Deep Learning

Abdelhak Mahmoudi
abdelhak.mahmoudi@um5.ac.ma

INPT- 2020

Content

1. Deep Artificial Neural Networks
2. **Convolutional Neural Networks**
3. Sequence Models
4. Generative Models

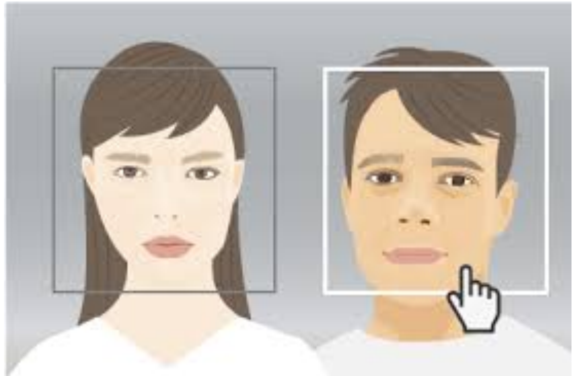
Convolutional Neural Networks

- Motivation
- General Architecture
- Existing Architectures
- Transfer Learning

Computer Vision Problems

Image Classification

→ Cat



Face detection

Object detection

Neural Style Transfer

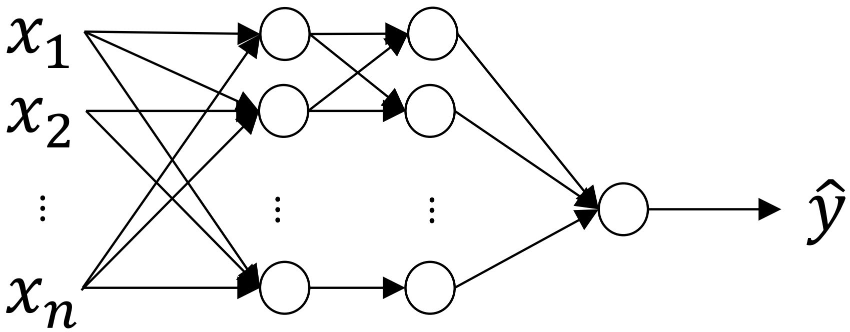
Deep Learning on Large Images

64x64x3

Cat? (0/1)

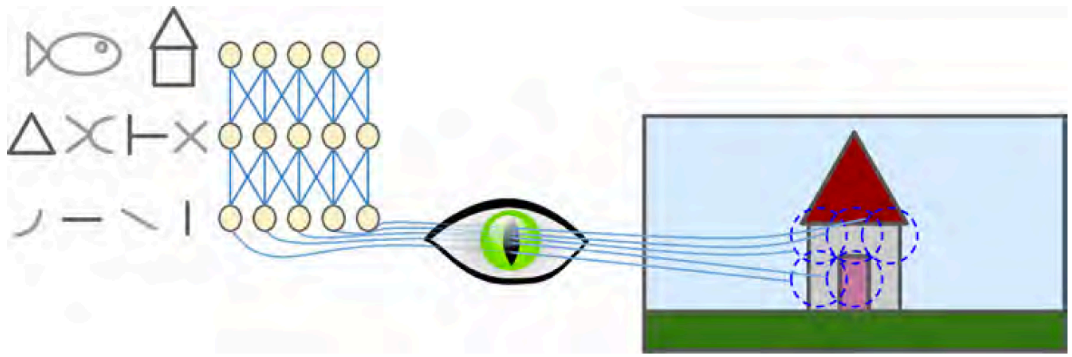
n = 1000x1000x3

$\mathbb{R}^n$ *Huge Dimension* !!

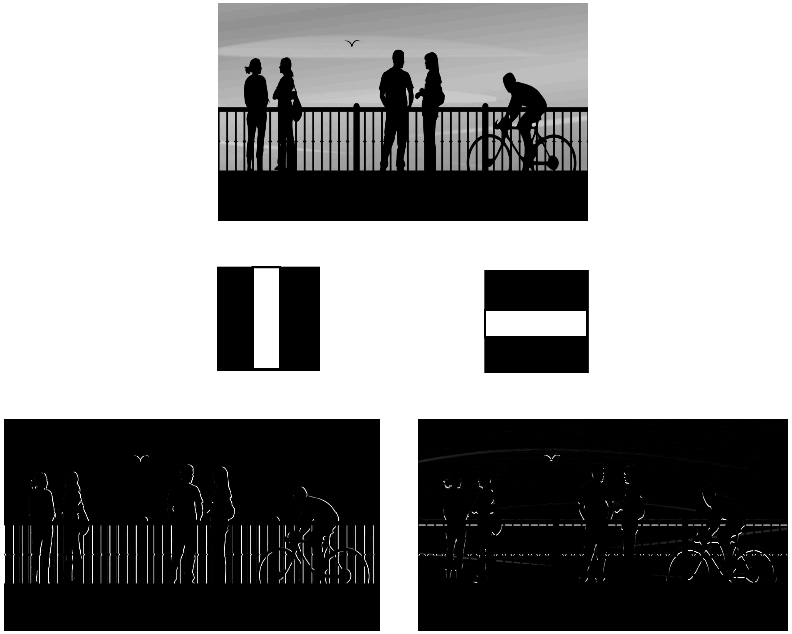


Visual Cortex

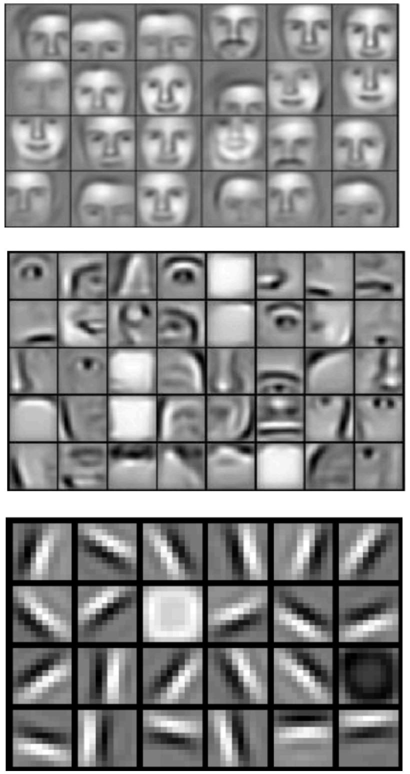
- **1958**, “Single Unit Activity in Striate Cortex of Unrestrained Cats”, D. Hubel and T. Wiesel.
- **1980**, “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” K. Fukushima.
- **1998**, Gradient-Based Learning Applied to Document Recognition,” Y. LeCun, L. Bottou, Y. Bengio, P. Haffner

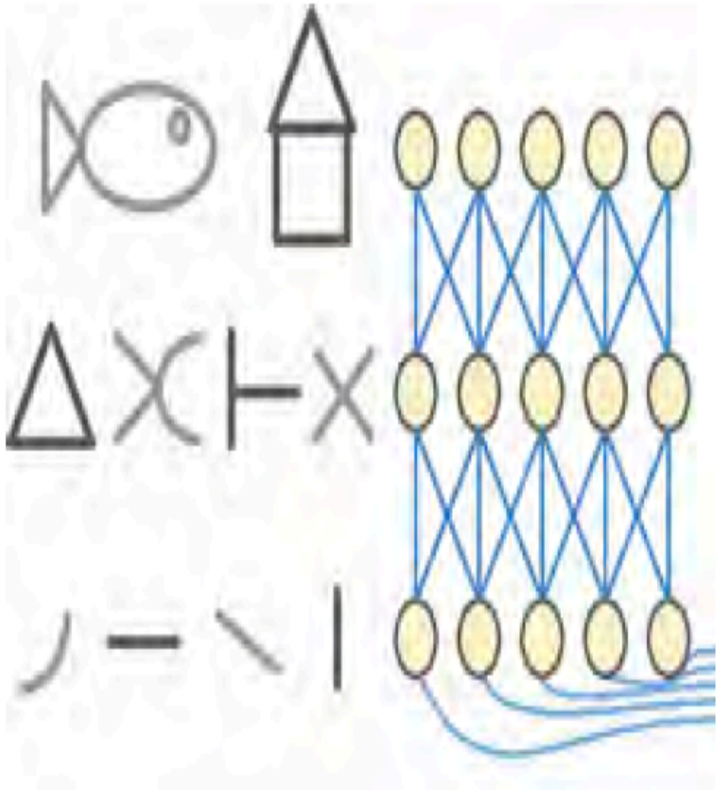


Learning Feature Maps

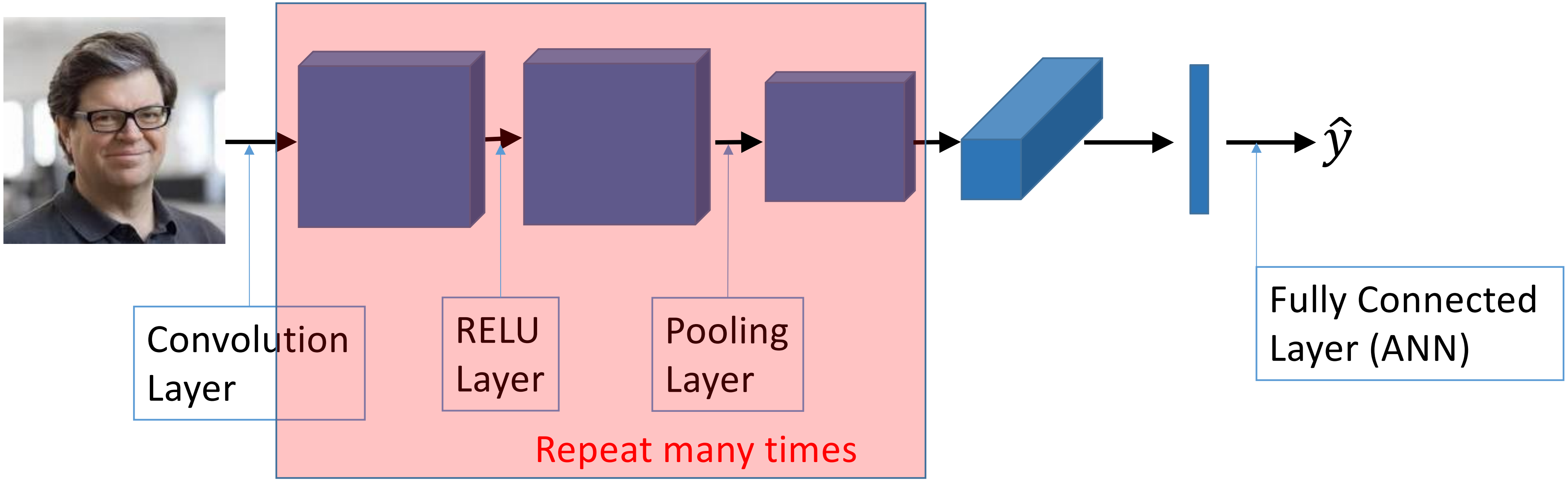


vertical edges horizontal edges





CNN: General Architecture



Convolution

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

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

Convolution

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

Vertical edge detection

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

Horizontal edge detection

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

*

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

=

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

Convolution

Vertical

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

Sobel

1	0	-1
2	0	-2
1	0	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

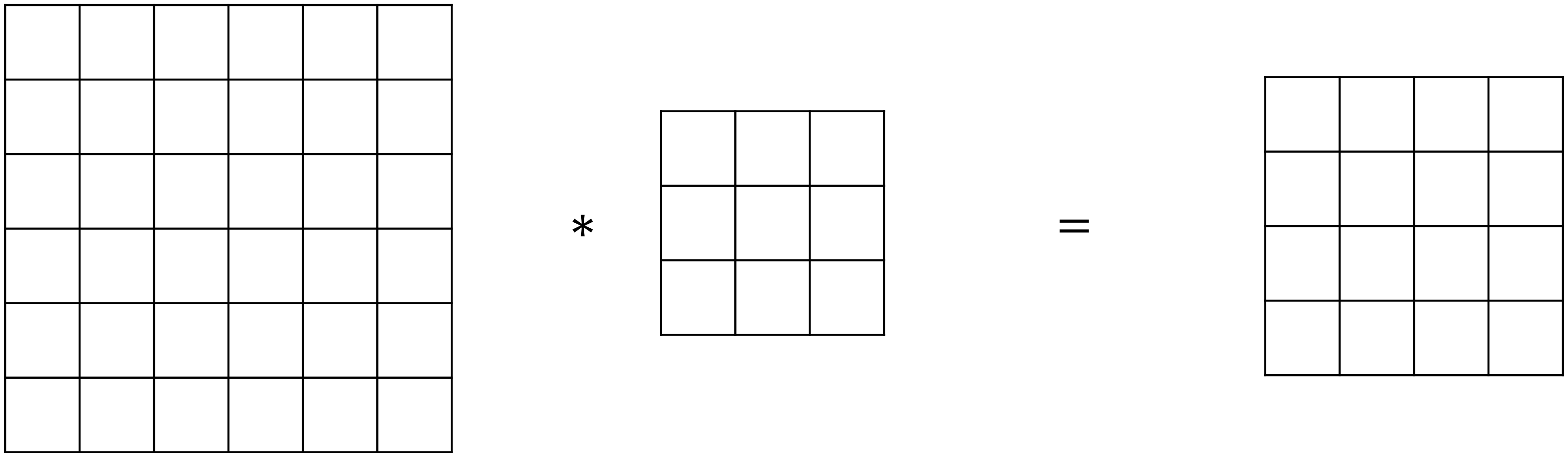
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Conv Net idea:

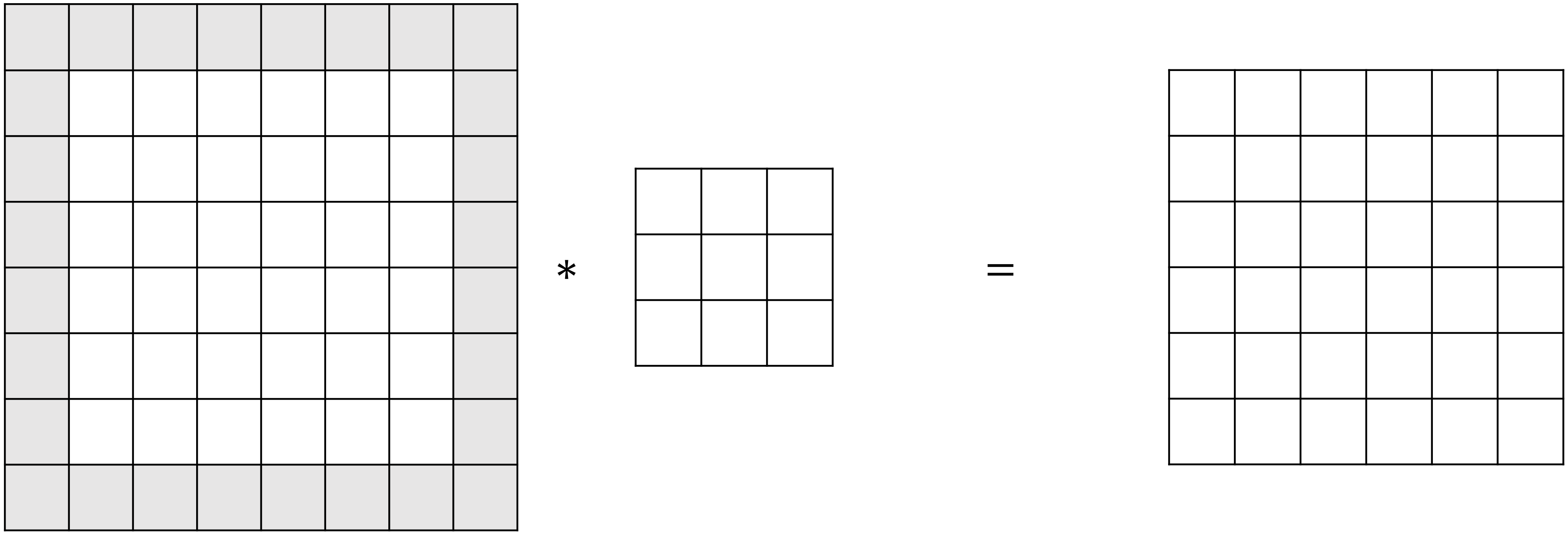
Learning to Detect Edges

Convolution: No Padding (Valid)



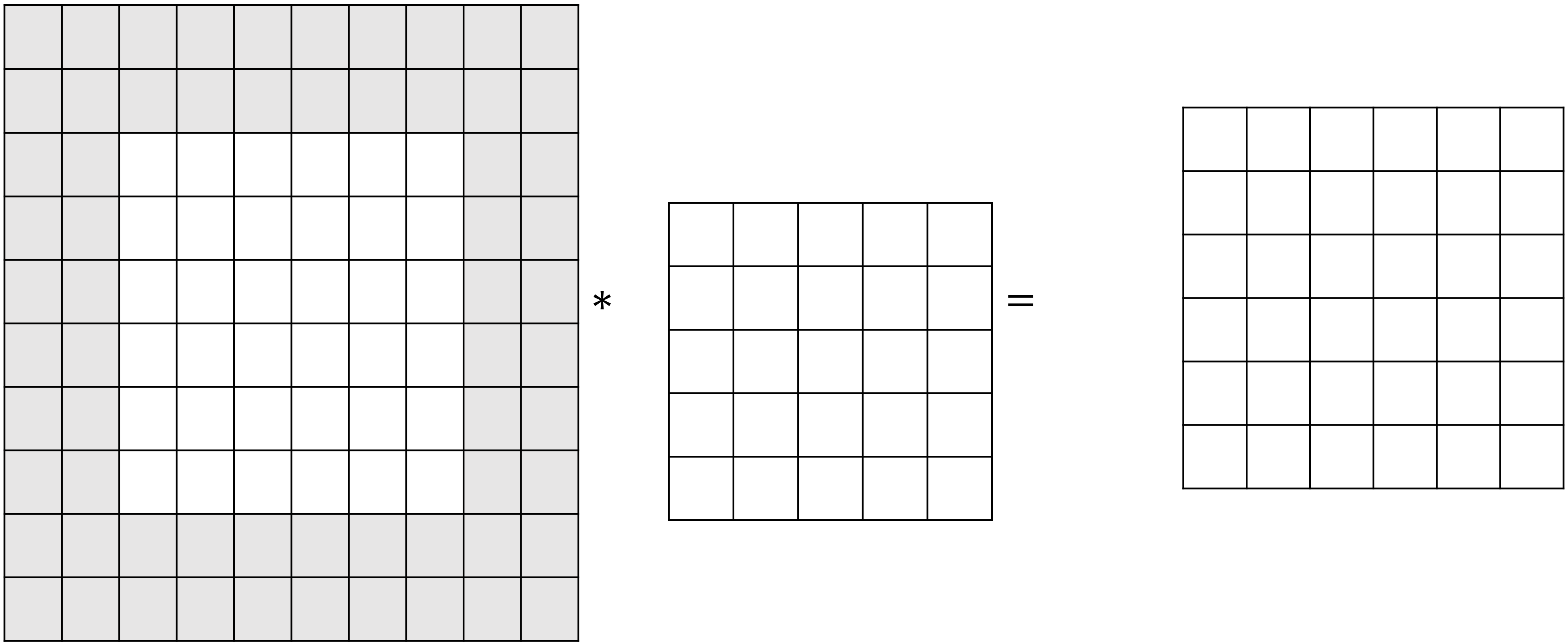
Valid 6x6 Filtre 3x3 Result 4x4

Convolution: Padding (Same)



Same 6x6 , p=1 => 8x8 Filtre 3x3 Result 6x6

Convolution: Padding (Same)



Same 6x6 , p=2 => 10x10

Filtre 5x5

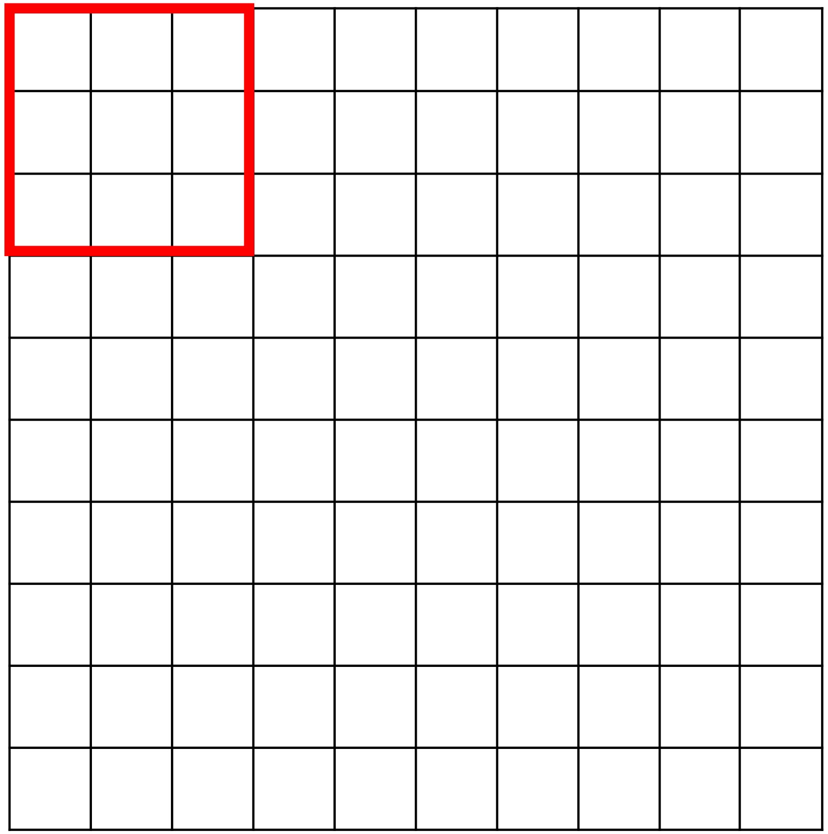
Result 6x6

Convolution: Padding, General Rule

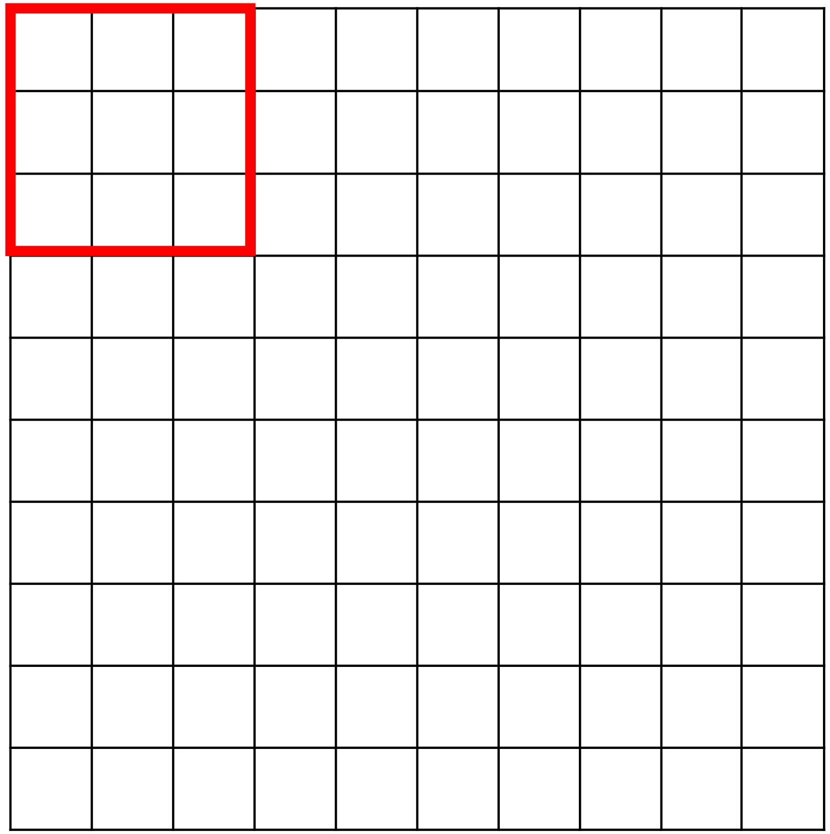
- Filter size f,
- with no padding (p =0:“Valid”):
  - $n \times n \rightarrow (n - f + 1) \times (n - f + 1)$
- with padding p
  - $n \times n \rightarrow (n + 2p - f + 1) \times (n + 2p - f + 1)$
- If we want output to be the “Same” as input:
  - $p = \frac{f-1}{2}$,
- f is alwayse odd

Convolution: Stride

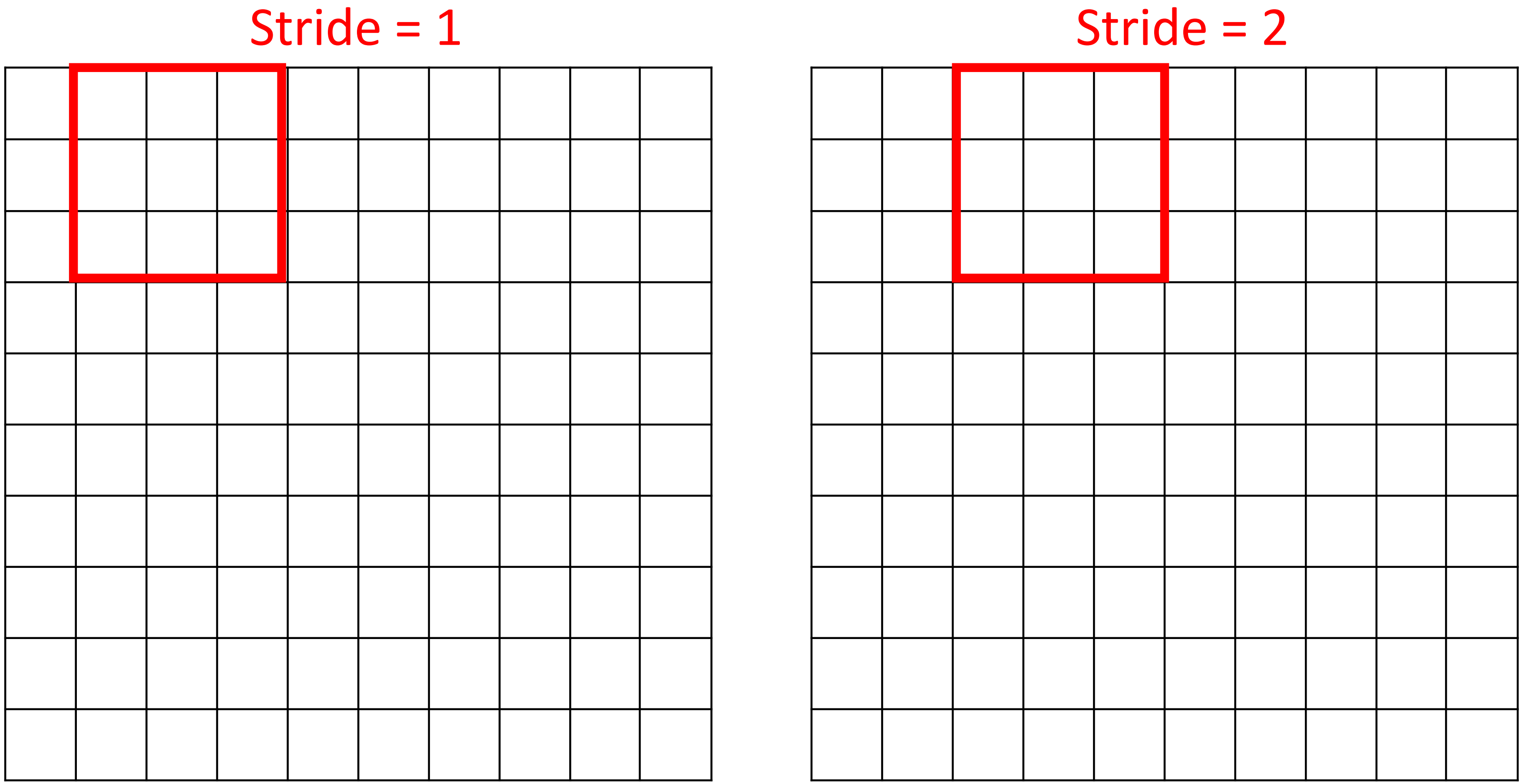
Stride = 1



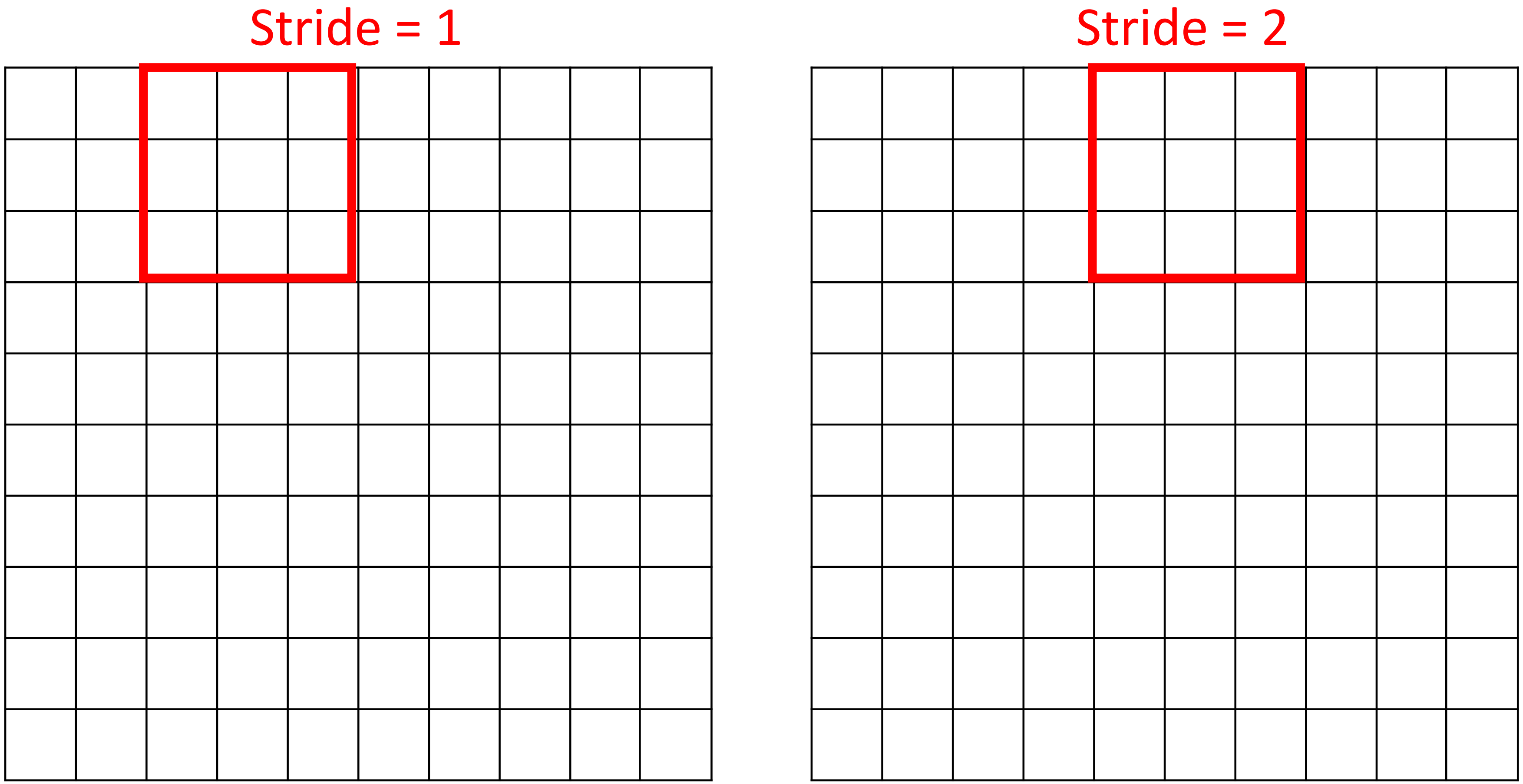
Stride = 2



Convolution: Stride



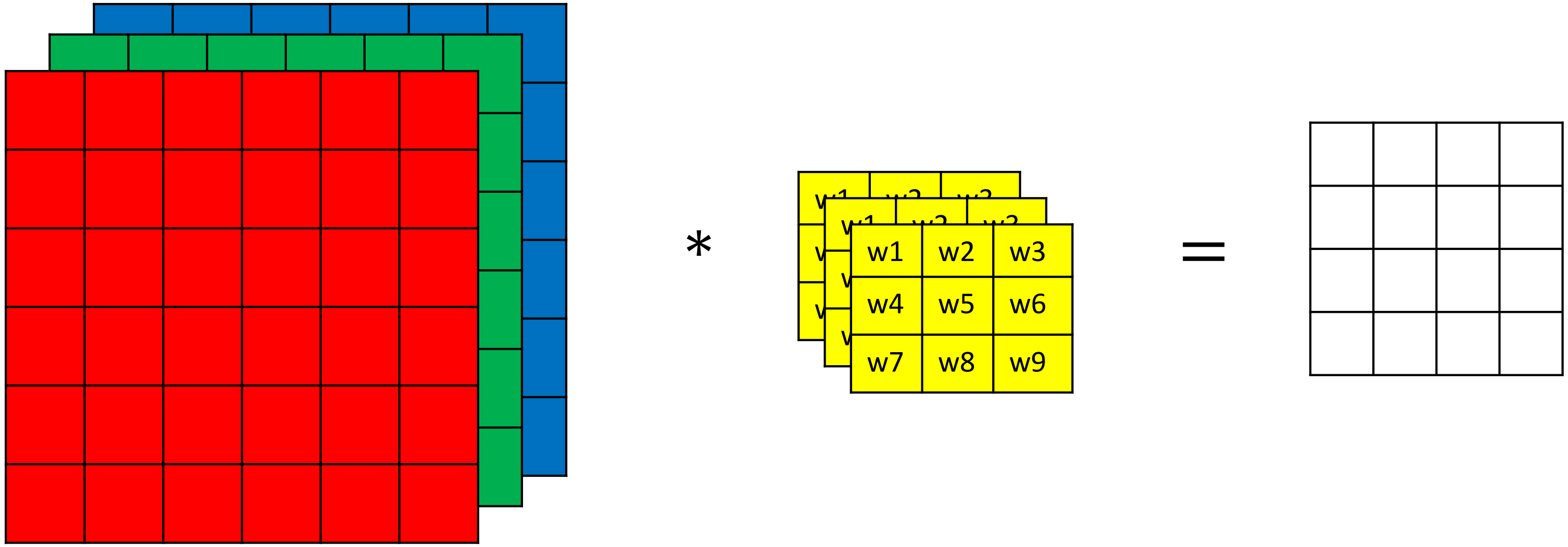
Convolution: Stride



Convolution: Padding and Stride, General Rule

- Image $n \times n$, Filter f x f, padding p, stride s
  - $n \times n \rightarrow \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$
- Example: f =3, p =2, s =2
  - 6x6 → 4x4

Convolution: Volumes



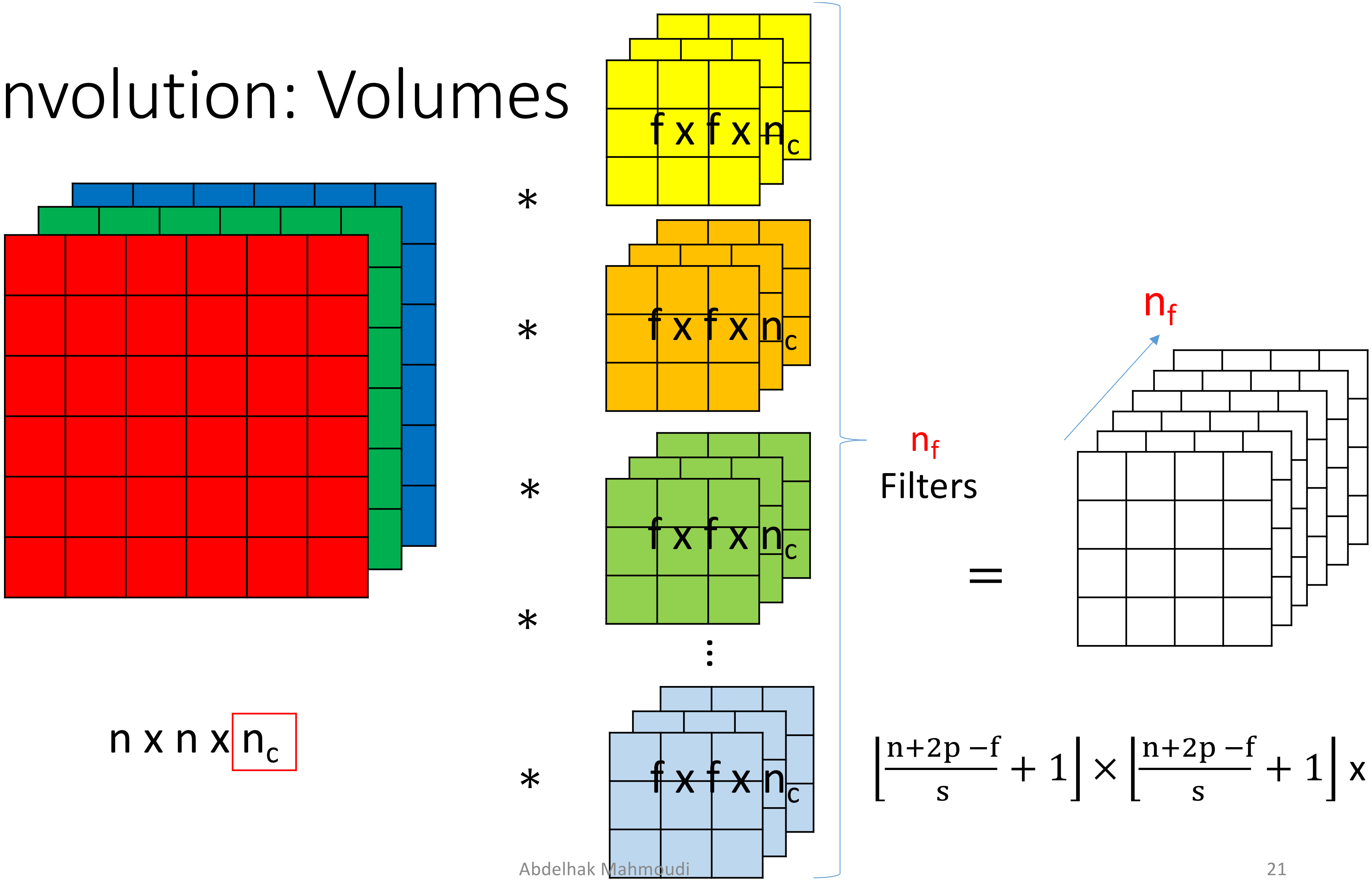
p=0, s=1   6 x 6 x 3   3 x 3 x 3   4 x 4

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

Depth
(# of channels)

Depth must be the same as
The number of channels)

Convolution: Volumes

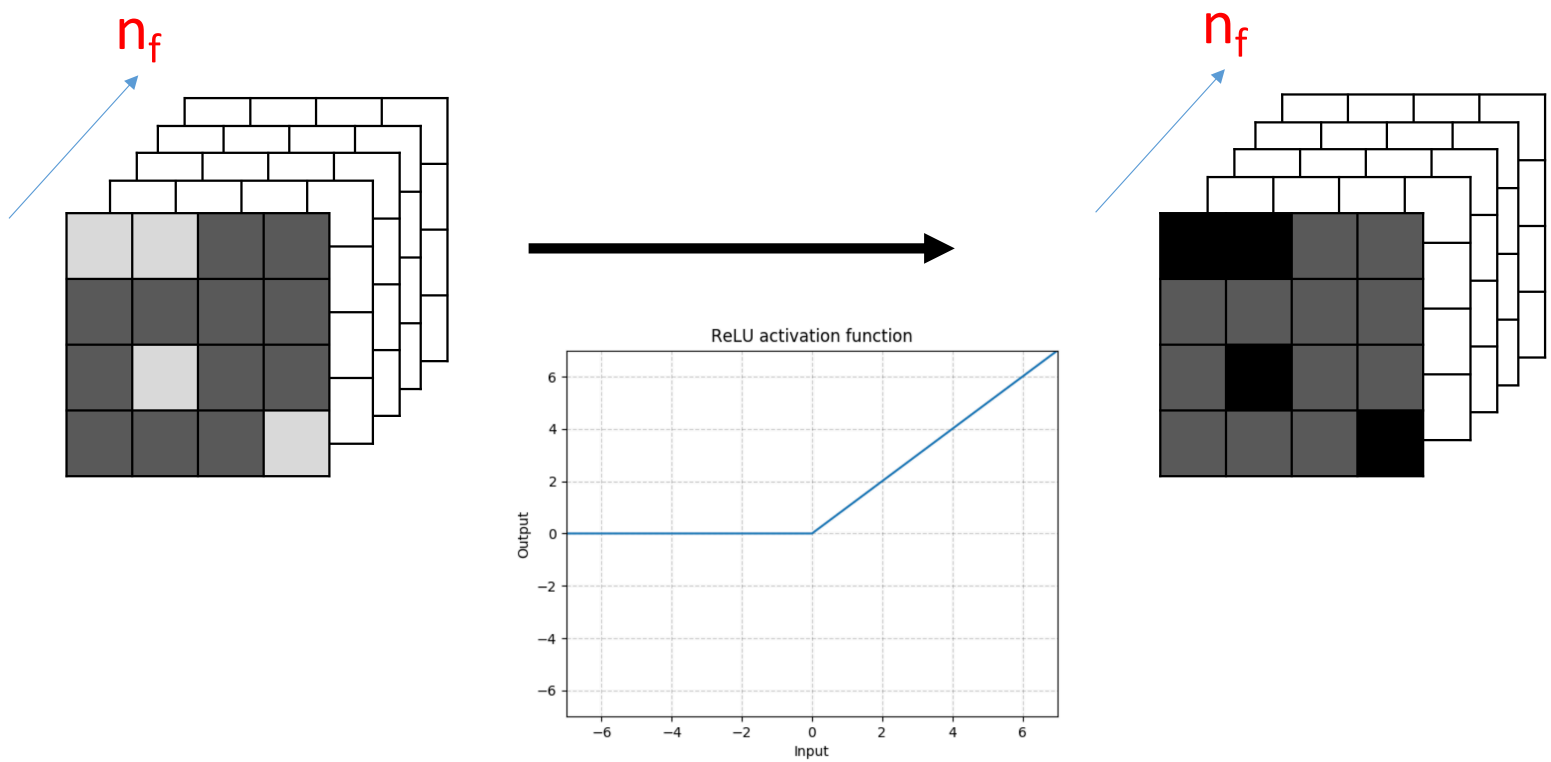


p, s

n x n x $\boxed{n_c}$

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

Non Linearity (RELU)



Pooling (Max)

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

Keep the most relevant values

9	2
6	3

$$\left\lfloor \frac{n - fpool}{s_{pool}} + 1 \right\rfloor \times \left\lfloor \frac{n - f_{pool}}{s_{pool}} + 1 \right.$$

f_{pool} = Pooling filter Size
s_{pool} = Pooling stride

No weights to learn

Pooling (Average)

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

Keep the average over all values

3.75	1.25
3.75	2

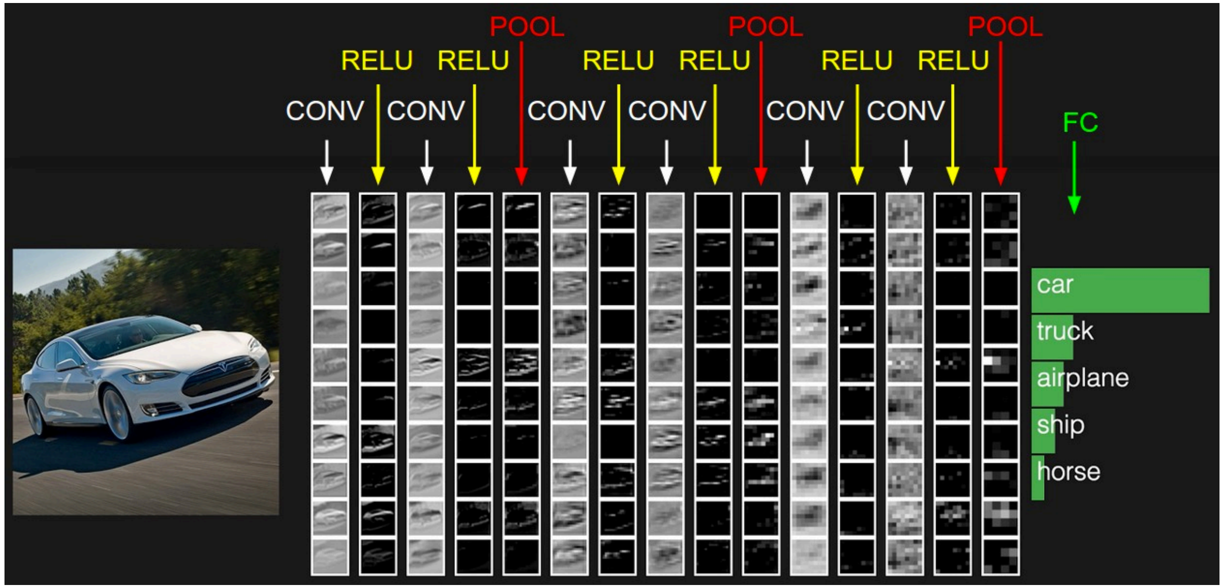
$$\left\lfloor \frac{n - fpool}{s_{pool}} + 1 \right\rfloor \times \left\lfloor \frac{n - f_{pool}}{s_{pool}} + 1 \right.$$

f_{pool} = Pooling filter Size
s_{pool} = Pooling stride

No weights to learn

ReLU-Pooling Order

- Pooling can be applied after successive conv-ReLU.
- With max-pooling, the order doesn't matter,
- Preferably use max-pooling and then ReLU for speed.

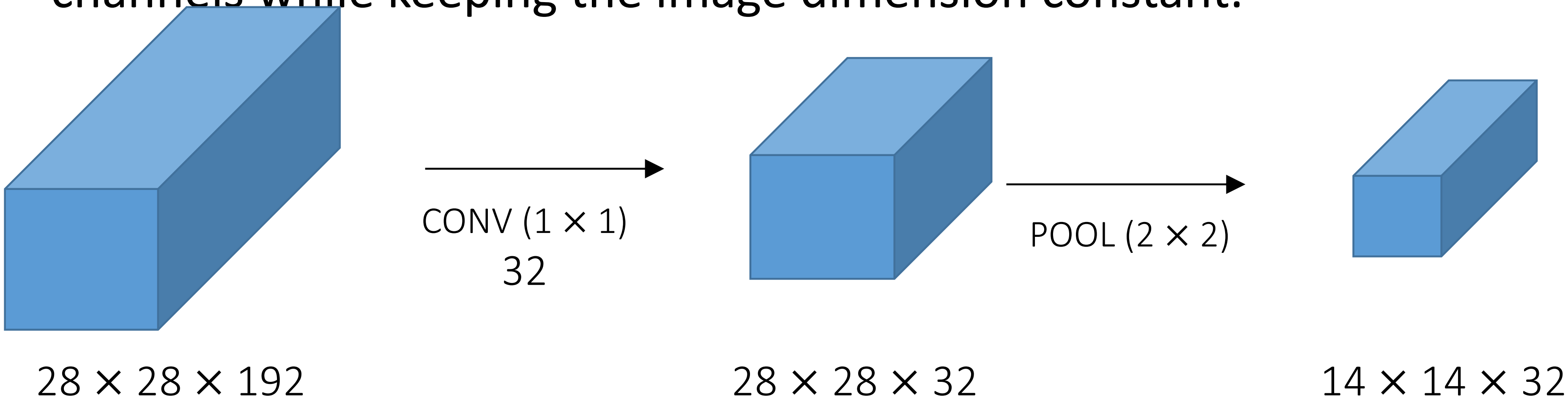


Memory and Parameter Sharing

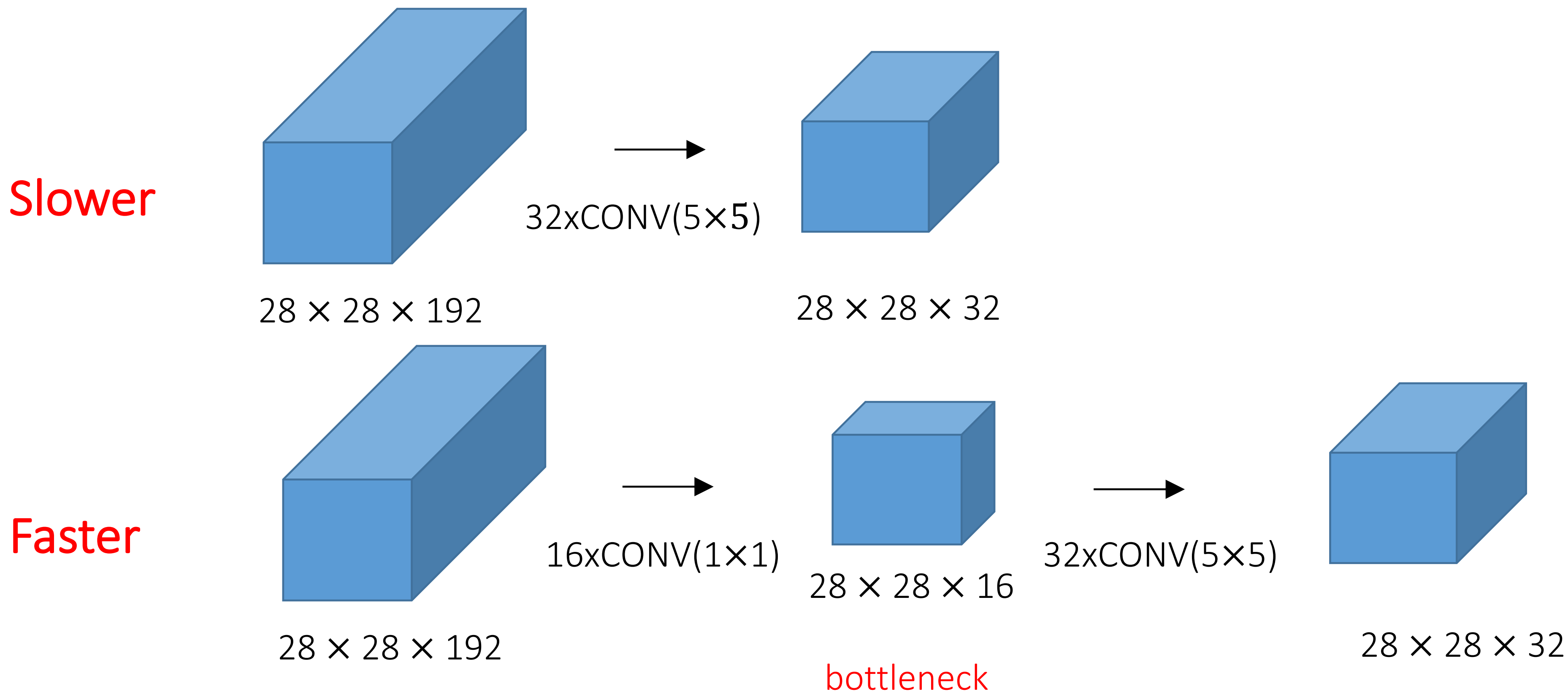
- Convolution
  - **Parameter sharing:** A feature detector (filter) that's useful in one part of the image is probably useful in another part of the image.
  - **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.
- Pooling
  - Reduce the computational load,
  - Reduce the memory usage,
  - Limit the risk of overfitting,
  - Tolerate a little bit of image shift (location invariance).

Control Dimensions

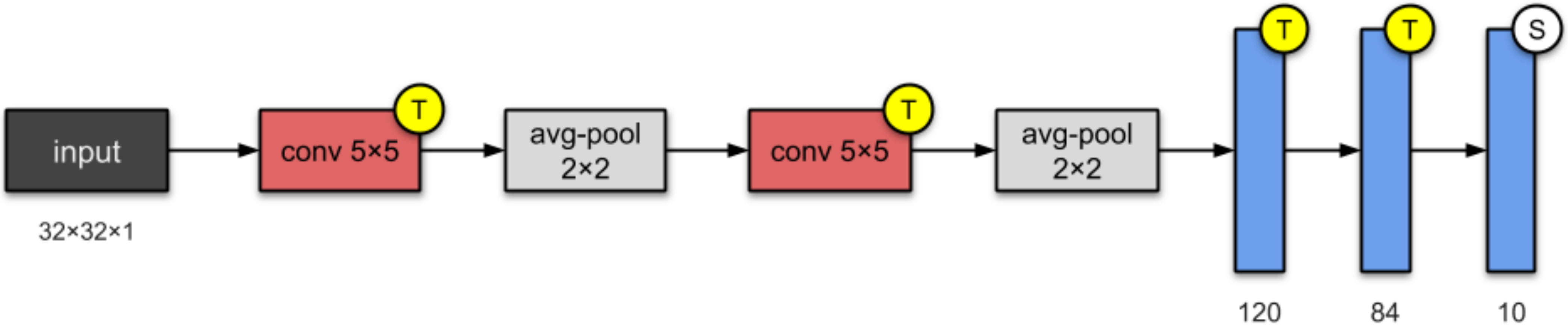
- With the pooling layer or a convolution with stride>1, we can reduce the dimension
- With a 1x1 convolution we can increase or decrease the number of channels while keeping the image dimension constant.



Computation cost: CONV(1x1)

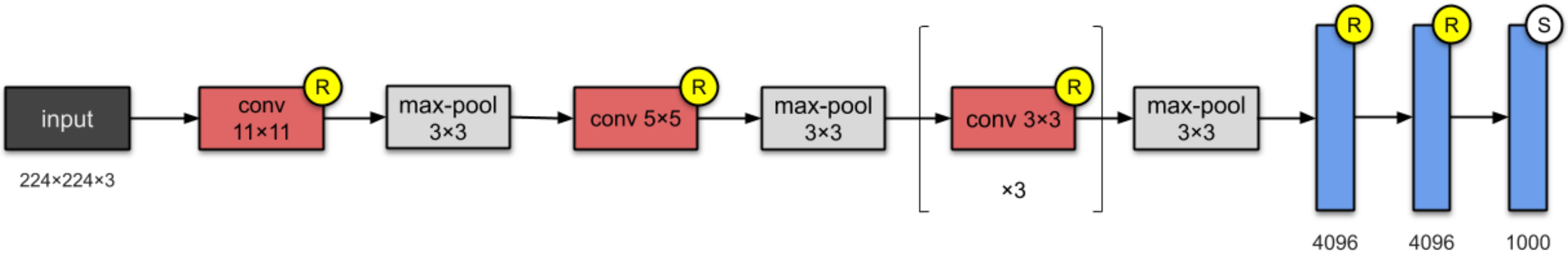


LeNet-5 (60K parameters, 1998)



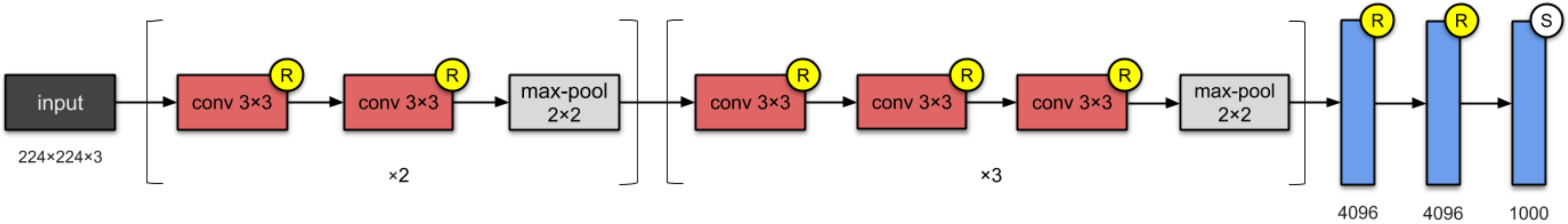
[LeCun et al., 1998. Gradient-based learning applied to document recognition]

AlexNet (60M parameters, 2012)



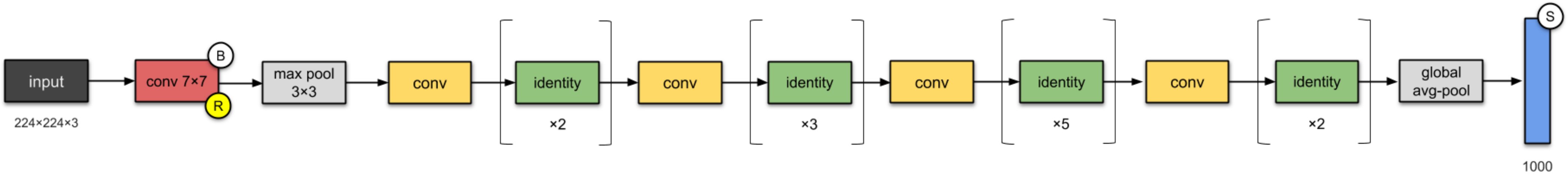
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

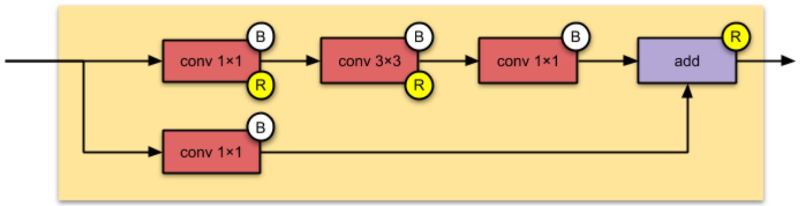
VGG-16 (138M parameters, 2015)



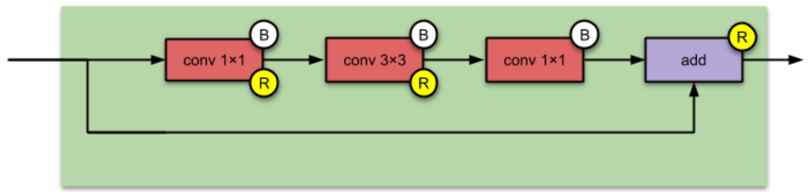
[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

ResNets50 (25.5M parameters, 2015)



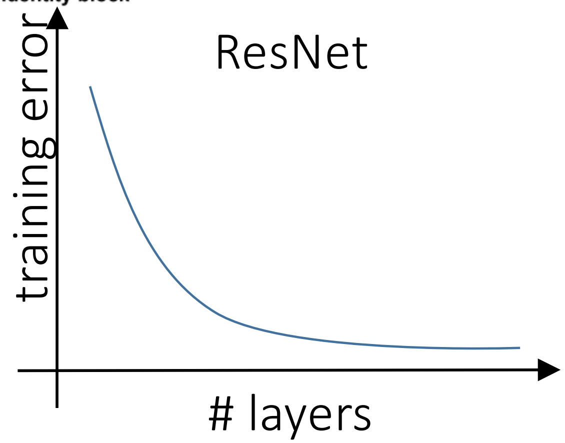


Conv block



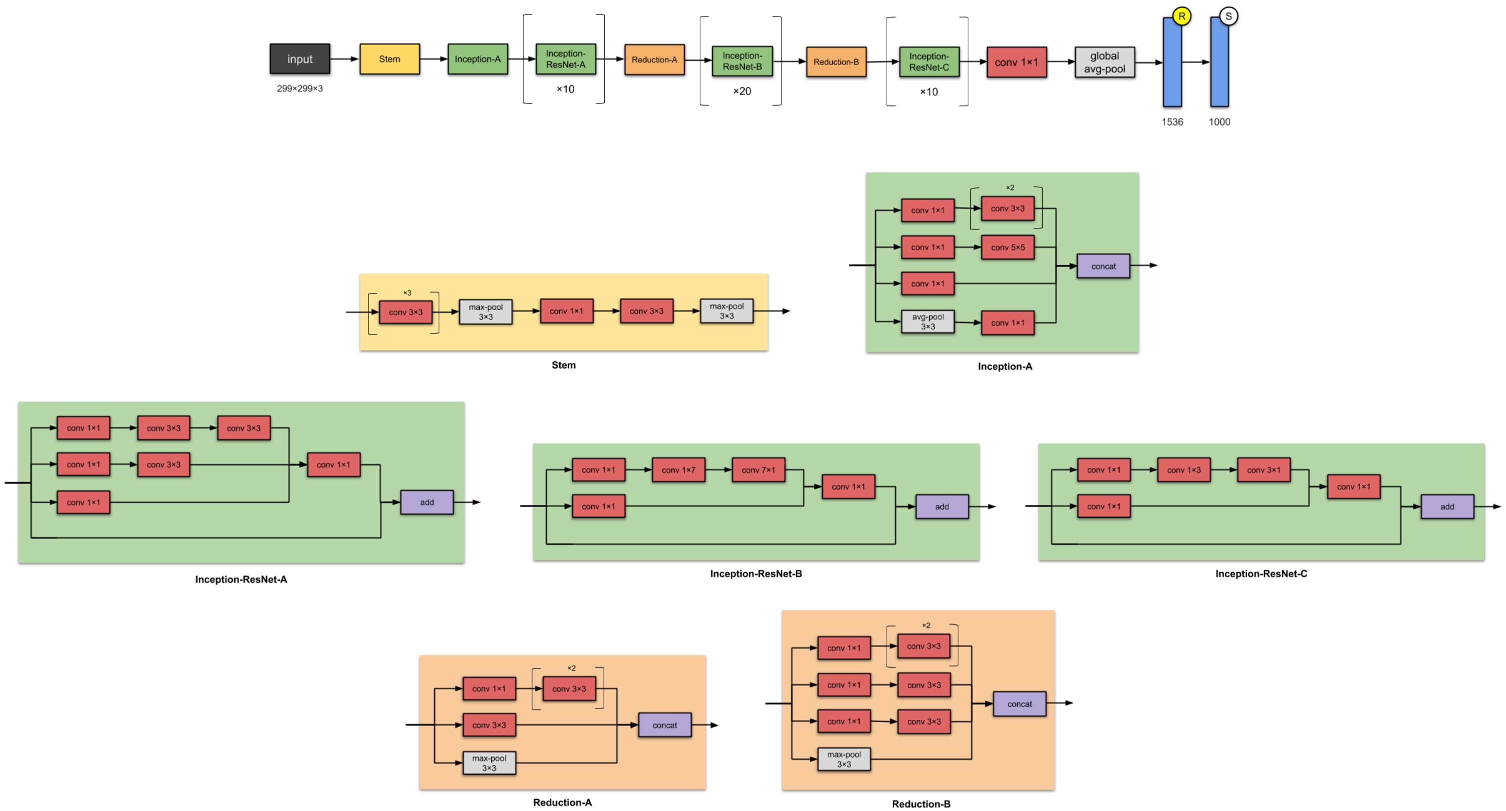
Identity block

- Skip Connections
- Train much deeper Nets

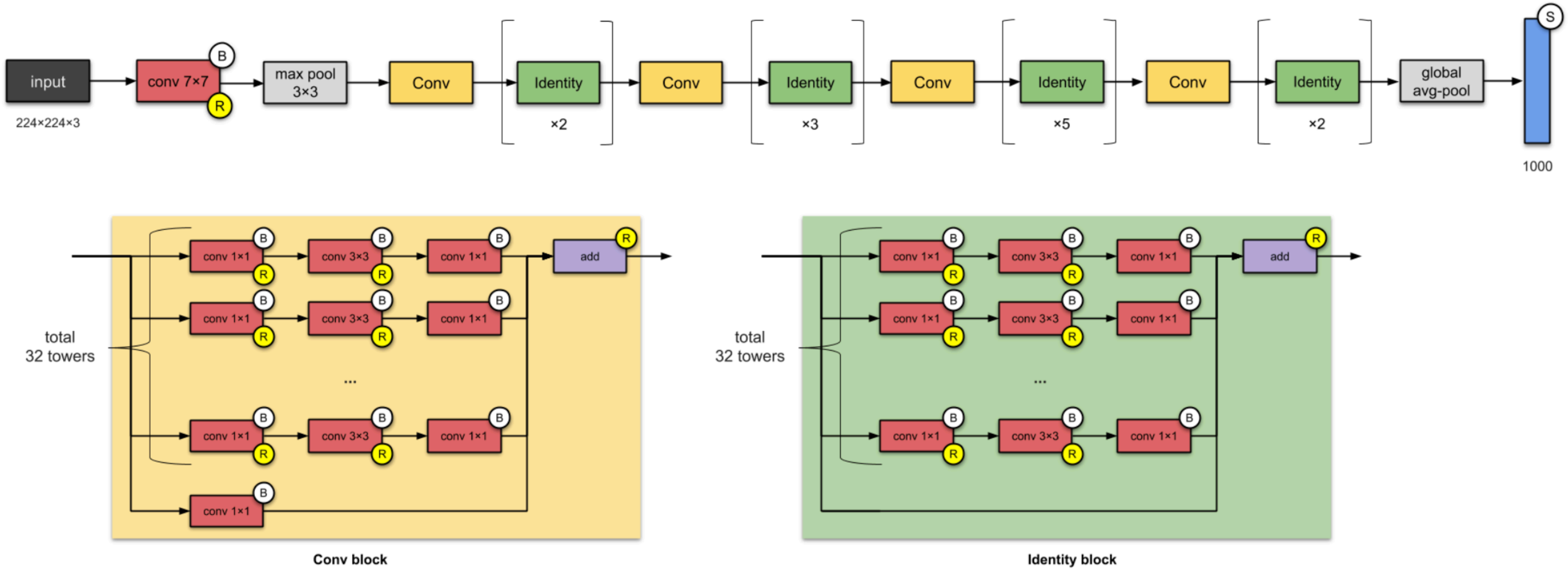


[He et al., 2015. Deep residual networks for image recognition]

Inception-ResNet-V2 (56M parameters, 2016)

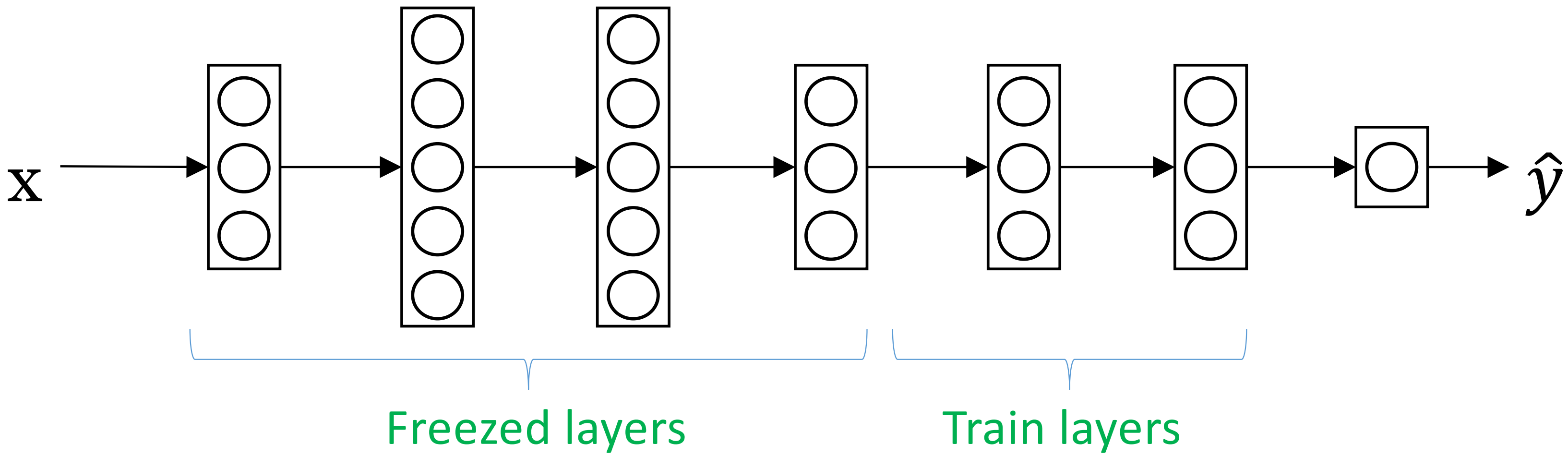


ResNeXt50(25M parameters, 2017)



Transfer learning

- Very few data: Rather than training the network from scratch, you can use weights already learned from another task.
- Not enough data: Download the architecture and the weights as well (Github)
- Enough data, use the transferred weights as initialization weights.



When to use Transfer Learning?

- Task A and B have the same input x
- You have a lot more data for Task A than Task B.
- Low level features from A could be helpful for learning B