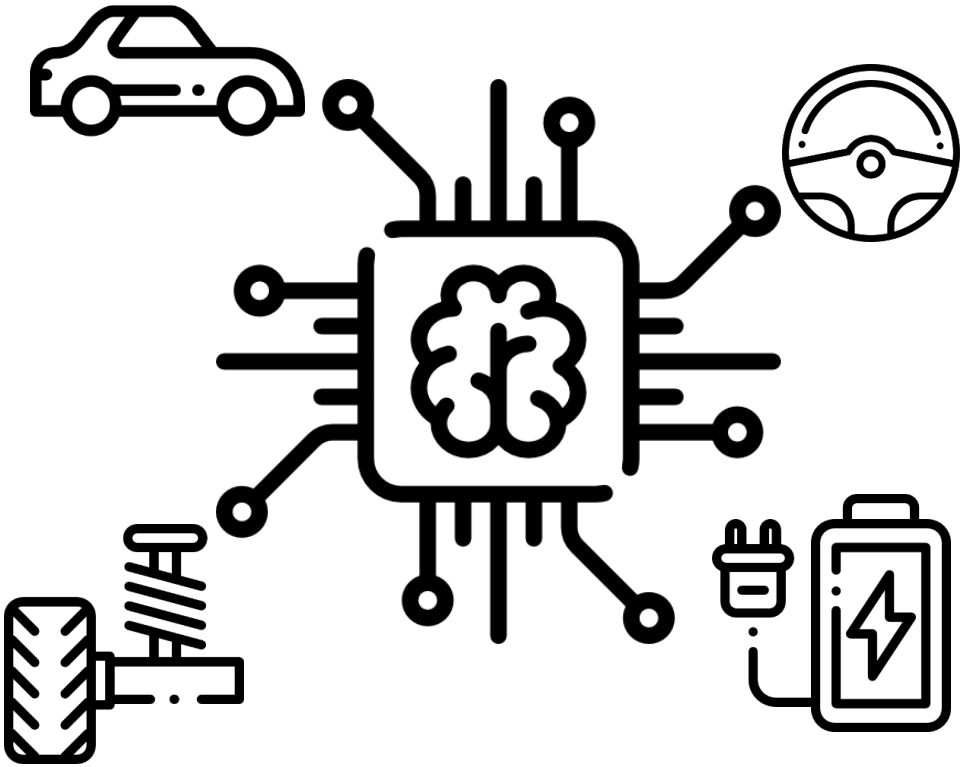
Institute of Automotive Technology
TUM School of Engineering and Design
Technical University of Munich

Artificial Intelligence in Automotive Technology

Maximilian Geißlinger / Fabian Netzler

Prof. Dr.-Ing. Markus Lienkamp

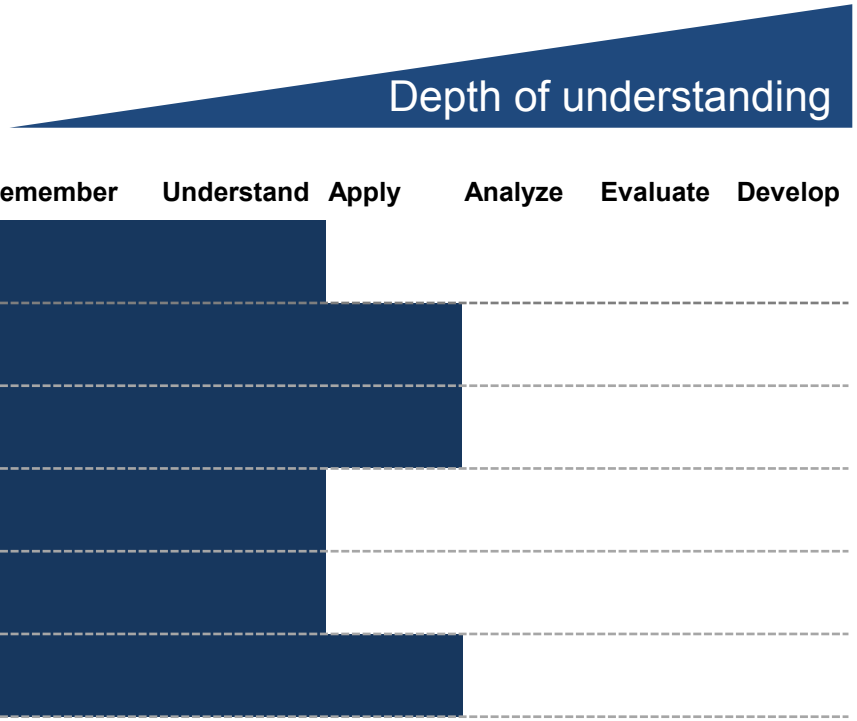


Lecture Overview

Lecture 16:15-17:45 | Practice 17:45-18:30	
1 Introduction: Artificial Intelligence	20.10.2022 – Maximilian Geißlinger
2 Perception	27.10.2022 – Sebastian Huber
3 Supervised Learning: Regression	03.11.2022 – Fabian Netzler
4 Supervised Learning: Classification	10.11.2022 – Andreas Schimpe
5 Unsupervised Learning: Clustering	17.11.2022 – Andreas Schimpe
6 Introduction: Artificial Neural Networks	24.11.2022 – Lennart Adenaw
7 Deep Neural Networks	08.12.2022 – Domagoj Majstorovic
8 Convolutional Neural Networks	15.12.2022 – Domagoj Majstorovic
9 Knowledge Graphs	12.01.2023 – Fabian Netzler
10 Recurrent Neural Networks	19.01.2023 – Matthias Rowold
11 Reinforcement Learning	26.01.2023 – Levent Ögretmen
12 AI-Development	02.02.2023 – Maximilian Geißlinger
13 Guest Lecture	09.02.2023 – to be announced

Objective for Lecture 8: CNNs

After the lecture you will be able to…



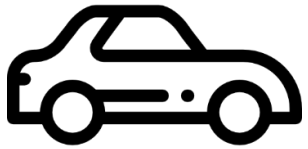
Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

Agenda

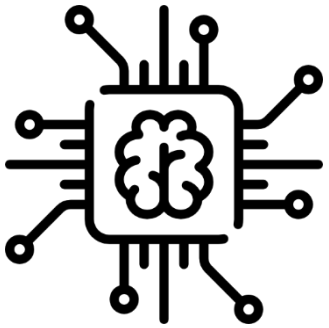
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride



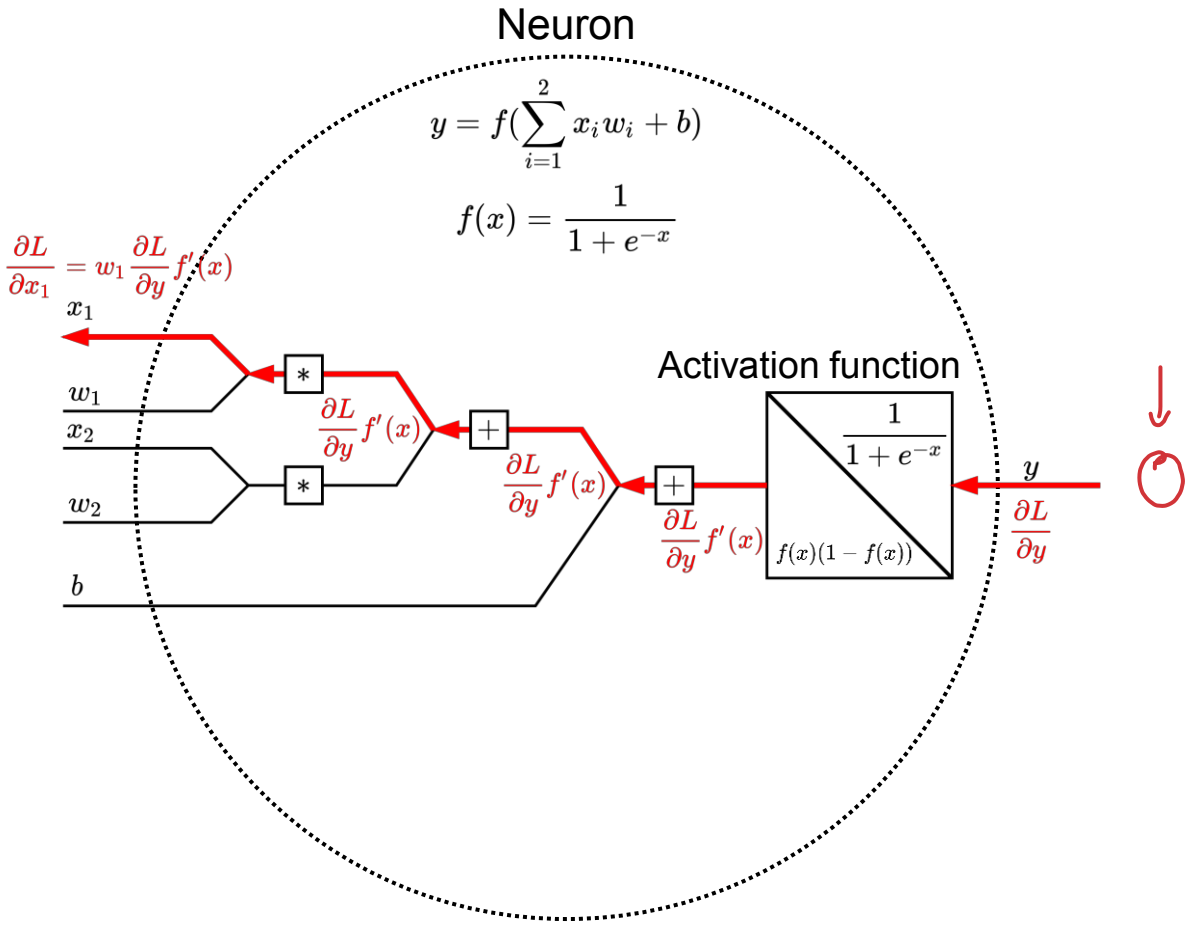
2. Chapter: Pooling

3. Chapter: CNN in Action



Computational Graph

Neuron

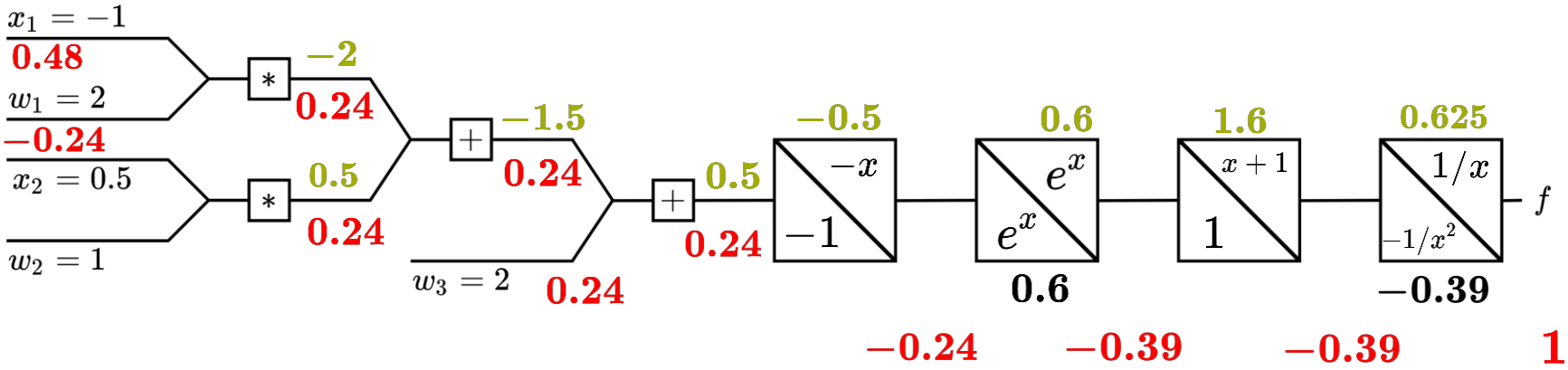


Computational Graph

Complex Example

$$f(x_1, x_2) = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2 + w_3)}}$$

Local gradient 

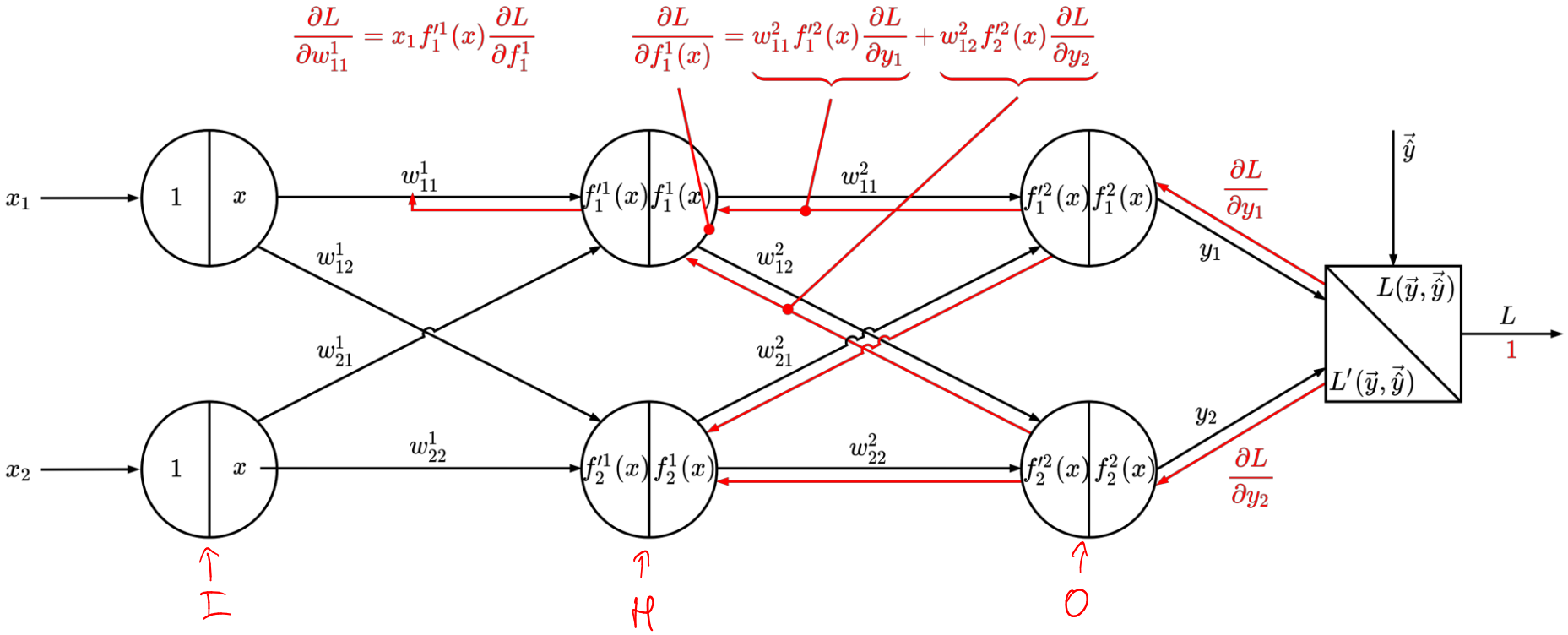


$$\frac{\partial f}{\partial x_1} = w_1 \frac{e^{-w_1 x_1 - w_2 x_2 - w_3}}{(e^{-w_1 x_1 - w_2 x_2 - w_3} + 1)^2} = 0.470074$$

$$\frac{\partial f}{\partial w_1} = x_1 \frac{e^{-w_1 x_1 - w_2 x_2 - w_3}}{(e^{-w_1 x_1 - w_2 x_2 - w_3} + 1)^2} = -0.235004$$

Backpropagation

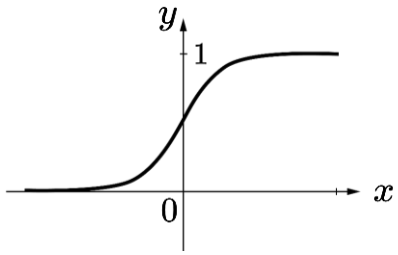
Neural Network, one hidden layer



$$\frac{\partial L}{\partial w_{11}^1} = x_1 f_1'^1(x) \frac{\partial L}{\partial f_1^1} \quad \frac{\partial L}{\partial w_{12}^1} = x_1 f_2'^1(x) \frac{\partial L}{\partial f_2^1} \quad \frac{\partial L}{\partial w_{21}^1} = x_2 f_1'^1(x) \frac{\partial L}{\partial f_1^1} \quad \frac{\partial L}{\partial w_{22}^1} = x_2 f_2'^1(x) \frac{\partial L}{\partial f_2^1}$$

Activation Functions

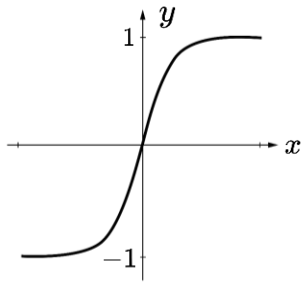
Sigmoid



$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = f(x)(1 - f(x))$$

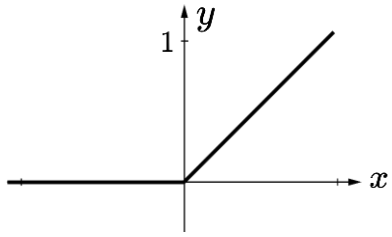
TanH



$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$f'(x) = \frac{4}{(e^{x} + e^{-x})^{2}}$$

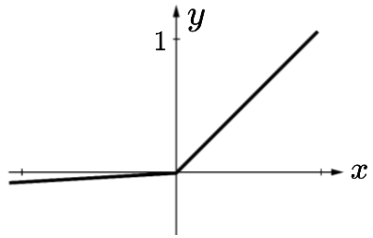
ReLU



$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

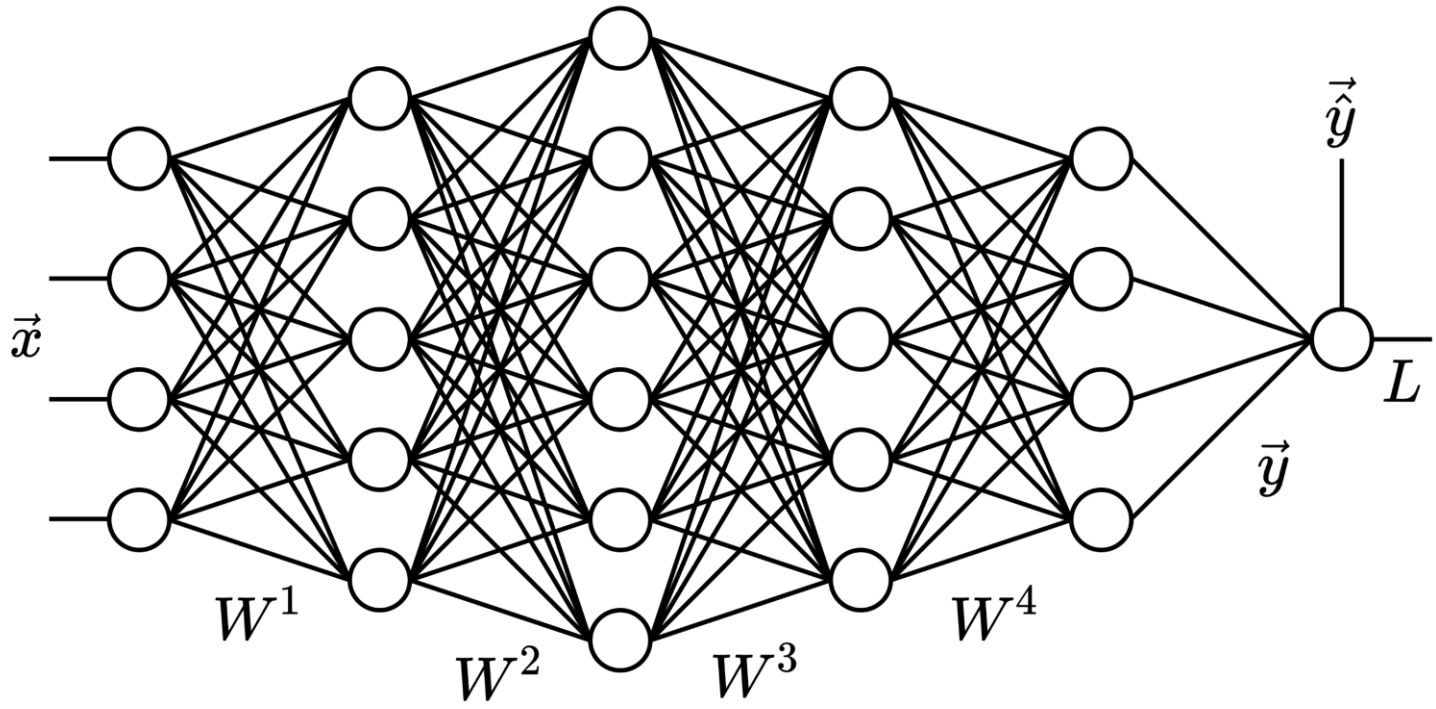
Leaky ReLU



$$f(x) = \begin{cases} x, x \geq 0 \\ ax, x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, x \geq 0 \\ a, x < 0 \end{cases}$$

Training Neural Networks



Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

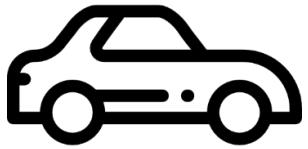
Agenda

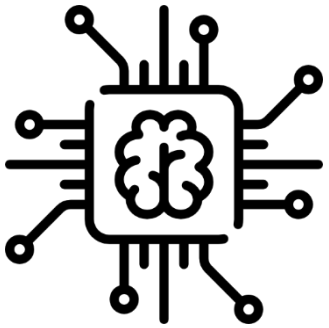
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride

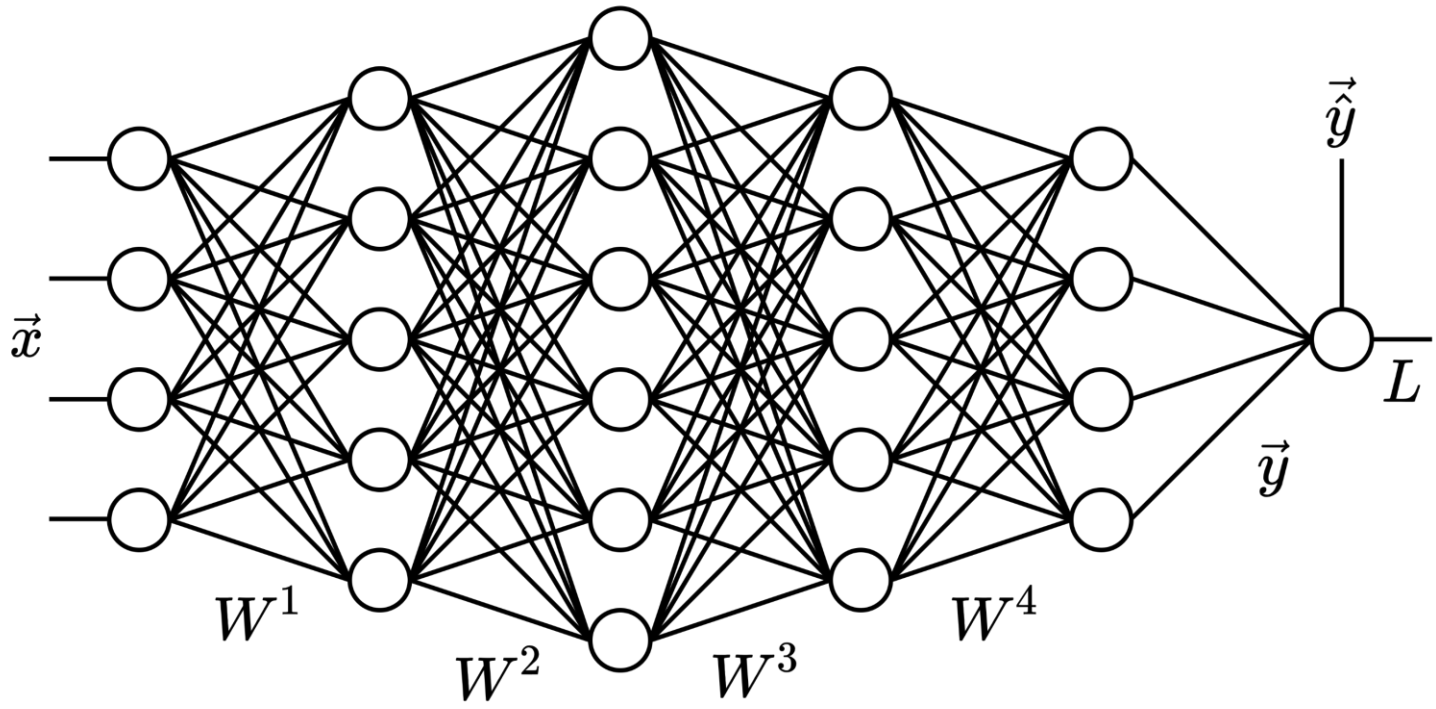
2. Chapter: Pooling

3. Chapter: CNN in Action



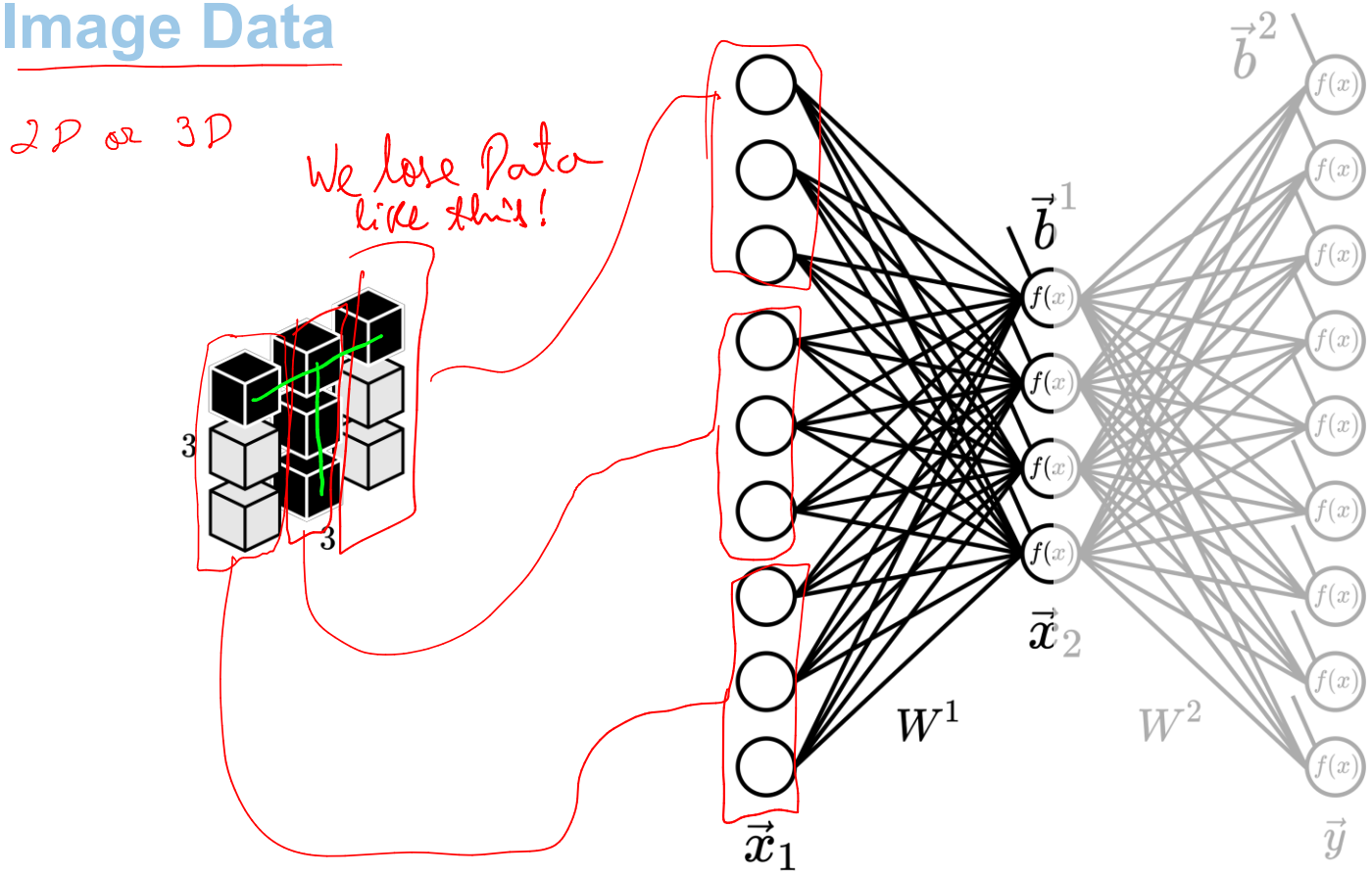


Fully Connected Neural Network and Image Data



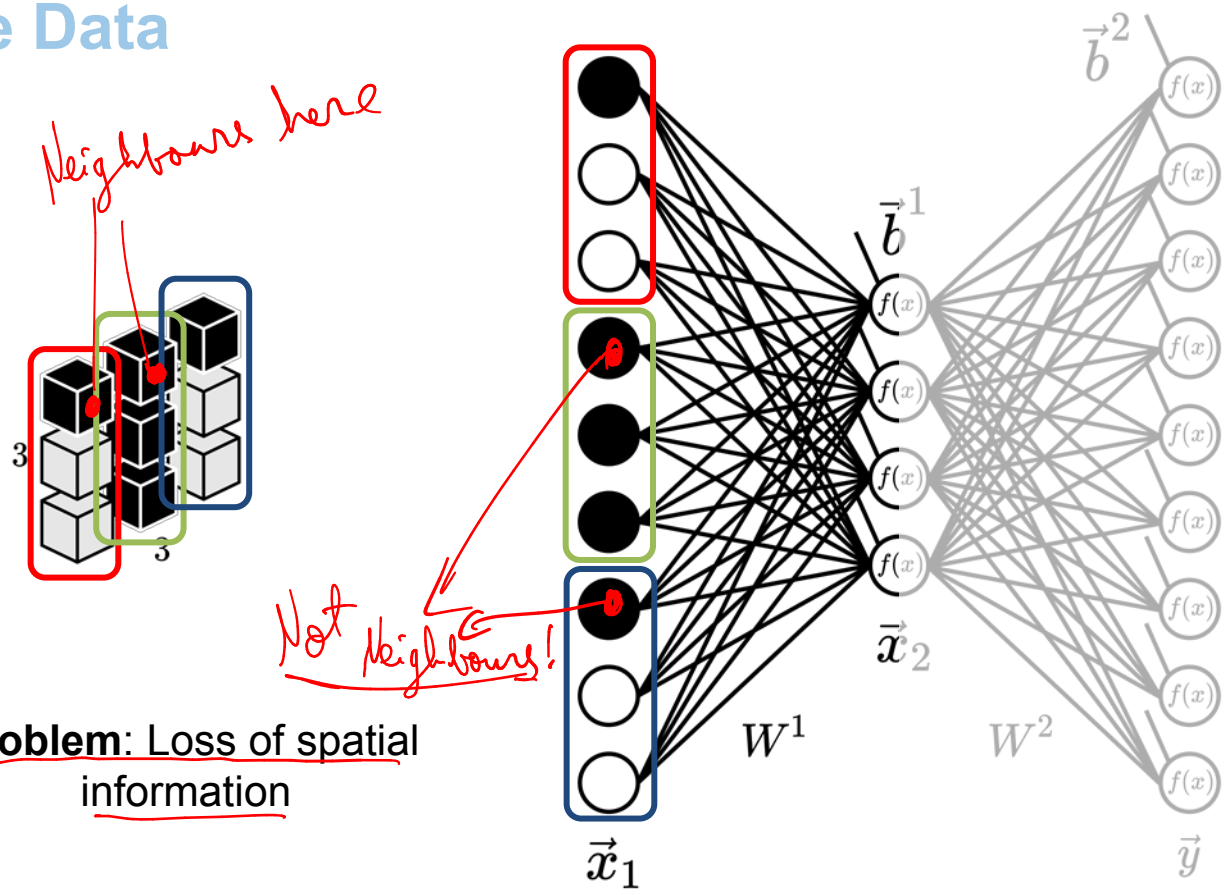
Fully Connected Layer

Image Data



Fully Connected Layer

Image Data

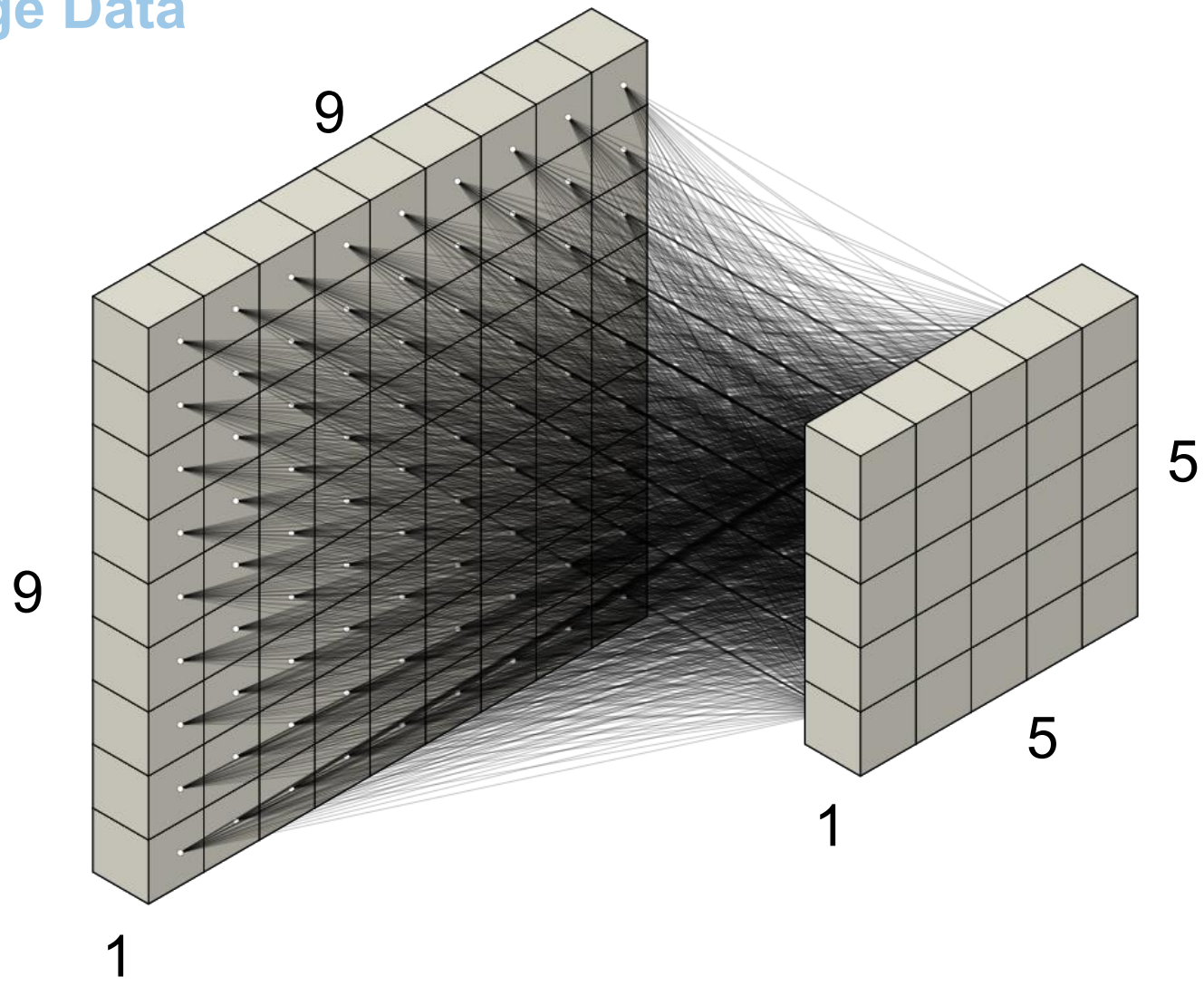


Problem: Loss of spatial information

In order to use images as an input layer to the fully connected network, 2D structure with the corresponding pixel values needs to be “flattened”, i.e., transformed into a column vector. This process causes some neighboring neurons from the original 2D structure not to be neighbors anymore. In other words, this side effect results in a loss of spatial information to some extent which makes the overall learning process much more difficult.

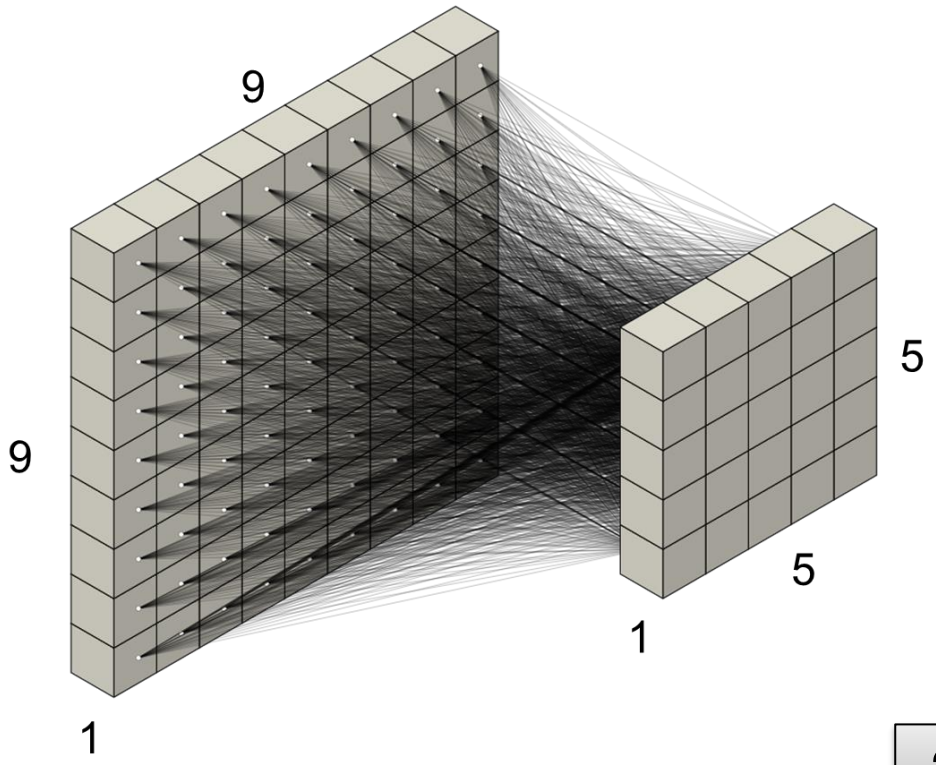
Fully Connected Layer

Image Data



Fully Connected Layer

Image Data



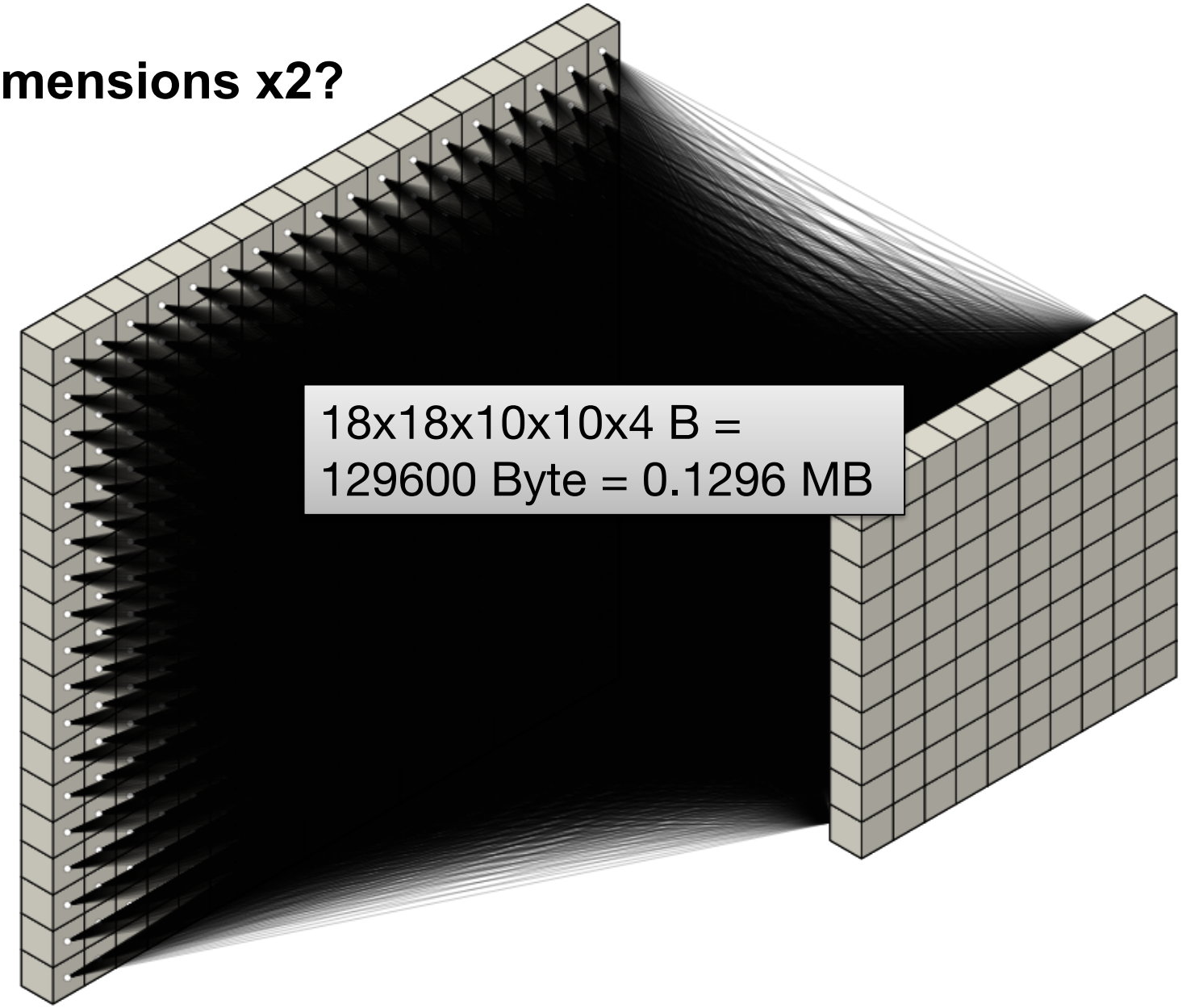
$$\vec{x} \in K^{1\times 81}$$
$$\vec{y} \in K^{1\times 25}$$
$$W \in K^{81\times 25}$$

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,25} \\ \vdots & \ddots & \vdots \\ w_{81,1} & \cdots & w_{81,25} \end{pmatrix}$$

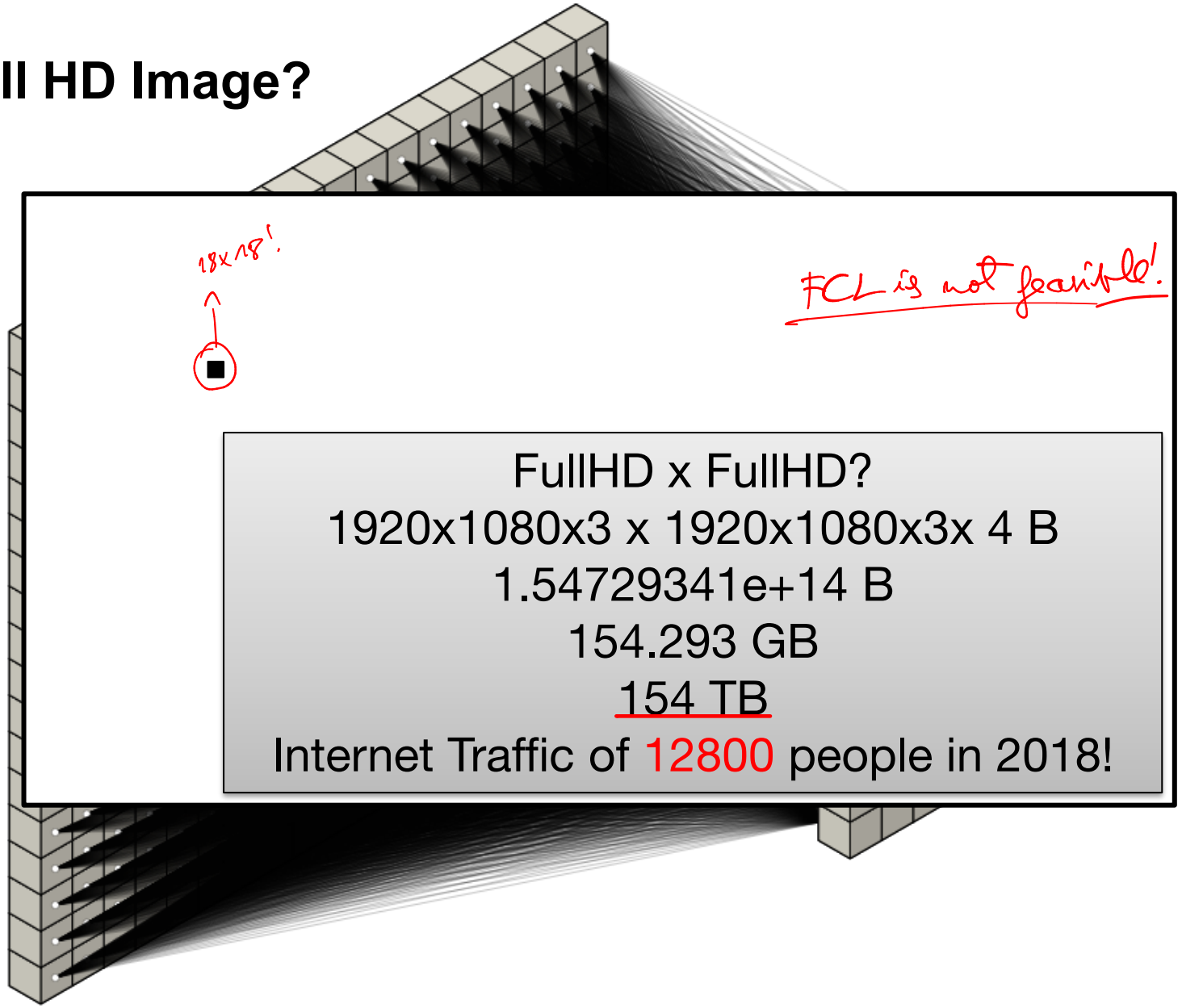
$$\Rightarrow 9 \cdot 9 \cdot 5 \cdot 5 = 2025 \text{ Parameter}$$

2025 x 4 B = 8100 Byte = 8 kB

Dimensions x2?



Full HD Image?



Images usually have a lot of pixels - we often refer to the image resolution in terms of megapixels which represent 10^6 pixels. Having images as input data results in having one neuron per each pixel, which means that the number of neurons will be huge. This also means that the memory required to store values for all these connections will also be huge, so using images on FCN is feasible only to some extent with a subpar learning performance.

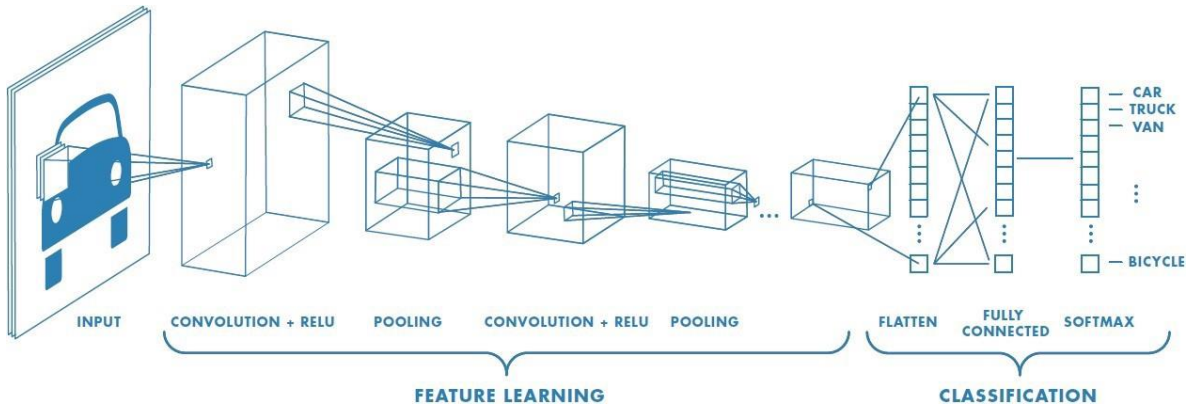
Fully Connected Layer

2 Main Issues

1. Loss of spatial information
2. Hungry for memory

Solution?

Convolutional Neural Networks



Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

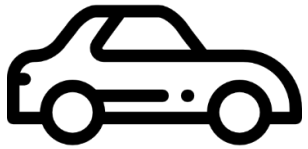
Agenda

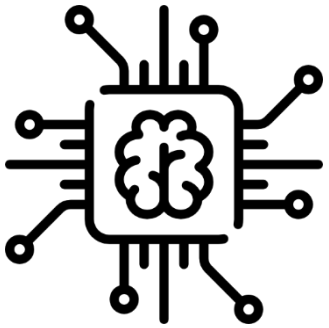
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride

2. Chapter: Pooling

3. Chapter: CNN in Action





Convolutional Neural Networks

Idea

1. How to avoid the loss of spatial information?

We need the ability to find and analyze the patterns/features.
What are images?
How do we humans understand and classify objects?

2. How to reduce the number of parameters/weights?

We need a new architecture (layer type)

Convolutional Neural Networks

Base Properties

- Same as ordinary neural networks (FCN/MLP), CNNs also **have neurons that have learnable weights and biases**.
- Each neuron still receives some inputs, **performs a dot product** and (optionally) follows it with a non-linearity, i.e., **activation** (function).
- The network still expresses a **single differentiable score function**: from the raw image pixels on one end to class probabilities at the other end.
- It **still has a loss function** on the last (fully-connected!) layer
- **All the tips/tricks** seen so far for learning regular (Deep) Neural Networks **still apply**.

Convolutional Neural Networks

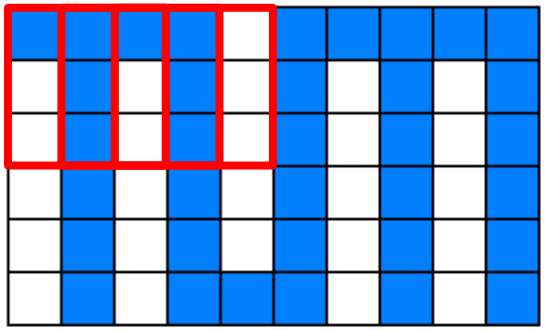
What is new then?

- CNNs make the explicit assumption that the **inputs are images** (2D, 3D inputs).
- This assumption **allows to encode certain properties into the architecture** which make the forward function much more efficient to implement.
- This **vastly reduces the number of parameters** in the network and **improves the learning performance** in general.

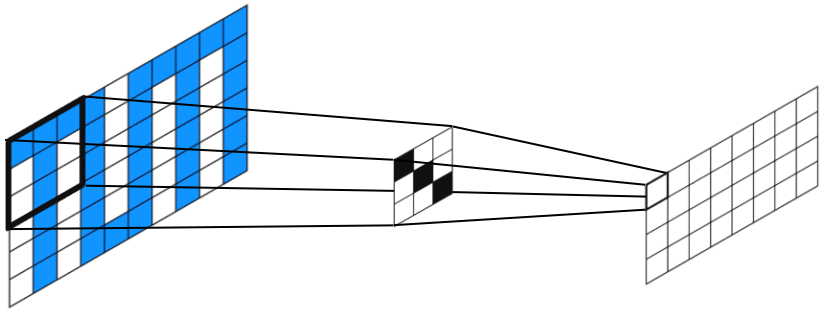
Convolutional Neural Networks

Idea

Find and analyze the patterns/features



Modified architecture that reduces the number of weights



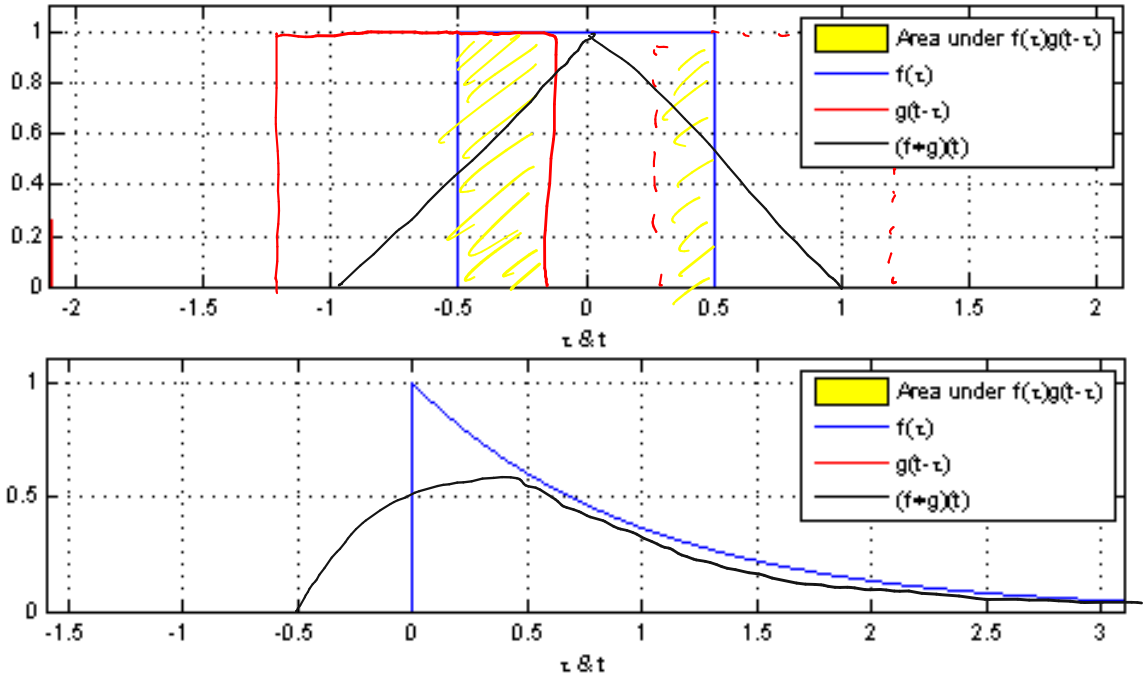
Instead of trying to learn and train parameters for all neuron connections, the concept of convolution is introduced. Now, the network will try to learn separate filters which are characteristic for different objects. For example, if we have an image of a car, our CNN network will try to come up with a set of different filters that describe a car, e.g., a filter which activates on tire detection, or window detection, etc. As the network gets deeper, filter values and dimensions will get more complex.

This way, the network needs to store only the corresponding weights of the filters, and since filters are usually much smaller than the image, this will result in the significant reduction of the memory need while have better learning performance.

Convolutional Layer

Convolution

Definition: $(f * g)(x) = \int_{\mathbb{R}^n} f(\tau) g(x - \tau) d\tau$



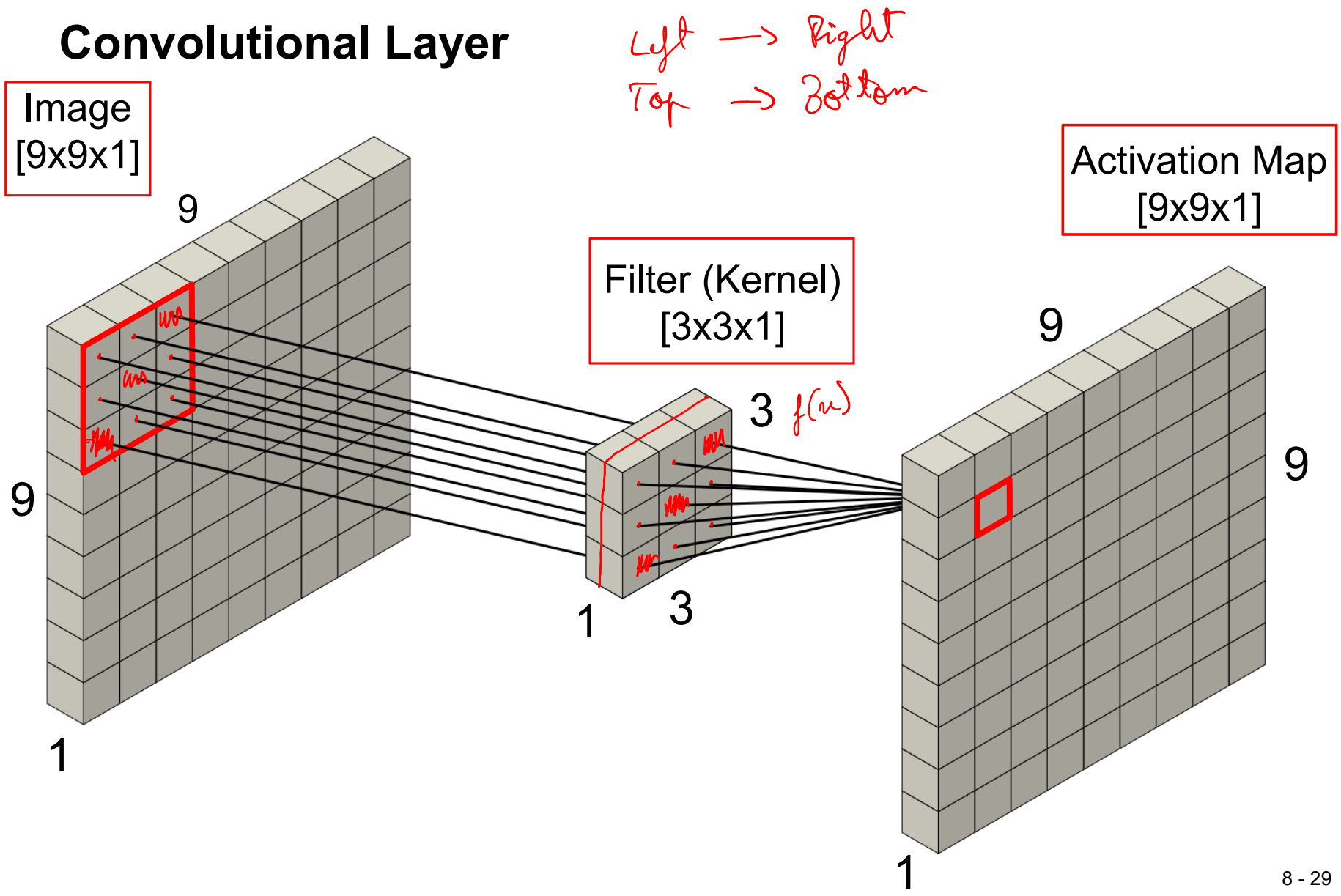
$$f(x, y) * g(x, y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f(n_1, n_2) \cdot g(x - n_1, y - n_2)$$

https://en.wikipedia.org/wiki/Convolution

Additional Slides

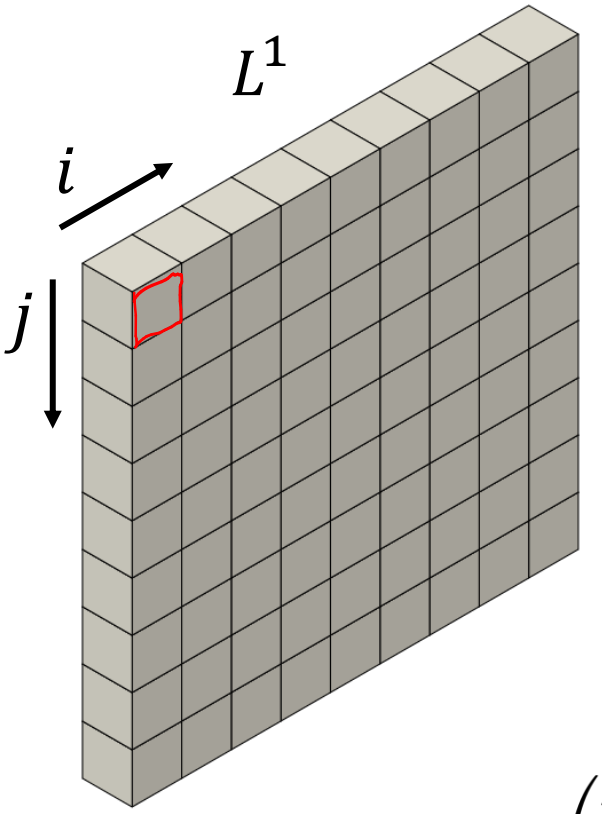
Convolution gives us a mathematical way to measure the similarity between two objects/signals/arrays.

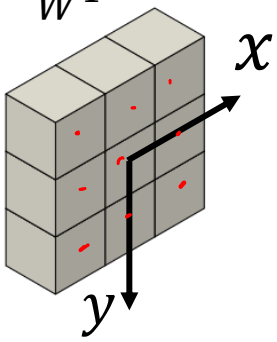
Convolutional Layer

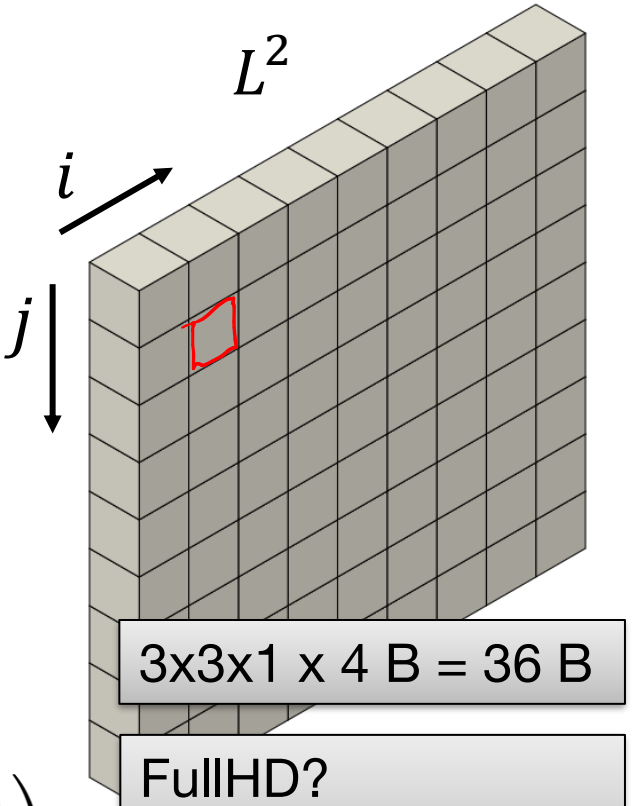


Convolutional Layer

$$L^2_{i,j} = f(\sum_{x=-1}^{1}\sum_{y=-1}^{1} L^1_{x+i,y+i} \cdot w_{x,y} + b_{i,j})$$





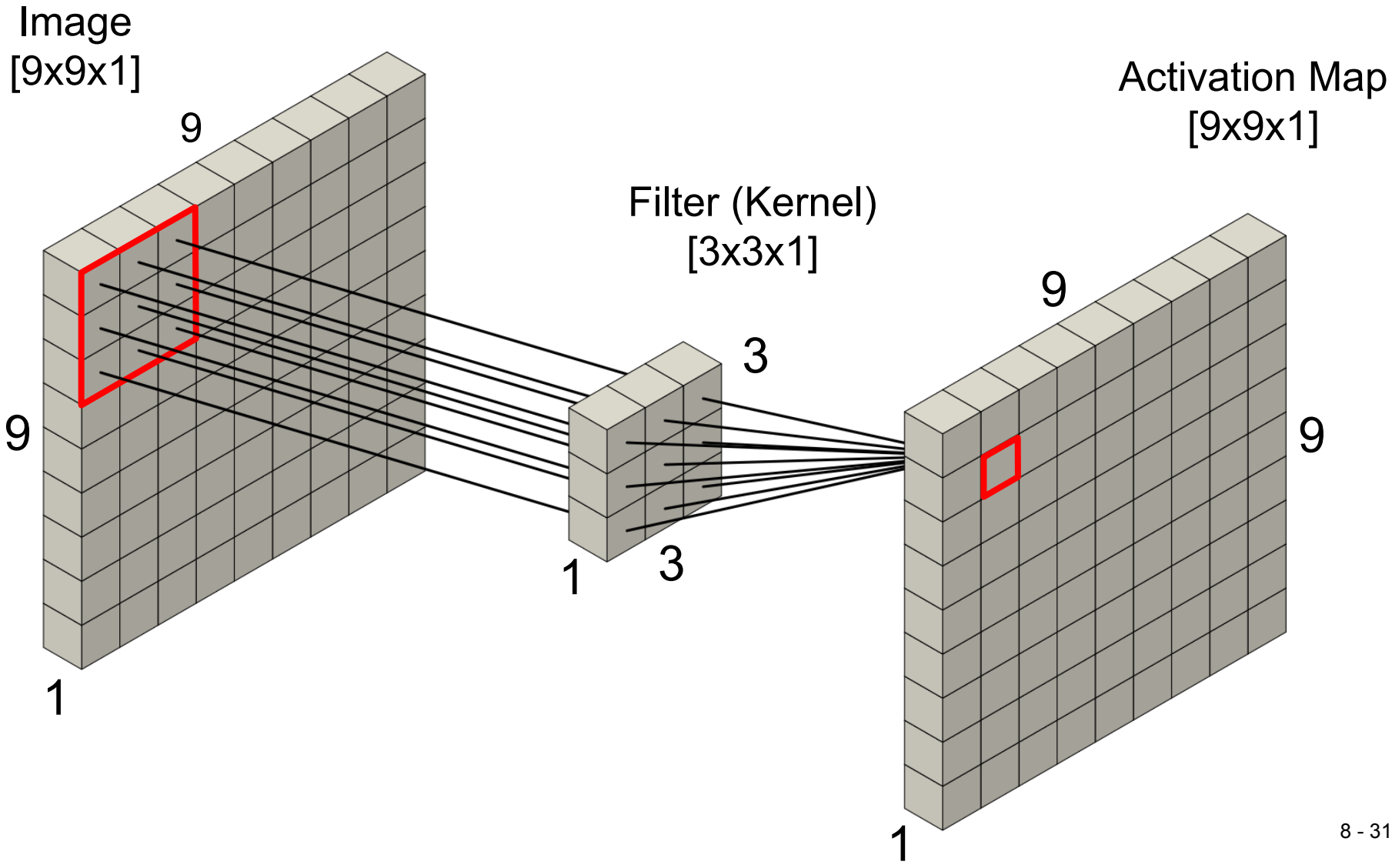


3x3x1 x 4 B = 36 B

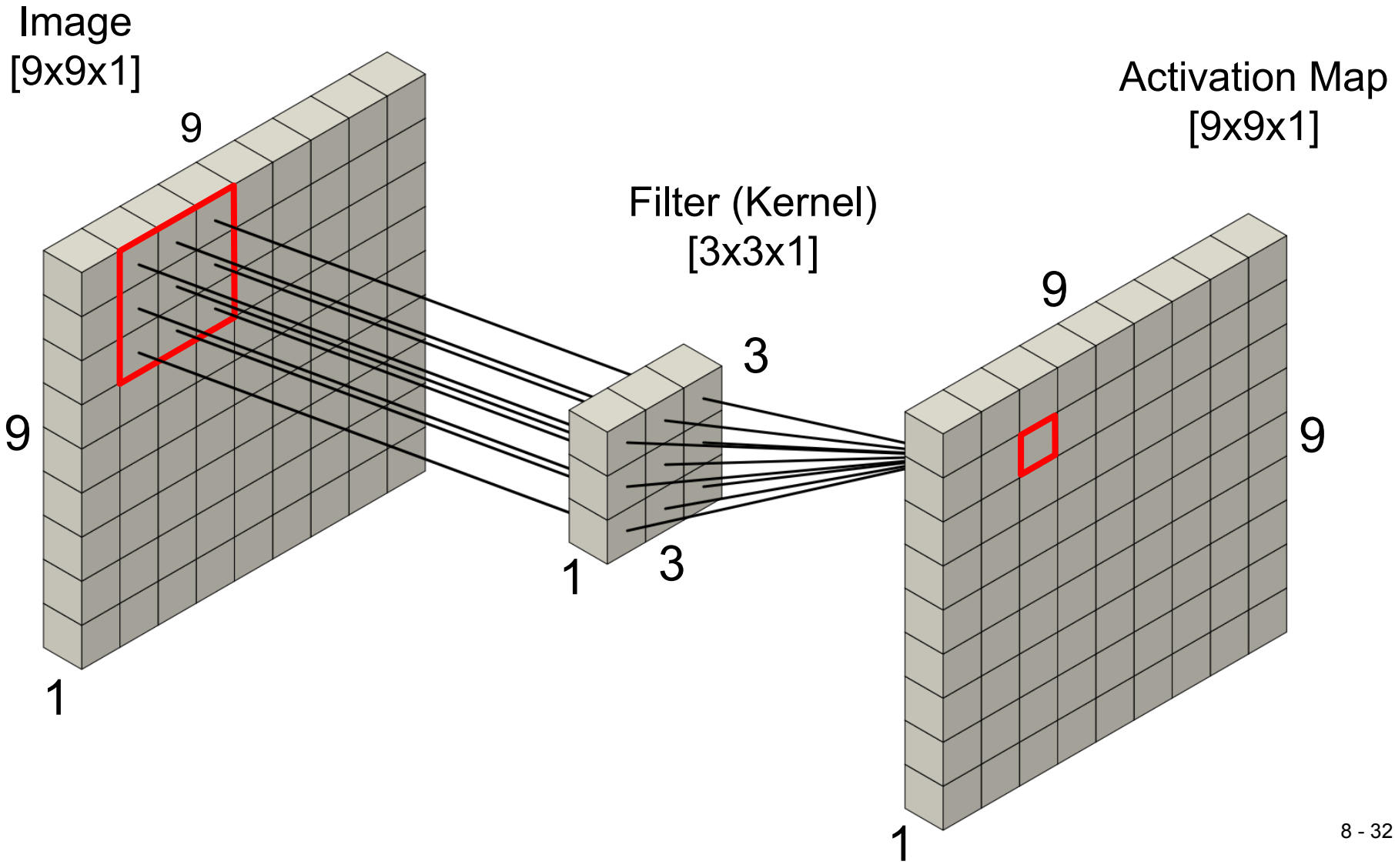
FullHD?
3x3x1 x 4 B = 36 B

$$W^1 = \begin{pmatrix} w_{-1,-1} & w_{0,-1} & w_{1,-1} \\ w_{-1,0} & w_{0,0} & w_{1,0} \\ w_{-1,1} & w_{0,1} & w_{1,1} \end{pmatrix}$$

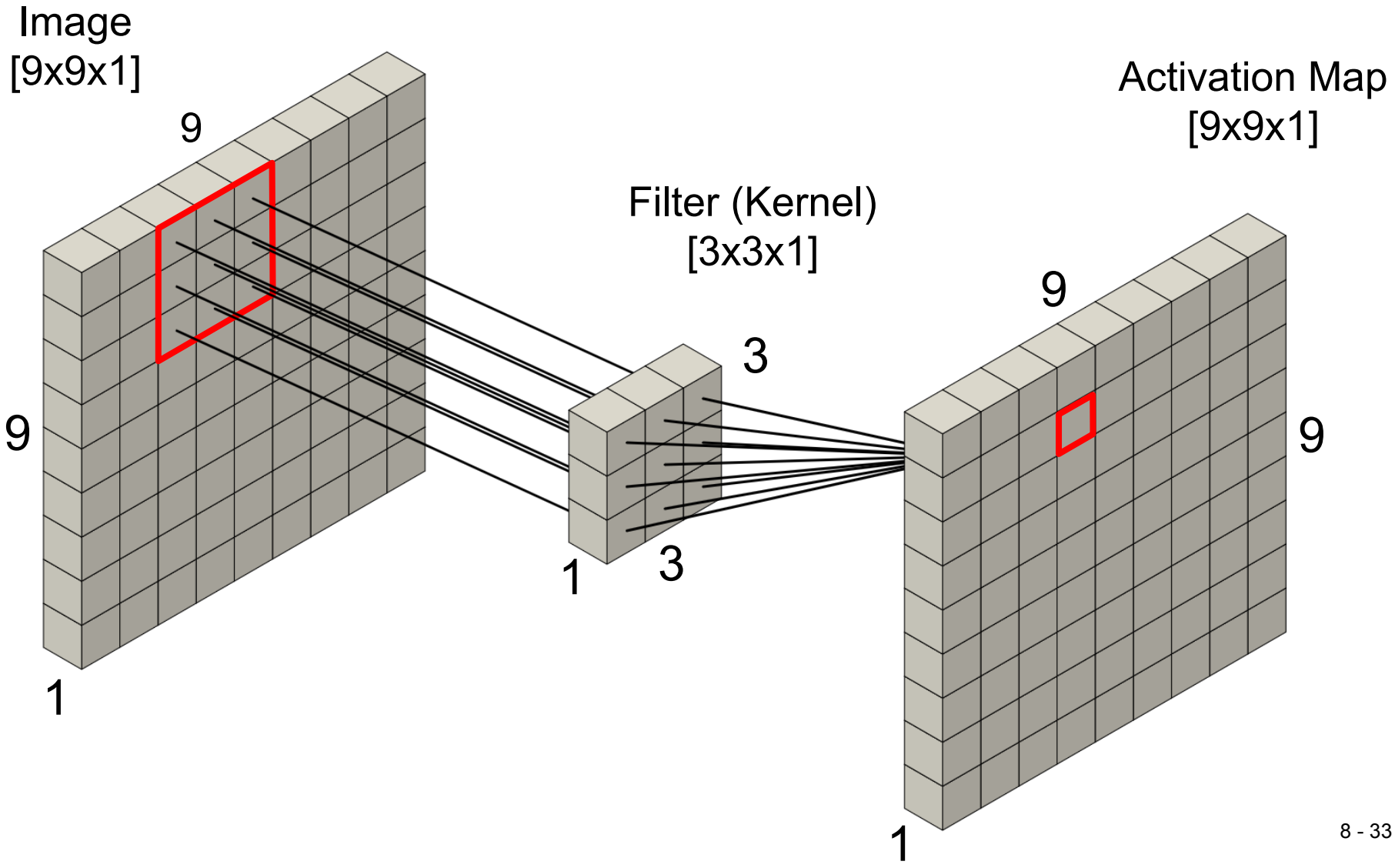
Convolutional Layer



Convolutional Layer



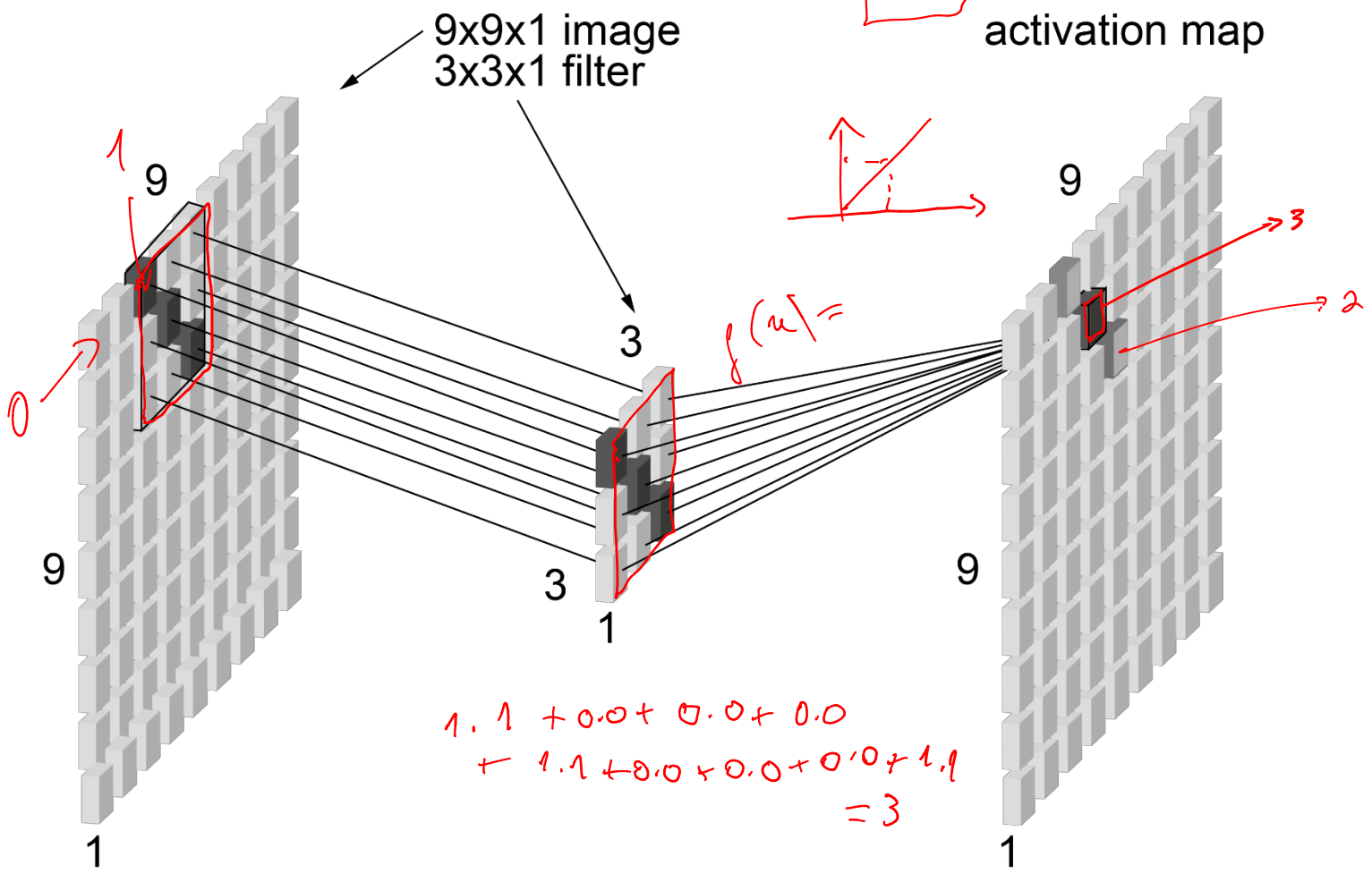
Convolutional Layer



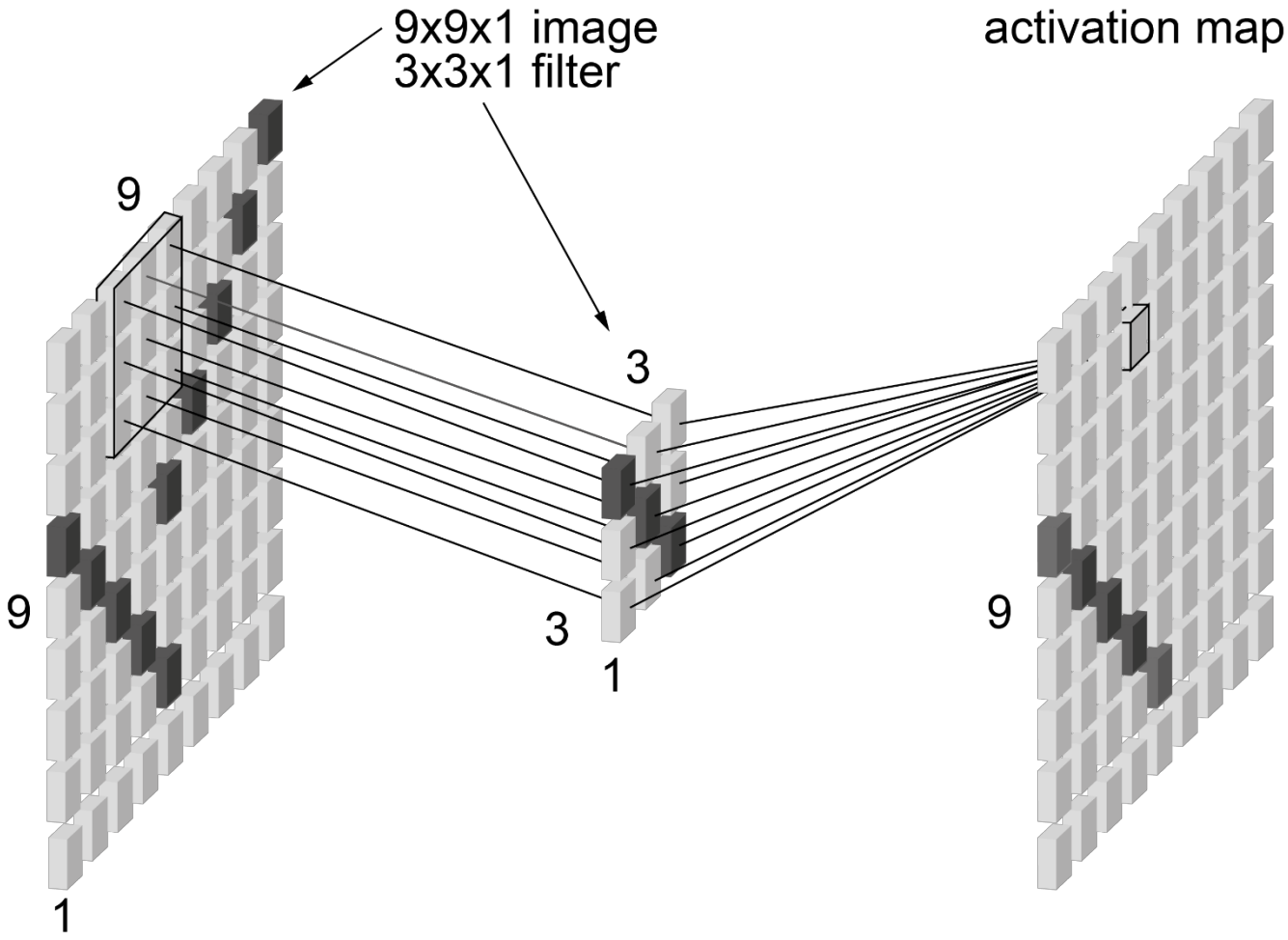
Kernel (filter) moves/convolves from left to right, and from top to bottom. Resulting value is stored into an activation map which represents particularly well activated the parts of the network (parts that have good overlap/correlation with the given filter). Each filter produces a separate activation map.

Convolutional Layer

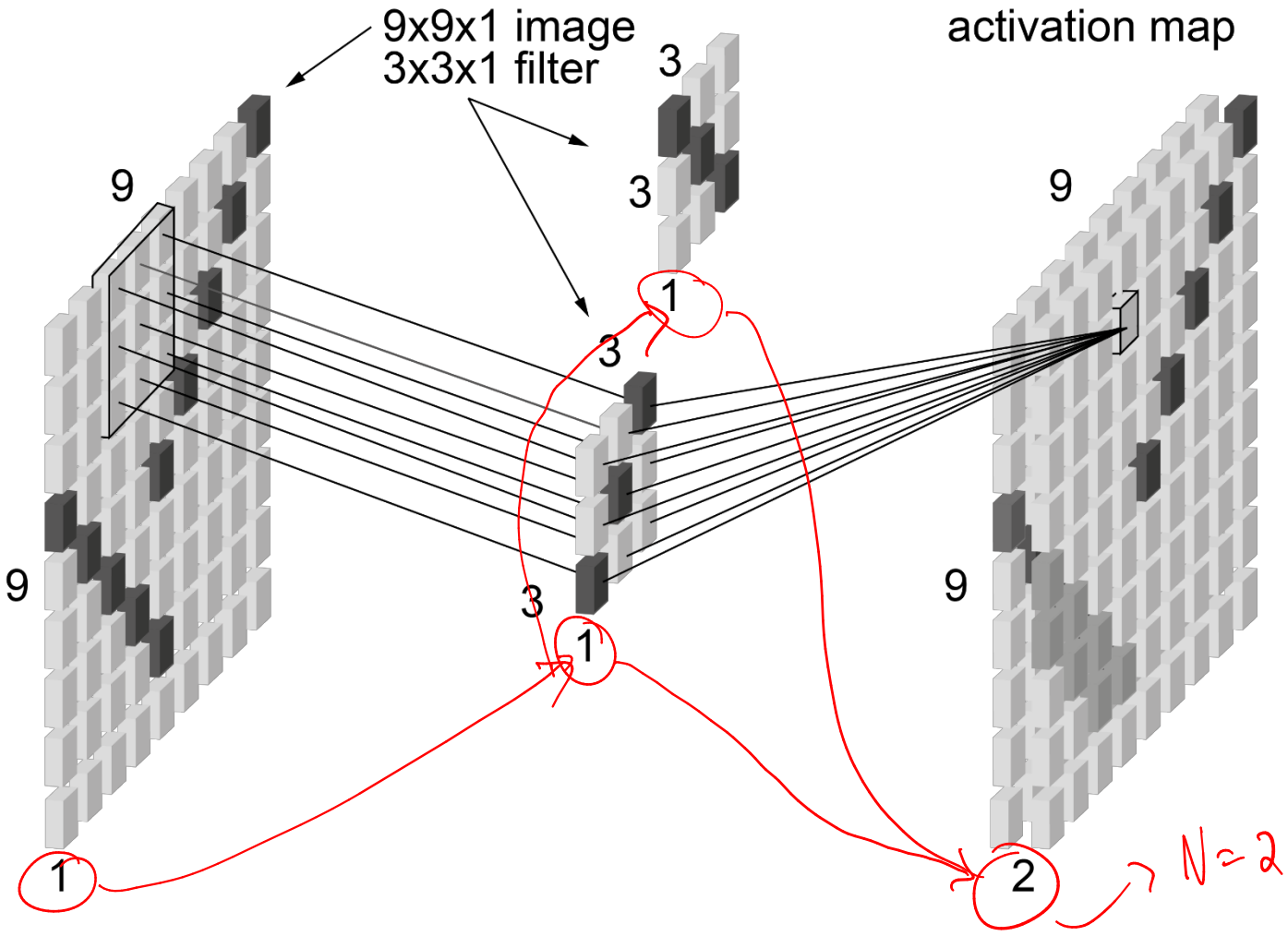
$$L^2_{i,j} = f(\sum_{x=-1}^{1} \sum_{y=-1}^{1} L^1_{x+i,y+i} \cdot w_{x,y} + b_{i,j})$$



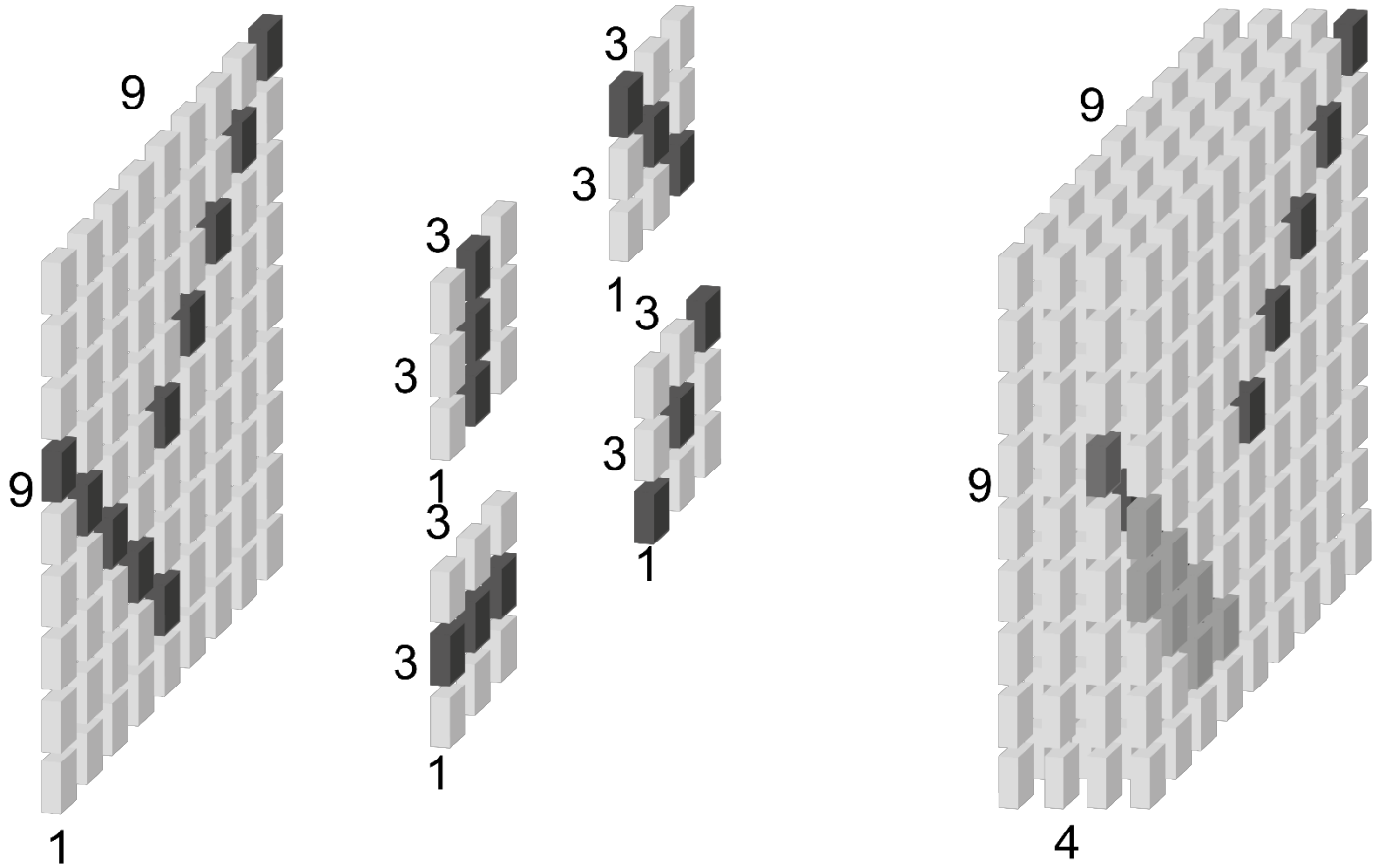
Convolutional Layer



Convolutional Layer

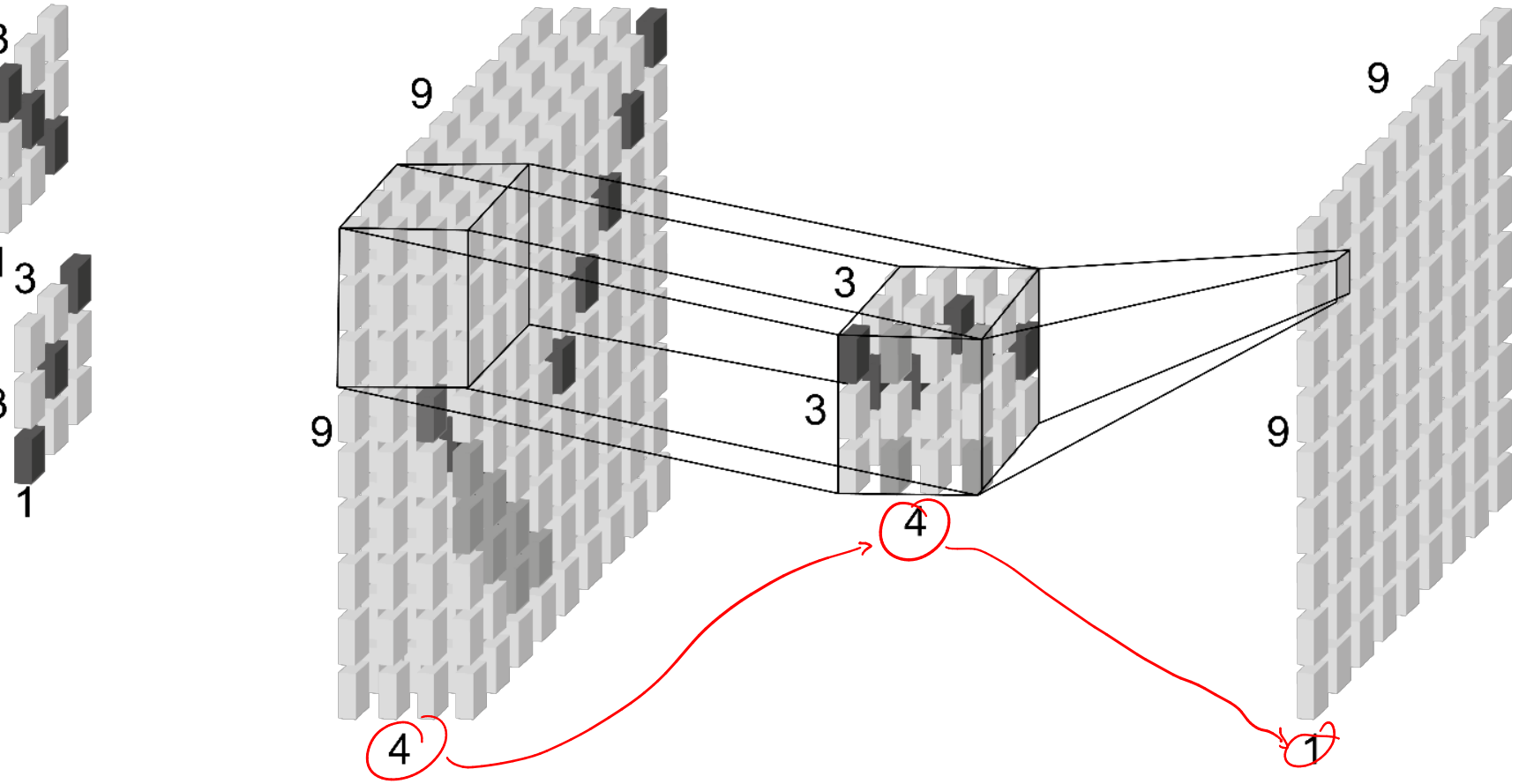


Convolutional Layer



Convolutional Layer

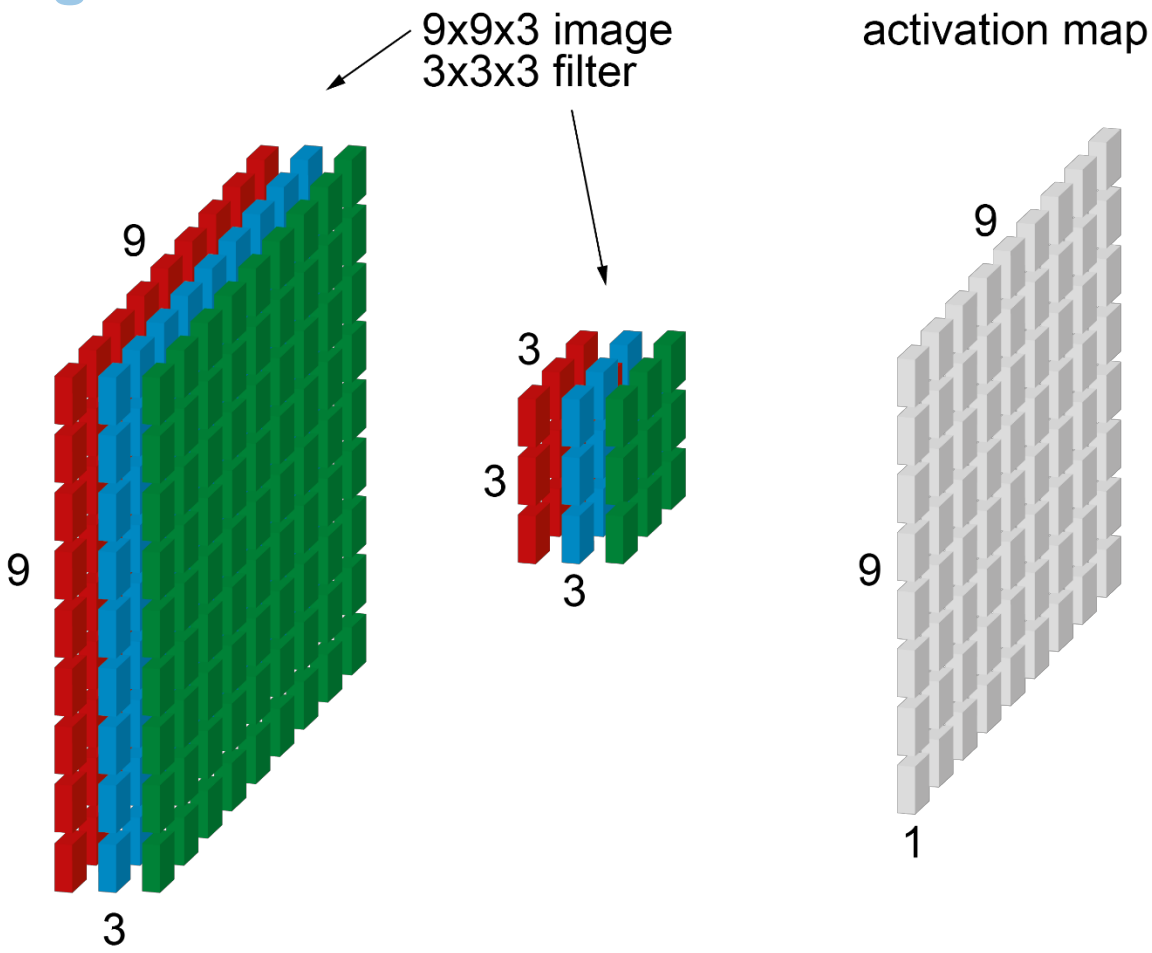
$$L^2_{i,j,k} = f(\sum_{x=-1}^{1}\sum_{y=-1}^{1}\sum_{z=1}^{4} L^1_{x+i,y+i,z} \cdot w^k_{i,j,z} + b^k_{i,j})$$



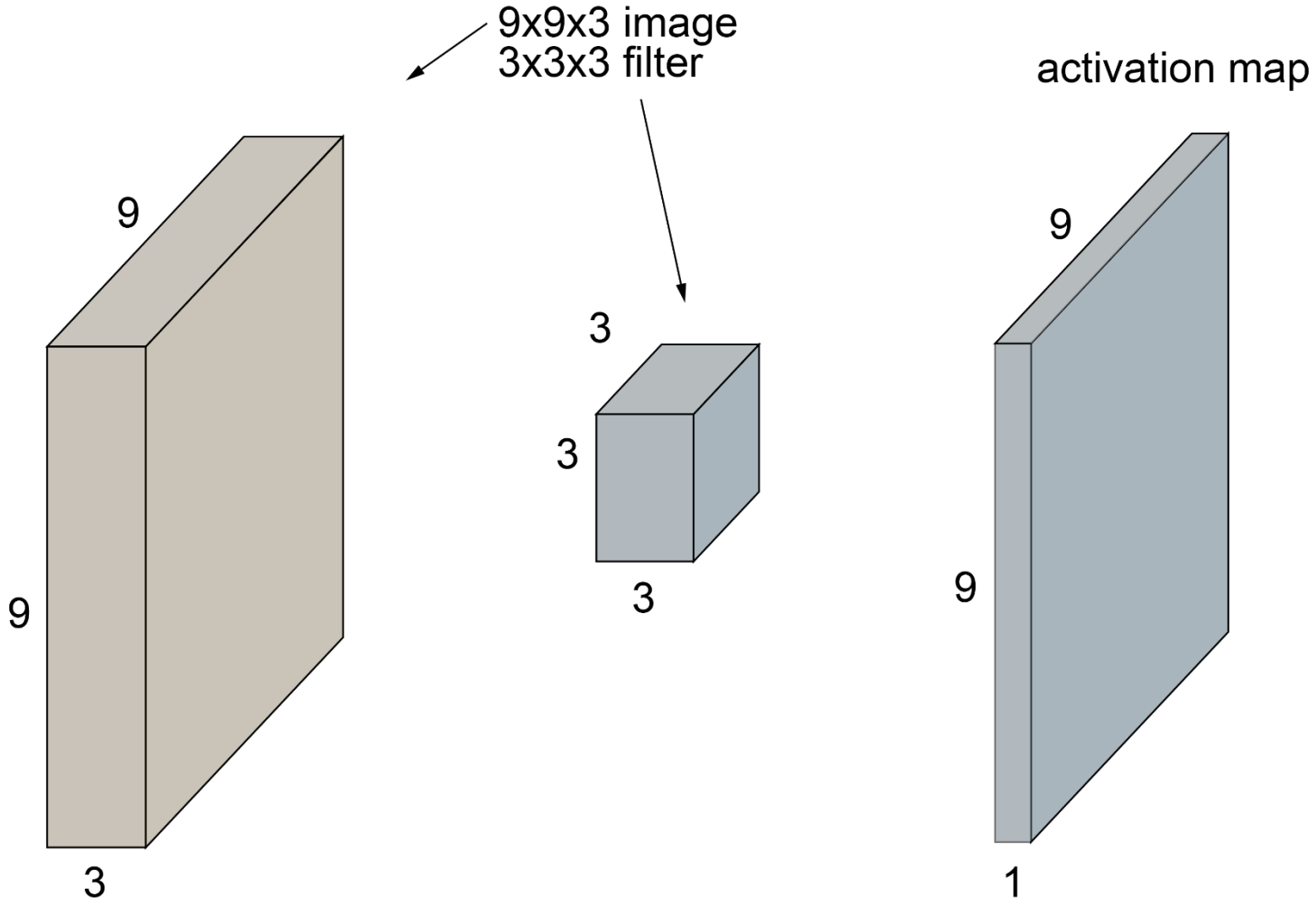
Dimensions are important. Number of filters defines the number of resulting activation maps. Newly produced depth of the activation maps, defines the depth size for new filters. As the filter depth gets more complex, it becomes more difficult for humans to interpret the filter tuning results.

Convolutional Layer

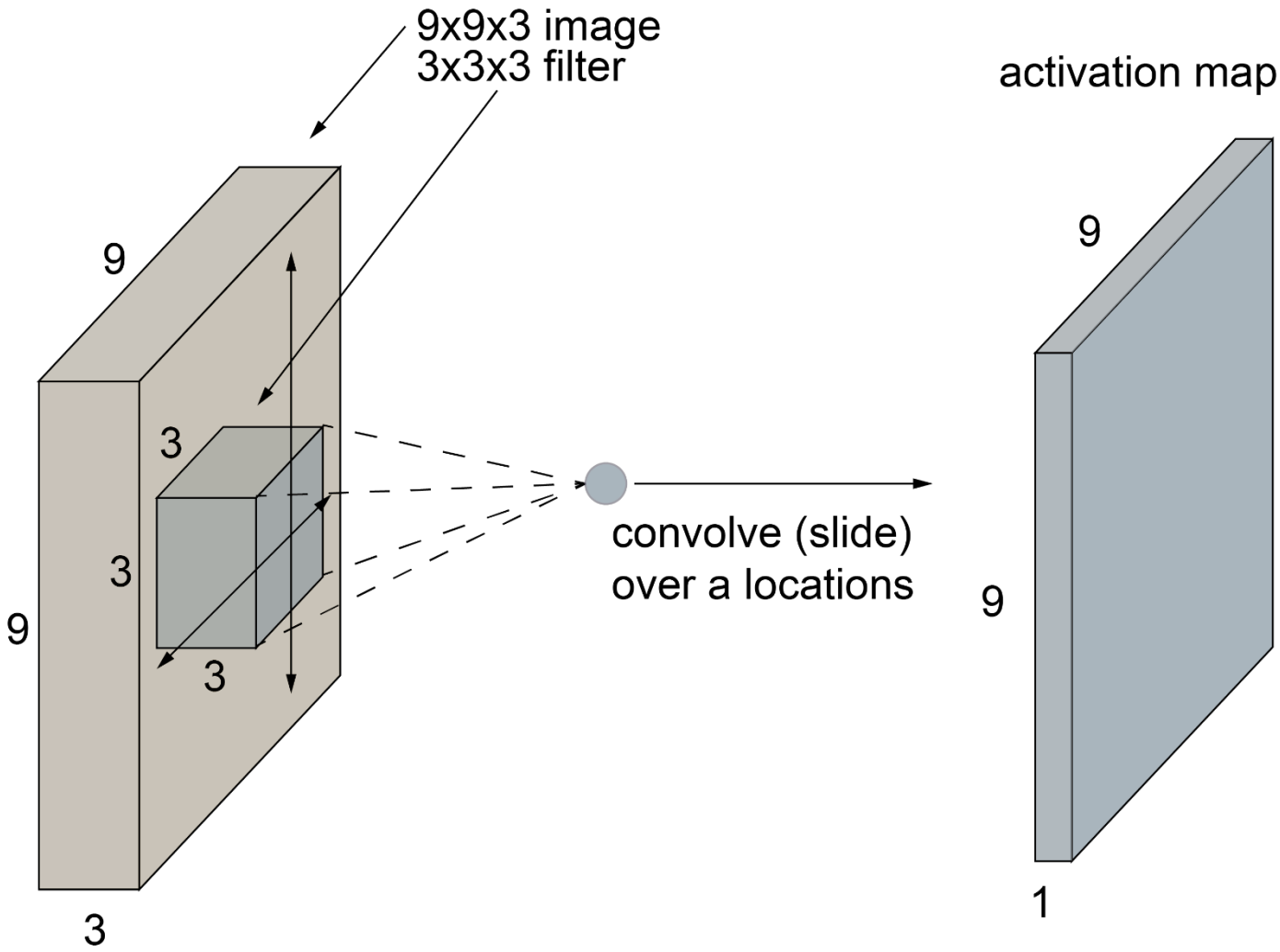
RGB Image



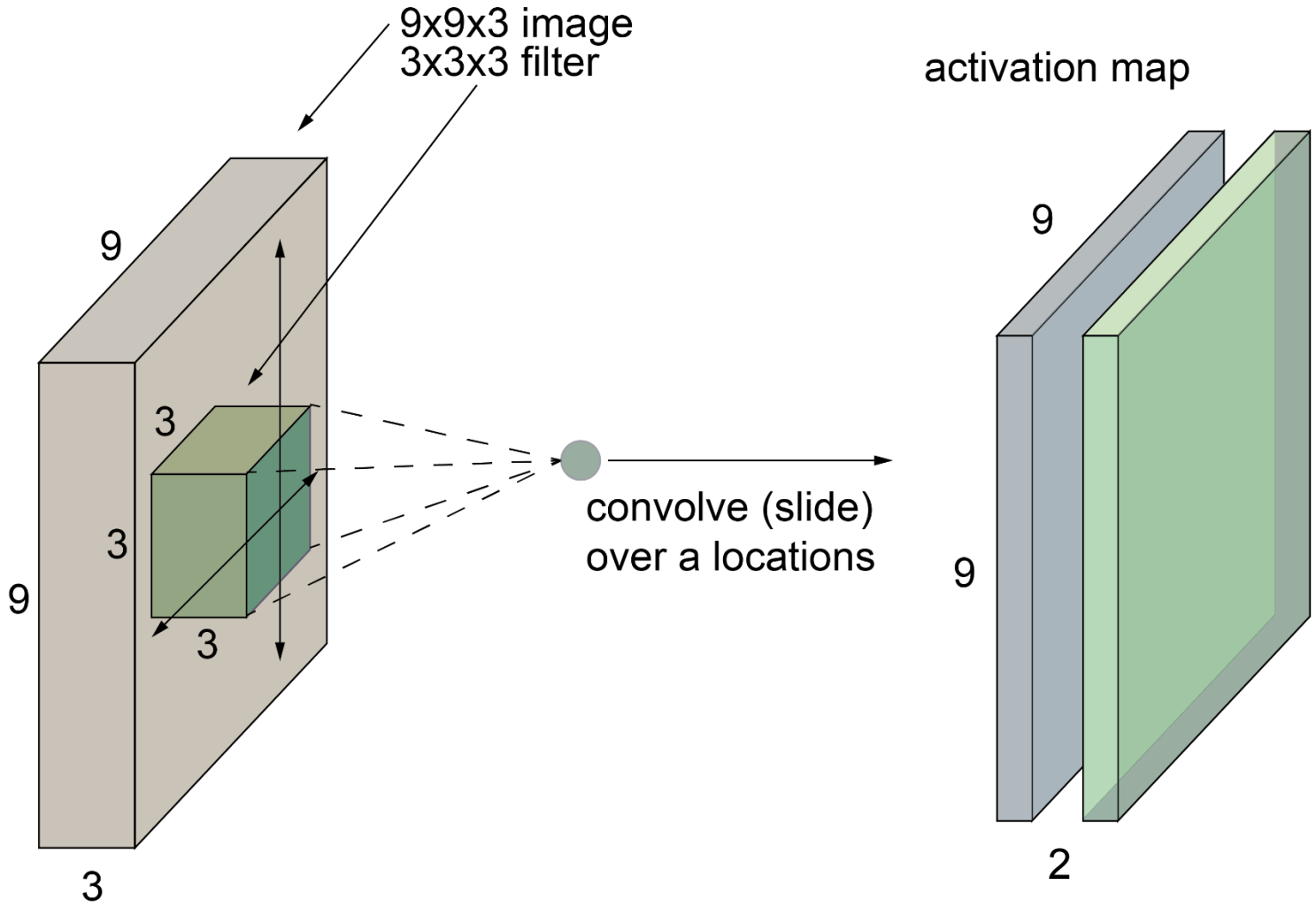
Convolutional Layer



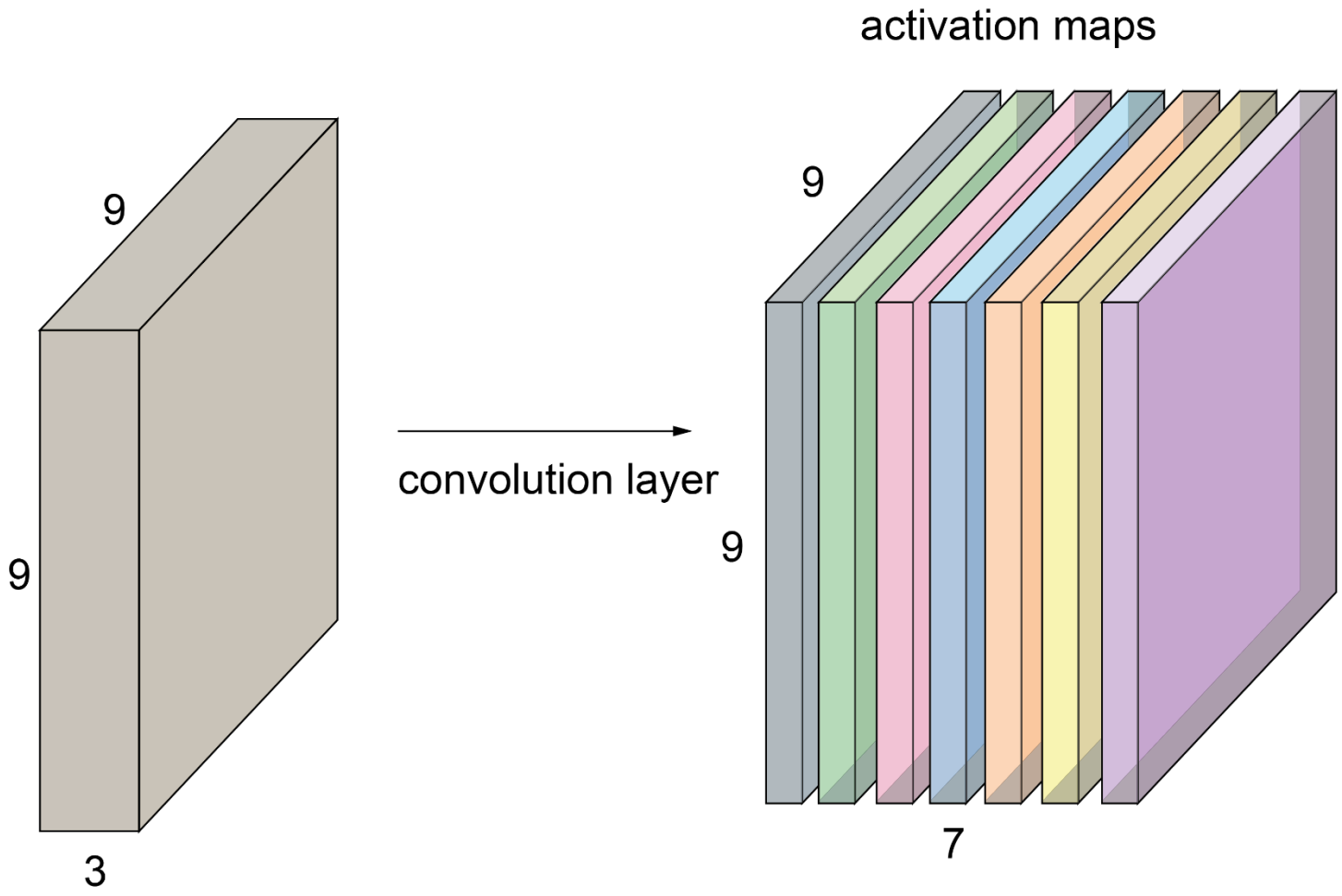
Convolutional Layer



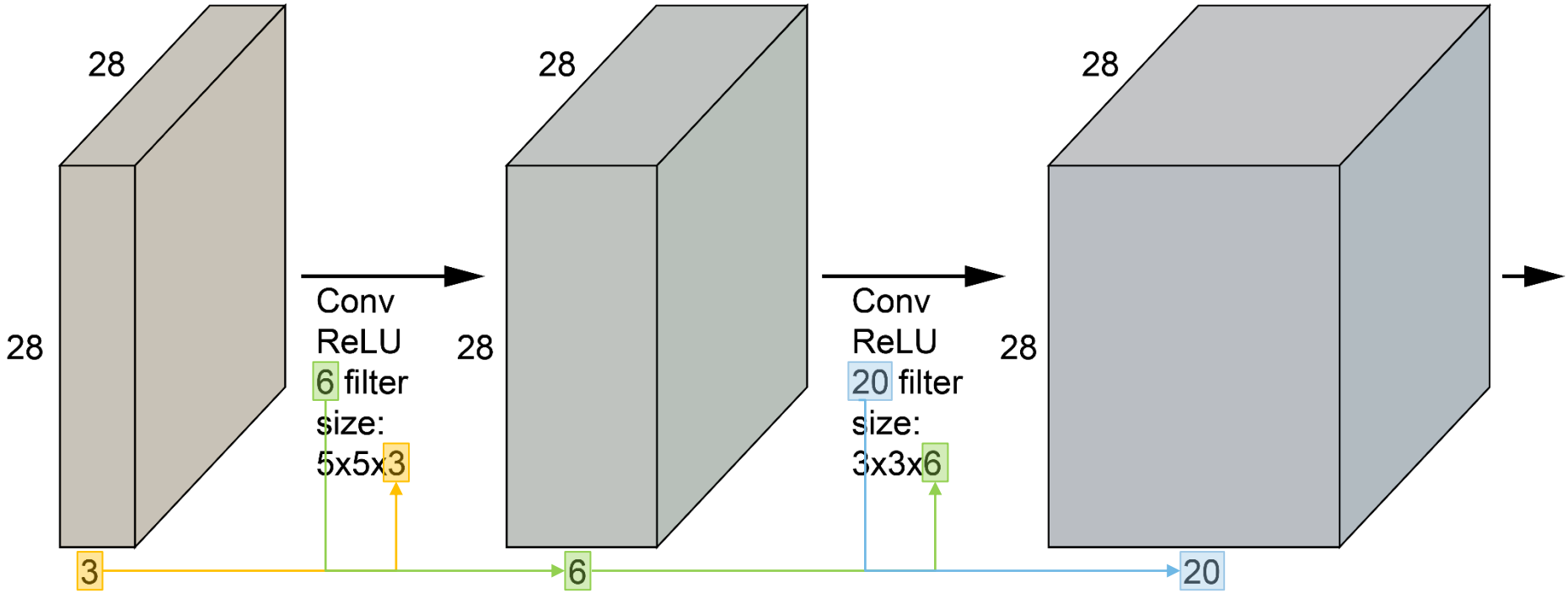
Convolutional Layer



Convolutional Layer

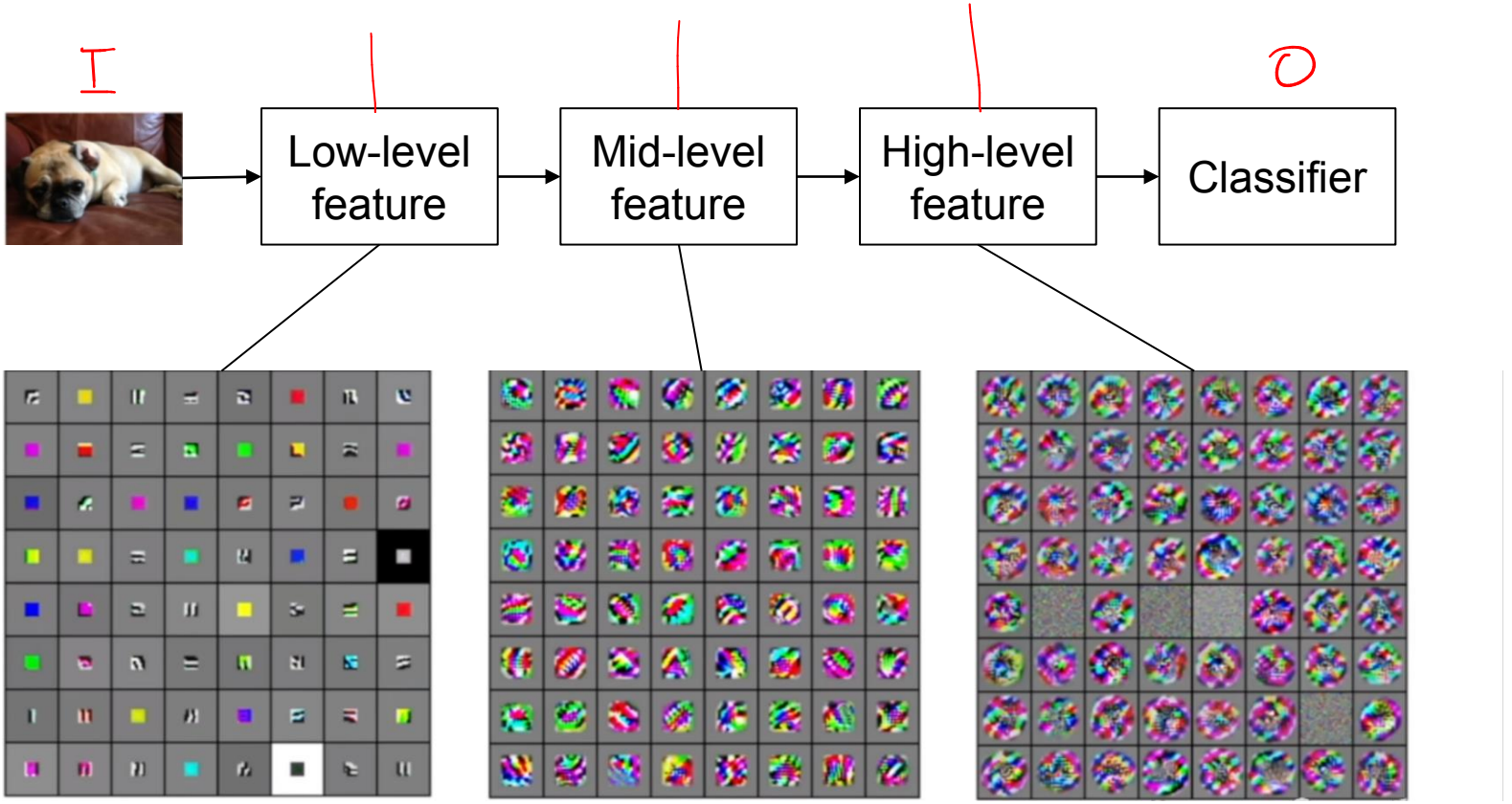


Convolutional Layer



Convolutional Layer

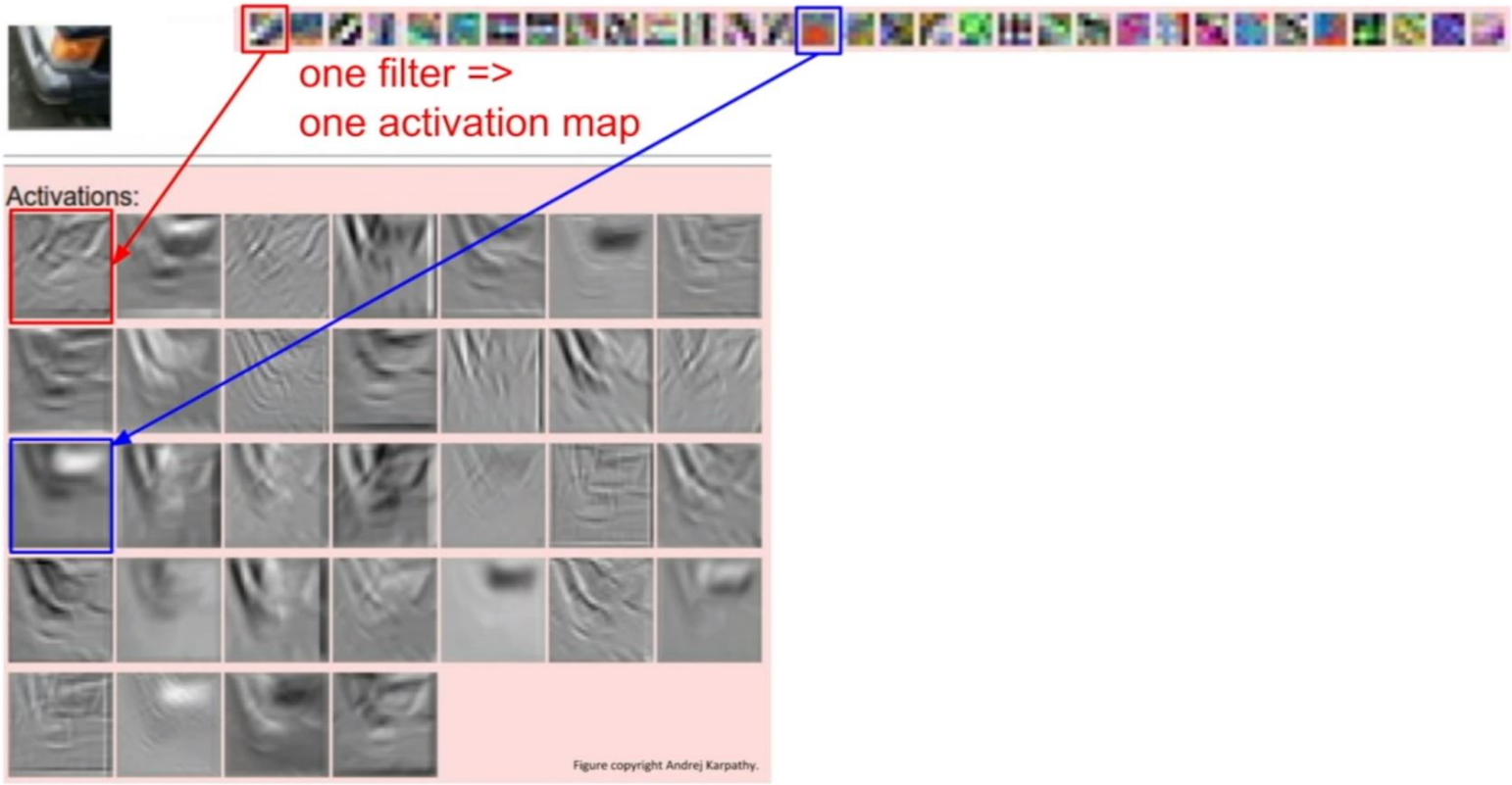
Filter Visualization



High level features are more complex and usually almost impossible for humans to interpret. On the other hand, low level features usually have some trivial features such as circles, squares, diagonal lines, etc., and can be identified.

Convolutional Layer

Activation Map Visualization



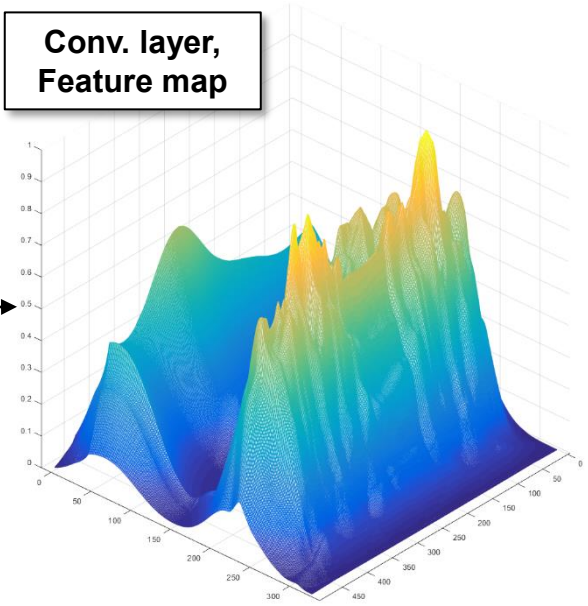
Convolutional Layer

Visualisation

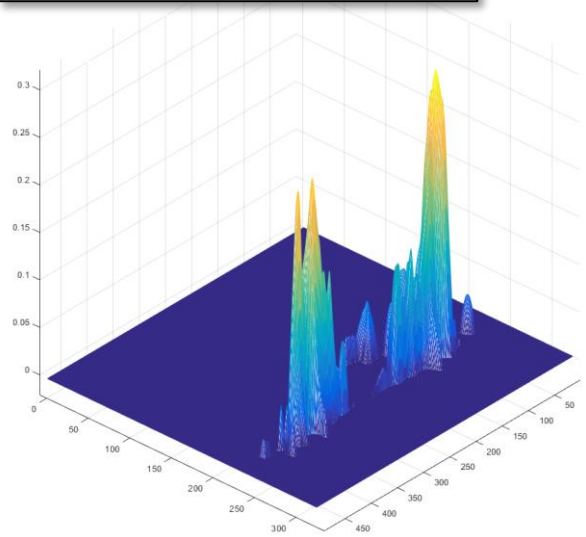
Input Layer

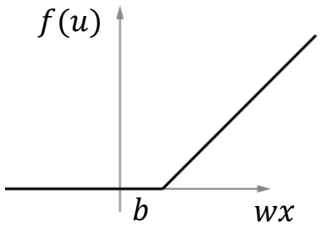
Feature 1

Conv. layer, Feature map



Relu Activation Function b=-0.7





Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

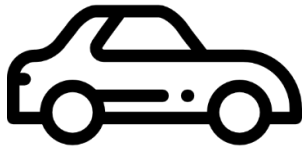
Agenda

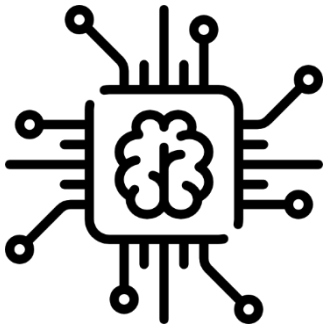
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride

2. Chapter: Pooling

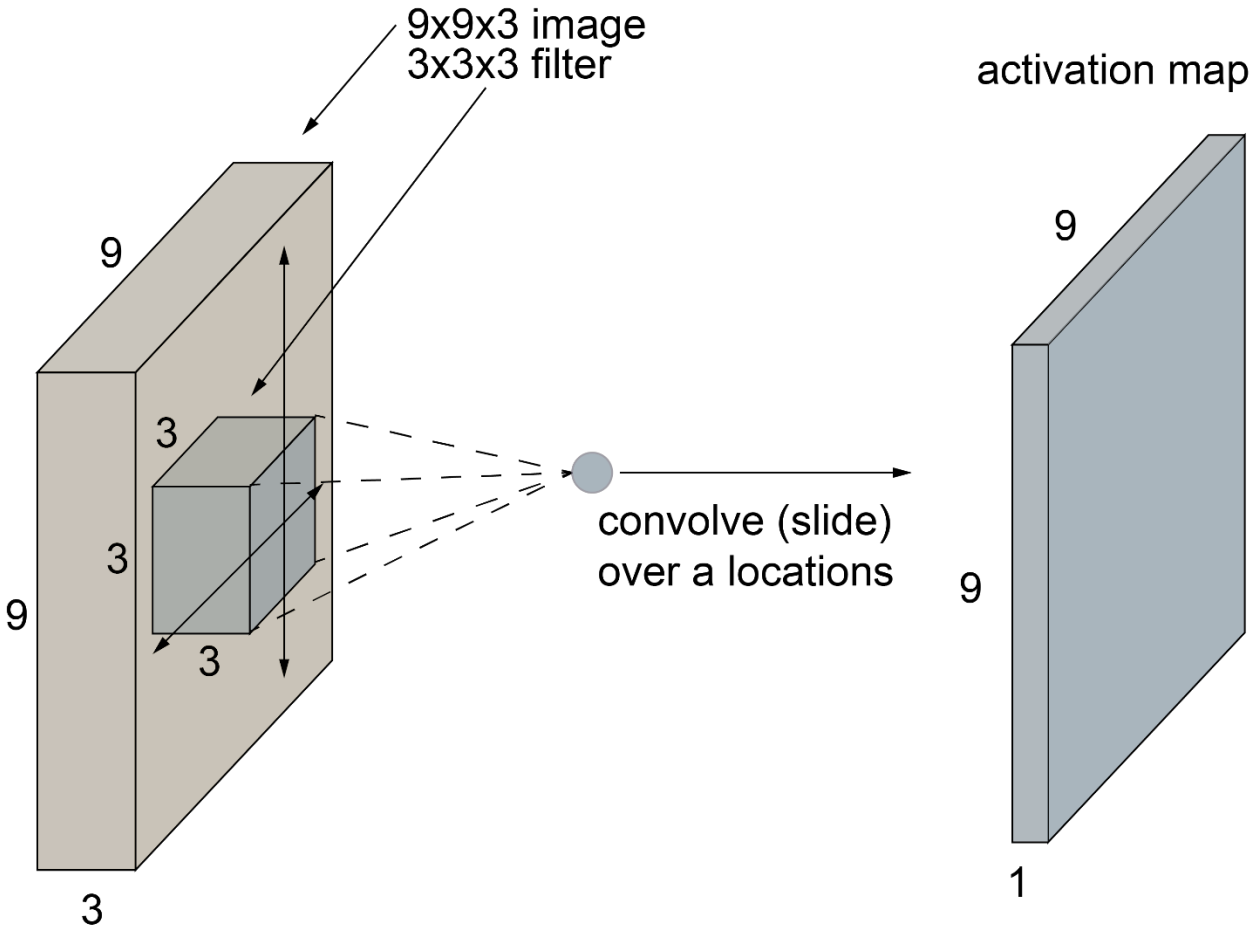
3. Chapter: CNN in Action





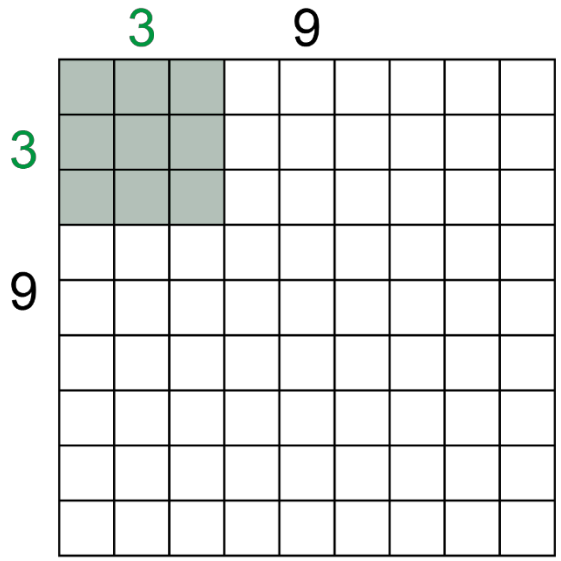
Convolutional Layer

Dimensions



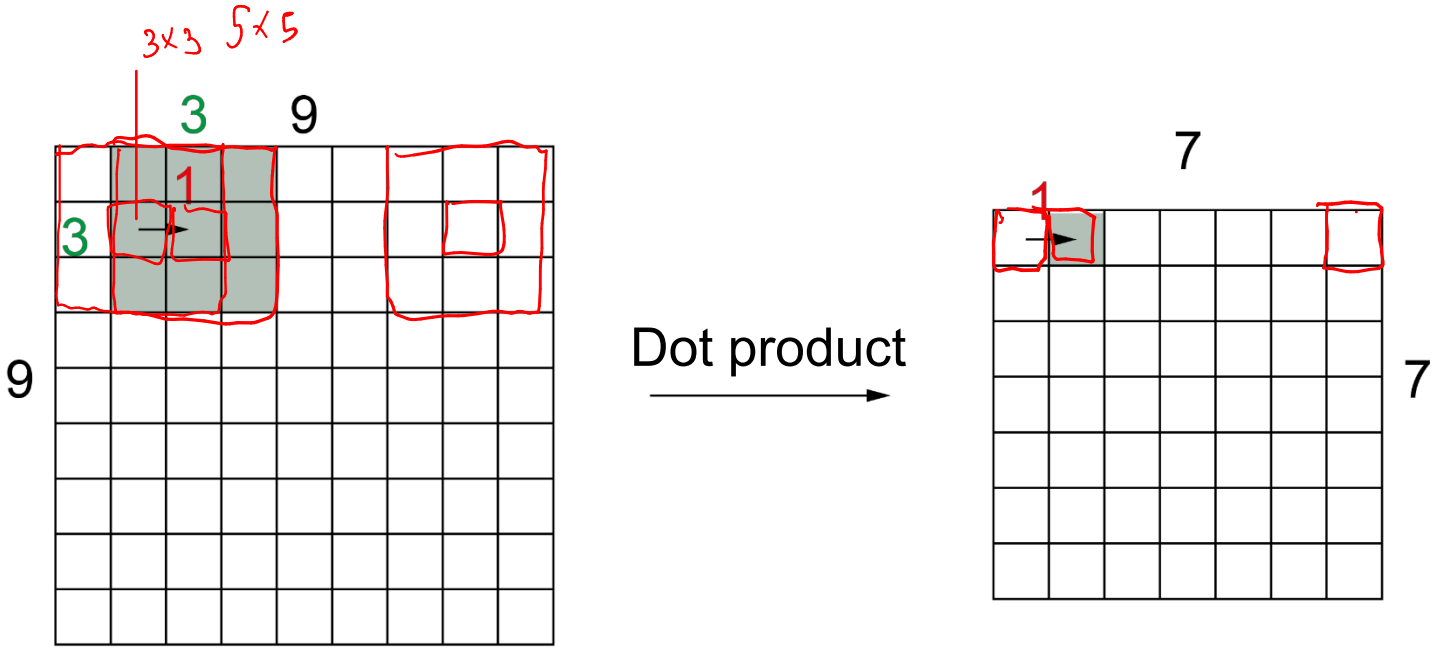
Convolutional Layer

Dimensions



Convolutional Layer

Dimensions



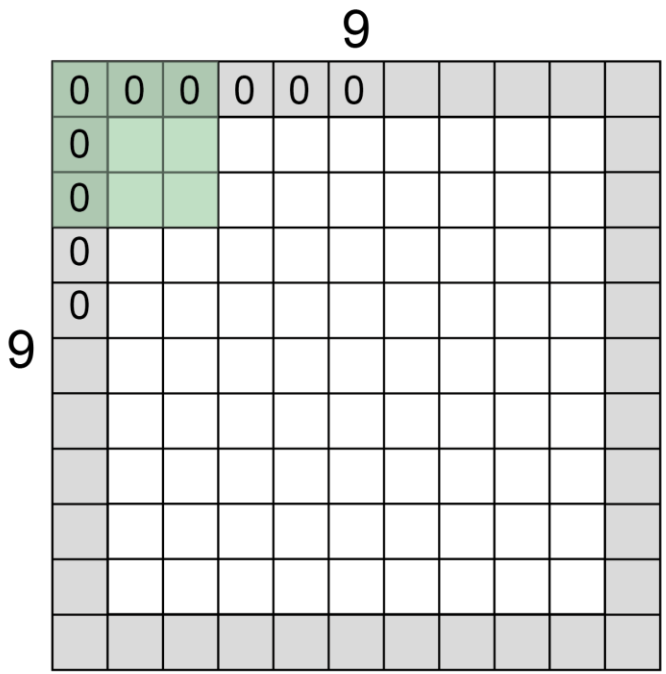
Applying convolution will by default result in some dimension reduction. Sometimes this might be ok, sometimes not. If it's necessary to keep the original dimension after the convolution process, it's possible to use Padding feature to control that behavior.

Padding will add additional "artificial values" around the layer (usually with 0 or 1 values) to extend the dimension in a way that the convolution produces the same dimension as the layer before. This is a very effective and inexpensive way to maintain the dimension.

Convolutional Layer

Dimensions: Padding

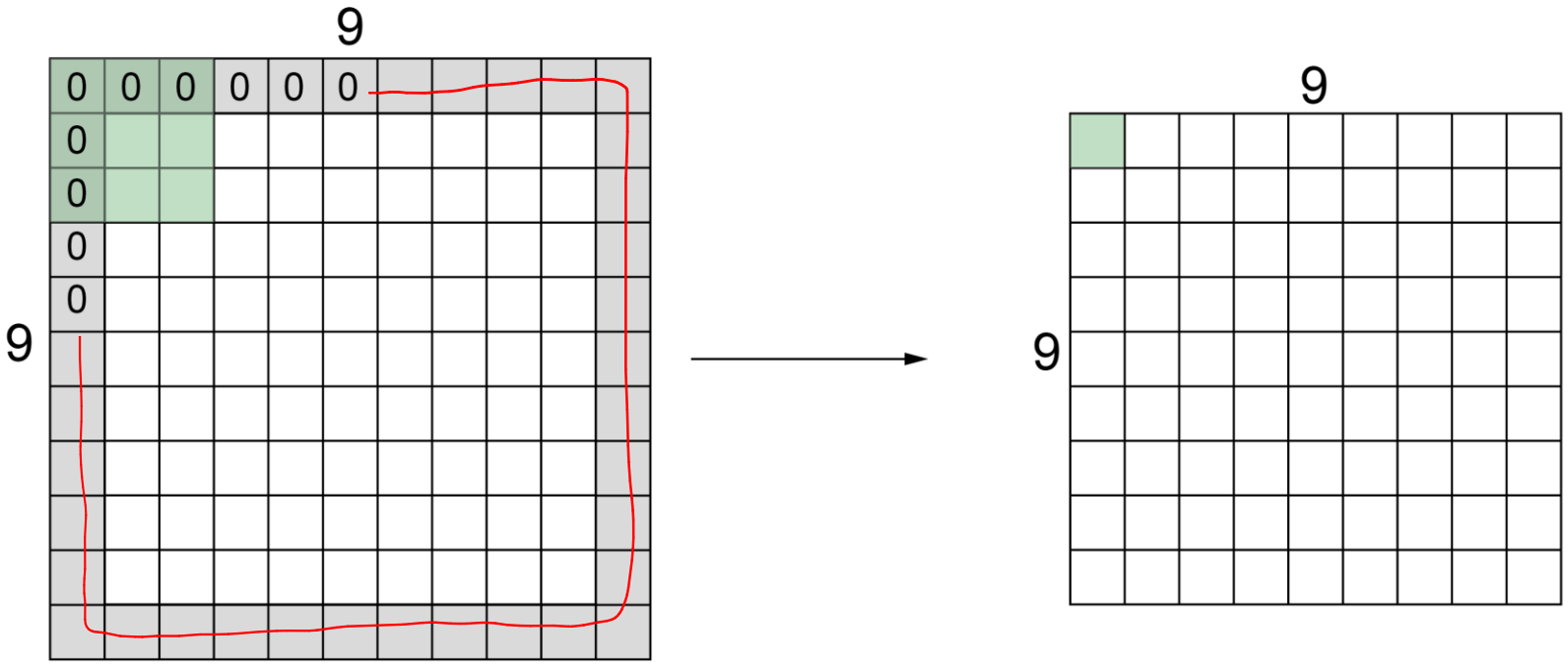
Padding = 1



input 9x9
3x3 filter, applied with stride 1
what is the output?

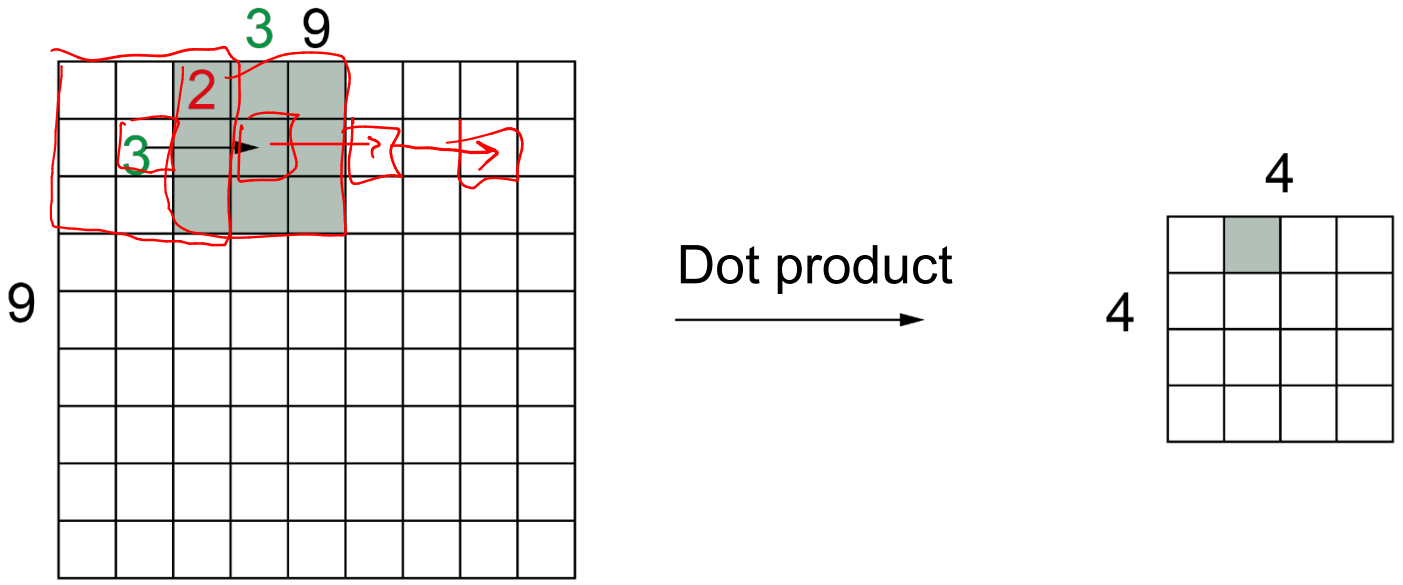
Convolutional Layer

Dimensions: Padding



Convolutional Layer

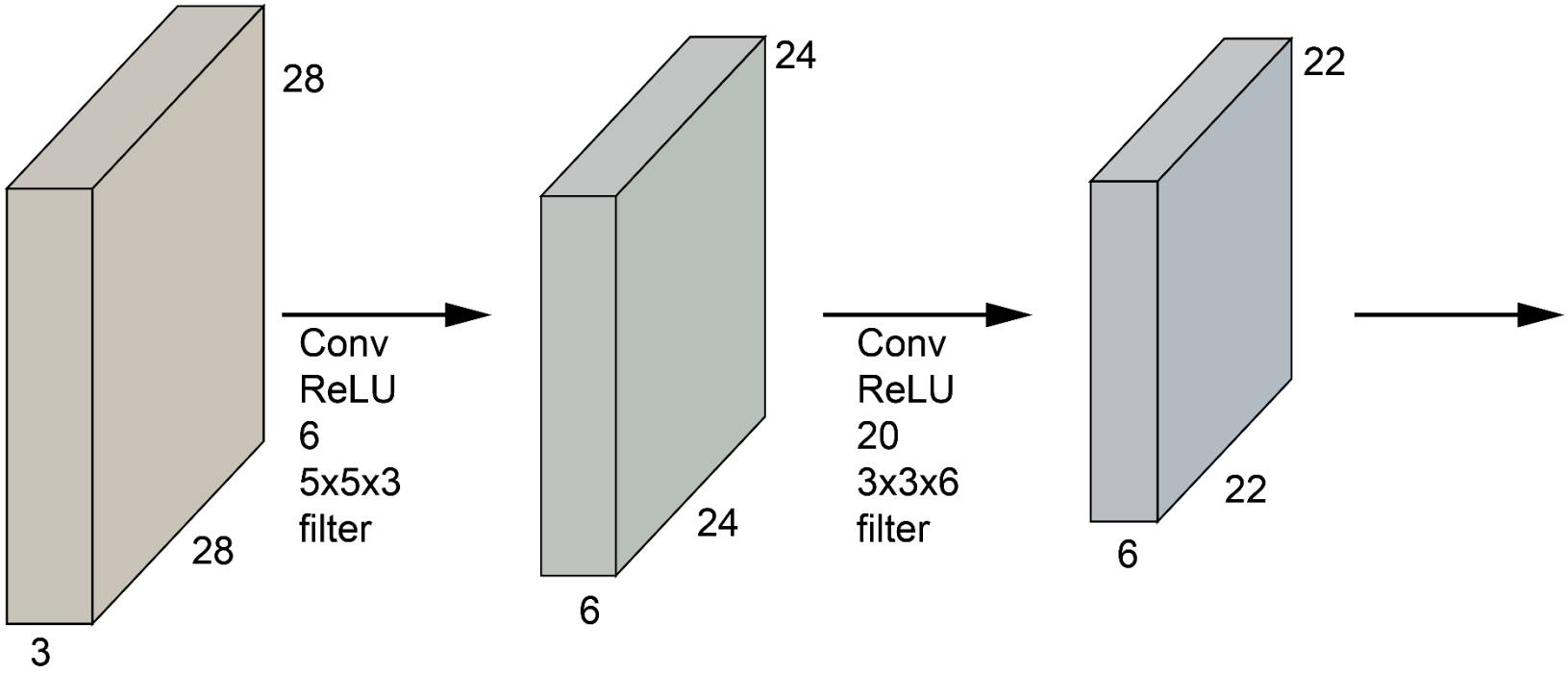
Dimensions: Stride



Convolutional Layer

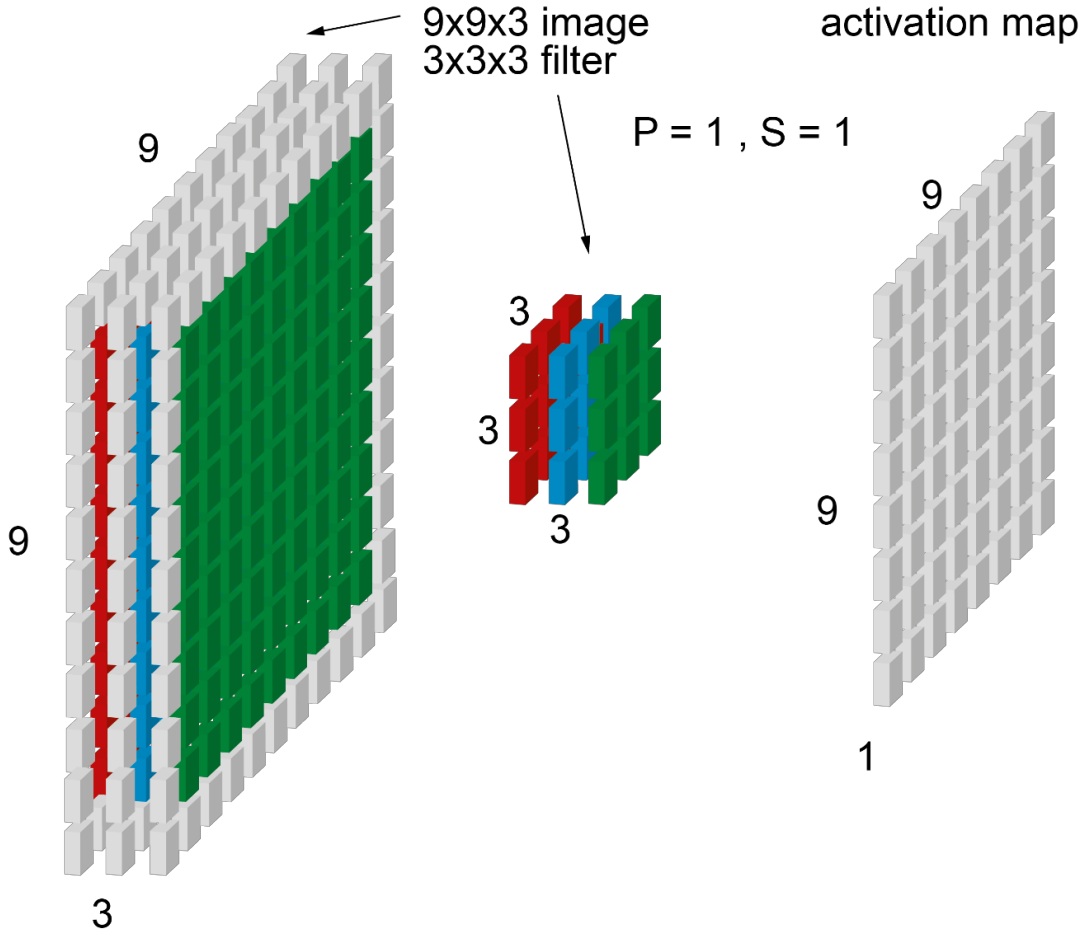
Padding

Without padding, volume shrinks with every convolution layer. This can cause problems in very deep neuronal networks. Use padding!



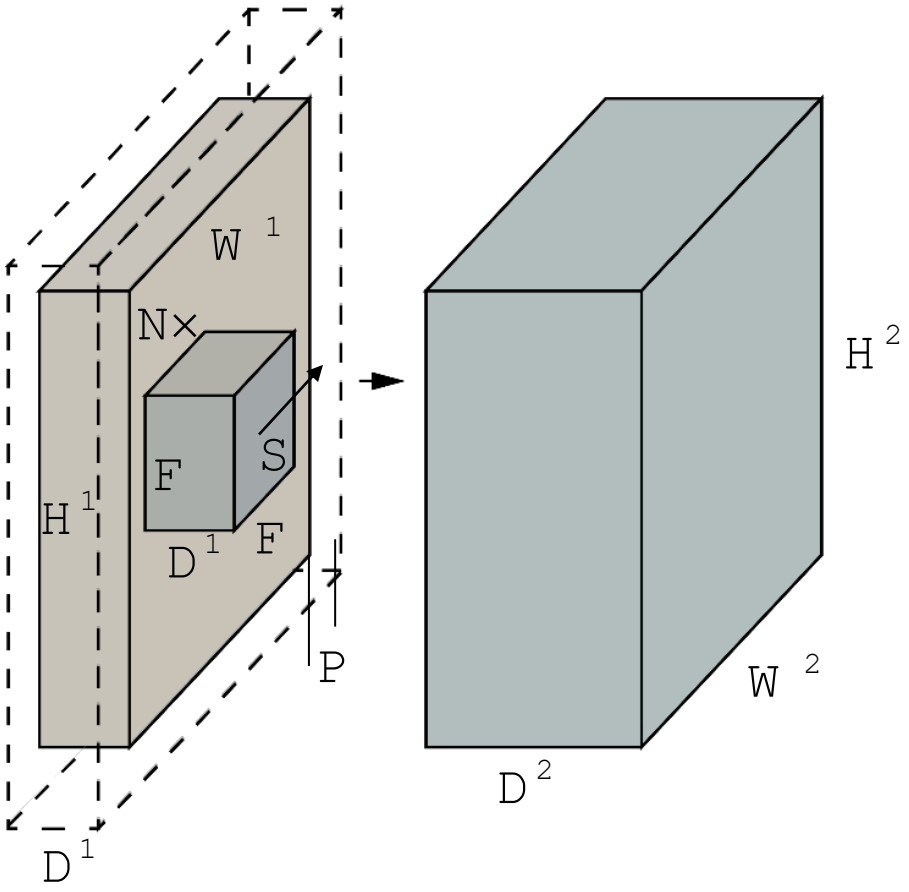
Convolutional Layer

Padding with RGB Image



Convolutional Layer

Dimension Formula



$$L^1 \in K^{D^1 \times W^1 \times H^1} \Rightarrow L^2 \in K^{D^2 \times W^2 \times H^2}$$

N : Number of Filters

F : Filter size

S : Stride

P : Padding

New Dimensions:

$$W^2 = \frac{W^1 + 2P - F}{S} + 1$$

$$H^2 = \frac{H^1 + 2P - F}{S} + 1$$

$$D^2 = N$$

Keep Dimensions:

$$F = 3 \Rightarrow P = 1$$

$$F = 5 \Rightarrow P = 2$$

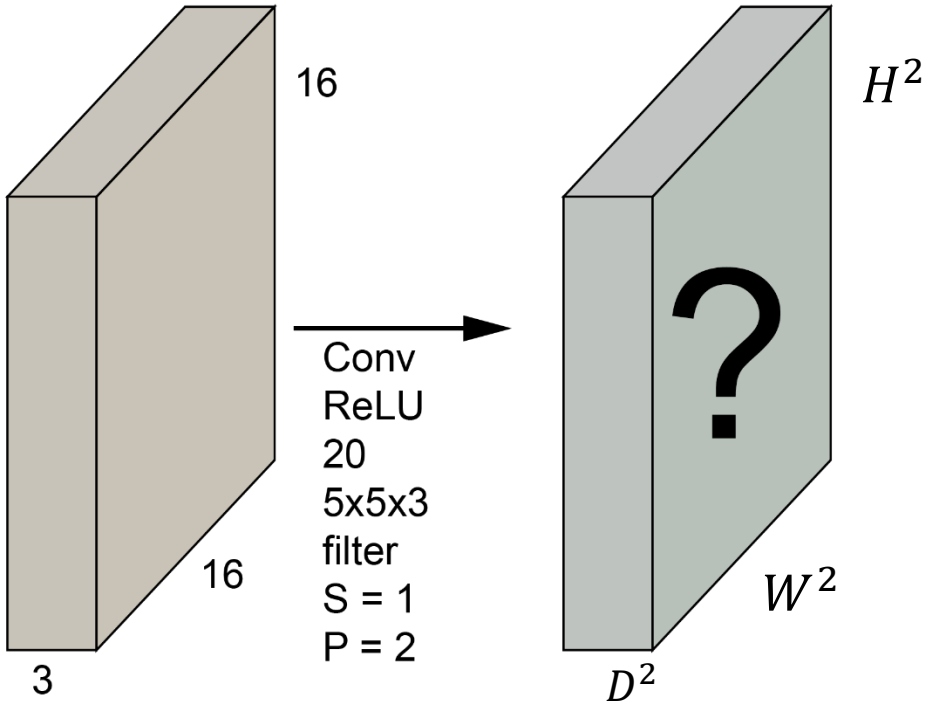
$$F = 7 \Rightarrow P = 3$$

Additional Slides

These formulas can be used to calculate resulting dimensions.

Convolutional Layer

Example



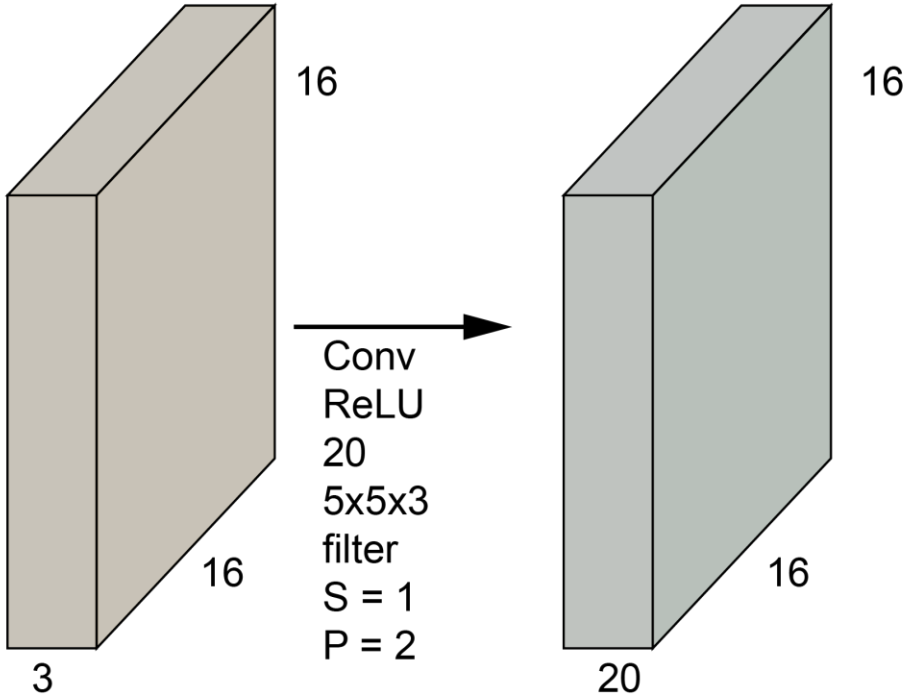
$$W^2 = \frac{W^1 + 2P - F}{S} + 1$$

$$H^2 = \frac{H^1 + 2P - F}{S} + 1$$

$$D^2 = N$$

Convolutional Layer

Example



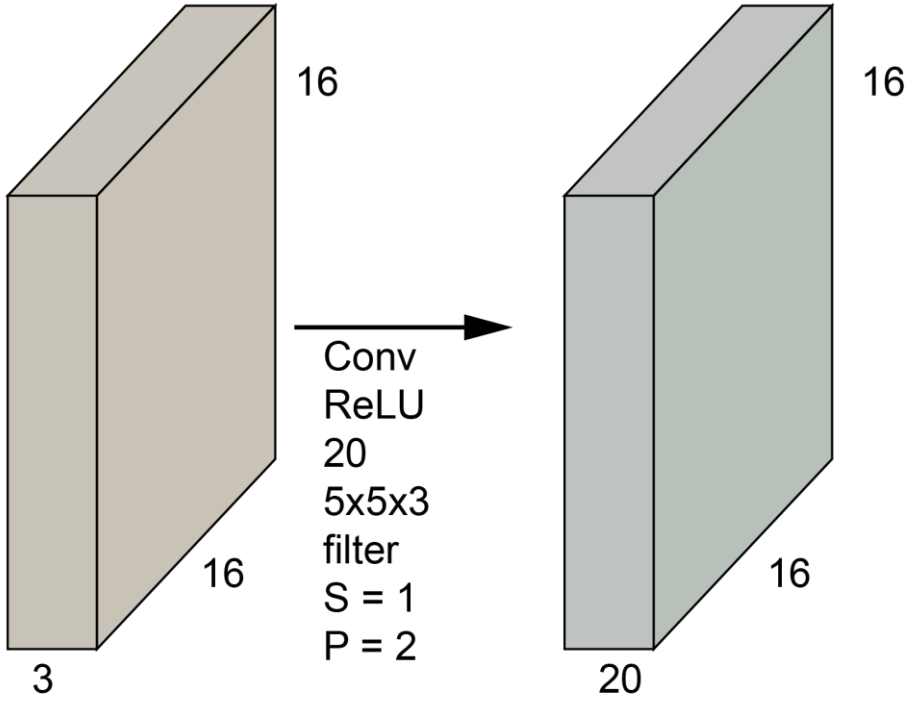
$$W^2 = \frac{16 + 2 \cdot 2 - 5}{1} + 1 = 16$$

$$H^2 = \frac{16 + 2 \cdot 2 - 5}{1} + 1 = 16$$

$$D^2 = 20$$

Convolutional Layer

Example



Number of parameters in this layer?

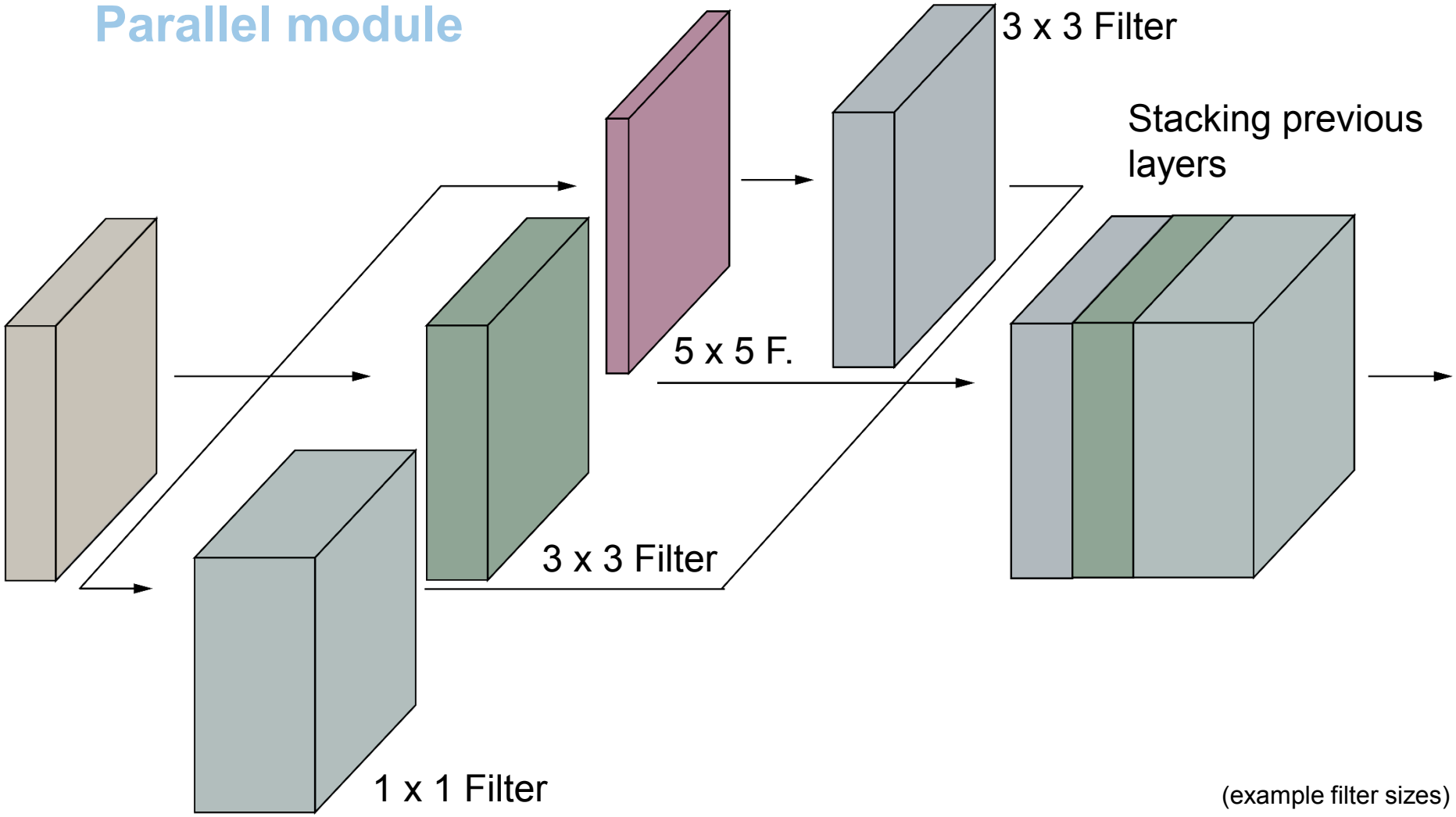
$5 \times 5 \times 3 + 1 = 76$ Parameters per filter

$20 \cdot 76 = 1520$ Parameters

In convolutional layer, the only parameters that are being learned are the weight values of the filter(s) with corresponding bias term(s).

Convolutional Layer

Parallel module



(example filter sizes)

We usually want to apply multiple different filters within the same layer to make sure that the network can learn to detect different object features.

Convolutional Layer

PyTorch Code Example

CLASS `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*`)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

PyTorch makes it really easy to code CNN layers.

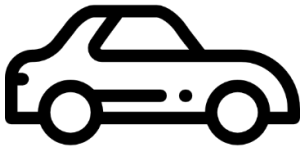
Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

Agenda

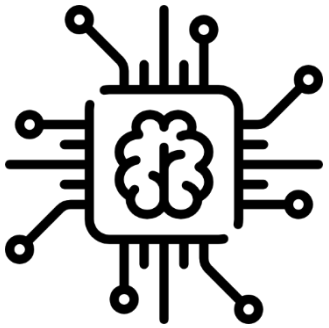
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride



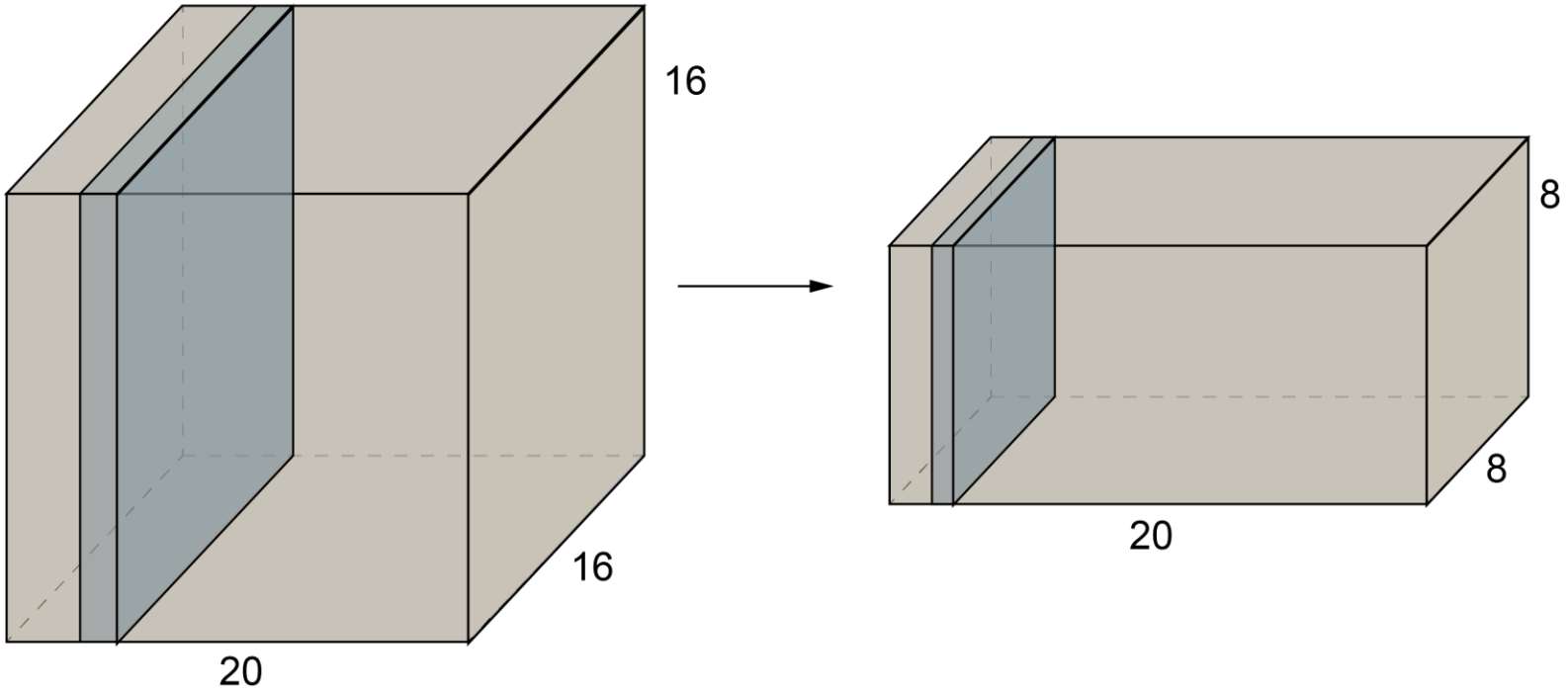
2. Chapter: Pooling

3. Chapter: CNN in Action

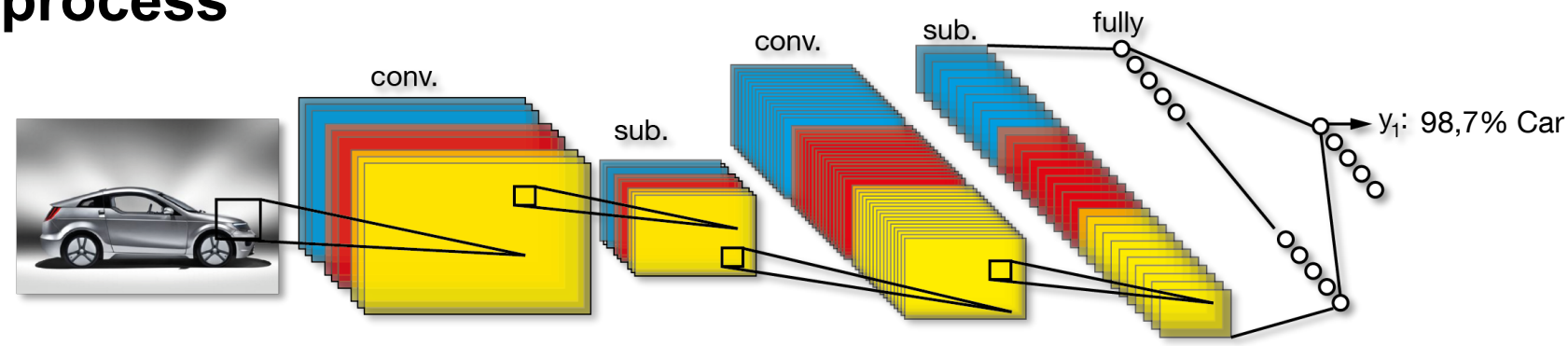


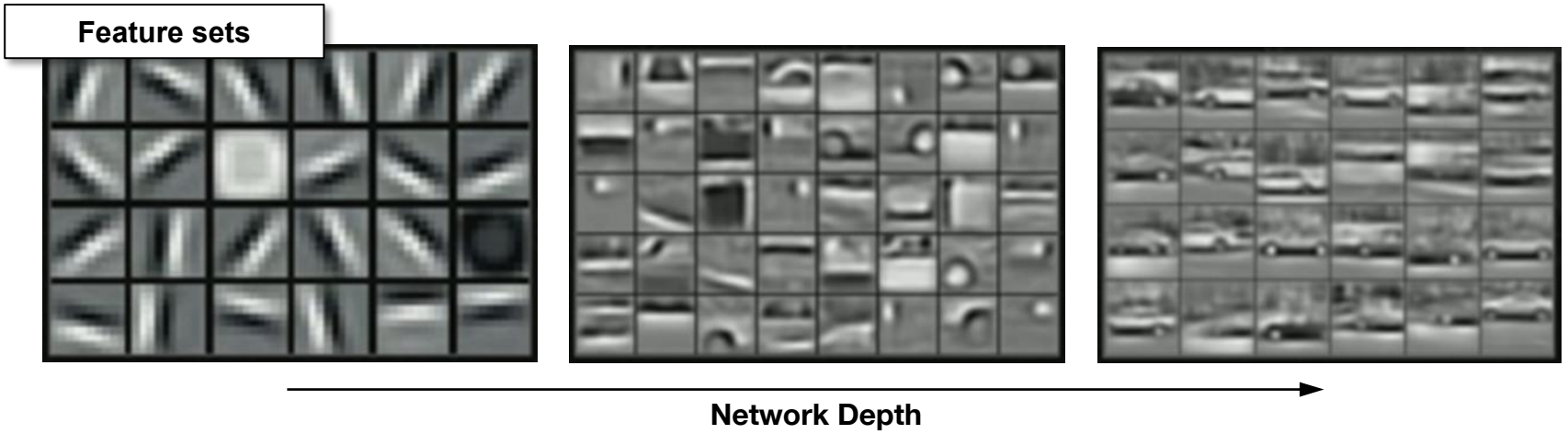
Pooling

- reduces the spatial output dimensions
- operates over each activation map independently



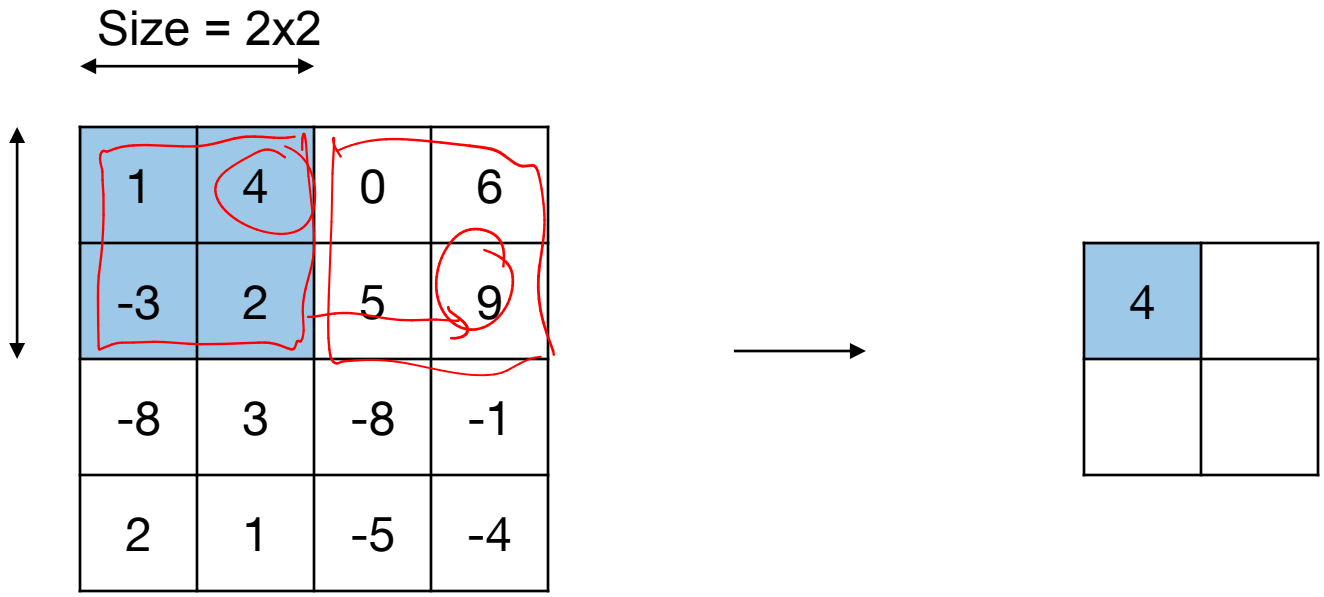
Pooling makes it possible to improve learning process





Pooling

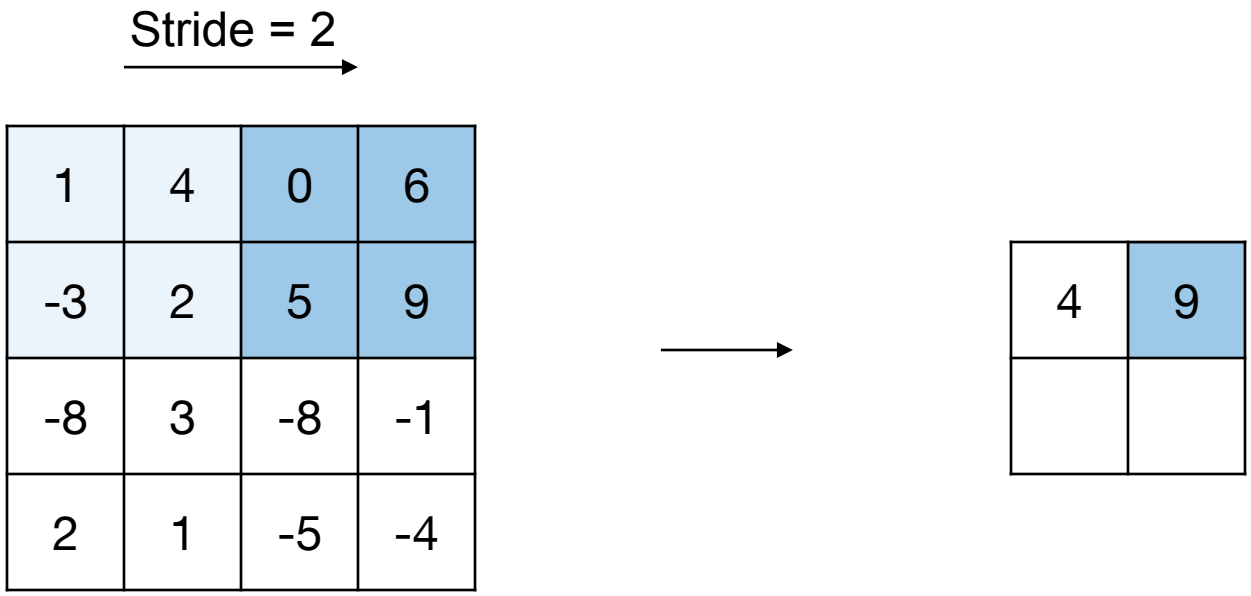
Max Pool



$$L^2_{i,j} = max(L^1_{i,j}, L^1_{i+1,j}, L^1_{i,j+1}, L^1_{i+1,j+1})$$

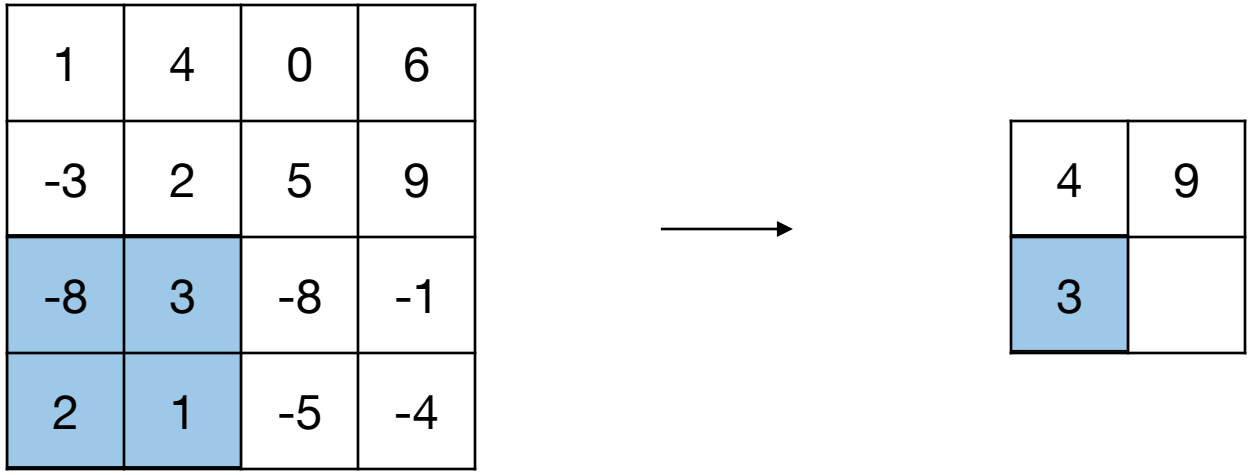
Pooling

Max Pool



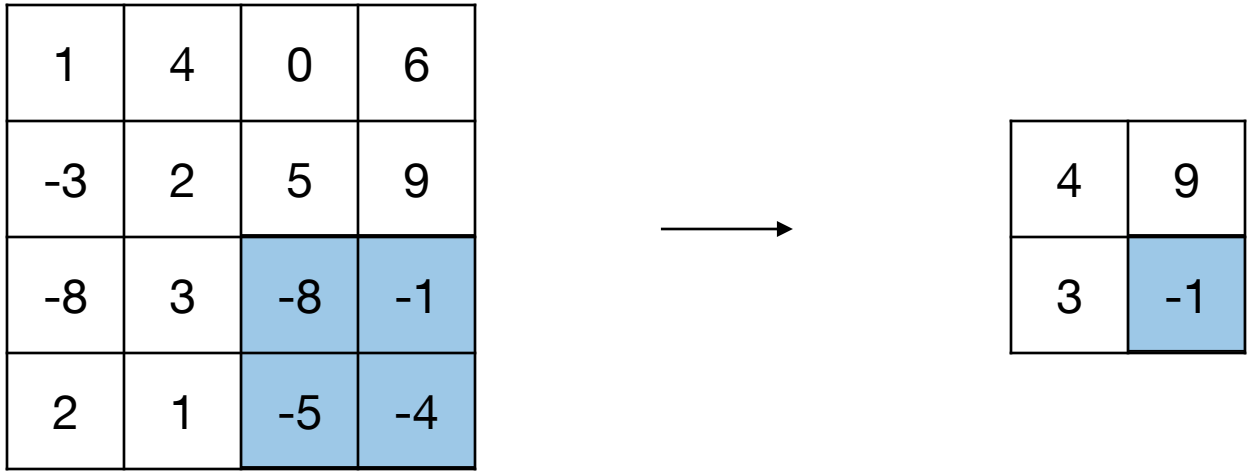
Pooling

Max Pool



Pooling

Max Pool



Pooling

Max Pool

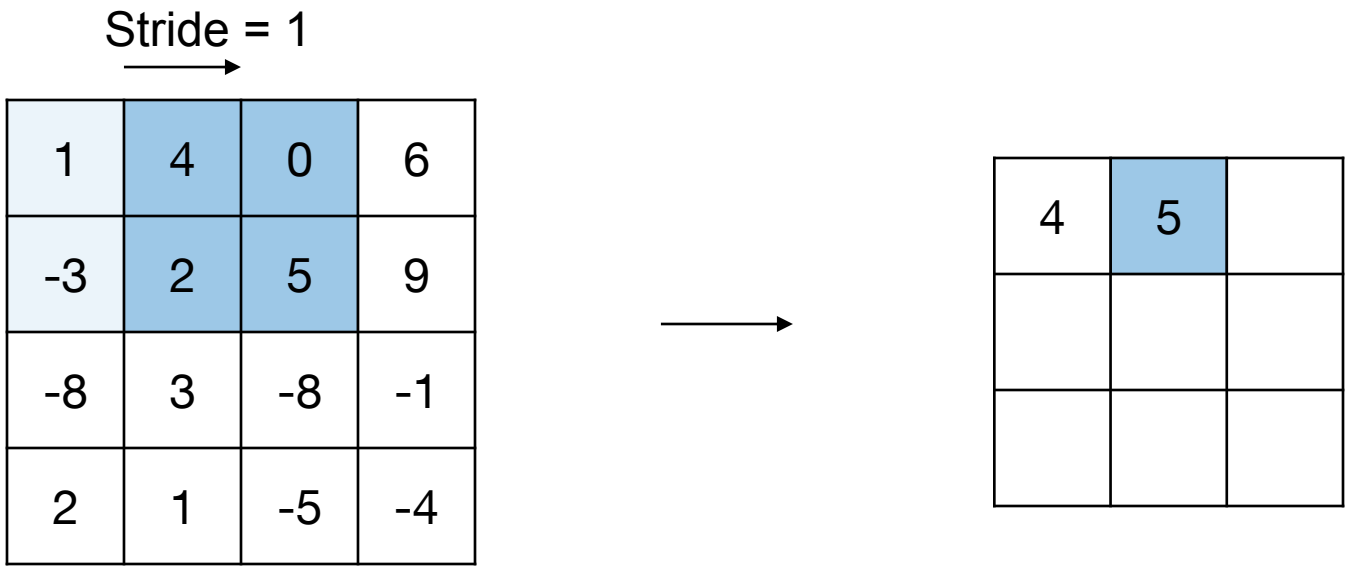
S = 1

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

4		

Pooling

Max Pool



Pooling

Max Pool

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

4	5	9
2	5	9
3	3	-1

Pooling

Average Pool

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

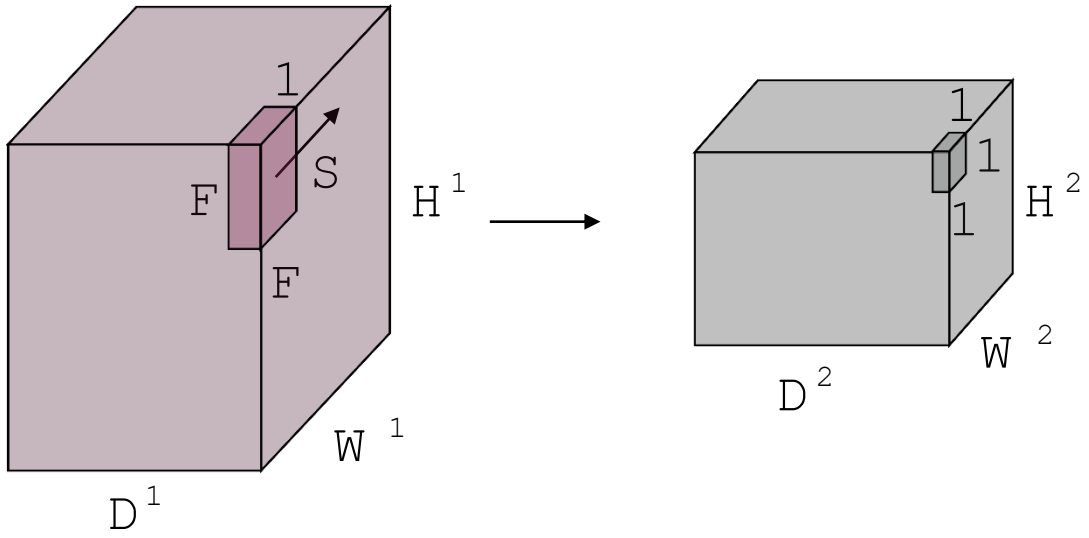
1	5
-0.5	-4.5

$$L^2_{i,j} = \frac{1}{4} \sum_{x=0}^{1} \sum_{y=0}^{1} L^1_{2i+x,2j+y}$$

There are different concepts for Pooling. MaxPool is typically a method of choice that produces good and stable results.

Pooling

Dimensions



$$L^1 \in K^{D^1 \times W^1 \times H^1} \Rightarrow L^2 \in K^{D^2 \times W^2 \times H^2}$$

New Dimensions:

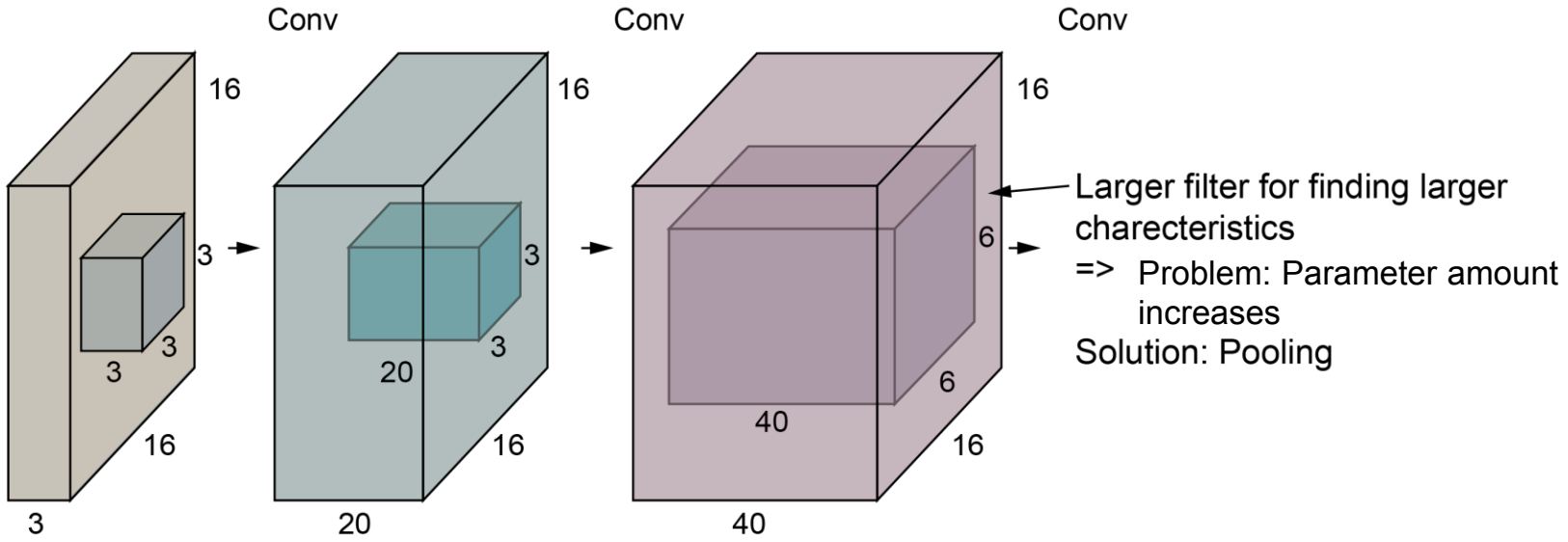
$$W^2 = \frac{W^1 - F}{S} + 1$$

$$H^2 = \frac{H^1 - F}{S} + 1$$

$$D^2 = D^1$$

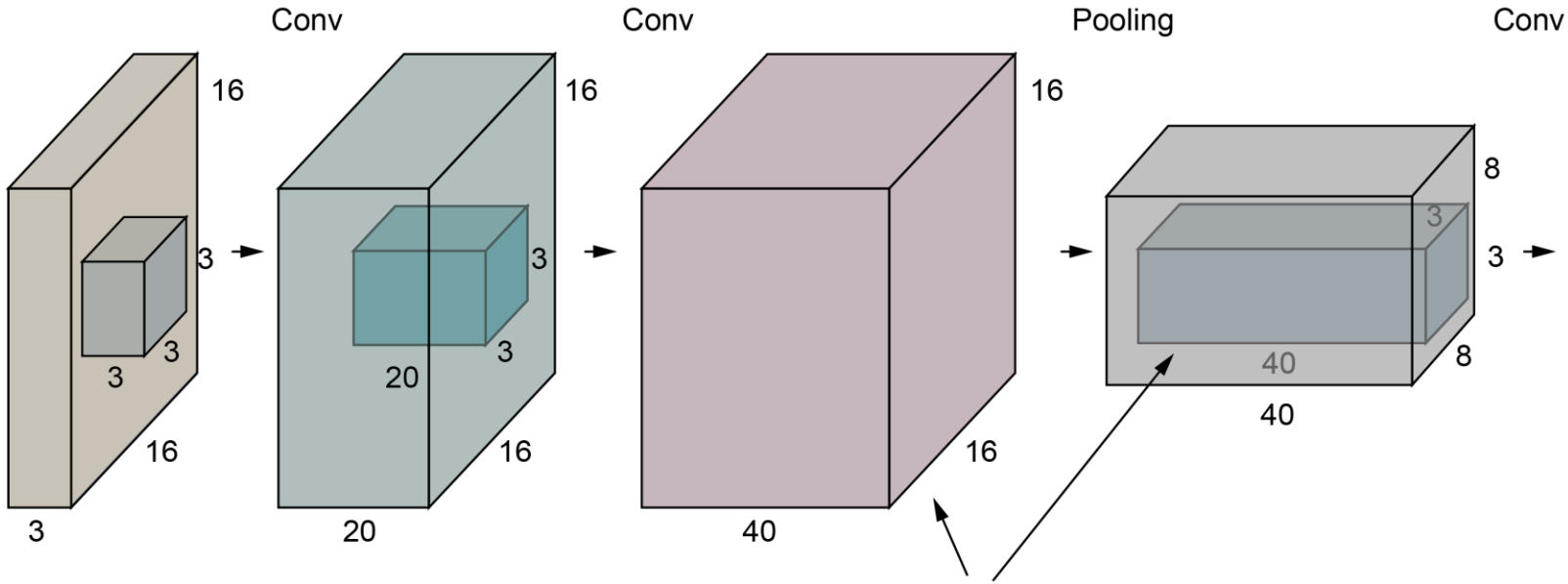
Pooling

With Convolution layer



Pooling

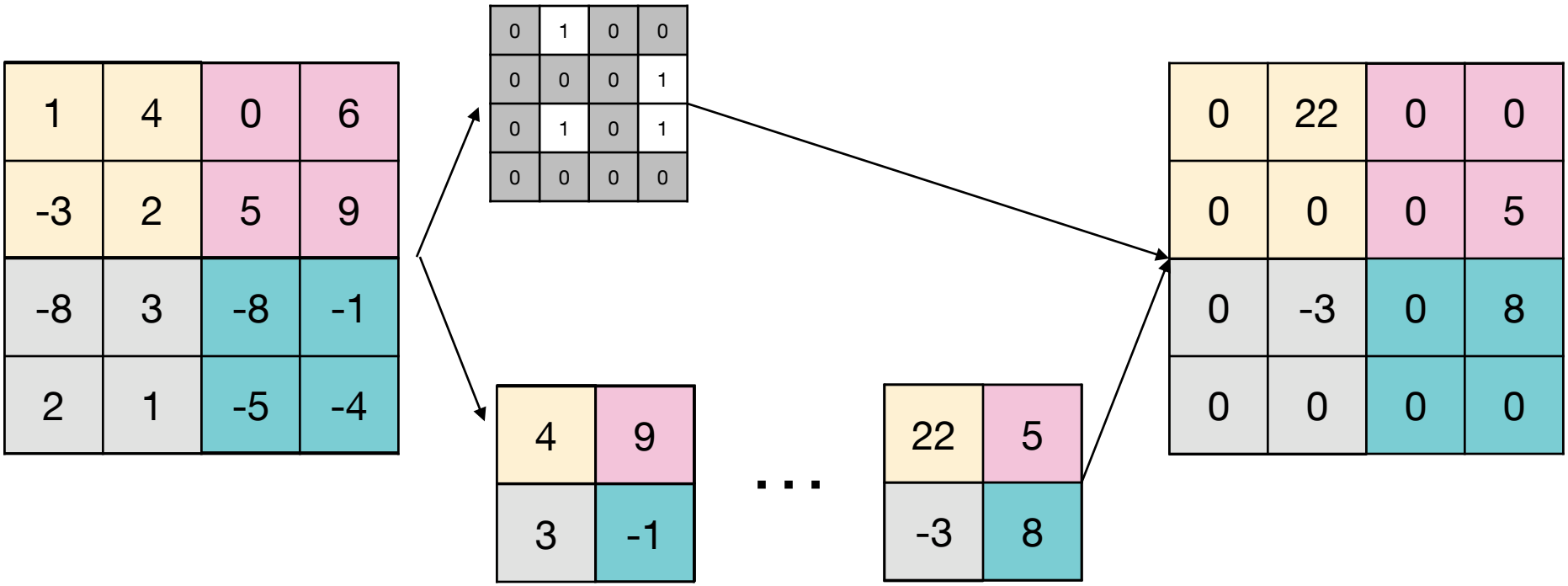
With Convolution layer



With the use of Pooling layers larger characteristics can be found with smaller filter sizes. Reducing the parameter amount in this example to 25%.

Unpooling

Max Unpool



Unpooling makes it possible to recover original dimensions. This is usually important for semantic segmentation where we want to classify each pixel of the original image. Using unpooling we can restore the original dimension and apply our network to predict and produce result for the input image with full resolution.

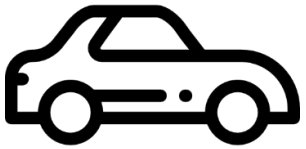
Convolutional Neural Networks
Prof. Dr. Markus Lienkamp

(Domagoj Majstorović, M. Sc.)

Agenda

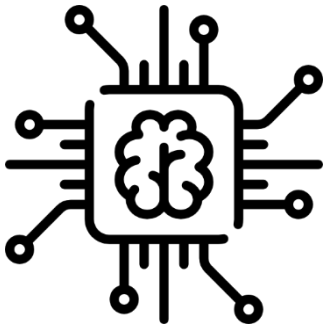
1. Chapter: Convolutional Neural Networks

- 1.1 Motivation
- 1.2 Introduction
- 1.3 Dimensions, Padding, Stride

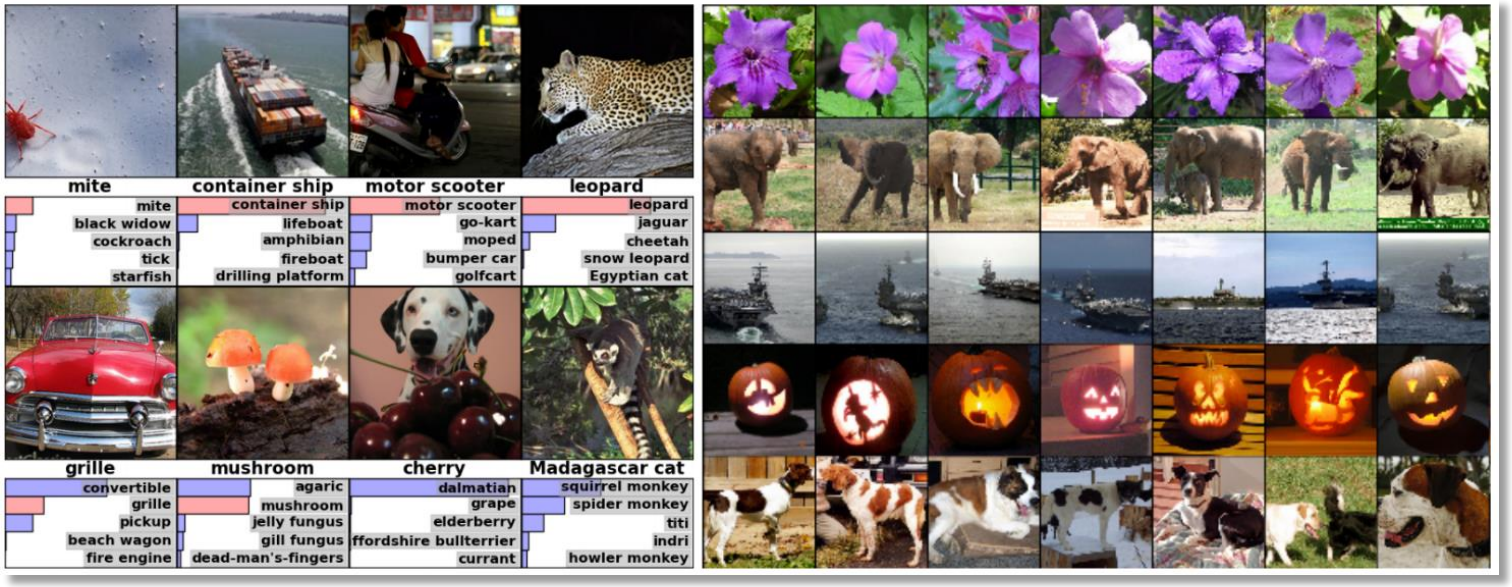


2. Chapter: Pooling

3. Chapter: CNN in Action

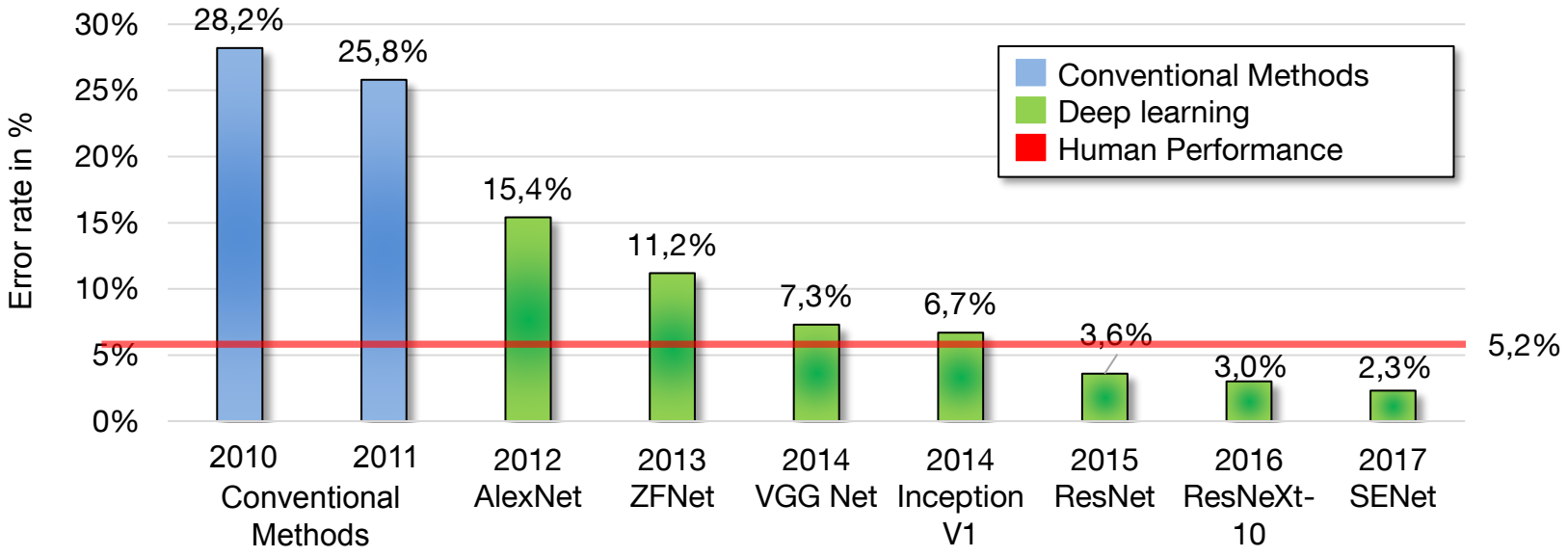


ImageNet Competition



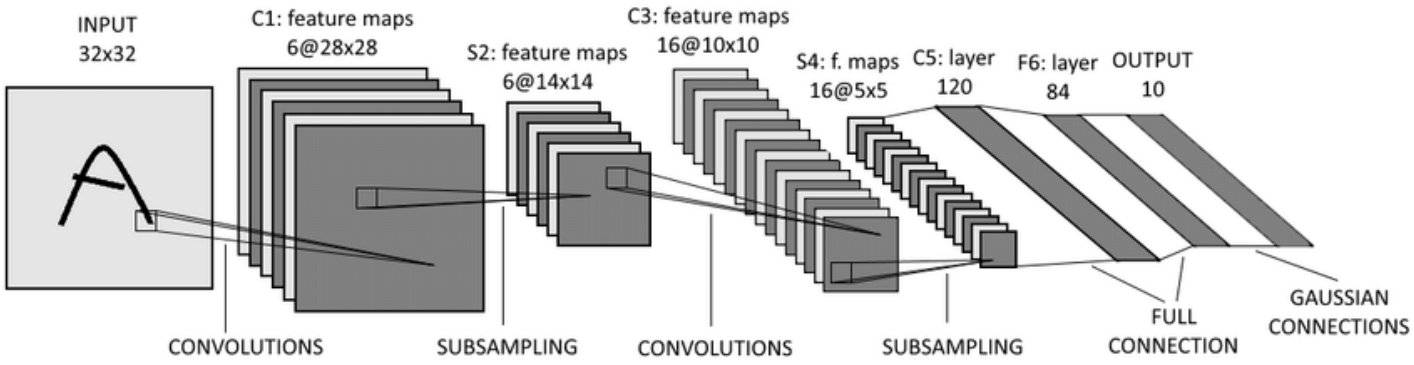
- ImageNet is a database that contains 14M images with more than 20000 object classes. Due to its size, it is suitable benchmark tool for Deep Neural Networks.

ImageNet Competition



Popular Architectures

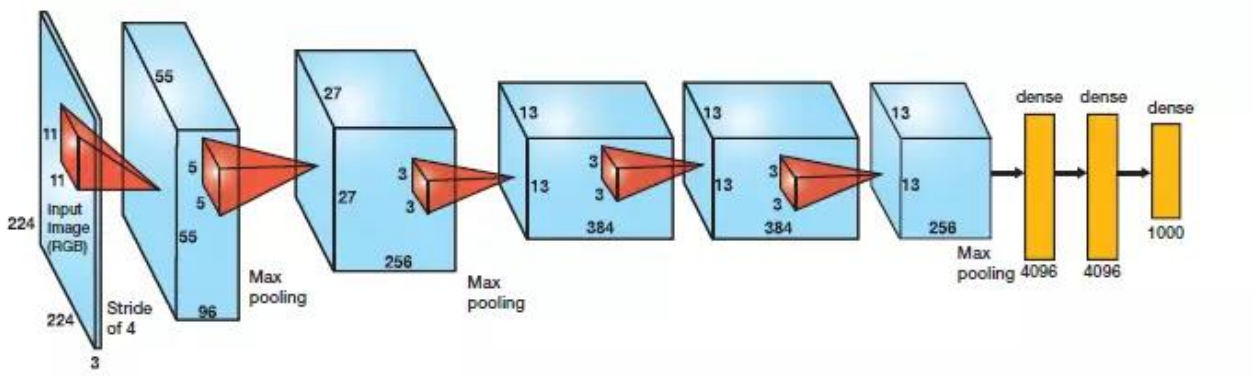
LeNet – Recognizing simple digit images



- Every convolutional layer includes three parts: convolution, pooling, and nonlinear activation functions
- Using convolution to extract spatial features (Convolution was called receptive fields originally)
- Subsampling average pooling layer
- TanH activation function
- Using MLP as the last classifier
- Sparse connection between layers to reduce the complexity of computation

Popular Architectures

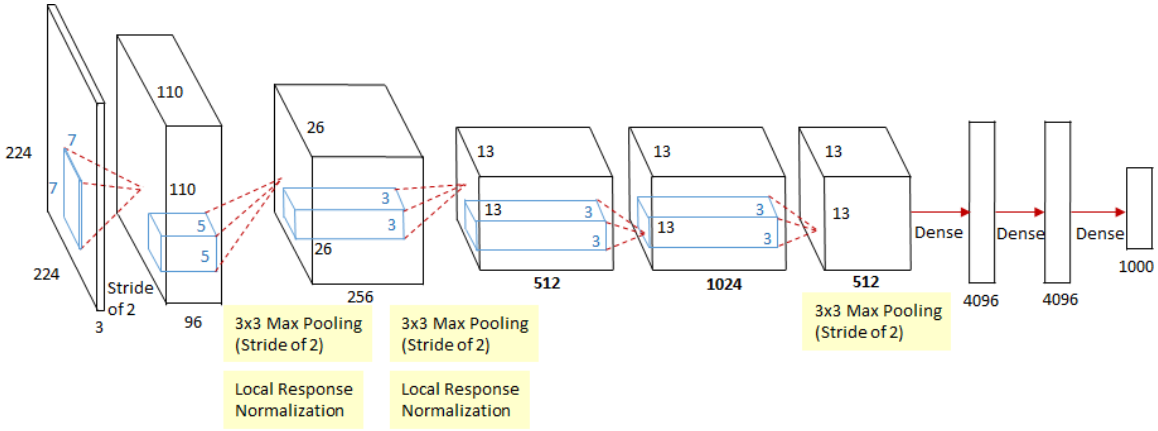
AlexNet – Image Classification



- AlexNet competed and won the ImageNet Large Scale Visual Recognition Challenge back in 2012 and achieved a Top-5 error of 15.3%.
- AlexNet contained eight layers:
  - The first five were convolutional layers, some of them (3) followed by max-pooling layers, and the last three were fully connected layers (2 FC + Softmax)
- It used the non-saturating ReLU activation function, which showed improved training performance over TanH and Sigmoid. SGD + Momentum as a learning algorithm.
- Computationally expensive (62M parameters), but feasible due to the utilization of graphics processing units (GPUs) during training.

Popular Architectures

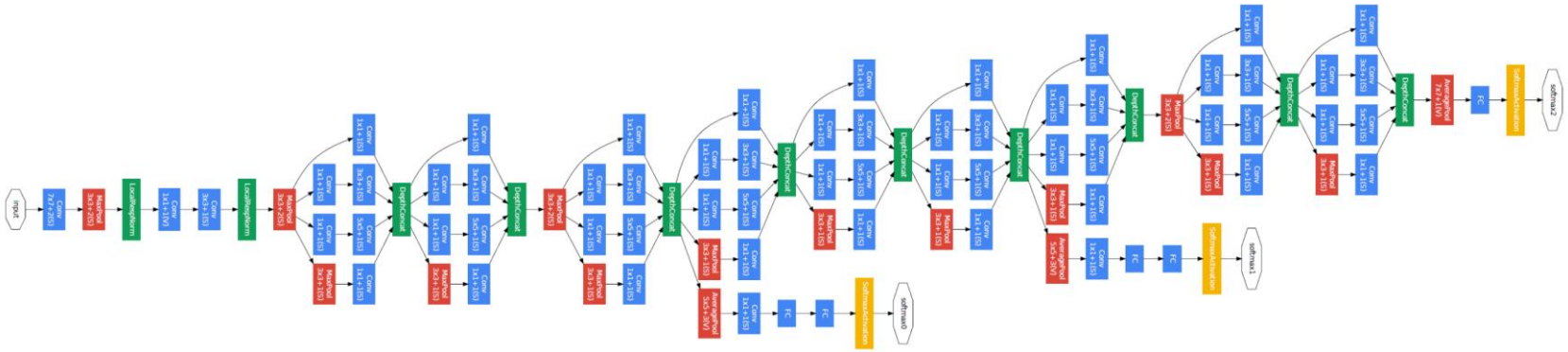
ZFNet – Image Classification



- ZFNet competed and won the ImageNet Large Scale Visual Recognition Challenge back in 2013.
- Significantly improved the image classification error rate compared to AlexNet
- The architecture of the network is an optimized version of AlexNet
  - Filter size in the first convolutional layer was reduced (11x11 to 7x7) and the overall number of filters in all convolutional layers has been doubled, as well as the number of neurons in the FC layers.
  - They decreased the error rate from 15.4% to 14.8% and won the competition.-

Popular Architectures

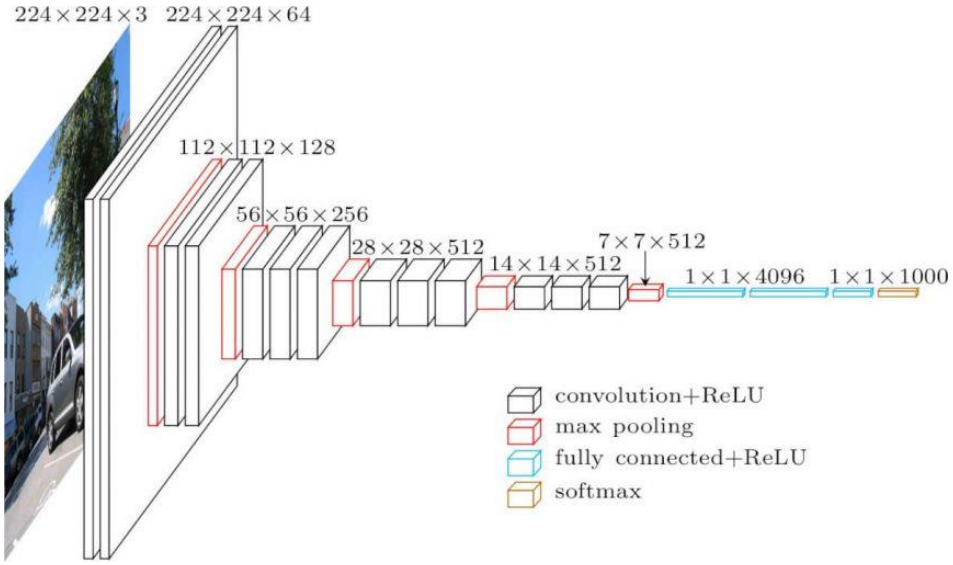
GoogLeNet – Image Classification



- GoogLeNet competed and won ImageNet competition back in 2014.
- Also known as Inception V1
- GoogLeNet contained 22 layers:
  - Introduced the *inception module*
  - Introduced *1x1 convolution* and *global average pooling* instead of FC layers
- It contained almost 7M parameters.

Popular Architectures

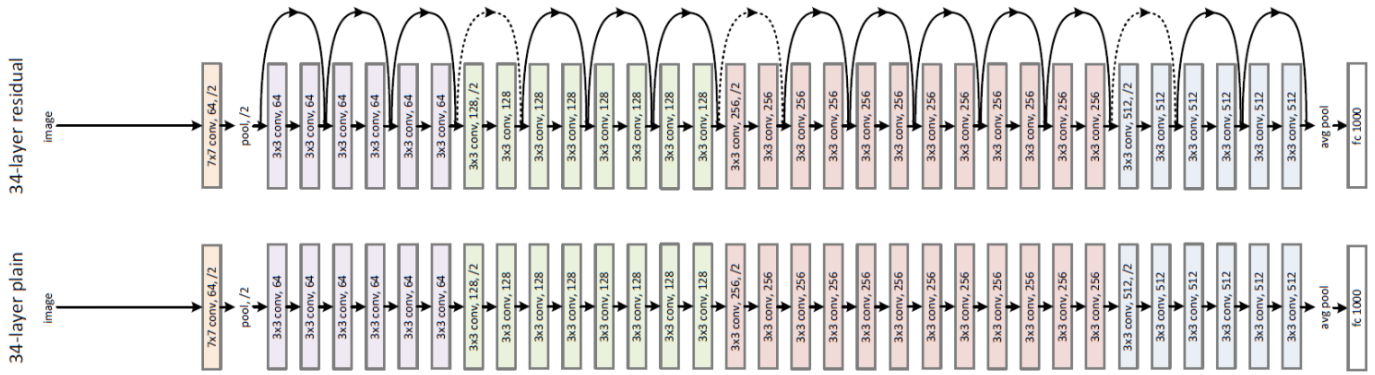
VGGNet – Image Classification



- VGGNet competed and won a second place in ImageNet competition back in 2014.
- 2 Versions: VGG-16 or VGG-19 consisting of 16 (13 ConvL + 3 FCL) and 19 convolutional layers.
- Simple architecture, yet a huge network with 138M Parameters (VGG-16)

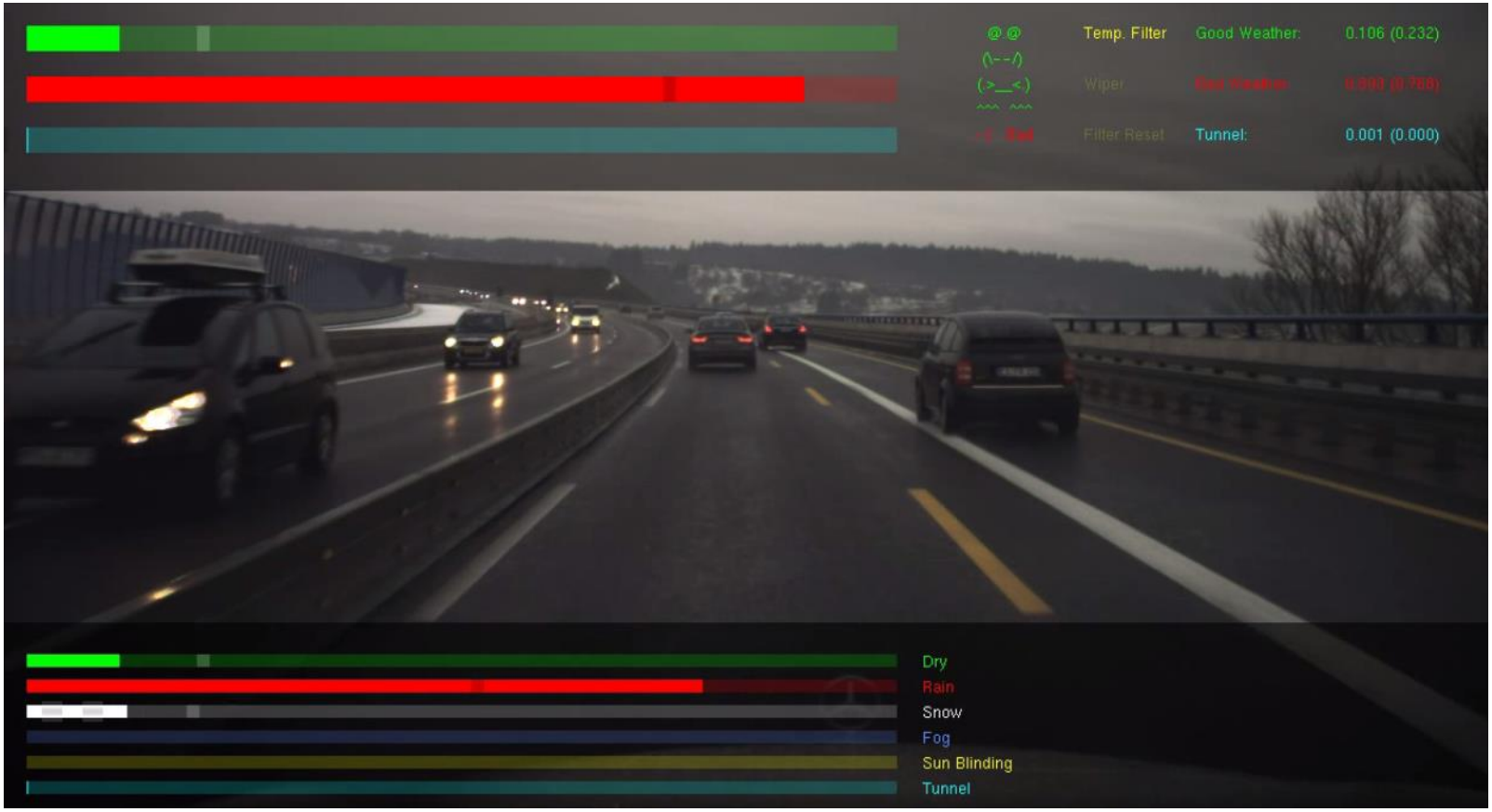
Popular Architectures

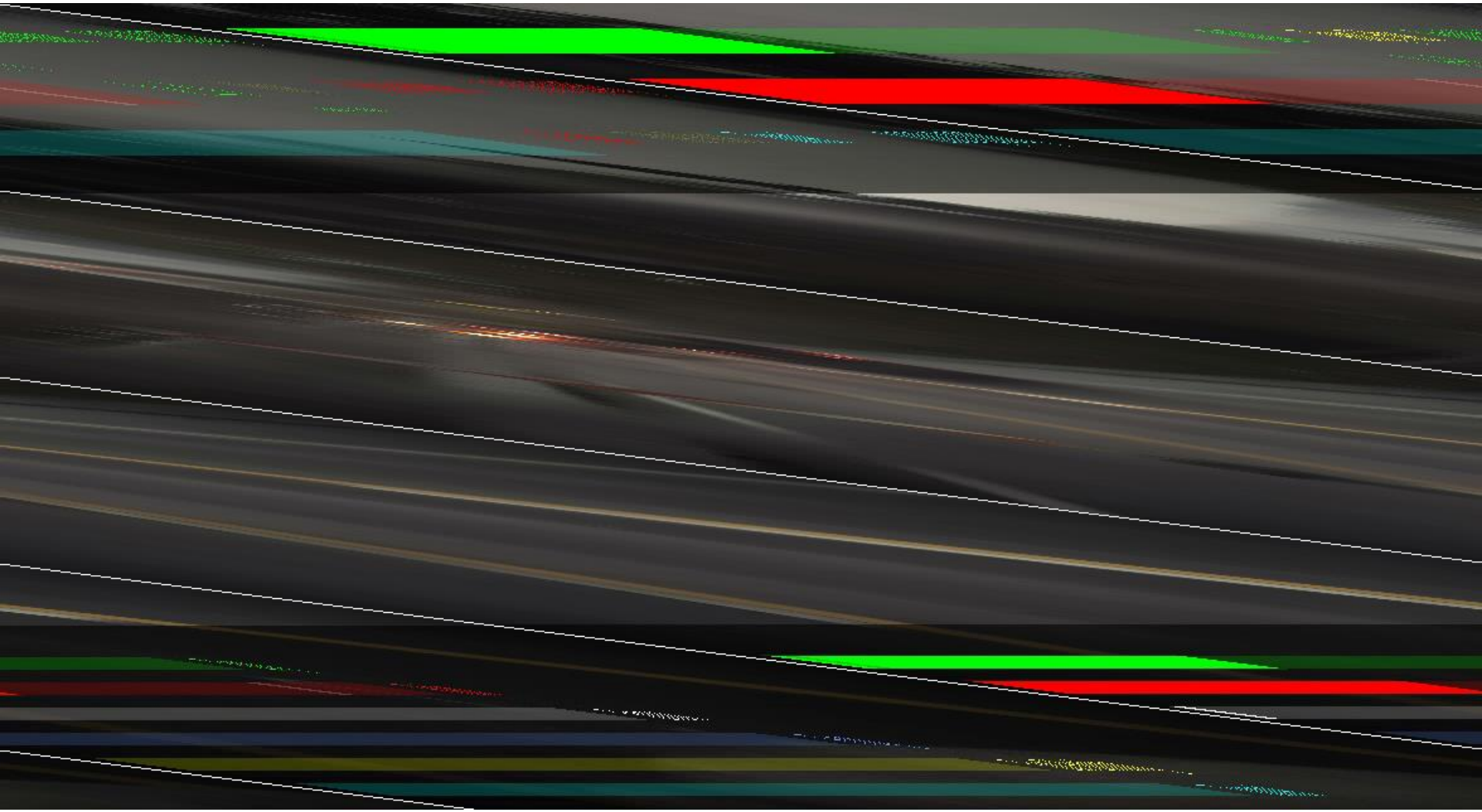
ResNet – Image Classification



- ResNet competed and won ImageNet competition back in 2015.
- A very deep network – up to 152 layers with the “bottleneck” design.
- Introduces “*skip connection*” (or *shortcut connection*) to fit the input from the previous layer to the next layer without any modification of the input. This made possible to have a very deep network while avoiding the vanishing gradient problem.
- With a depth of 152 layers, the network achieved the Top-5 error rate of 5.71%.

Daimler: Weather Prediction

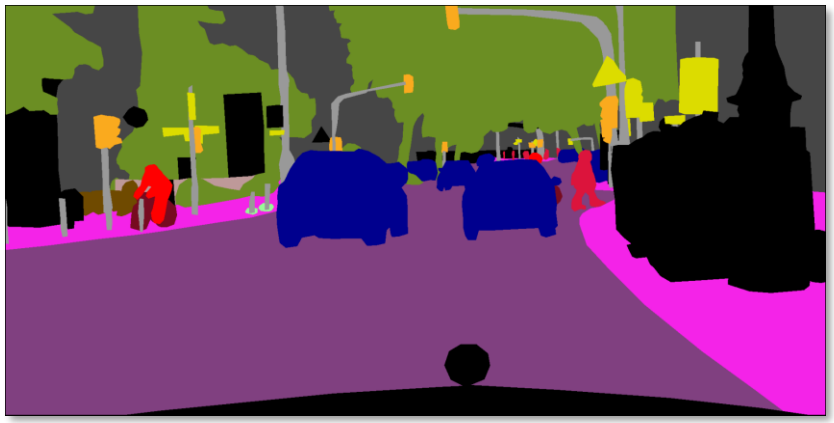




Daimler: Cityscapes Dataset

x

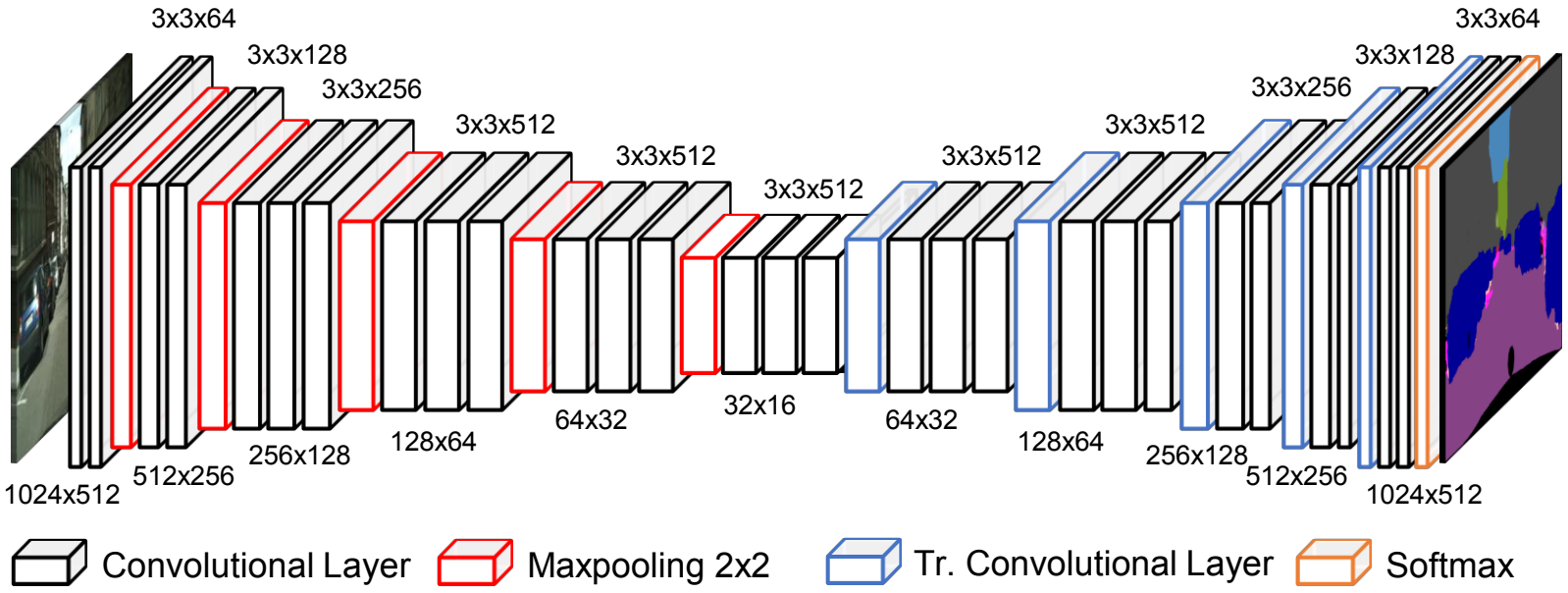
$\hat{y}$



Sample Scene

Sample Scene

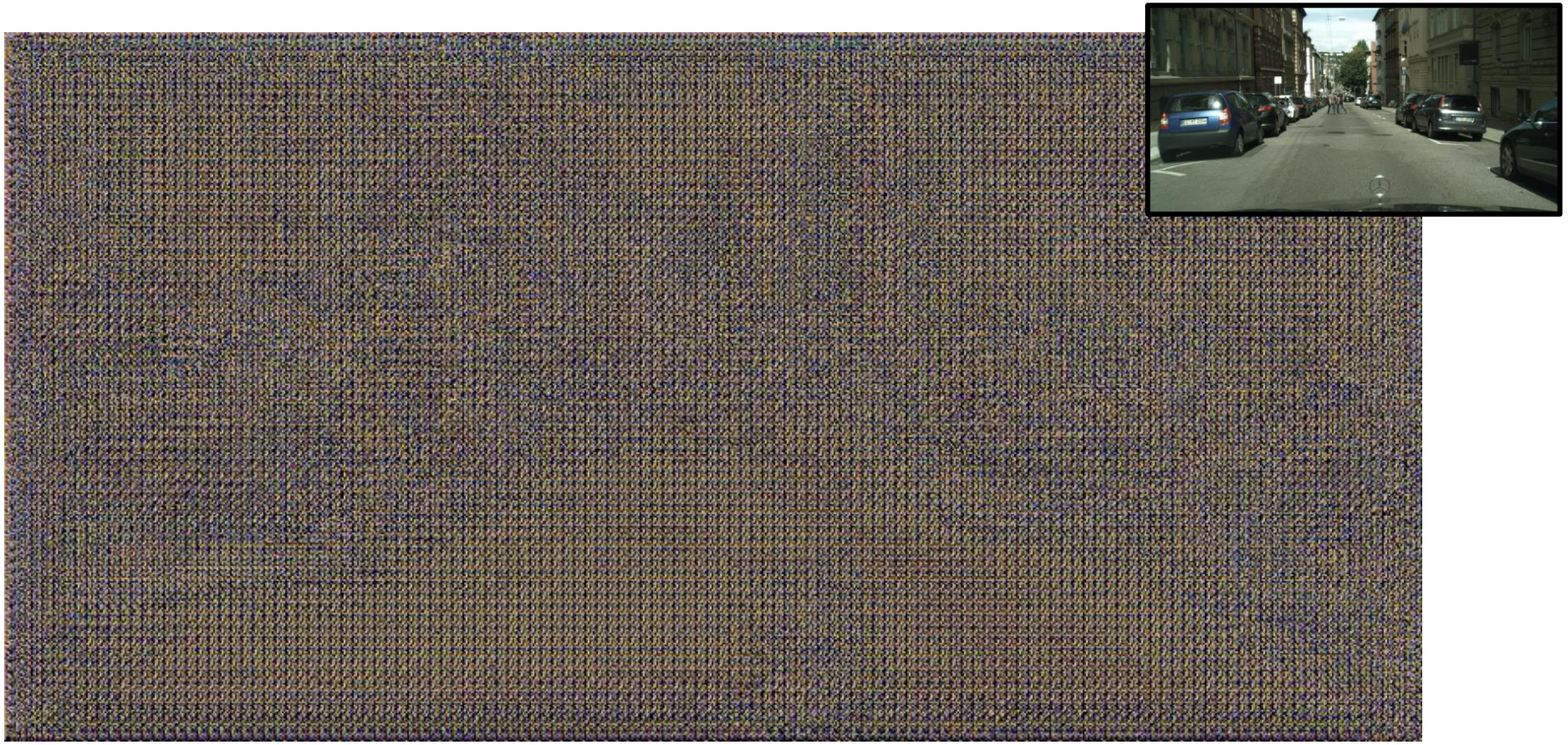
CNN-Structure



This is an example of the semantic segmentation, where each pixel of the input image is classified w.r.t. one of the expected classes.

Cityscapes FCN: Learning Process

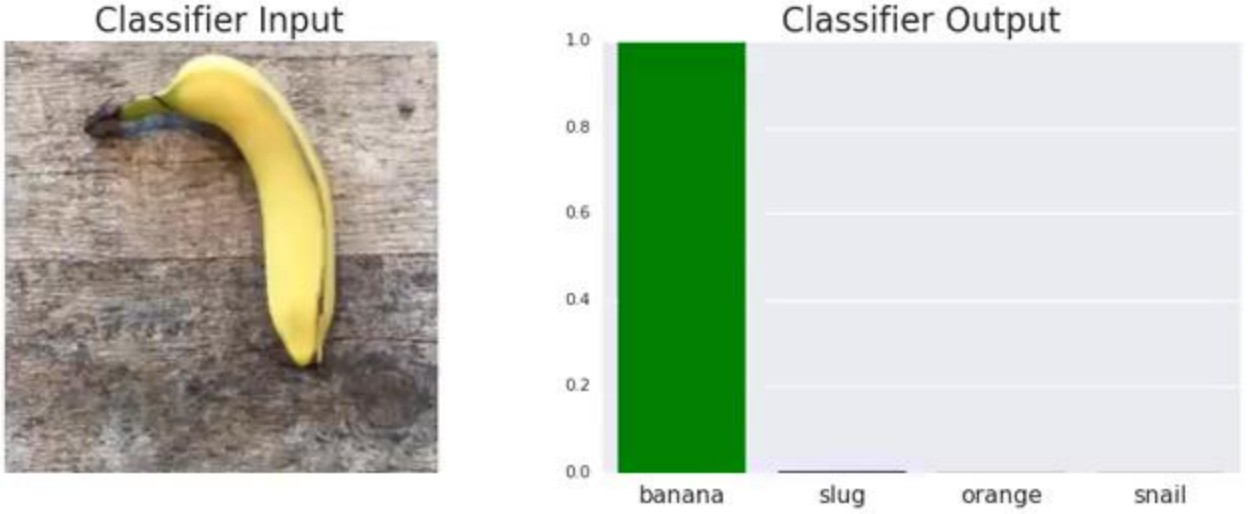
Cityscapes CNN: Learning Process



Galaxy S7 Results

Nvidia: Semantic Segmentation

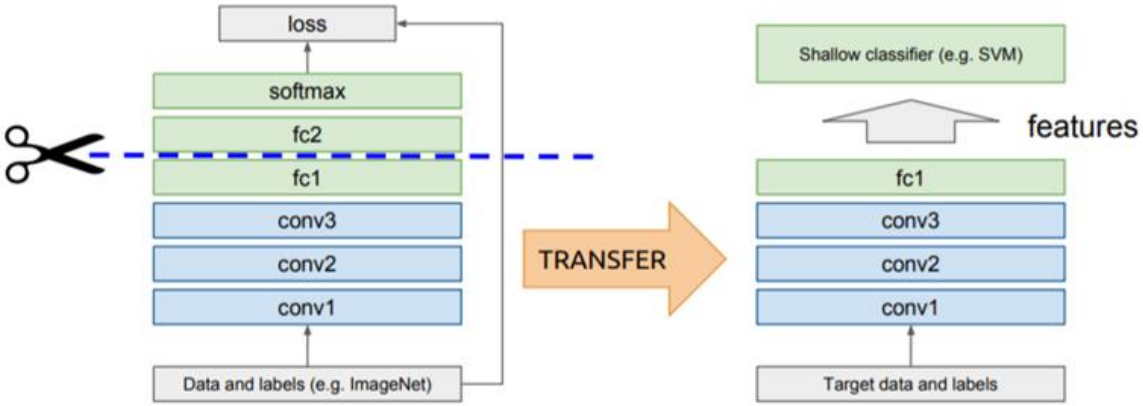
Video: Adversial Patch



Adverserial Attacks are currently a hot topic since this field deals with different safety concerns. One method how deal with this issue is to run redundant networks at the same time and analyze the result.

Transfer Learning

- Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.
- The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.



Transfer learning is a very useful concept where a pretrained network can be used as a baseline, and the latter part of the network is designed to be tuned for a new application.

Summary

What we learned today:

- Why are convolutional neural networks useful
- How standard CNN architecture looks like
- How to calculate network dimensions
- How to use different layer types to improve the learning performance
- What are the most popular architectures today

End