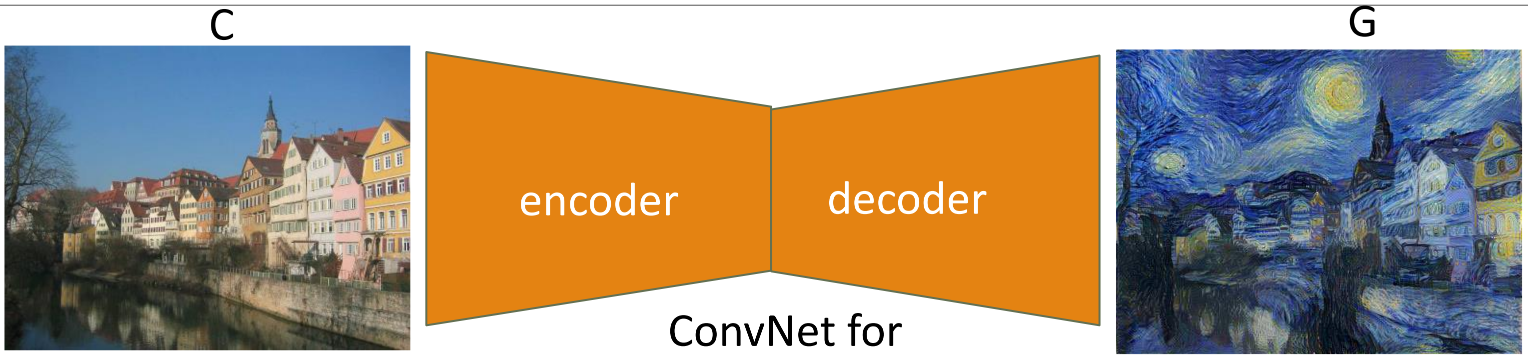
- Focal loss (classification at the pixel level)
- Dice loss: 1- $\frac{2|A \cap B|}{|A|+|B|}$

U-net

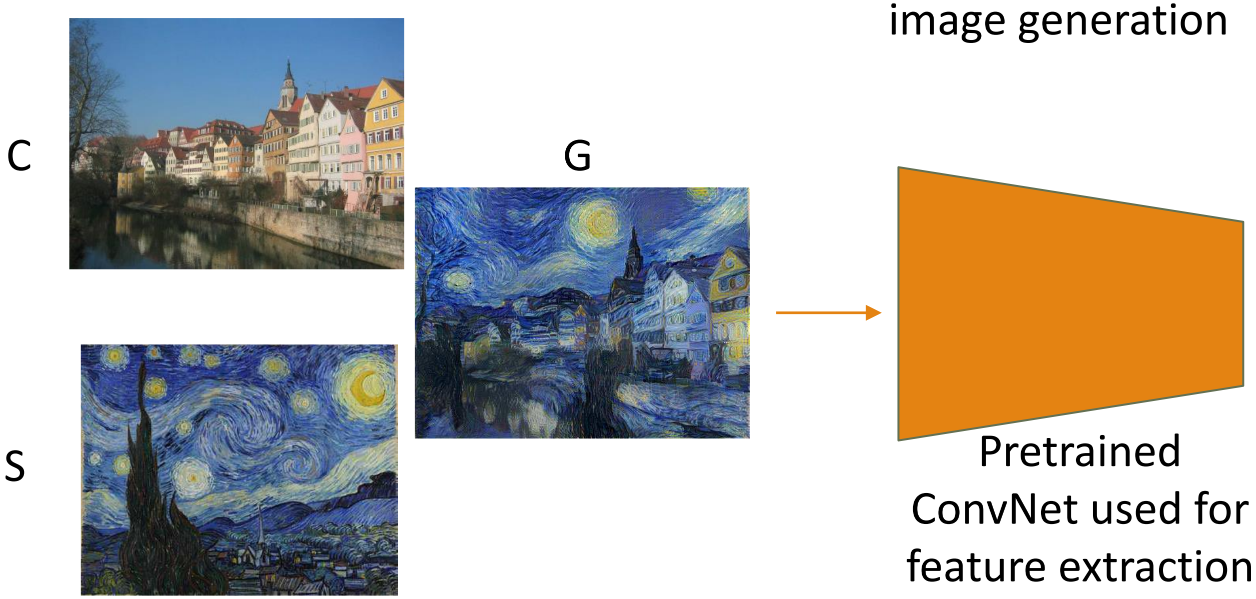


Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*.

Style transfer



ConvNet for
image generation



Pretrained
ConvNet used for
feature extraction

Loss function:

- $L(G) = L_S(G) + L_C(G)$

L_S - the loss with respect to the style, measured as correlation of the features between S and G

L_G - the loss with respect to the content, measured as Euclidean distance at specific layers between C and G

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

References

- Chapter 9 in **Goodfellow Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.**

- Krizhevsky Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).
- Simonyan Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
- He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- Howard Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).