KÜNSTLICHE NEURONALE NETZE & CONV-NETS

KÜNSTLICHE NEURONALE NETZE

(KNN)

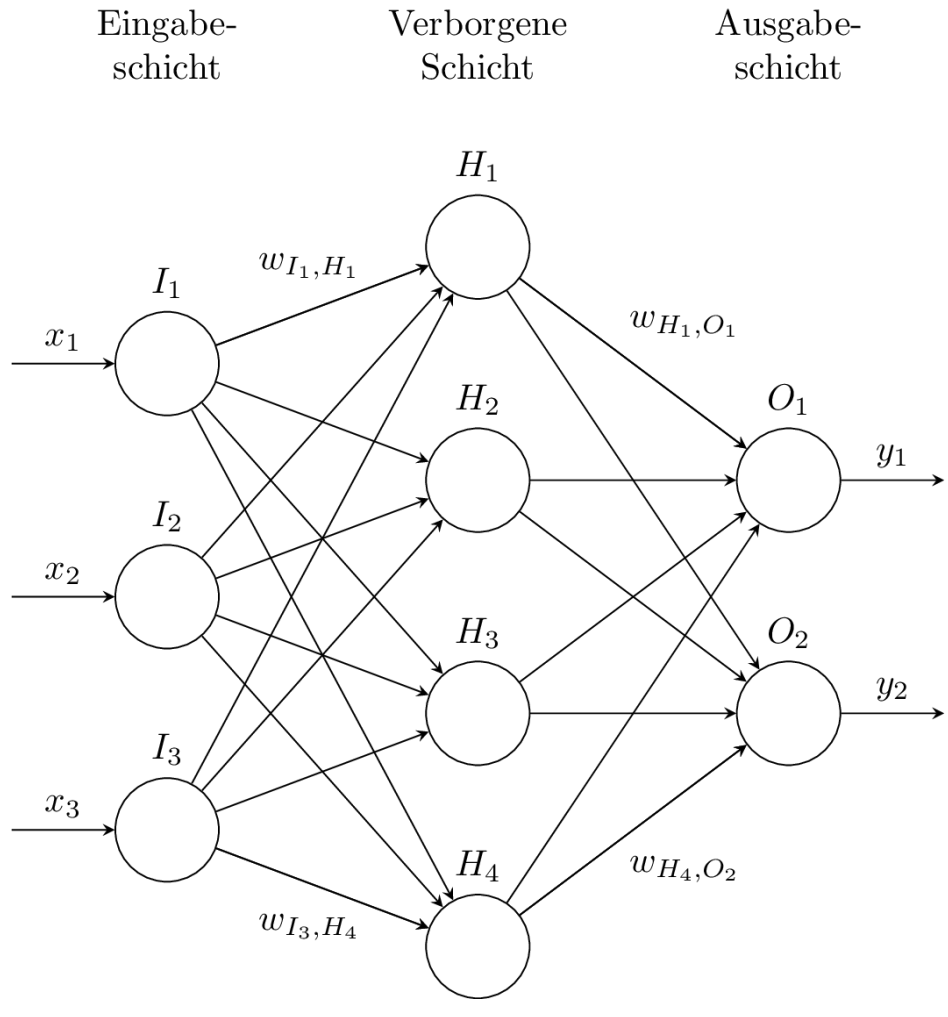
Modellieren **nicht** das Gehirn

Nur inspiriert von natürlichen NN

Bestehen aus *Schichten von Neuronen*

Approximieren Funktionen

Mustererkennung, Sprachsynthese, Textsynthese, etc.


Eingabe-
schicht
Verborgene
Schicht
Ausgabe-
schicht
H_1
w_{I_1,H_1}
I_1
x_1
w_{H_1,O_1}
H_2
O_1
y_1
I_2
x_2
H_3
O_2
y_2
I_3
x_3
H_4
w_{H_4,O_2}
w_{I_3,H_4}

KNN - APPROXIMATION VON FUNKTIONEN

Gesucht: $y = f^*(x)$

Gefunden: $y \approx f(x, \theta) := f^3(f^2(f^1(x, \theta_1), \theta_2), \theta_3)$

$f^k = k$-te Schicht

Lernen: θ finden

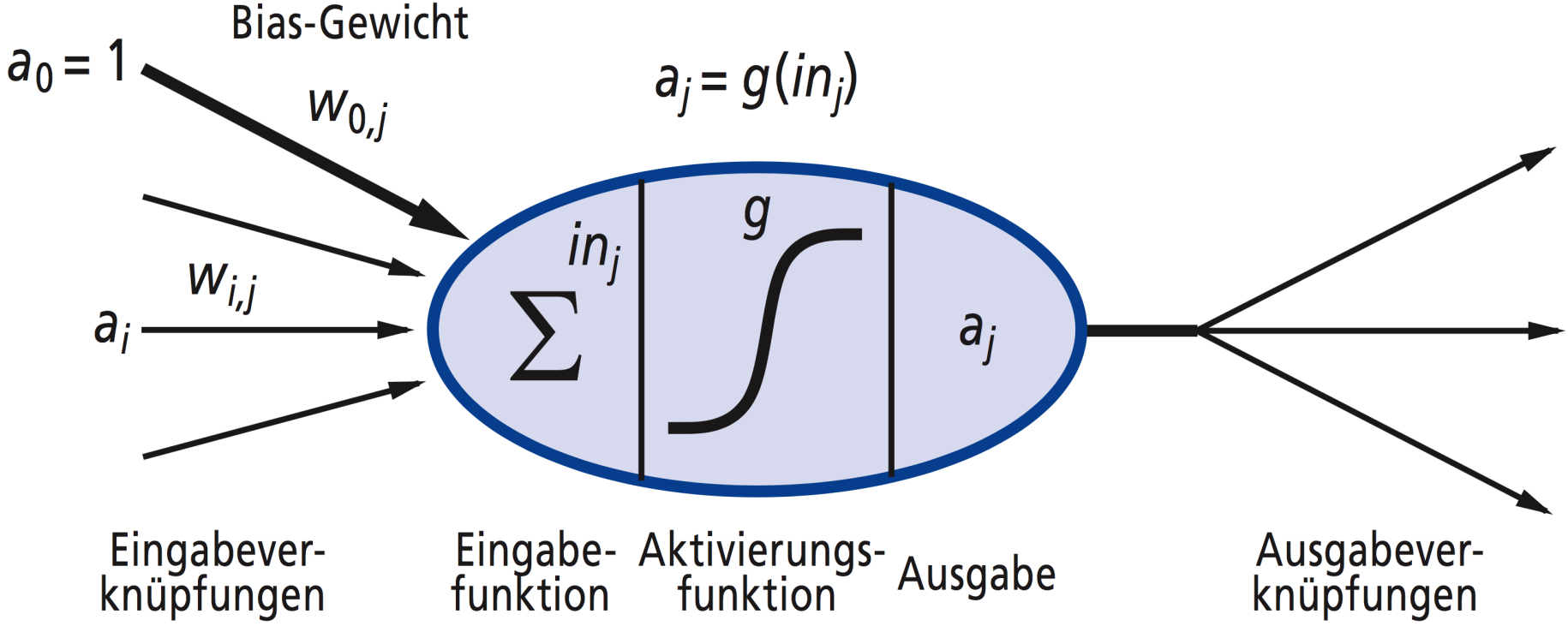
KNN - NEURON

Auch “Einheit” oder “Knoten”

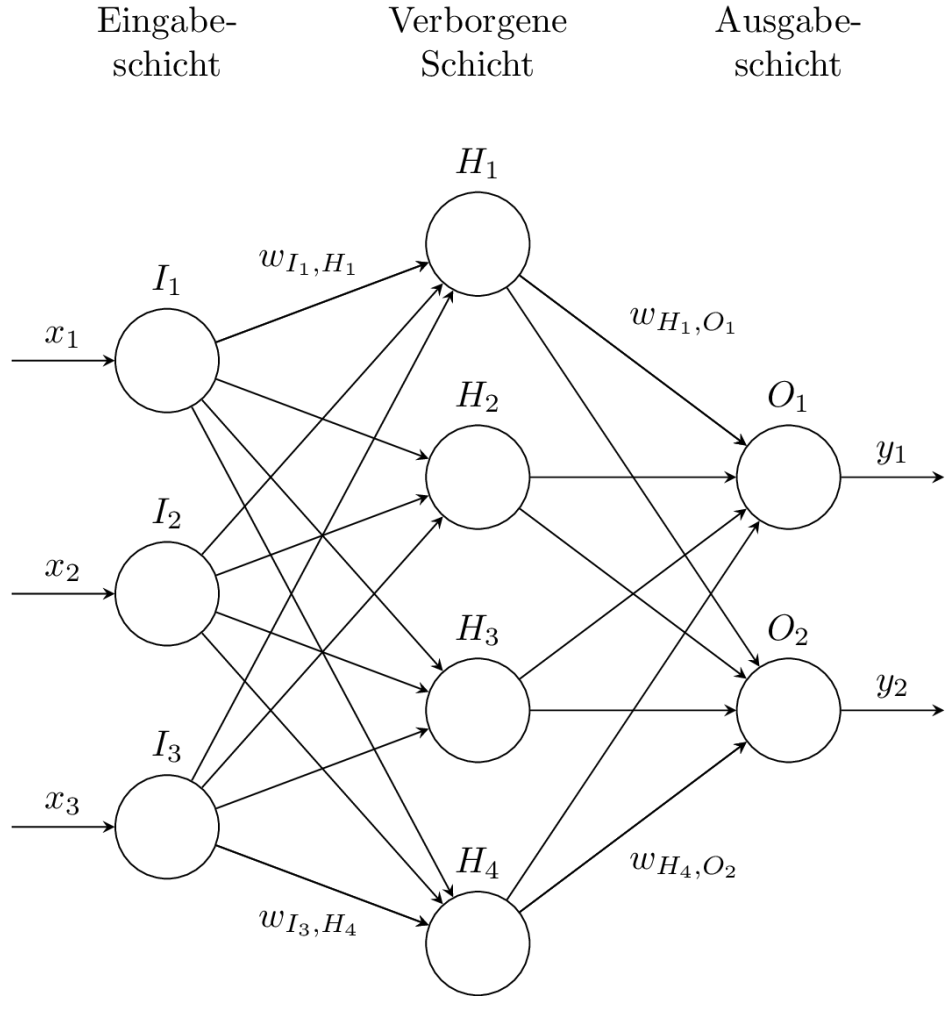
Inspiriert von natürlichen Neuronen

KNN - NEURON

[Russel & Norvig, 2012]



$$a_j = g(\sum w_{i,j} a_i)$$


Eingabe-
schicht
Verborgene
Schicht
Ausgabe-
schicht
H_1
w_{I_1,H_1}
I_1
x_1
w_{H_1,O_1}
H_2
O_1
y_1
I_2
x_2
H_3
O_2
y_2
I_3
x_3
H_4
w_{H_4,O_2}
w_{I_3,H_4}

KNN - LERNEN

$$y \approx f(x, \theta)$$

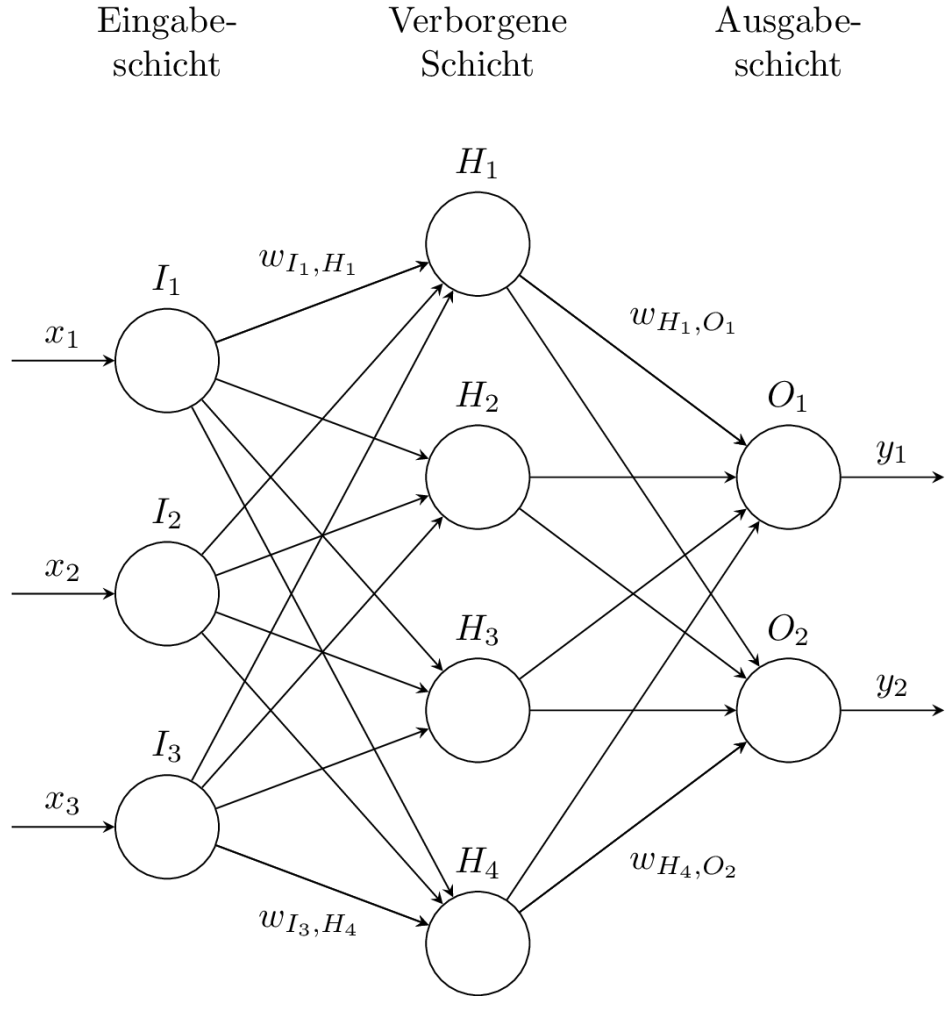
θ mit niedrigen zufälligen Werten initialisiert

Gradient Descent

Sehr aufwändiger Prozess → GPUs!

BACKPROPAGATION ALGORITHMUS

Führt Fehler der Ausgabe
auf verborgene Schichten **zurück**


Eingabe-
schicht
Verborgene
Schicht
Ausgabe-
schicht
H_1
w_{I_1,H_1}
I_1
x_1
w_{H_1,O_1}
H_2
O_1
y_1
I_2
x_2
H_3
O_2
y_2
I_3
x_3
H_4
w_{H_4,O_2}
w_{I_3,H_4}

BACKPROPAGATION ALGORITHMUS

$$\Delta_k = g'(in_j) \times (t_j - y_j)$$

$$\Delta_j = g'(in_j) \times \sum_k w_{j,k} \Delta_j$$

$$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta_j$$

KNN - LERNEN

Supervised Learning

Unsupervised Learning

Reinforcement Learning

KNN - DEEP LEARNING

KNN mit vielen Schichten

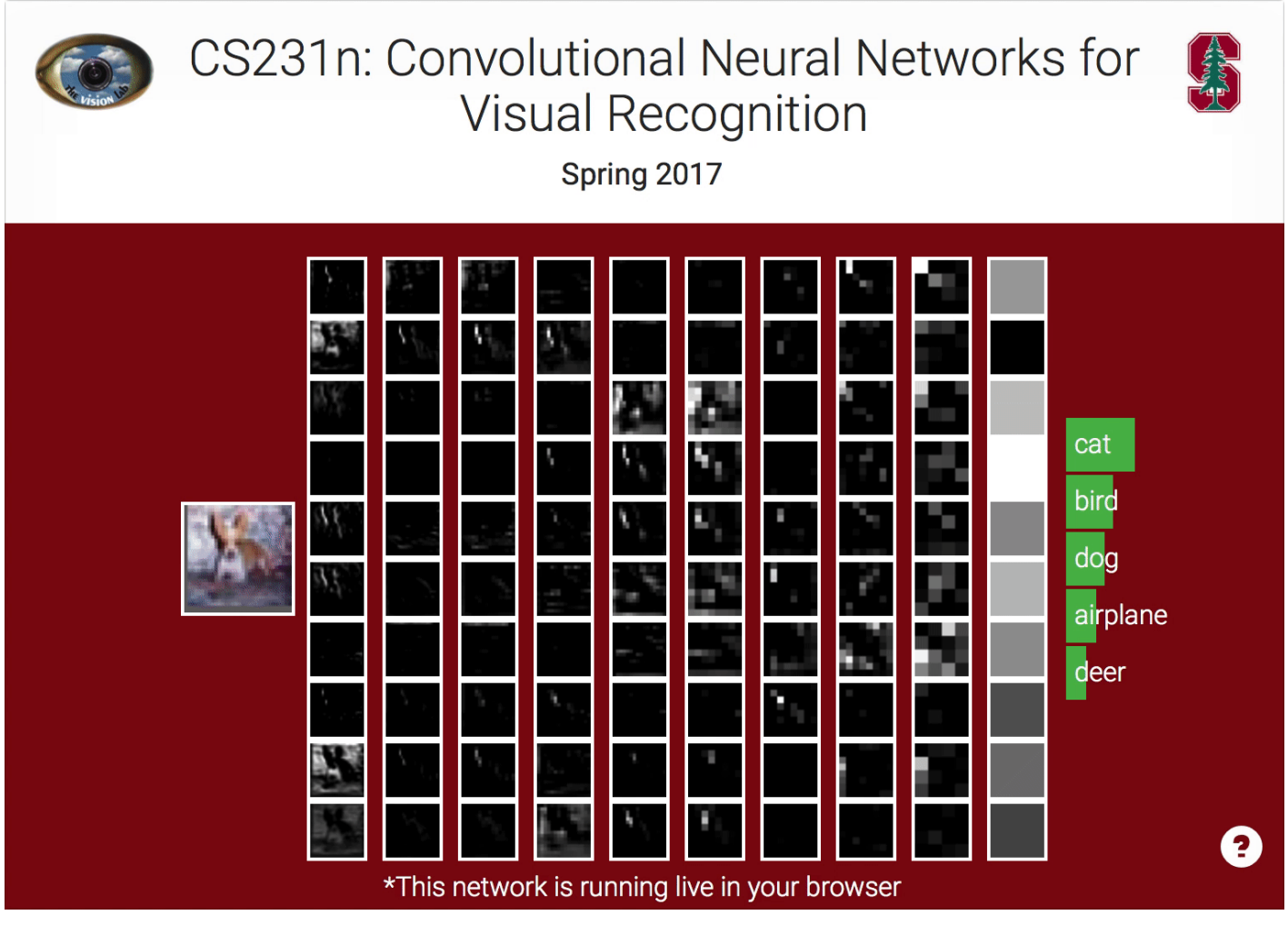
Overfitting Gefahr

“Entwurfsmuster” wie CNNs, RNNs, etc.

CONVOLUTIONAL NEURAL NETWORKS

[LeCun et al., 1993; http://youtu.be/FwFduRA_L6Q, 2017]

[http://cs231n.stanford.edu, 2017]



[http://cs231n.stanford.edu, 2017]

CONVOLUTIONAL NEURAL NETWORKS

(CNN)

“Gitterartige” Inhalte

Verwendet *Convolution*

Ansonsten ähnlich zu KNN

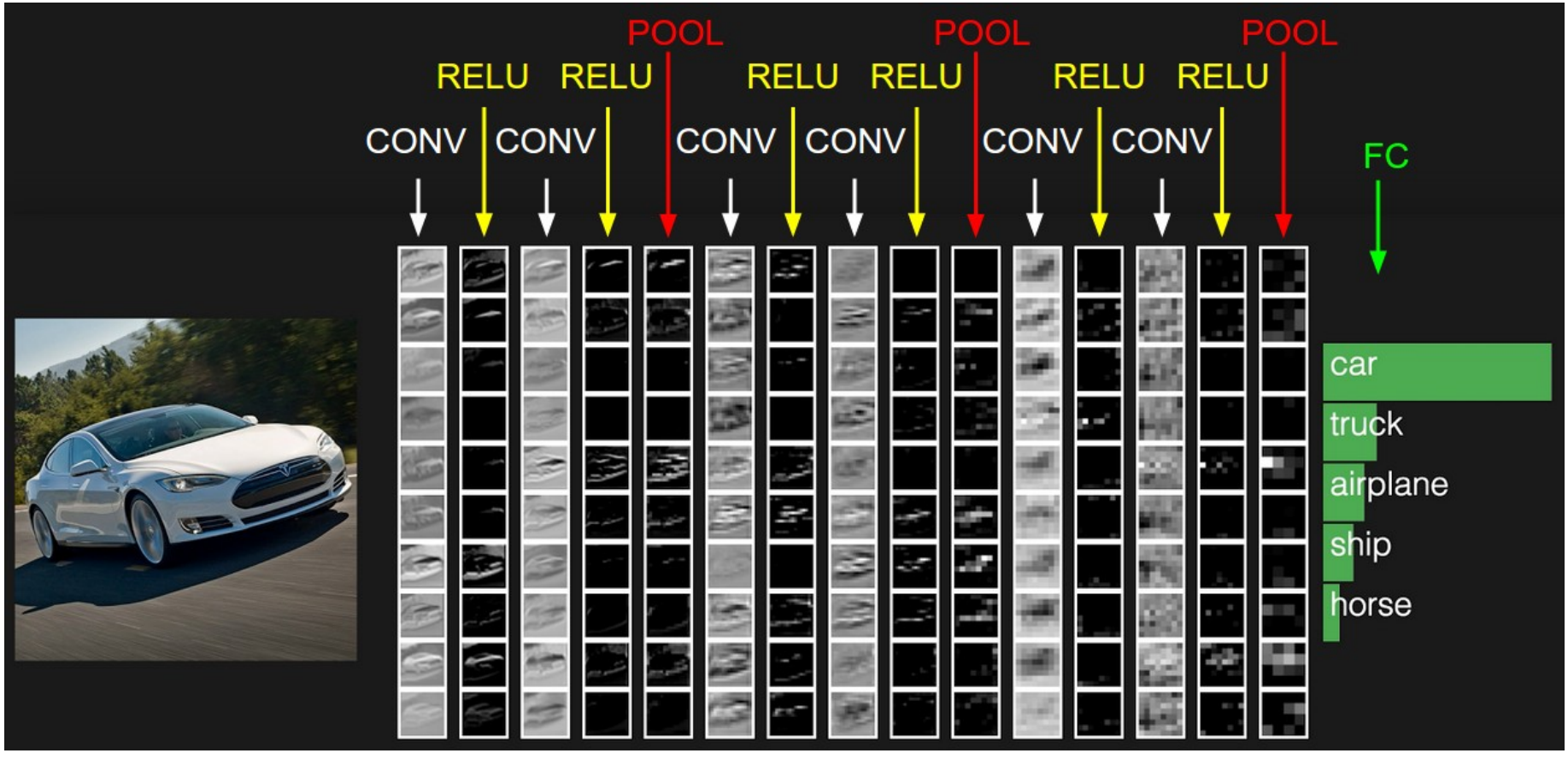
CNN - AUFBAU

Bestehen ebenfalls aus *Schichten*

Arten von Schichten:

- Input
- Convolution
- Activation
- Pooling
- Fully-Connected
- Output

CNN - AUFBAU



[http://cs231n.github.io/convolutional-networks/, 2017]

CONVOLUTION-LAYER

“Kernel” suchen nach Merkmalen im Bild

- sind einzelne Neuronen

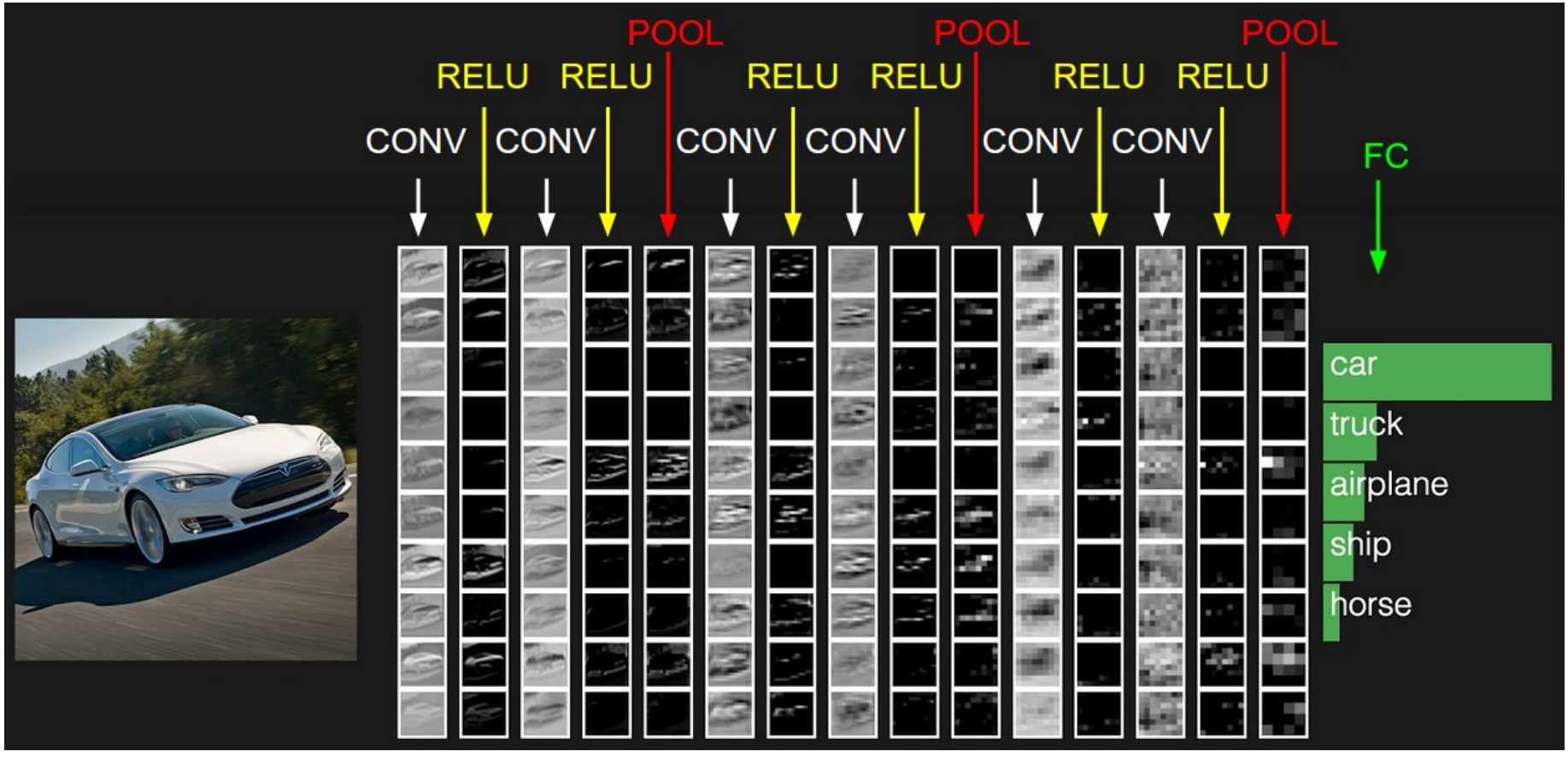
Kleiner Bereich des Bildes als Eingabe

CONVOLUTION

Bereich kann z.B. sein: $3 \times 3 \times 3$

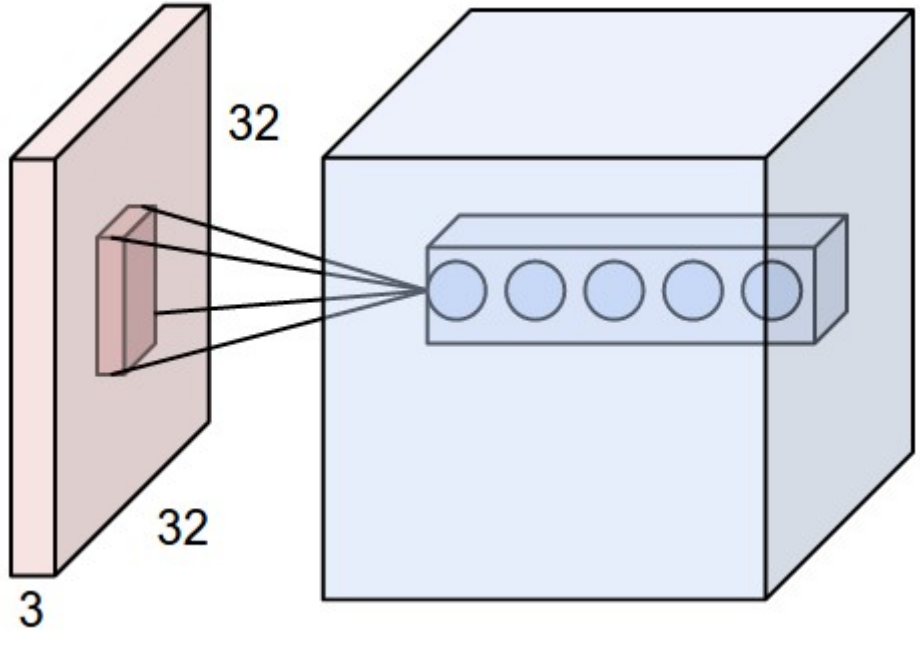
$= 27$ Gewichte für einen Kernel

$= 27$ “Pixel” aus dem Bild



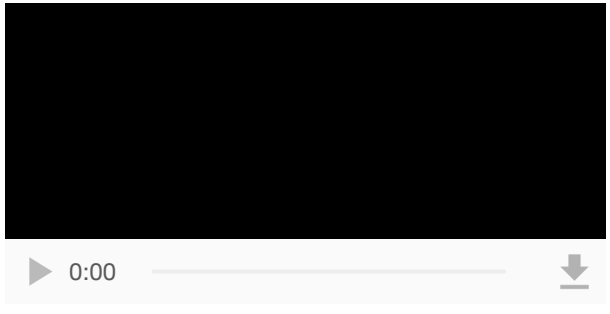
[http://cs231n.github.io/convolutional-networks/, 2017]

CONVOLUTION



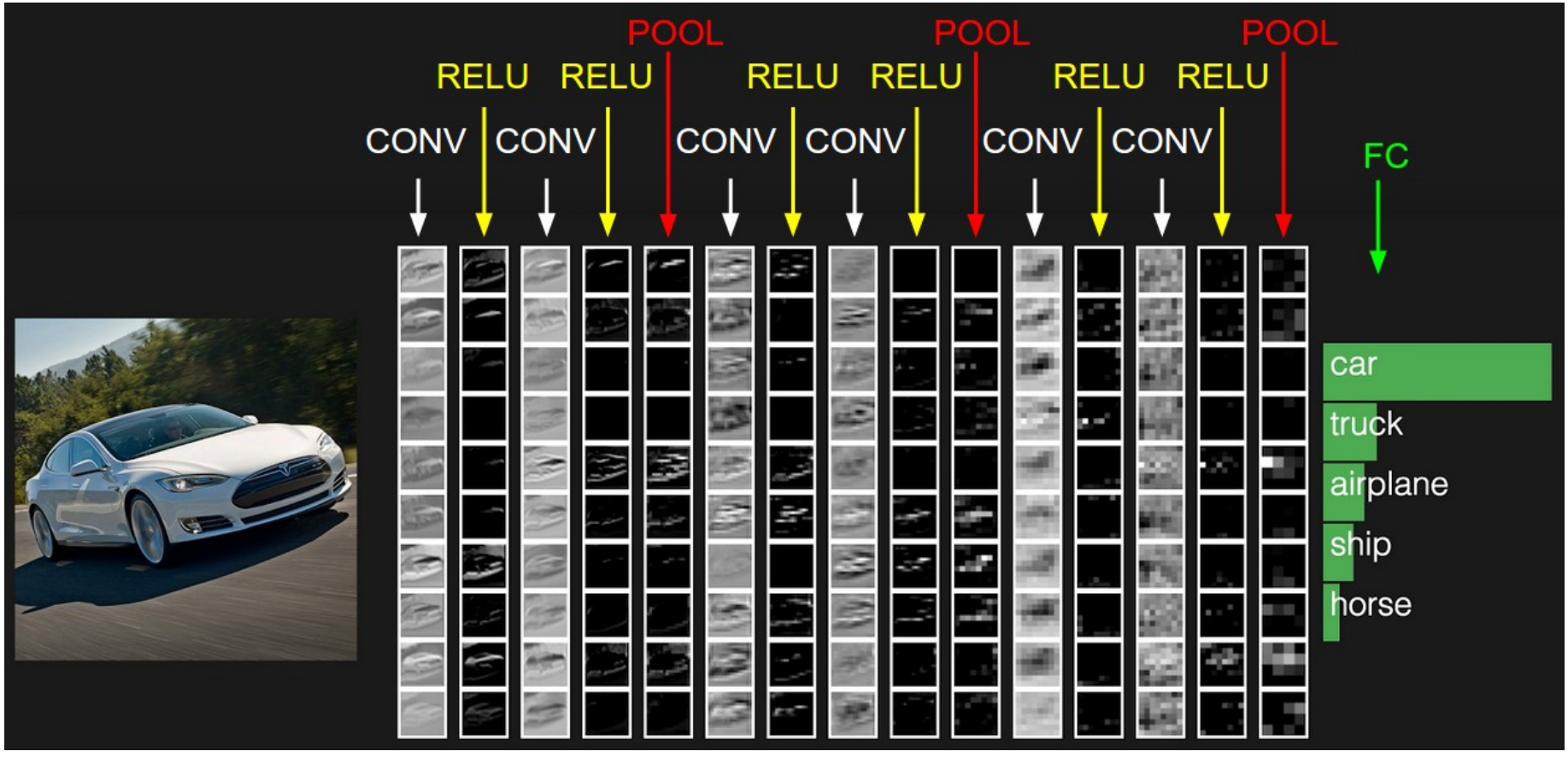
[http://cs231n.github.io/convolutional-networks/, 2017]

CONVOLUTION



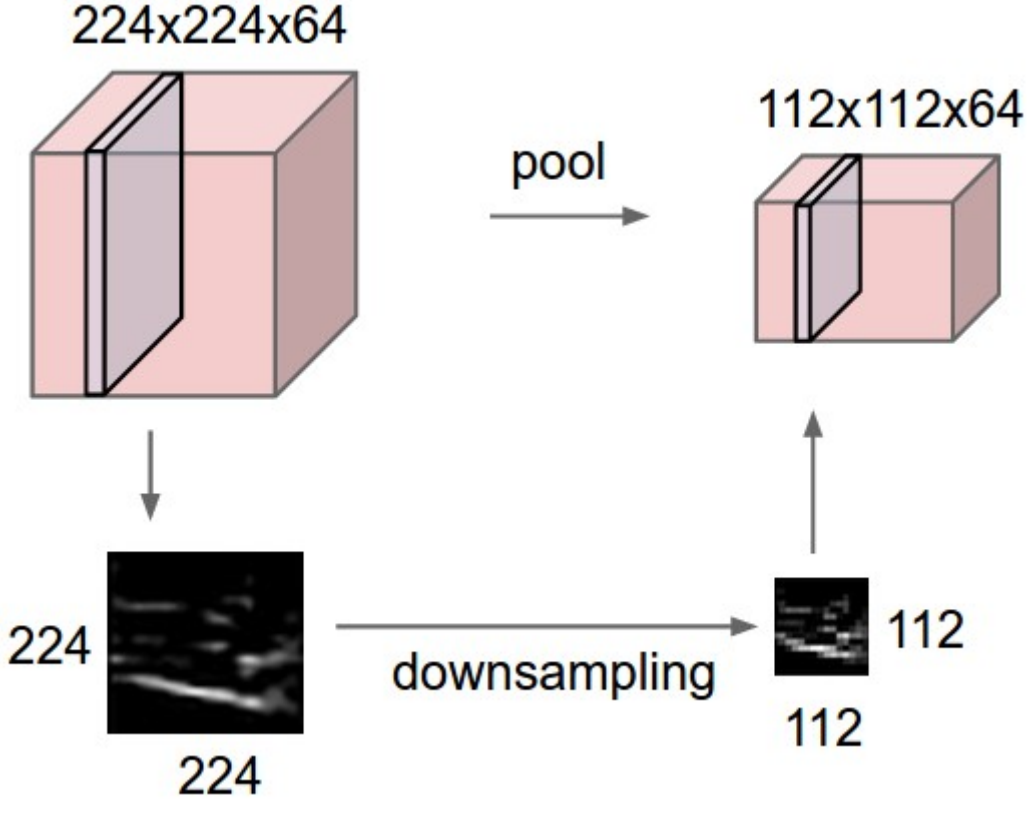
[http://cs231n.github.io/convolutional-networks/, 2017]

$$a_j = g(\sum w_{i,j} a_i)$$



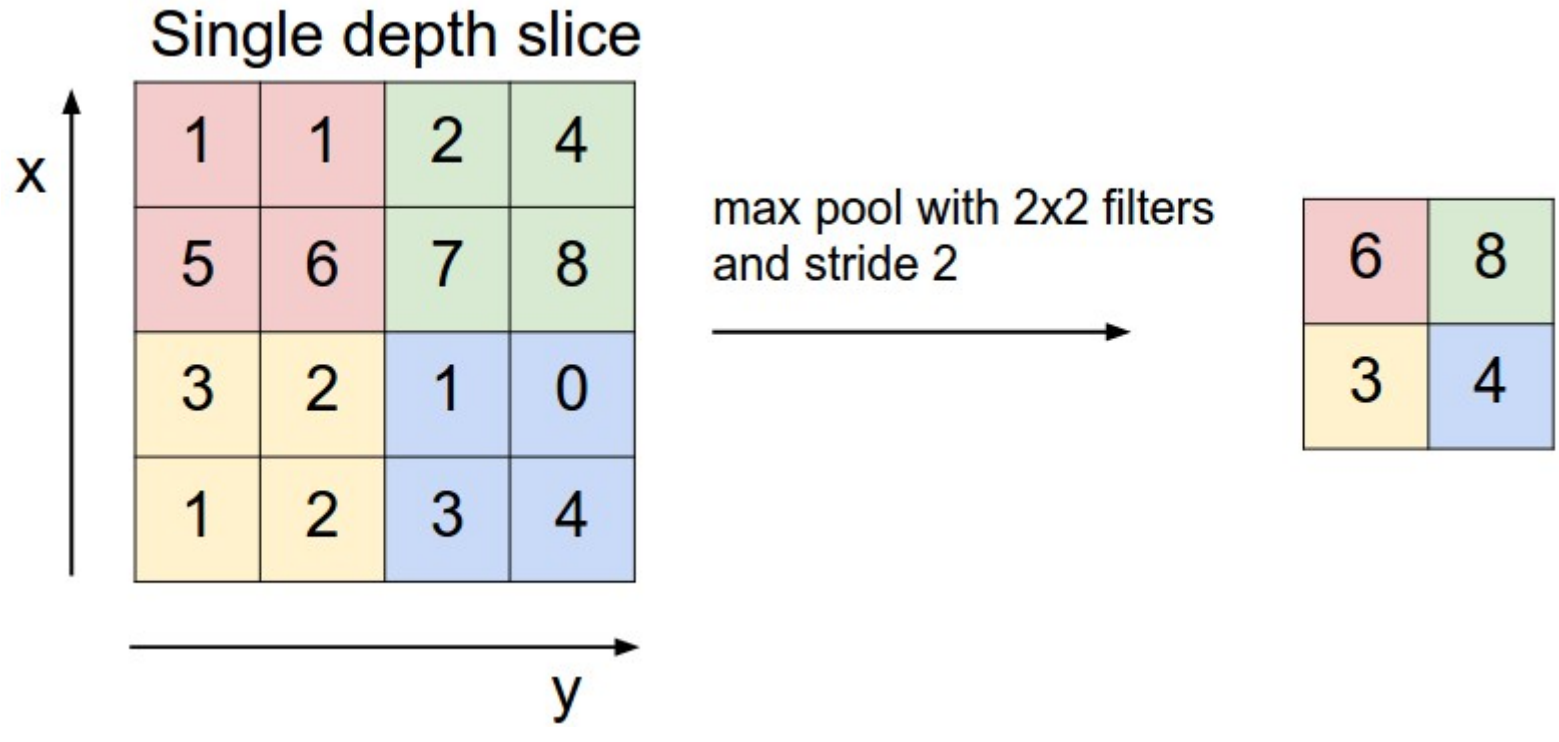
[http://cs231n.github.io/convolutional-networks/, 2017]

POOLING



[http://cs231n.github.io/convolutional-networks/, 2017]

MAX-POOLING



[http://cs231n.github.io/convolutional-networks/, 2017]

ALEXNET

Krizhevsky et al., 2012

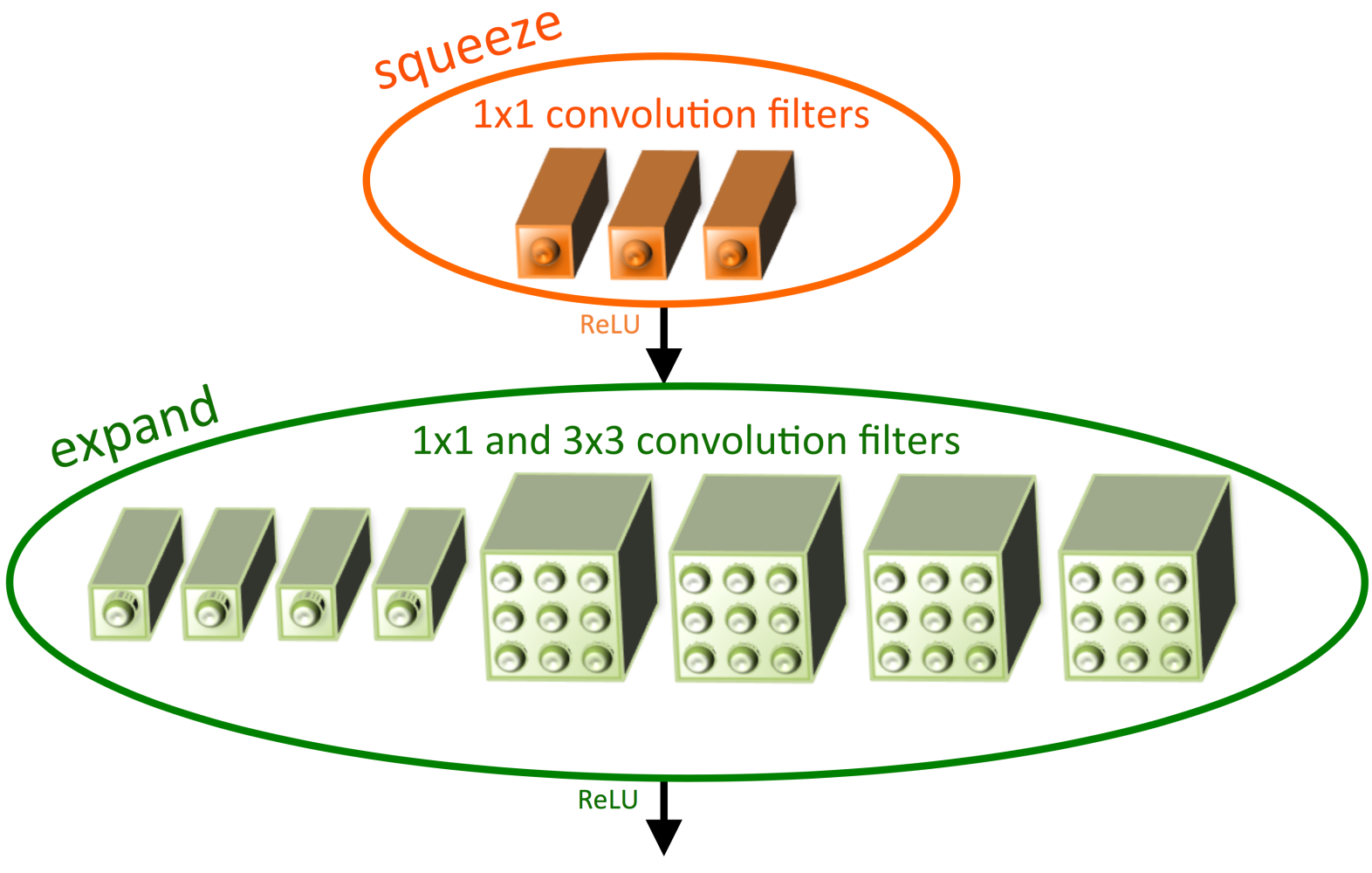
- Erstes CNN das die ILSVRC gewann (2012; 15% gegen 26%)
- 60 **mil.** Gewichte (auch *Parameter* genannt)
- 5 Conv-Layer, insgesamt 1376 Kernel
- 240MB
- Gilt als Auslöser des heutigen Deep Learning Booms

SQUEEZENET

Iandola et al., 2016

- “Kleines AlexNet”
- 1.248.424 Parameter
- 18 Conv-Layer, insgesamt 2976 Kernel
- 4,8MB
- 50x kleiner
- Gleiche Genauigkeit wie AlexNet
- Fire-Modul

FIRE-MODULE


squeeze
1x1 convolution filters
ReLU
expand
1x1 and 3x3 convolution filters
ReLU

MEINE ARBEIT

SqueezeCar: Fire-Modul basierte Deep Learning Architektur für eine autonome Fahrsimulation

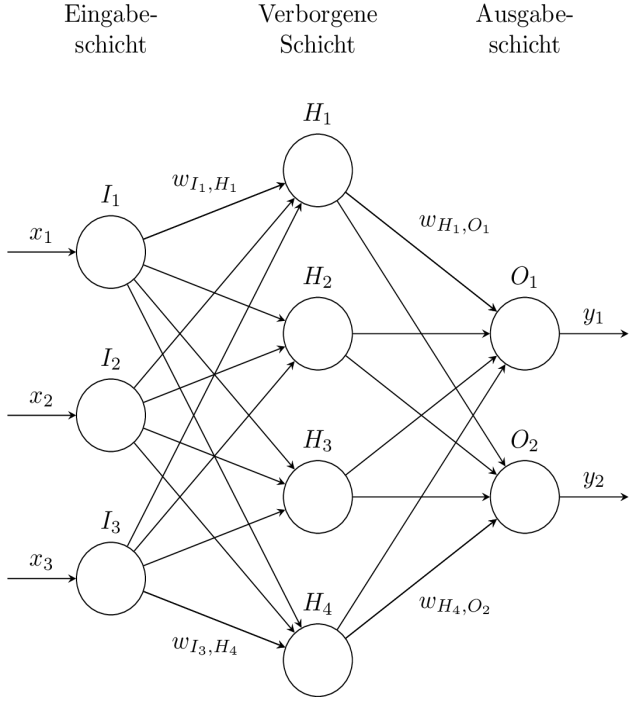
MEINE ARBEIT

SqueezeCar: Fire-Modul basierte Deep Learning Architektur für eine autonome Fahrsimulation

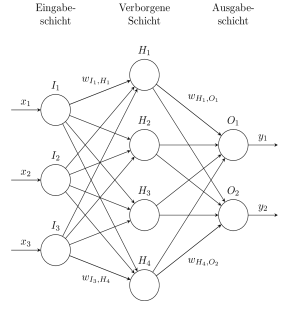
MEINE ARBEIT

SqueezeCar: Fire-Modul basierte Deep Learning Architektur für eine autonome Fahrsimulation

Andere:



Ich:



FAZIT

KNNs approximieren Funktionen

Kein Gehirn!

Lernen ist Konfiguration θ finden

Backpropagation Algorithmus

CNNs erkennen Merkmale in Bildern

De facto Standard

Arbeiten mit Convolution …und Kernels

Sehr präzise

Können sehr groß werden