

# Machine Learning Applications



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Tiefes Lernen — Deep Learning  
Am Beispiel von Faltende Neuronale Netzwerke



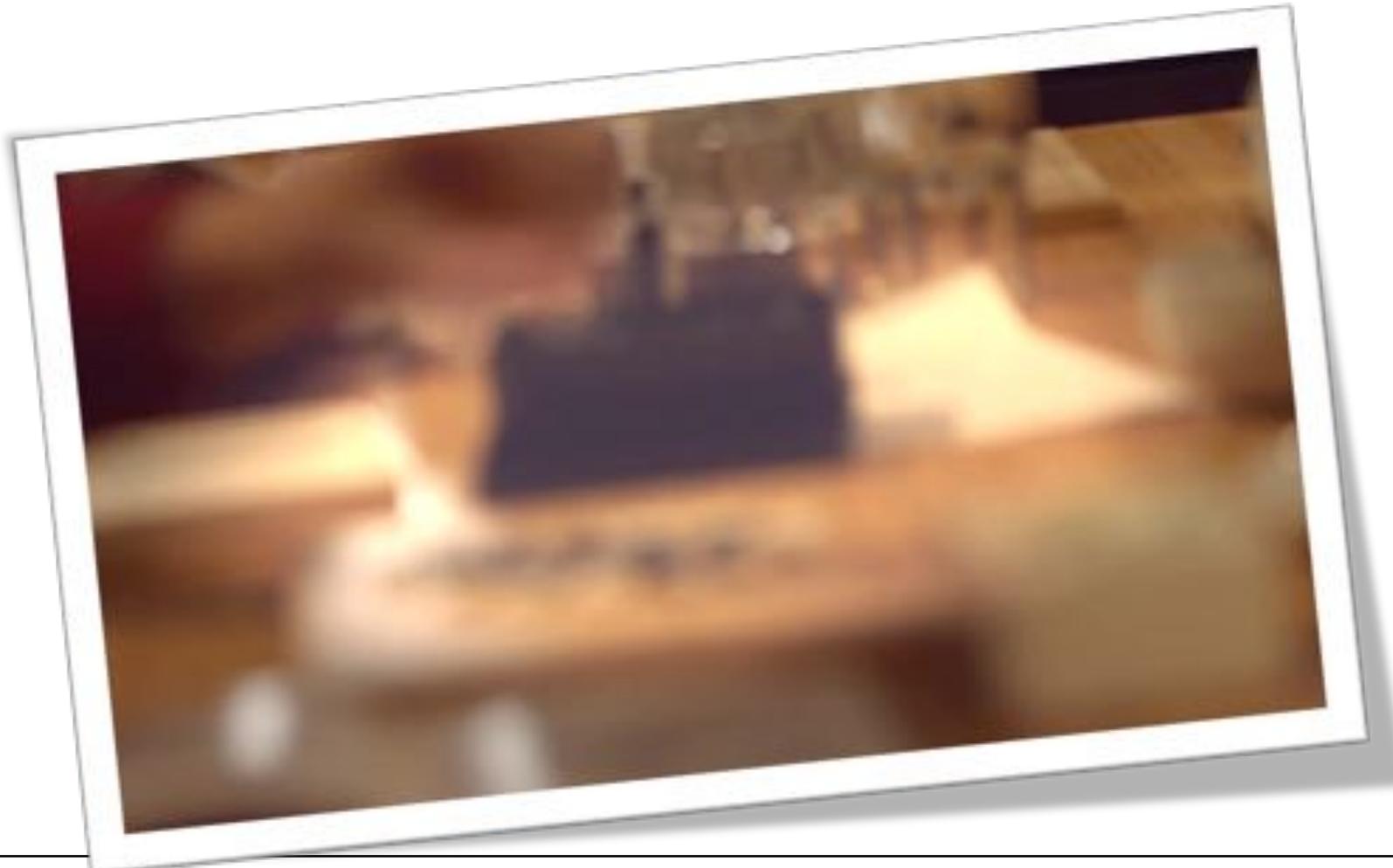
Basierende auf Folien von Viktoriia Sharmanska, Geoff Hinton, Fei-Fei Li und viele anderen.  
Danke fürs Offenlegen ihrer Folien



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

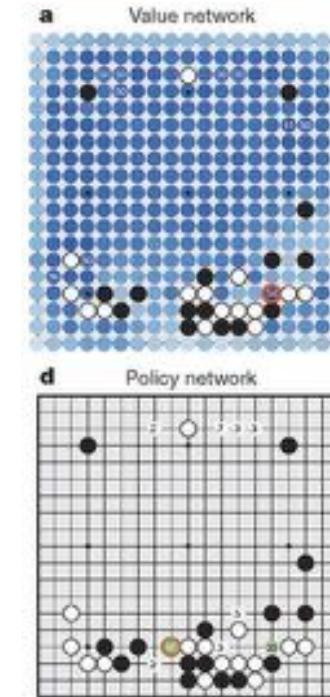
# DeepMinds AlphaGo

NATURE Video: <https://www.youtube.com/watch?v=g-dKXOlsf98>





# DeepMinds AlphaGo



Deep Policy Netzwerk wird so trainiert, dass es die Wahrscheinlichkeitskarte vielversprechender Züge darstellt. Das Deep Value Netzwerk wird zum Stutzen des MCMC Suchbaums benutzt.

# Maschine können auch andere Spiele spielen





# Das Ziel von tiefen Architekturen

Dazu stapeln die meisten Ansätze neuronale Netzwerke

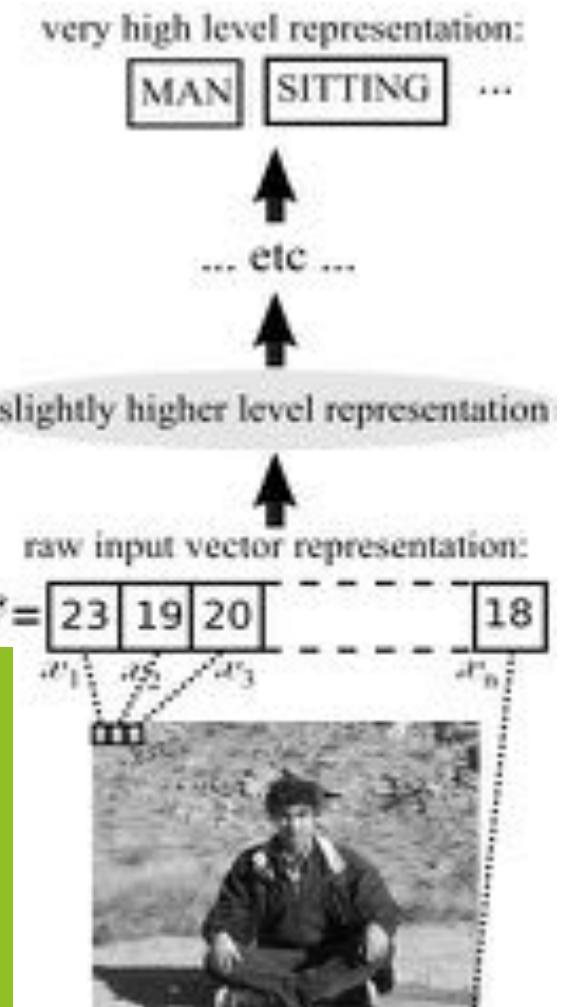
High-level, semantische Repräsentationen

Kanten, lokale Formen, Teile von Objekten

Low level Repräsentation

Tiefes Lernen zielt darauf ab,

- Hierarchien von Merkmalen zu lernen,
- wobei die Merkmale auf höheren Ebenen aus Merkmalen auf niederen Ebenen gebildet werden



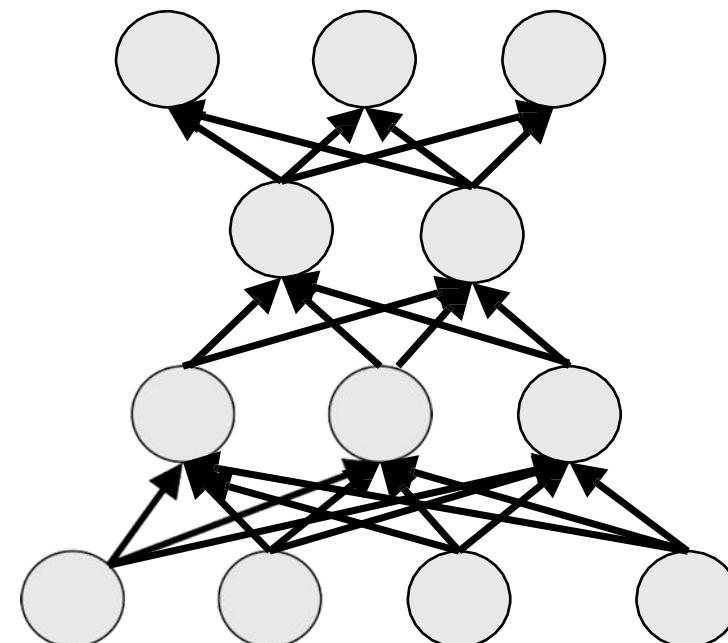
# Tiefe Architekturen

Tiefe Architekturen bestehen aus mehreren Schichten von nicht-linearen Berechnungen, wie z.B. neuronalen Netzwerken mit mehreren unbeobachteten Schichten

Ausgabeschicht

Unbeobachtete Schichten

Eingabeschicht



Beispiele für nicht-lineare Aktivierungsfunktionen:

$$\tanh(x)$$

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

$$\max(0, x)$$

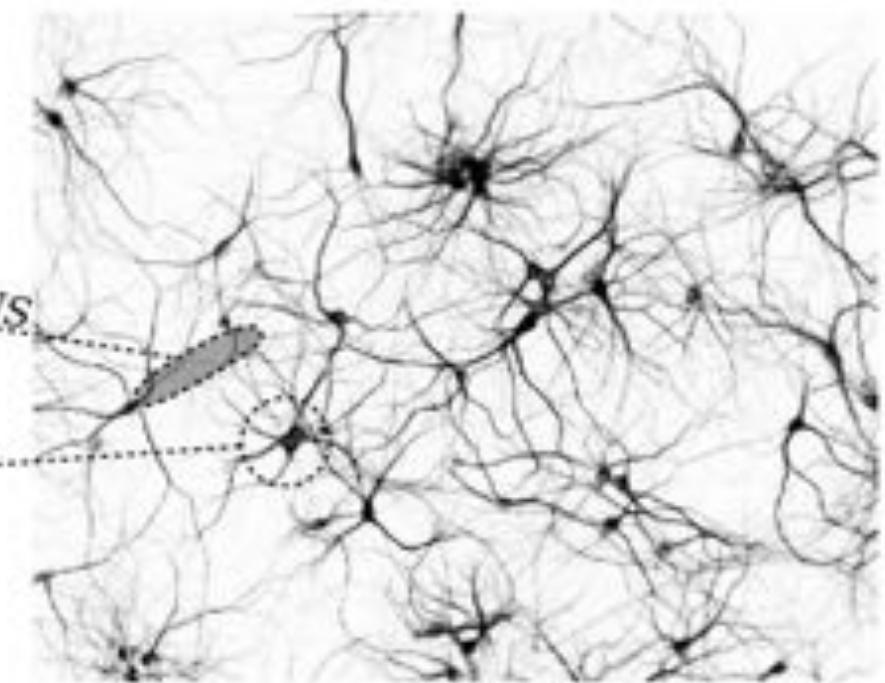
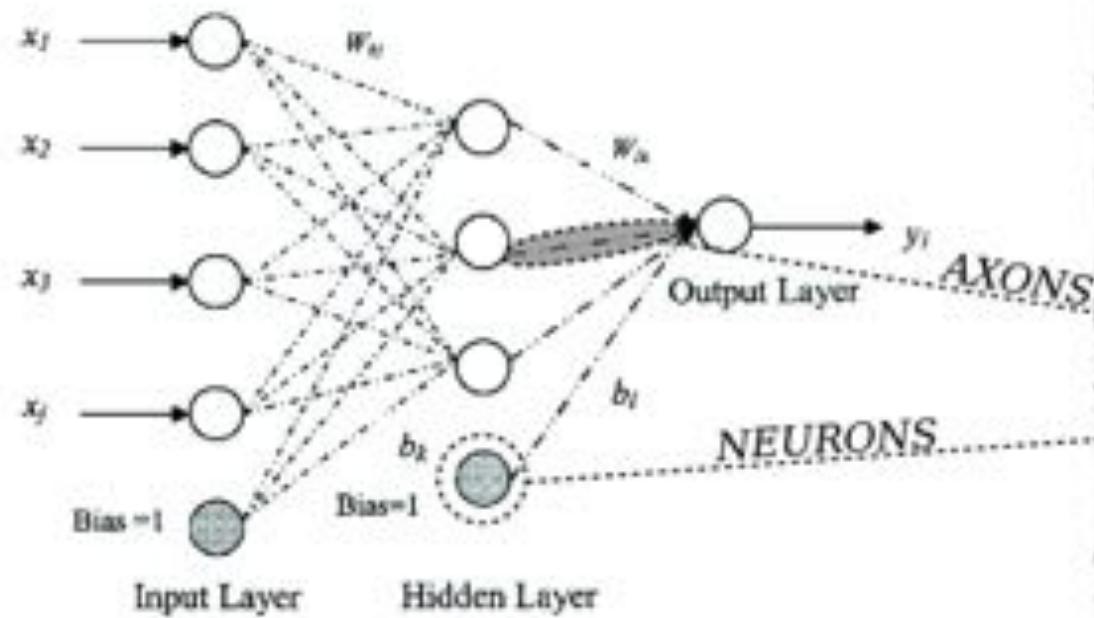
**Empirisch verhalten sich NN mit mehreren unbeobachteten Schichten besser als NN mit einer unbeobachteten Schicht.**

# Künstliche Neuronale Netzwerke (NN) sind durch biologische neuronale Netzwerke inspiriert

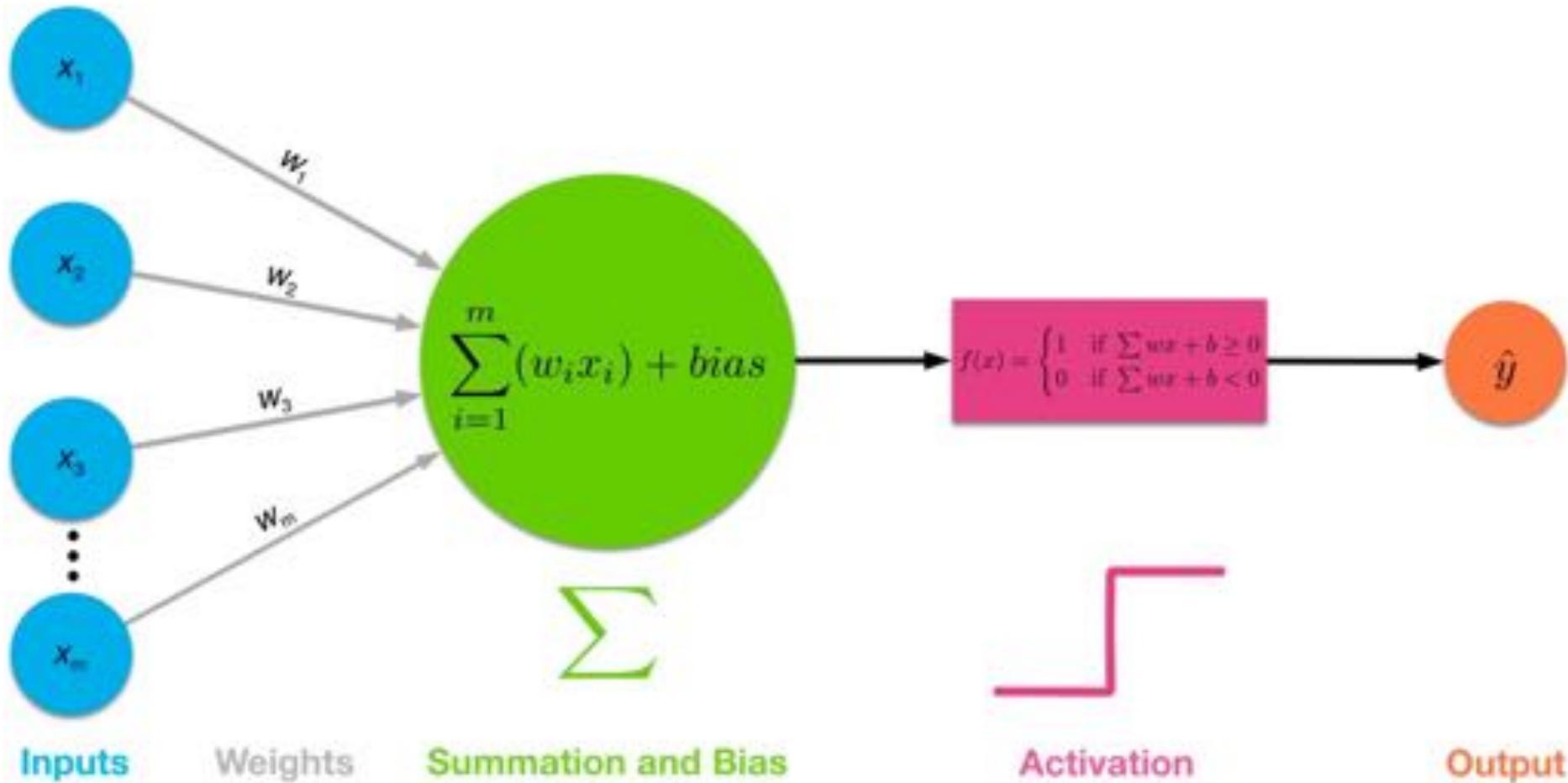


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## NEURAL NETWORK MAPPING



# Abstract Neural Unit





Godzilla vs. Trumplum:  
Some Suggestions to Add  
to the Periodic Table



To Protect Against Zika  
Virus, Pregnant Women  
Are Warned About Latin  
American Trips



THE NEW OLD  
F.T.C.'s Lure  
Doesn't End  
Training De

**nature**

international weekly journal of science

SCIEN

# Und hat ein großes Echo in der Presse

*Scientists See Promise in Deep-Learning Progr*

By JOHN MARKOFF NOV. 23, 2012

BBC

Sign in

News Sport Weather Sci

NEWS

Home Video World UK Business Tech Science Magazin

NATURE | NEWS



Game-playing software holds lessons  
for neuroscience

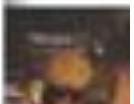
DeepMind computer provides new way to investigate how the brain

Forbes / Tech

NOV. 29, 2014 @ 11:37 AM 89,473 views

Top 20 Stocks For 2014

Tech 2015: Deep Learning And Machine Intelligence Will Eat The World



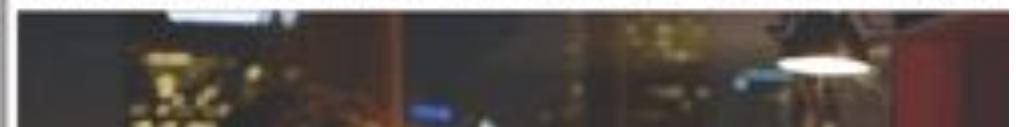
'Deep learning' technology inspired by human brain

culture business lifestyle fashion environment tech travel

Droids do dream of electric sheep

Google a step closer to developing machines with human-like intell

Algorithms developed by Google designed to encode thoughts, could computers with 'common sense' within a decade, says leading AI



# Auch wenn sie nicht wie Menschen generalisieren



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

The screenshot shows a research article from Current Biology, Volume 27, Issue 18, p2827–2832.e3, 25 September 2017. The article is titled "Humans, but Not Deep Neural Networks, Often Miss Giant Targets in Scenes". It is a REPORT by Miguel P. Eckstein, Kathryn Koehler, Lauren E. Welbourne, Emre Akbas. The article discusses how humans often miss targets when their size is inconsistent with the rest of the scene, even when the targets were made larger and more salient. In contrast, deep neural networks do not exhibit such deficits. The page includes a sidebar with options like "Switch to Standard View", "PDF (1 MB)", "Download Images (JPG)", "Email Article", "Add to My Reading List", and "Export Citation".

## Summary

Even with great advances in machine vision, animals are still unmatched in their ability to visually search complex scenes. Animals from bees [1, 2] to birds [3] to humans [4, 5, 6, 7, 8, 9, 10, 11, 12] learn about the statistical relations in visual environments to guide and aid their search for targets. Here, we investigate a novel manner in which humans utilize rapidly acquired information about scenes by guiding search toward likely target sizes. We show that humans often miss targets when their size is inconsistent with the rest of the scene, even when the targets were made larger and more salient and observers fixated the target. In contrast, we show that state-of-the-art deep neural networks do not exhibit such deficits in finding mis-scaled targets but, unlike humans, can be fooled by target-shaped distractors that are inconsistent with the expected target's size within the scene. Thus, it is not a human deficiency to miss targets when they are inconsistent in size with the scene; instead, it is a byproduct of a useful strategy that the brain has implemented to rapidly discount potential distractors.



# Wie dem auch sei, es hat die KI-Diskussion befeuert



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



## Stephen Hawking

"Success in creating AI would be the biggest event in human history..."

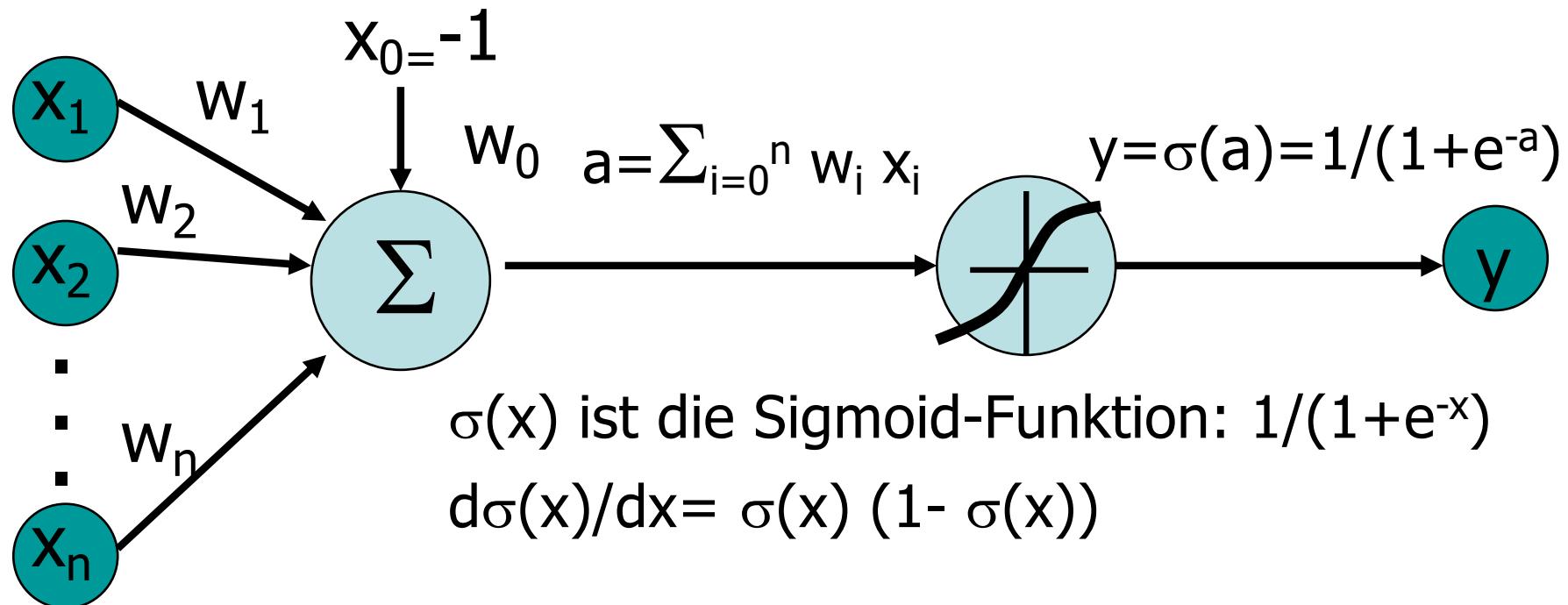
"Unfortunately, it might also be the last, unless we learn how to avoid the risks. In the near term, world militaries are considering autonomous-weapon systems that can choose and eliminate targets."

"...humans, limited by slow biological evolution, couldn't compete and would be superseded by A.I."



# Einzelne Neuronen sind typischerweise als Sigmoid-Einheiten modelliert

(aber andere Einheiten sind möglich)

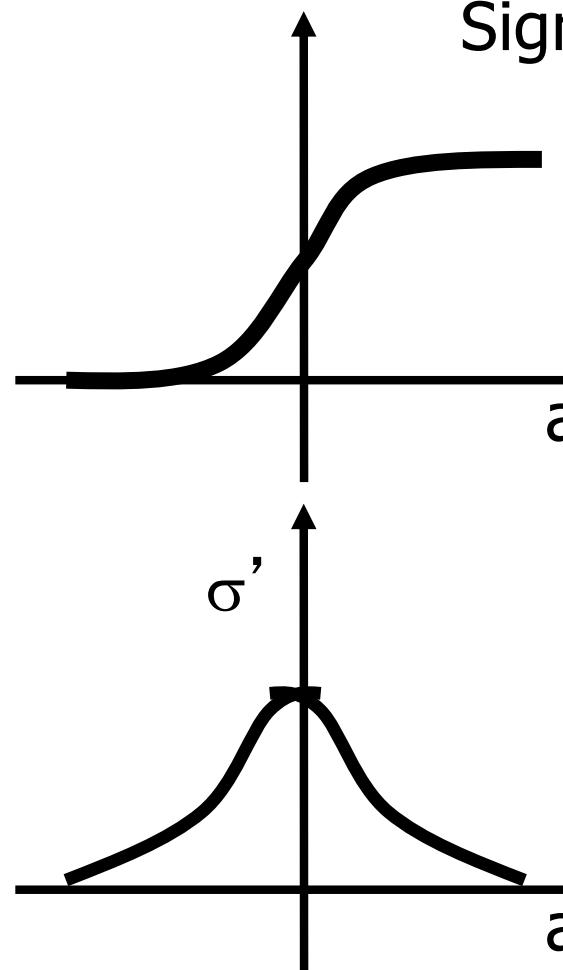


Zum Trainieren leiten wir das Gradient-Abstiegsverfahren her:

- Einzelne Sigmoid-Funktion:  $\partial E / \partial w_i = -\sum_p (t^p - y) y (1-y) x_i^p$
- Mehrschichtnetzwerk von Sigmoid-Einheiten: benutzte Backpropagation



# Gradientenabstieg für sigmoide Ausgabefkt.



Sigmoid

$$E^p[w_1, \dots, w_n] = \frac{1}{2} (t^p - y^p)^2$$

$$\partial E^p / \partial w_i = \partial / \partial w_i \frac{1}{2} (t^p - y^p)^2$$

$$= \partial / \partial w_i \frac{1}{2} (t^p - \sigma(\sum_i w_i x_i^p))^2$$

$$= (t^p - y^p) \sigma'(\sum_i w_i x_i^p) (-x_i^p)$$

$$\text{für } y = \sigma(a) = 1/(1+e^{-a})$$

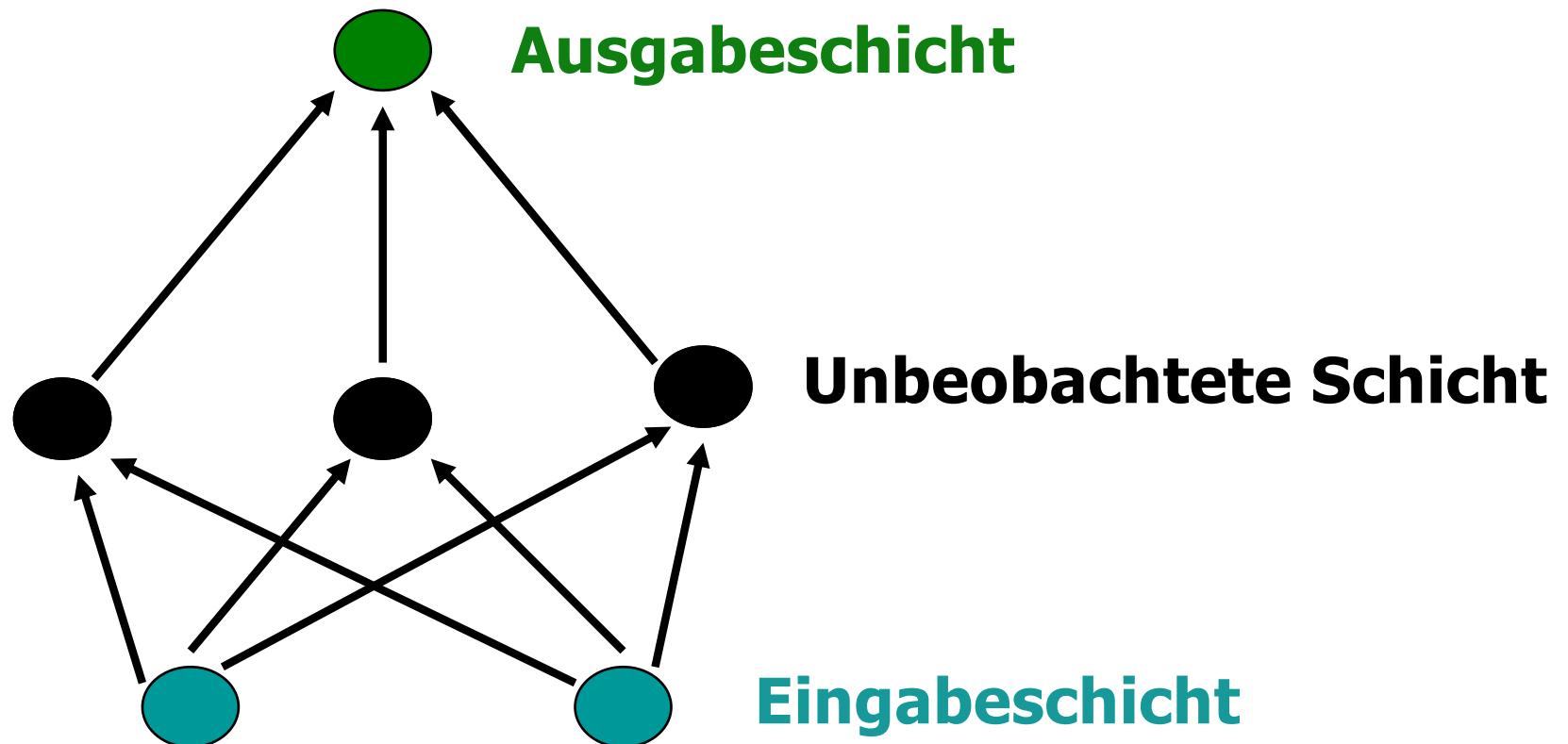
$$\sigma'(a) = e^{-a}/(1+e^{-a})^2 = \sigma(a) (1-\sigma(a))$$

$$w'_i = w_i + \Delta w_i = w_i + \alpha y(1-y)(t^p - y^p) x_i^p$$

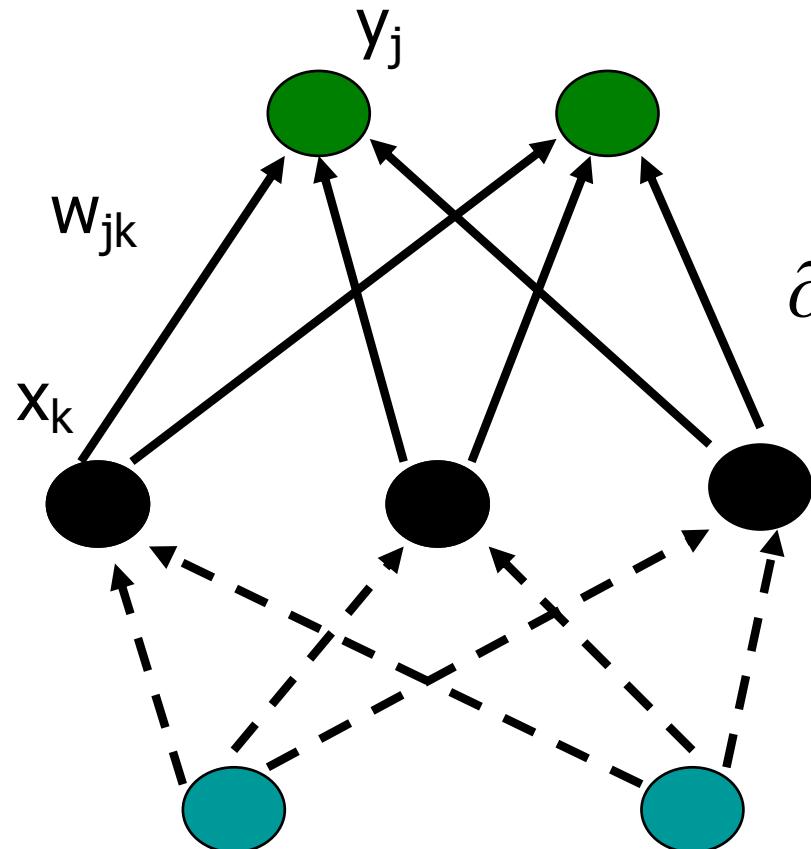
# Das Aufeinanderschichten von Einheiten ergibt (vorwärtsgerichtete) Mehrschichtnetzwerke



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



# Update-Regel für die Gewichte der Ausgabeschicht



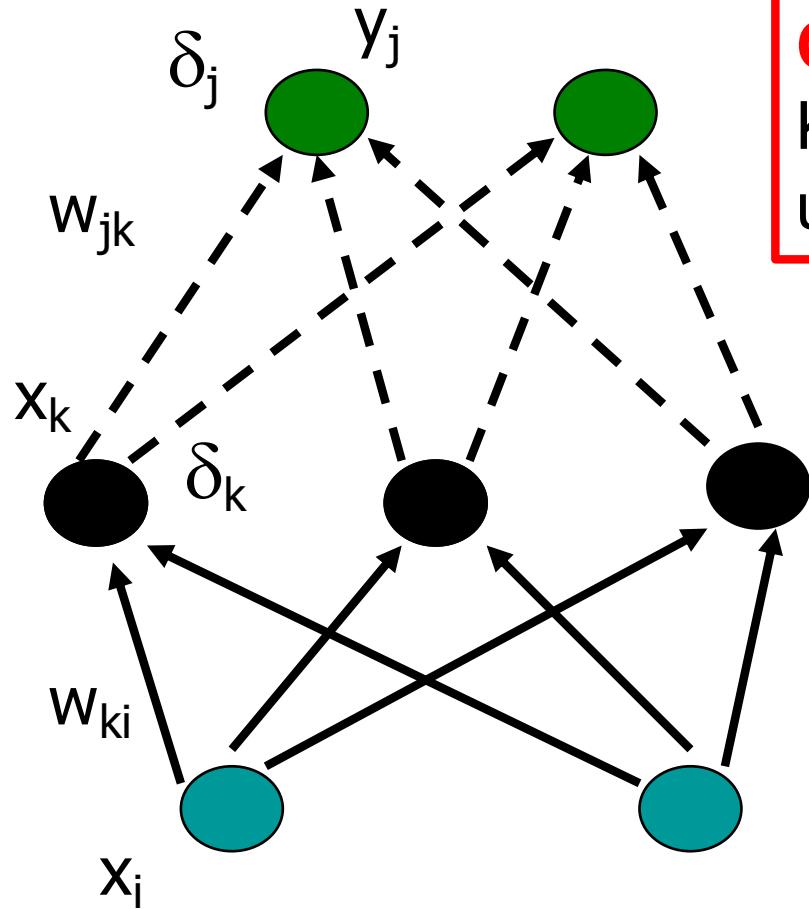
$$E^p[w_{ij}] = \frac{1}{2} \sum_j (t_j^p - y_j^p)^2$$

$$\begin{aligned}\partial E^p / \partial w_{ji} &= \partial / \partial w_{ji} \frac{1}{2} \sum_j (t_j^p - y_j^p)^2 \\ &= \dots \\ &= -y_j^p(1-y_j^p)(t_j^p - y_j^p) x_i^p\end{aligned}$$

$$\begin{aligned}\Delta w_{ji} &= \alpha y_j^p(1-y_j^p)(t_j^p - y_j^p) x_i^p \\ &= \alpha \delta_j^p x_i^p\end{aligned}$$

$$\text{mit } \delta_j^p := y_j^p(1-y_j^p)(t_j^p - y_j^p)$$

# Update-Regel für die Gewichte der unbeobachteten Schichten



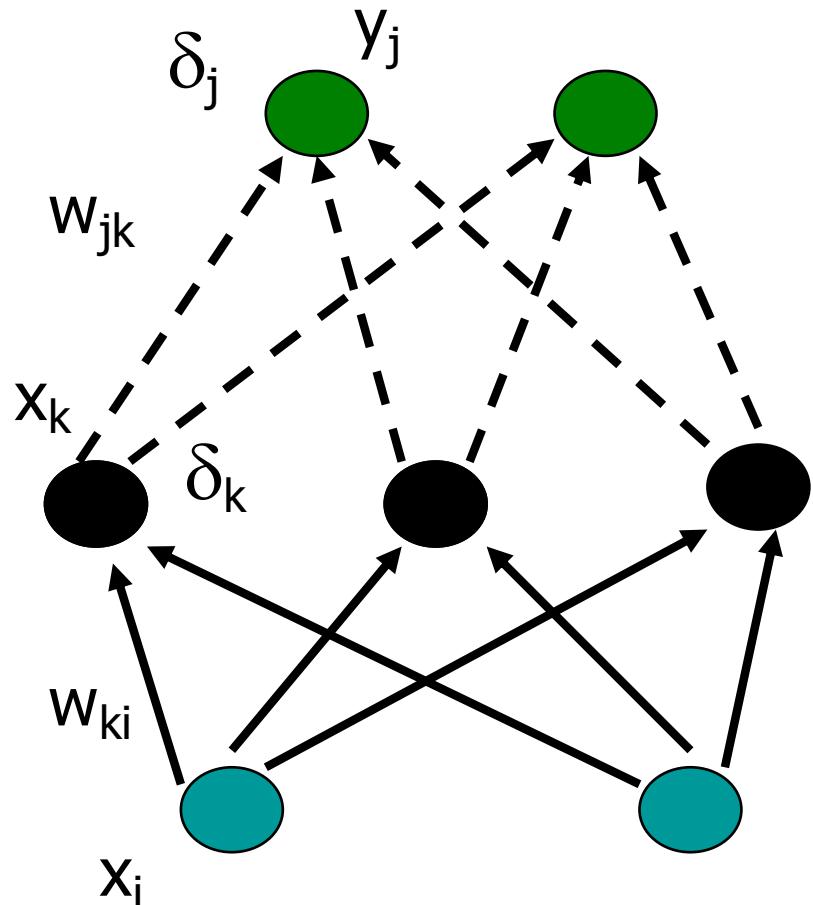
**Credit assignment problem:**  
Keine Zeilwerte  $t$  für die  
unbeobachteten Schichten

Wie sieht der Fehler für die  
unbeobachteten Einheiten aus?

$$\delta_k = \sum_j w_{jk} \delta_j y_j (1-y_j)$$

$$\Delta w_{ki} = \alpha x_k^p (1-x_k^p) \delta_k^p x_i^p$$

# Update-Regel für die Gewichte der unbeobachteten Schichten



$$E^p[w_{ki}] = \frac{1}{2} \sum_j (t_j^p - y_j^p)^2$$

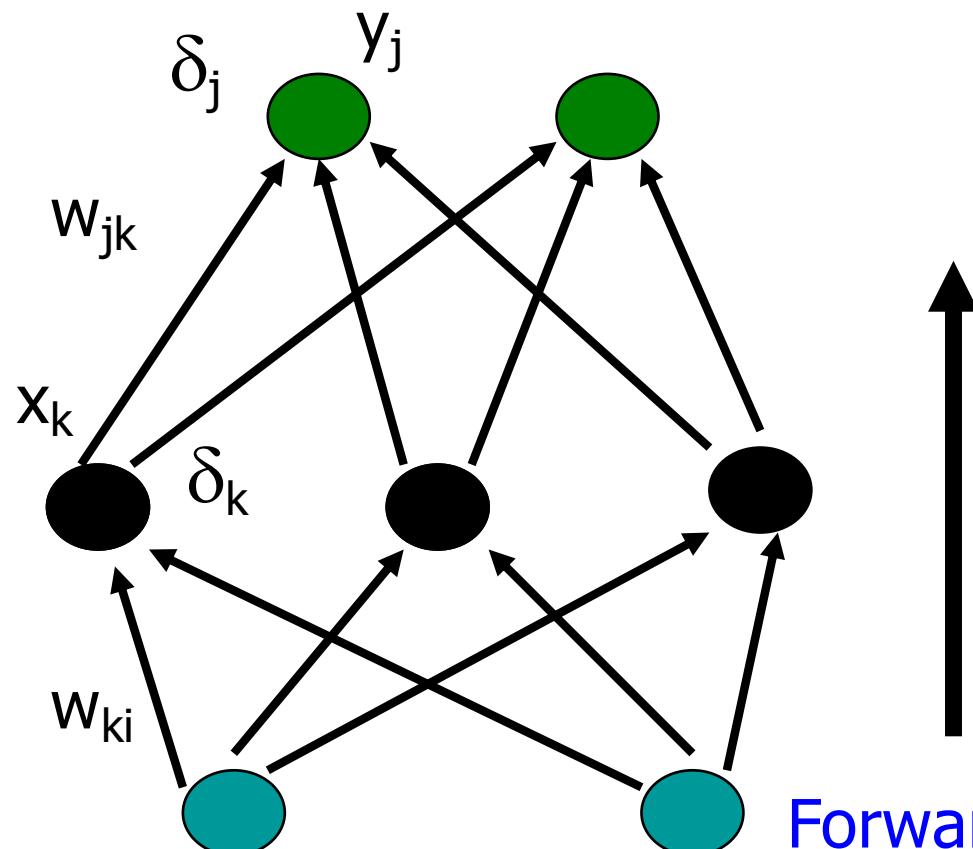
## Kettenregel

$$\begin{aligned}\partial E^p / \partial w_{ki} &= \partial / \partial w_{ki} \frac{1}{2} \sum_j (t_j^p - y_j^p)^2 \\&= \partial / \partial w_{ki} \frac{1}{2} \sum_j (t_j^p - \sigma(\sum_k w_{jk} x_k^p))^2 \\&= \partial / \partial w_{ki} \frac{1}{2} \sum_j (t_j^p - \sigma(\sum_k w_{jk} \sigma(\sum_i w_{ki} x_i^p)))^2 \\&= -\sum_j (t_j^p - y_j^p) \sigma'(a) w_{jk} \sigma'(a) x_i^p \\&= -\sum_j \delta_j w_{jk} \sigma'(a) x_i^p \\&= -\sum_j \delta_j w_{jk} x_k (1-x_k) x_i^p\end{aligned}$$

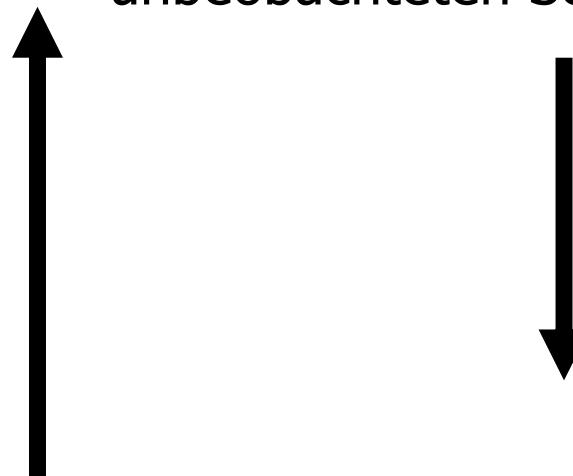
$$\Delta w_{ki} = \alpha \delta_k x_i^p \quad \text{with } \delta_k = \sum_j \delta_j w_{jk} x_k (1-x_k)$$

# Backpropagation

## Fehlerrückmeldungsverfahren



**Backward step / Rückmeldung:**  
Meldet den Fehler von der Ausgabeschicht (sukzessive) an die unbeobachteten Schichten zurück



**Forward step / Vorwärtsmeldung :**  
Propagiert Aktivierungen von der Eingabeschicht zur Ausgabeschicht

# Deep Convolutional NN (DCNs)

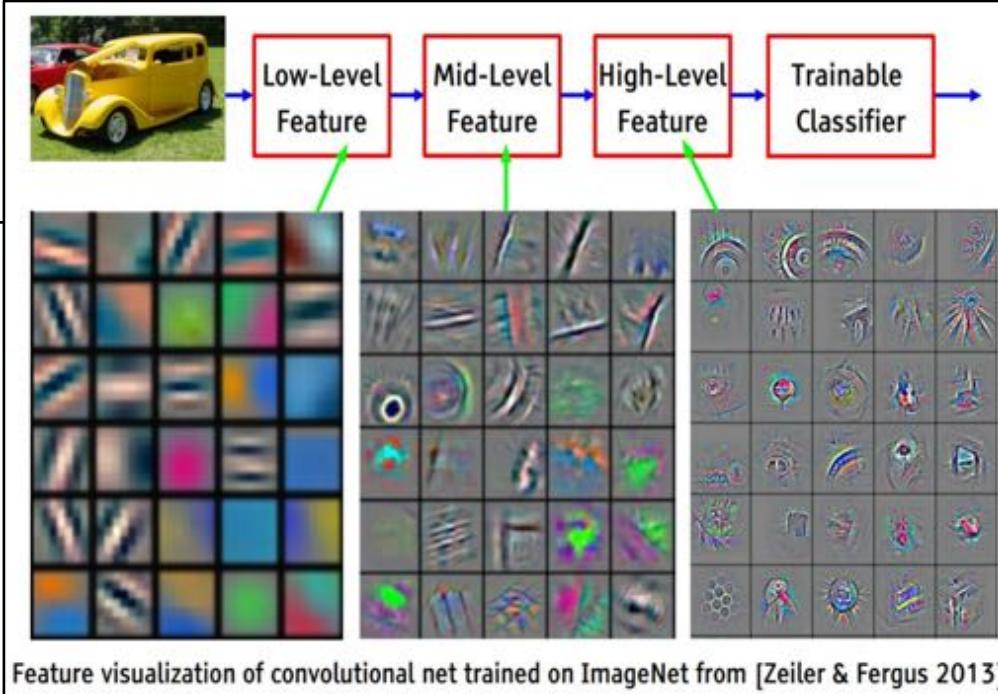
## Faltendes neuronales Netzwerk



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Im Vergleich zu normalen neuronalen Netzwerken mit ähnlich großen Schichten haben

- DCNs wesentlich weniger Koppelungen (Verbindungen) und daher auch Parameter,
- Sind damit einfacher zu trainieren (Schätzung der Parameter aus Daten)
- Und haben üblicherweise mehr als 5 Schichten (eine Anzahl die bei vollständig verbundene, neuronale Netzwerke dazu führt, dass diese im Grunde nicht mehr trainierbar sind).

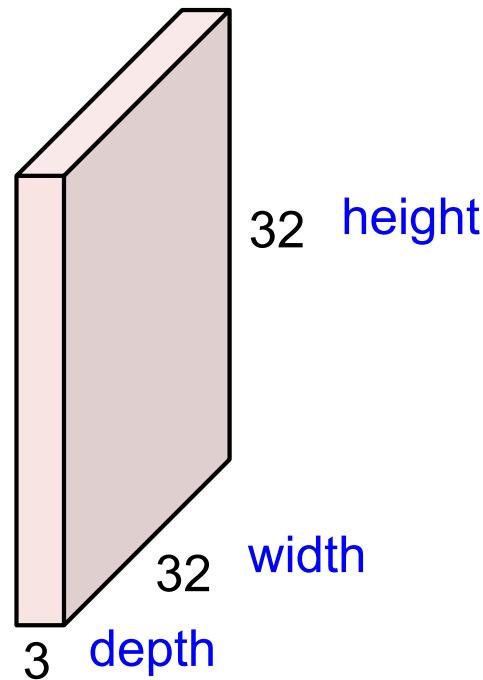


Einer der ersten Durchbrüche von DCNs war die  
**BILDKLASSIFIKATION**

- Faltungsschichten
- Nicht-lineare Aktivierungsfunktion ReLU
- Max pooling Schicht
- Vollständig verbundene Schicht

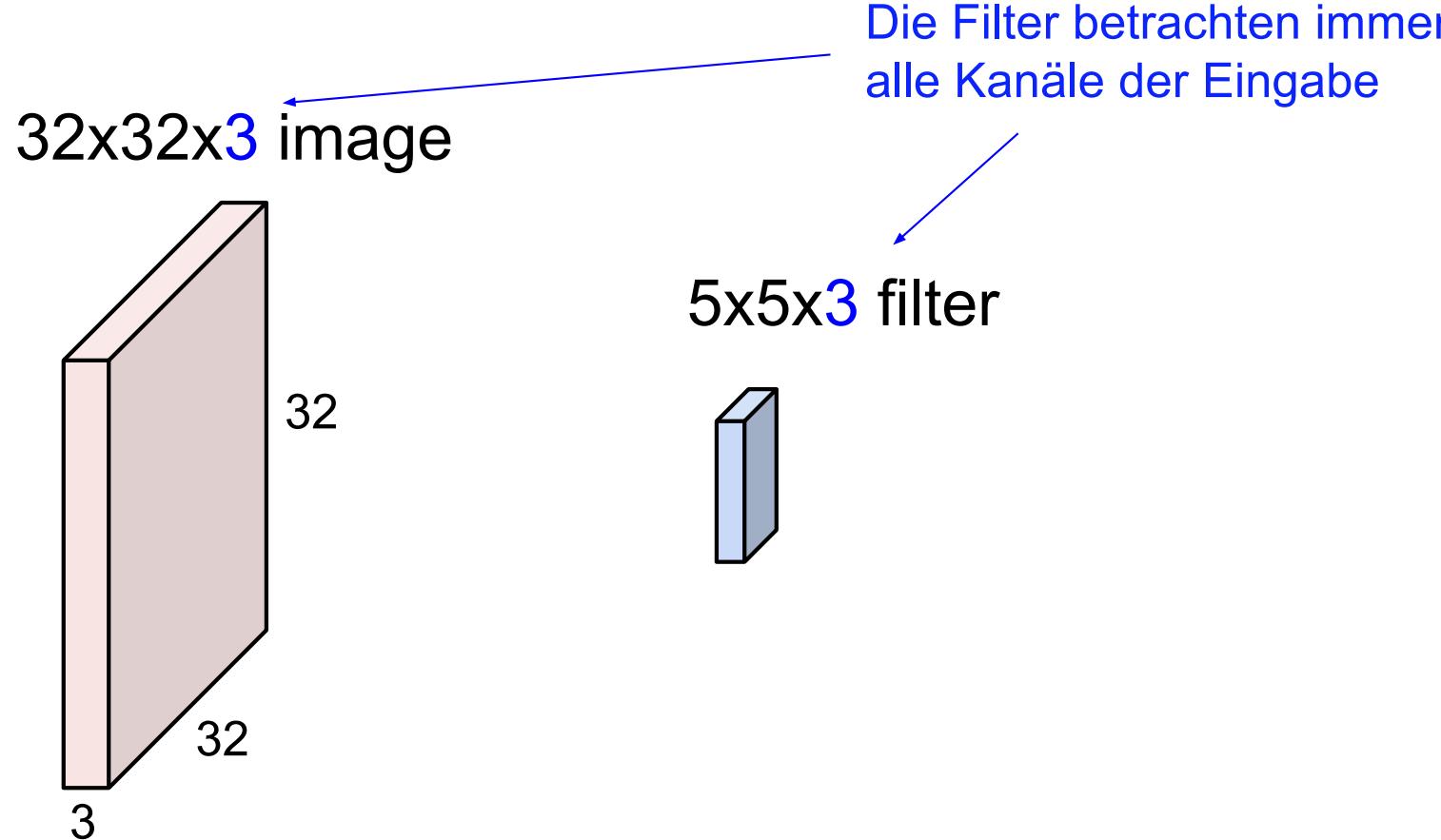
# Faltungsschicht (Convolutional Layer)

32x32x3 image



Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

# Faltungsschicht (Convolutional Layer)

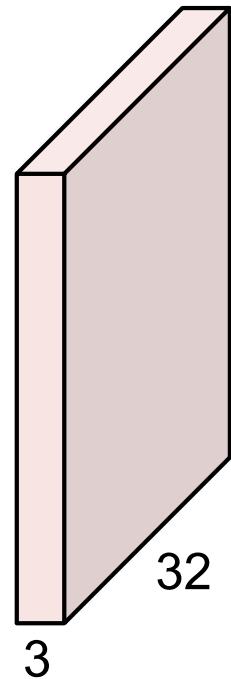


Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

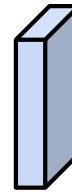


# Faltungsschicht (Convolutional Layer)

32x32x3 image



5x5x3 filter

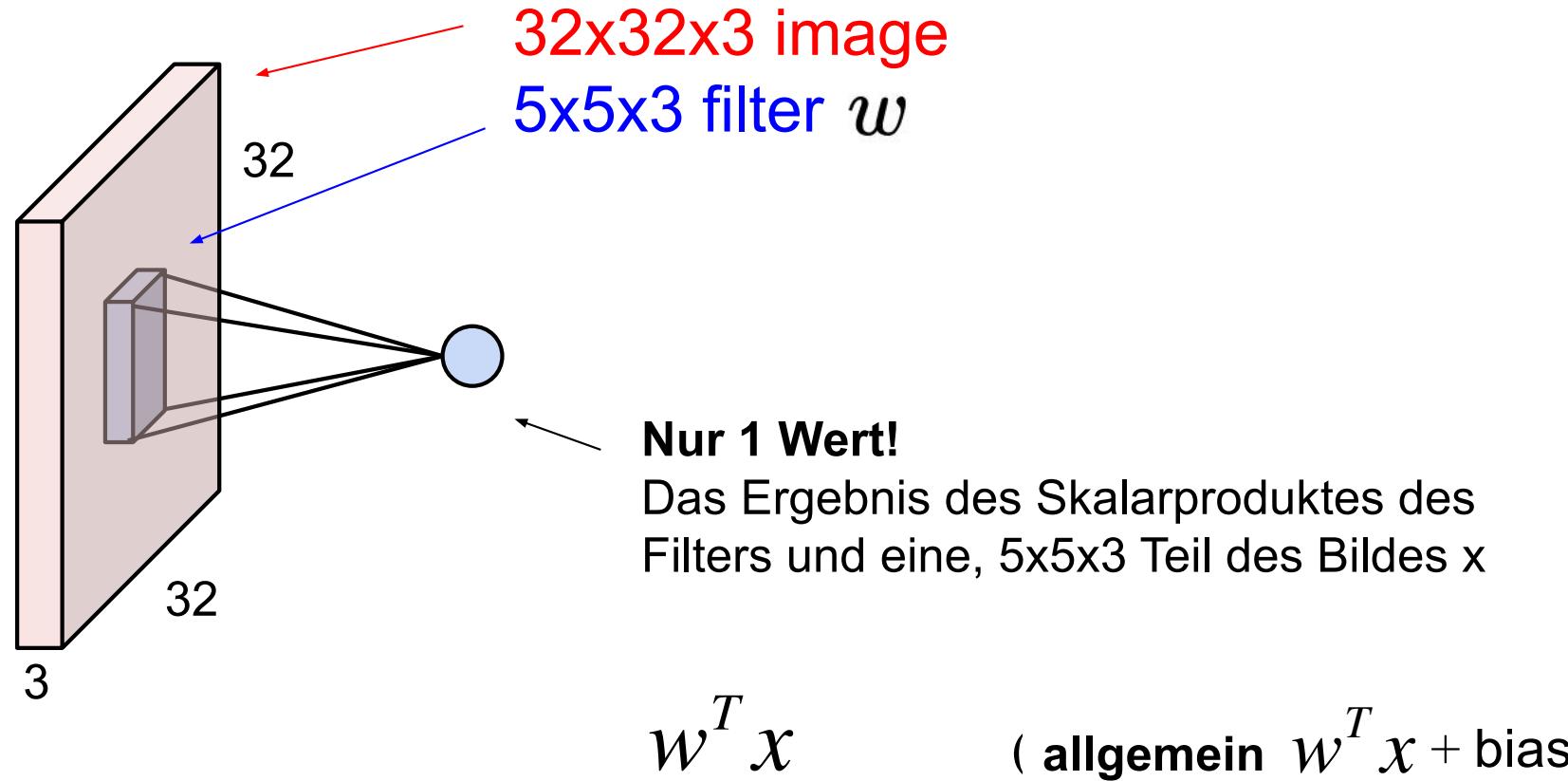


Die Filter betrachten immer alle Kanäle der Eingabe

Falte Bild mit dem Filter, d.h., „schiebe den Filter über das Bild und berechne an jeder Stelle das Skalarprodukt“

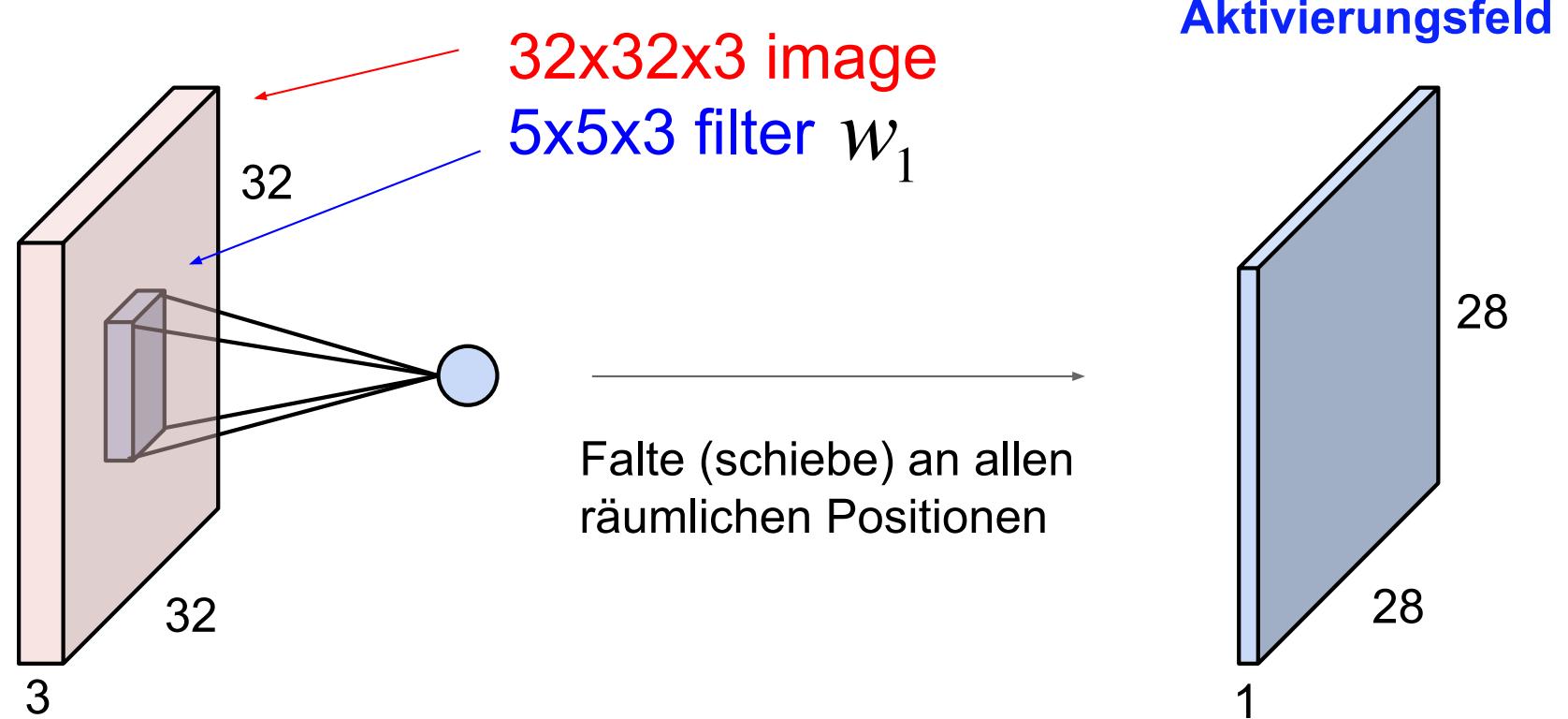
Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

# Faltungsschicht (Convolutional Layer)



Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

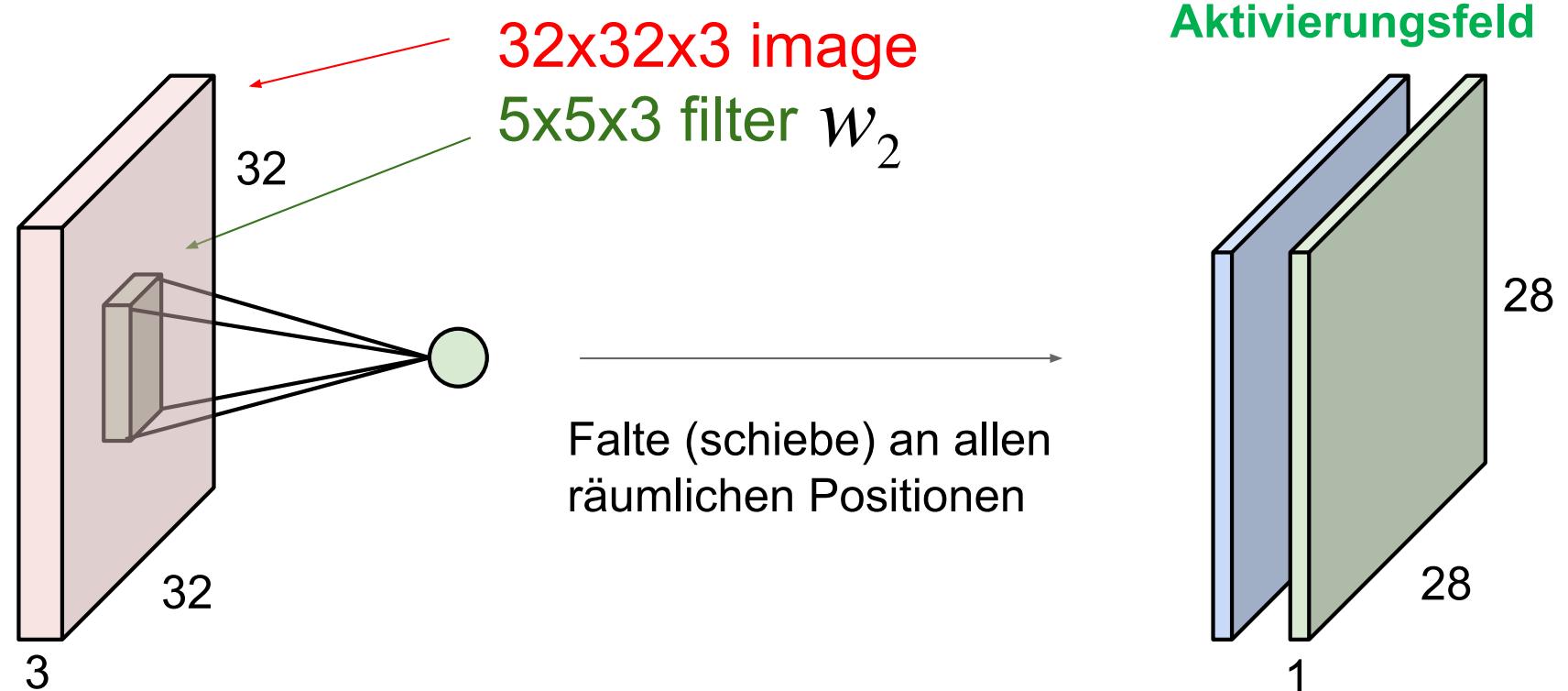
# Faltungsschicht (Convolutional Layer)



Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

# Faltungsschicht (Convolutional Layer)

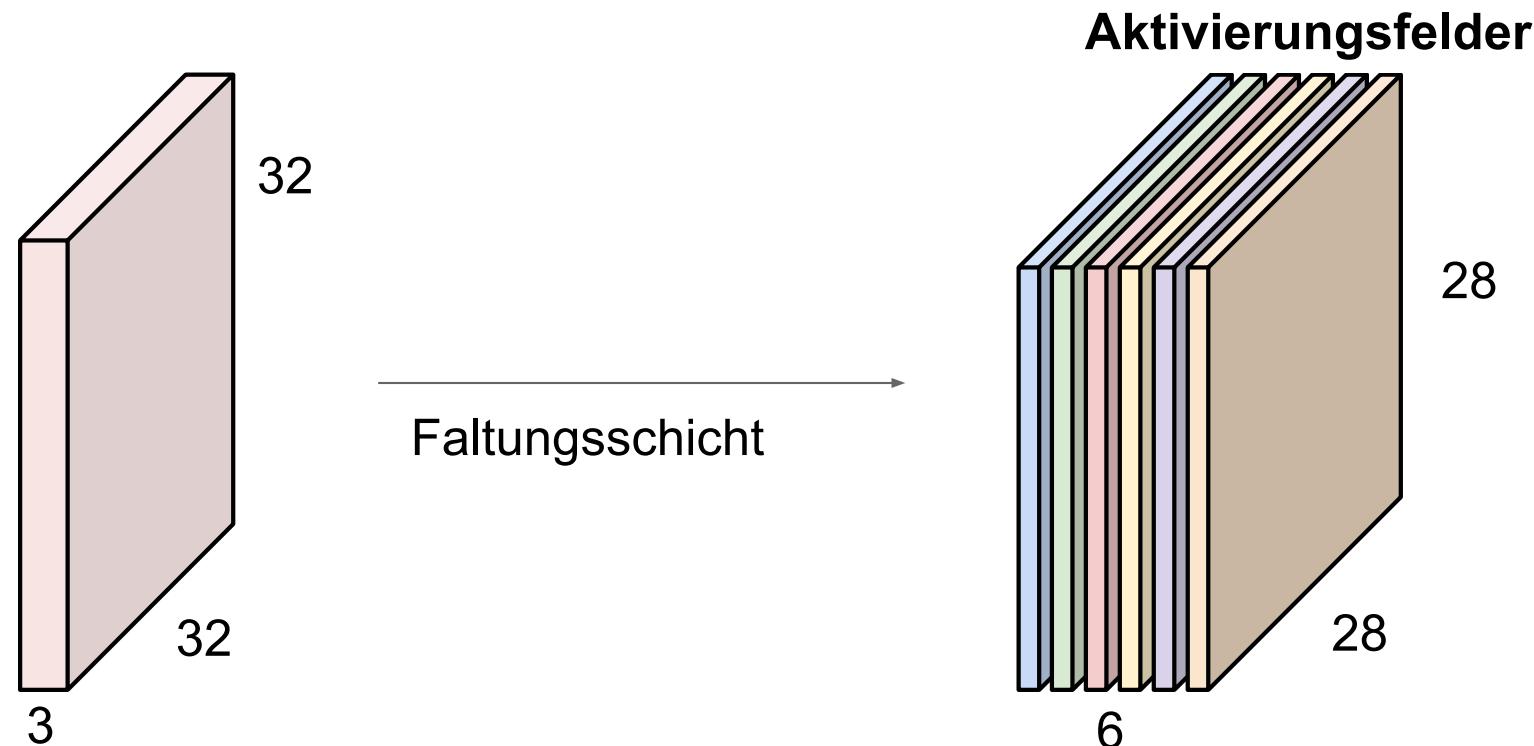
Benutze einen weiteren, grünen Filter



Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

# Faltungsschicht (Convolutional Layer)

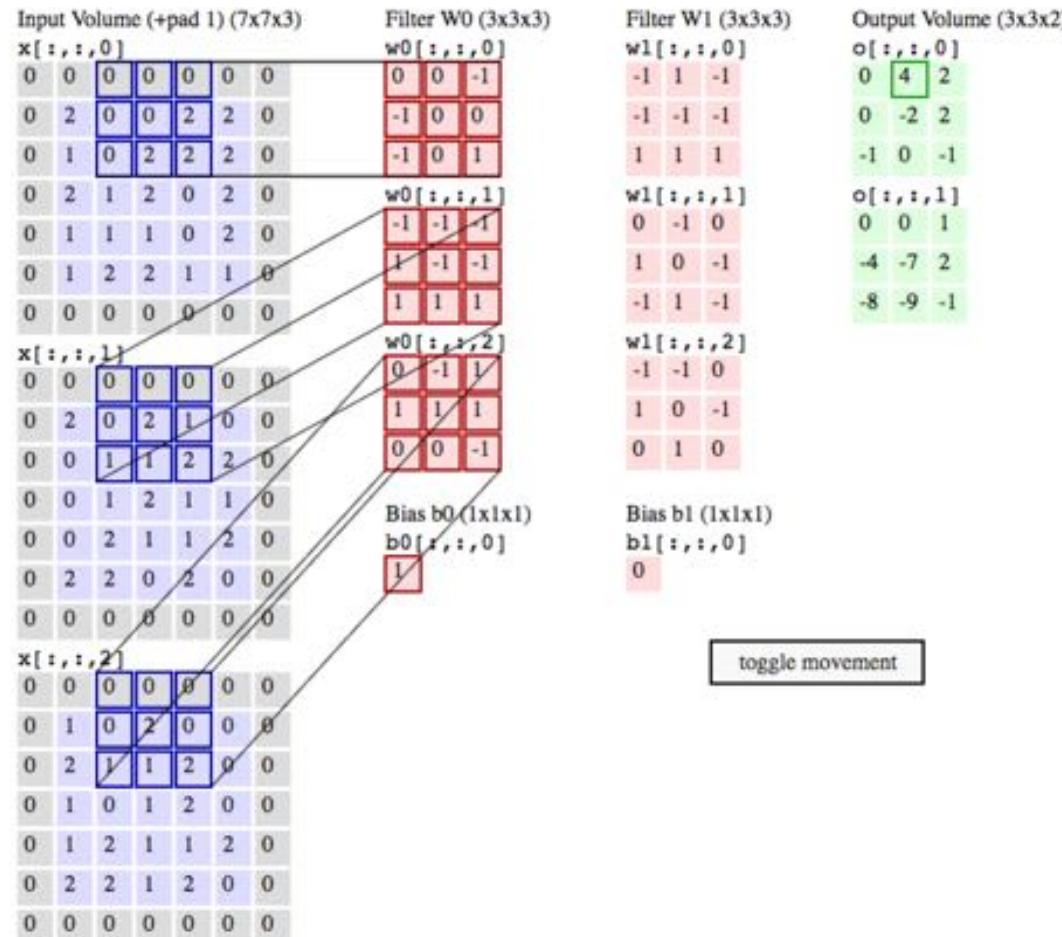
Wenn wir z.B. 6  $5 \times 5 \times 3$  Filter benutzen, bekommen wir 6 Aktivierungsfelder



Diese legen wir übereinander und erhalten ein “neues Bild“ der Größe  $28 \times 28 \times 6$

Filter detektieren lokale Muster wie z.B. Farbwerte, Kanten, ...

# Faltungsschicht in Aktion

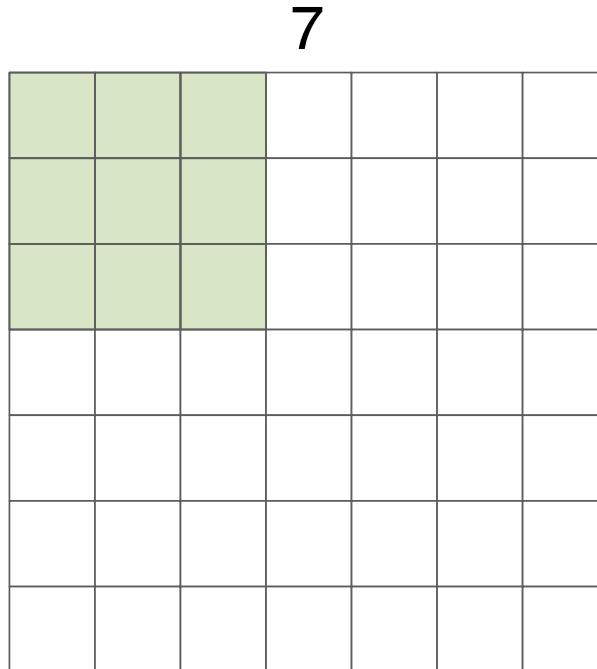


<http://cs231n.github.io/assets/conv-demo/index.html>



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



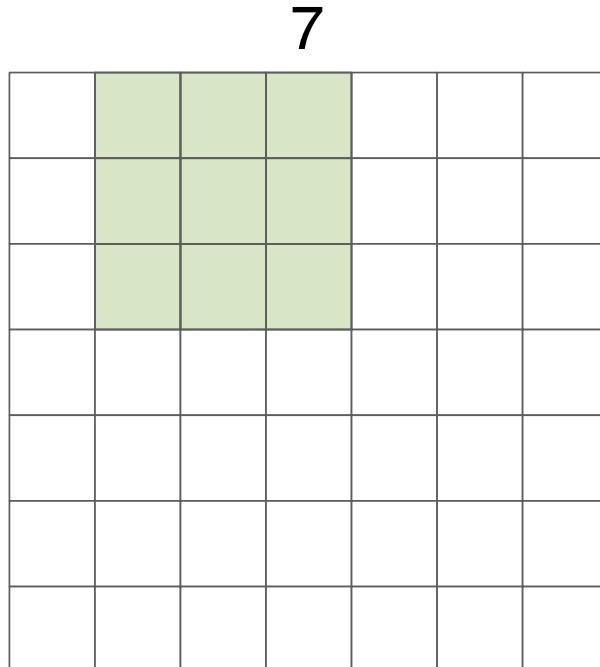
7x7x1 image

3x3x1 filter  $w$



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

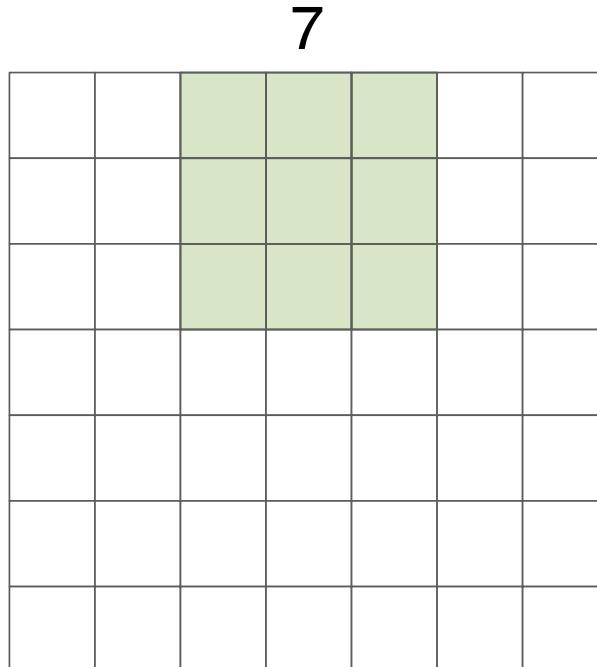
7

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

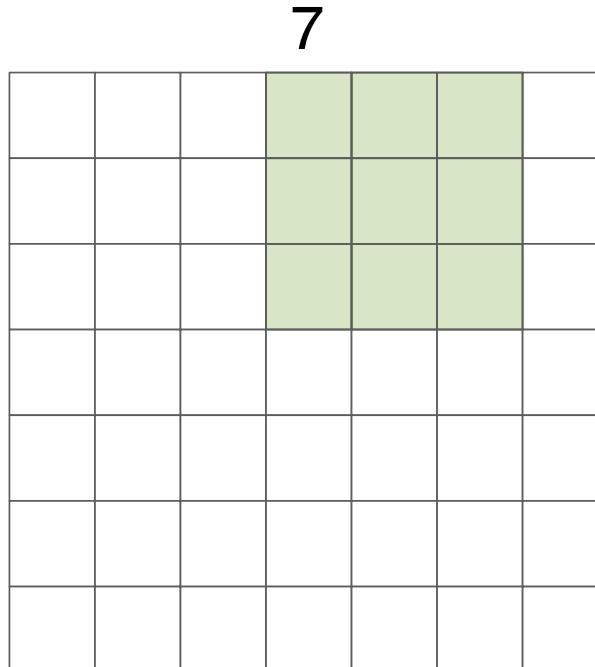
3x3x1 filter  $w$

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

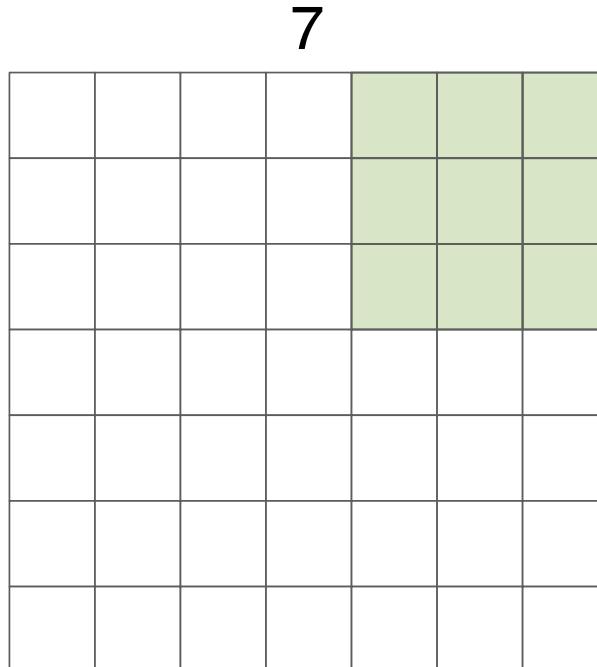
7

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

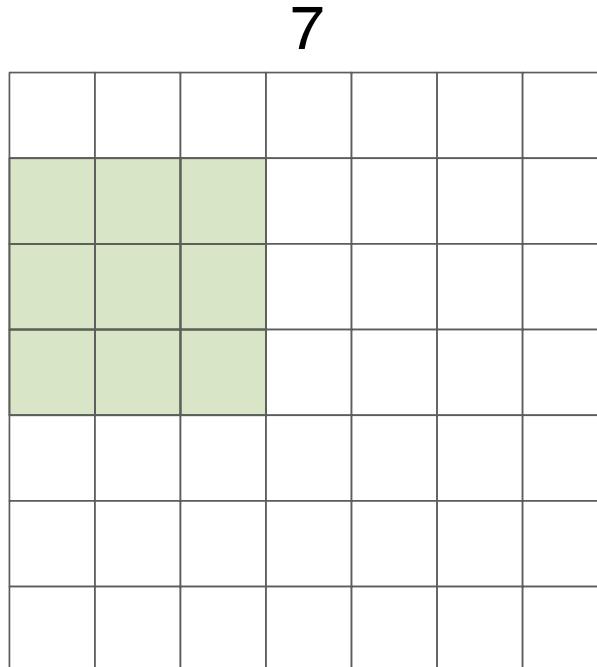
7

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

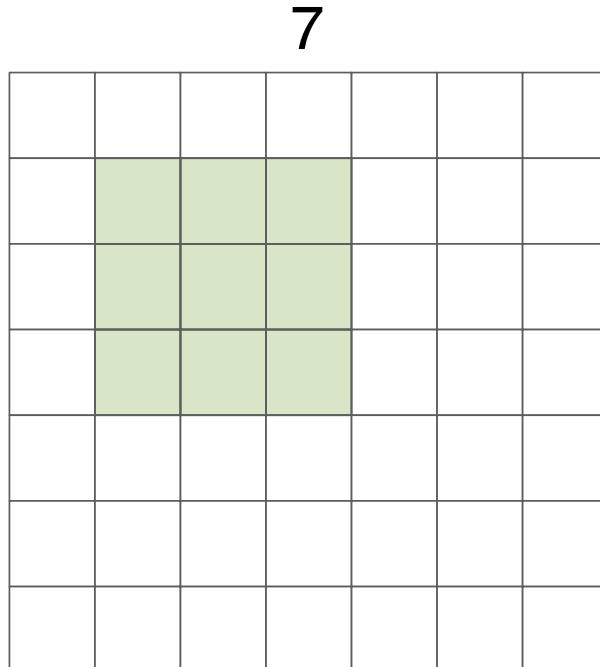
3x3x1 filter  $w$

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

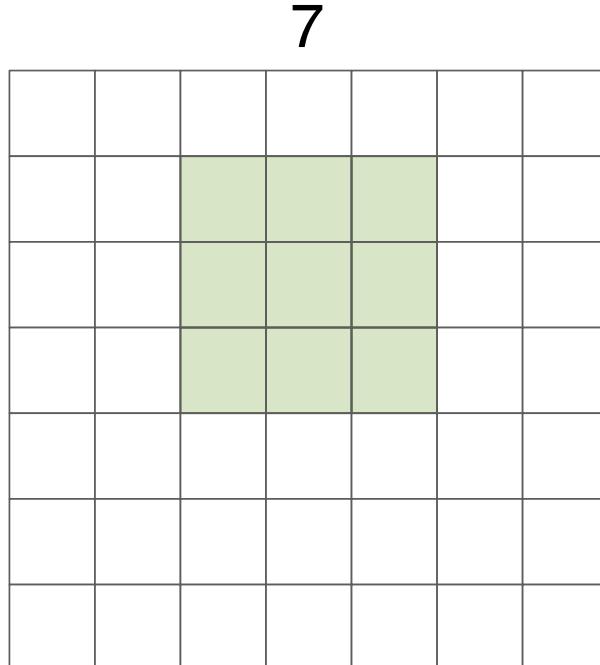
7

Schiebe den Filter um eine Position  
**(stride 1)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

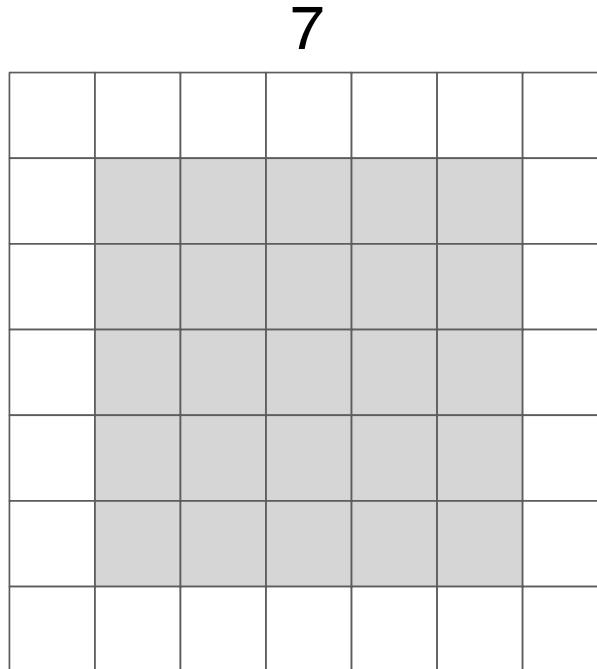
3x3x1 filter  $w$

Und so weiter ...



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

stride S=1

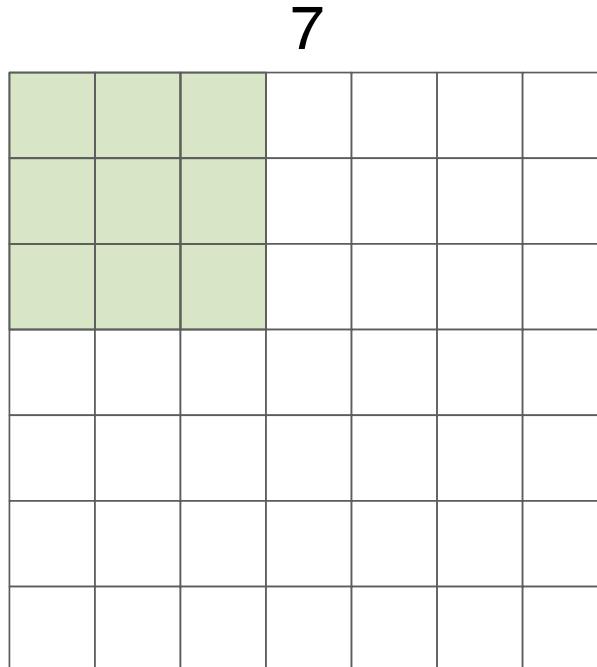
⇒ 5x5 Ausgabe

Aktivierungsfeld



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

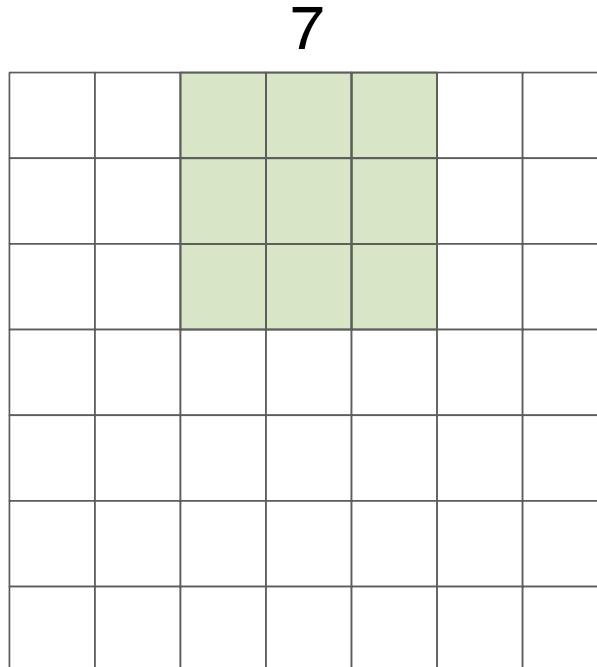
3x3x1 filter  $w$

Schiebe den Filter um zwei Positionen  
**(stride 2)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

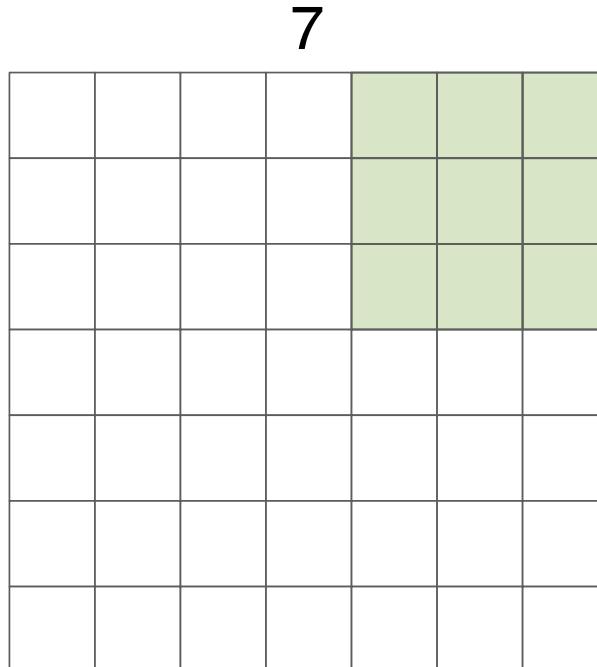
7

Schiebe den Filter um zwei Positionen  
**(stride 2)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

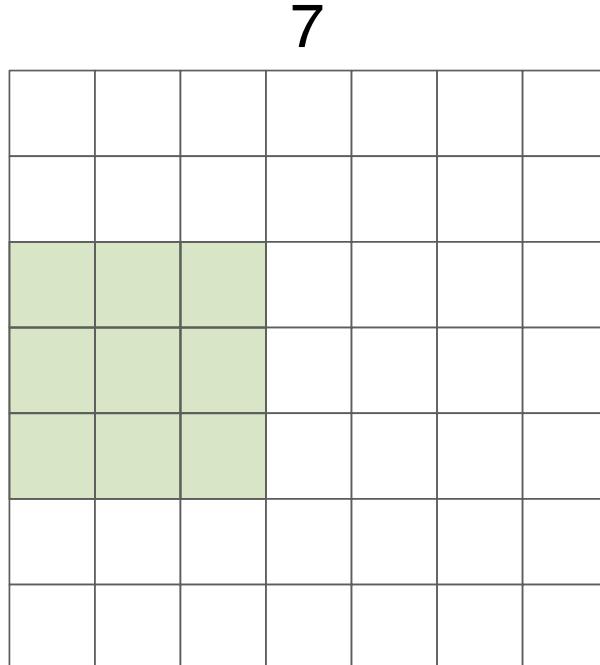
7

Schiebe den Filter um zwei Positionen  
**(stride 2)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

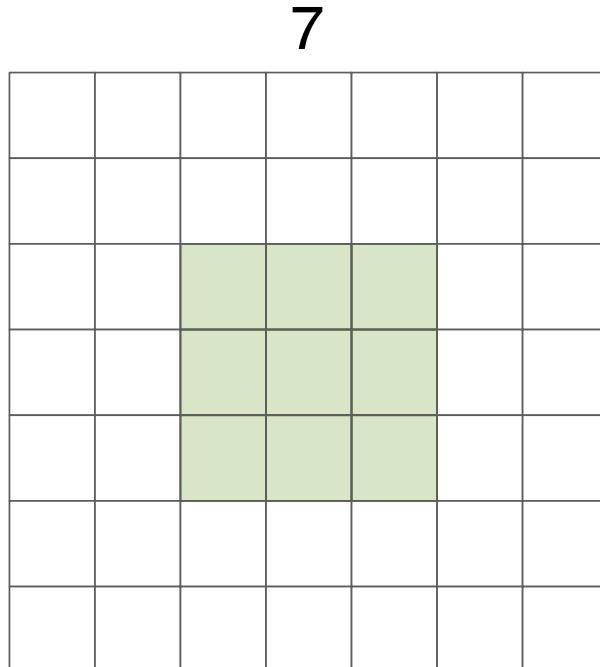
7

Schiebe den Filter um zwei Positionen  
**(stride 2)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

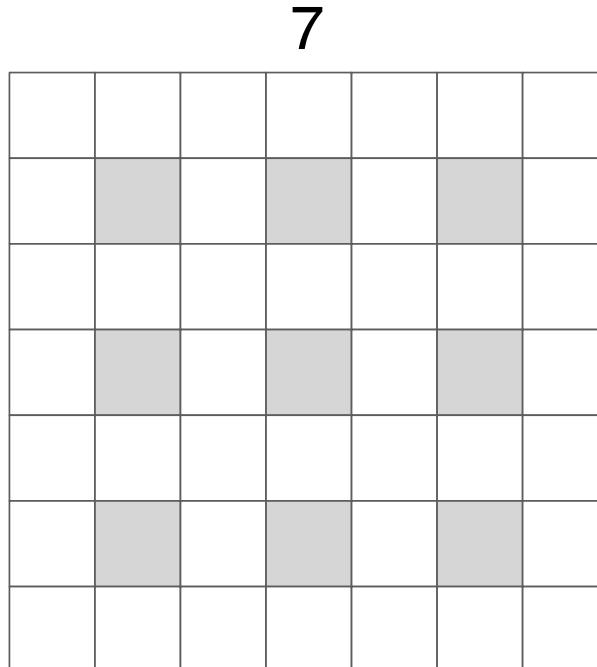
3x3x1 filter  $w$

Schiebe den Filter um zwei Positionen  
**(stride 2)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

stride S=2

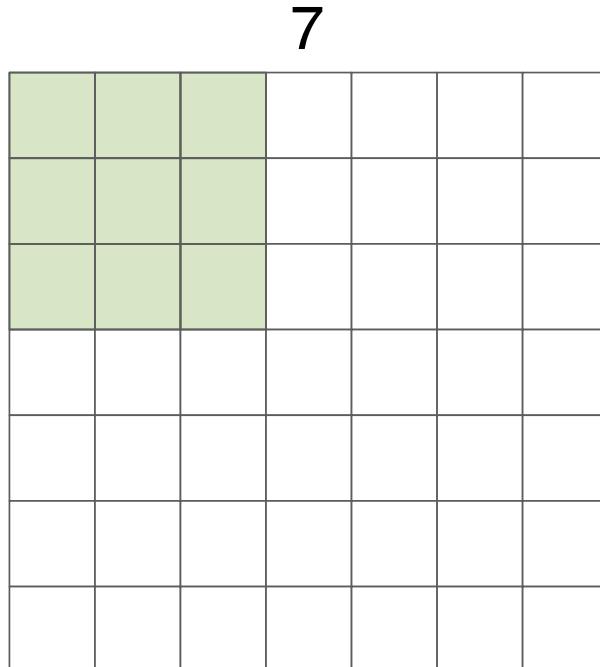
⇒ 3x3 Ausgabe

Aktivierungsfeld



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

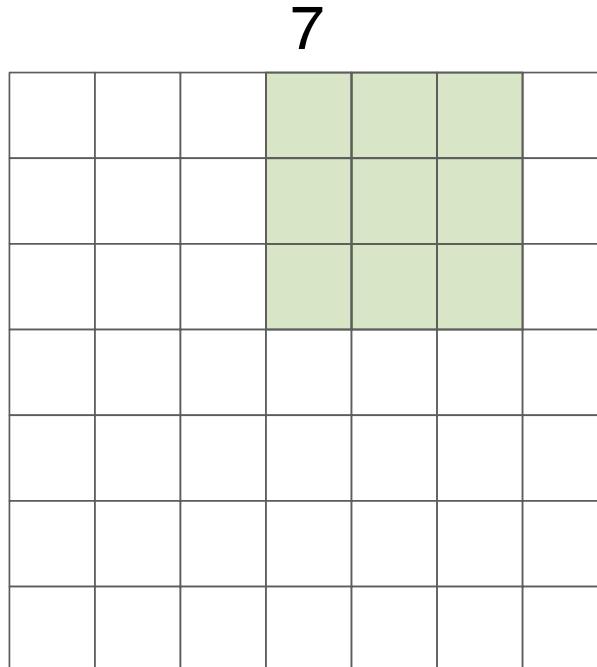
stride S=3

Schiebe den Filter um drei Positionen  
**(stride 3)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

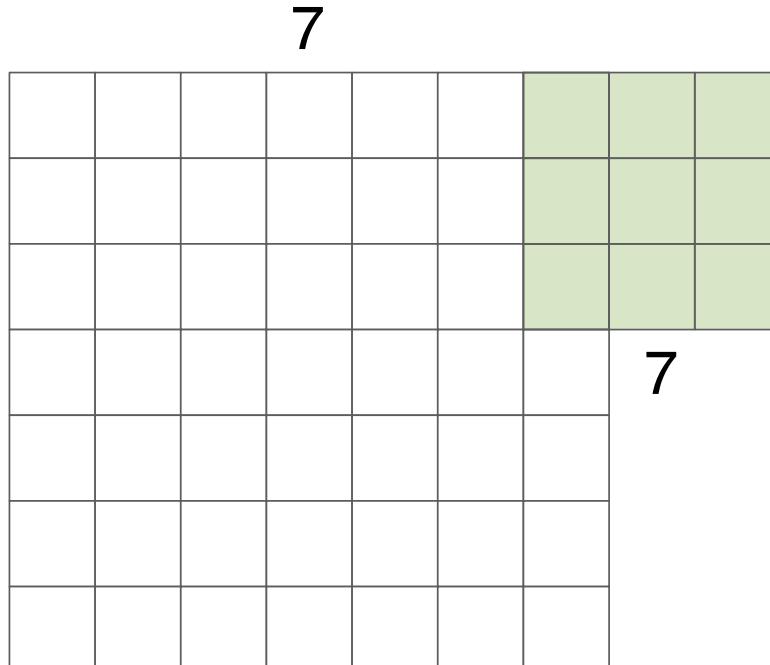
stride S=3

Schiebe den Filter um drei Positionen  
**(stride 3)** nach rechts (nach unten)



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?



7x7x1 image

3x3x1 filter  $w$

7  
stride S=3

Passt nicht!!!

Wir können keinen 3x3x1 Filter auf ein 7x7x1 Bild mit Stride 3 anwenden!



# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?

**Füge ein zero padding am Rand ein**

9	0	0	0	0	0	0	0	0	0
0				7					0
0									0
0									0
0						7	0		
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

7x7x1 image

3x3x1 filter  $w$

stride S=3

Padding = 1

$\Rightarrow$  3x3 Ausgabe

Aktivierungsfeld



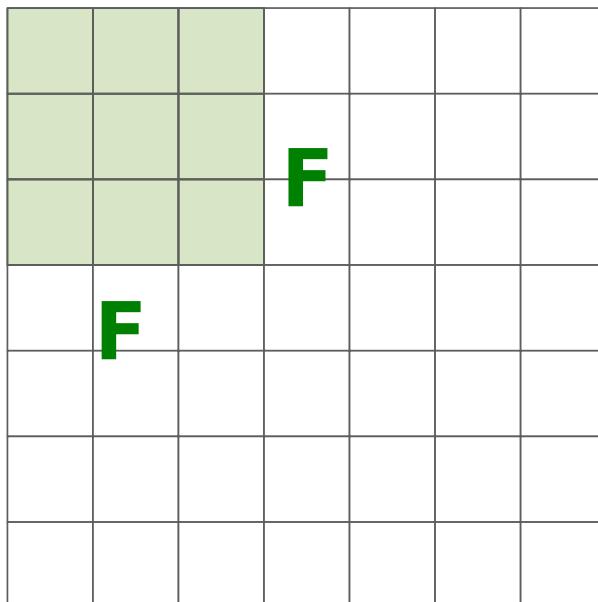
# Räumliche Auflösung

Wie verhält sich die Auflösung bei der Faltung?

Räumliche Auflösung der Ausgabe

$$\frac{I - F + 2P}{S} + 1$$

**I**



**IxIx d** input

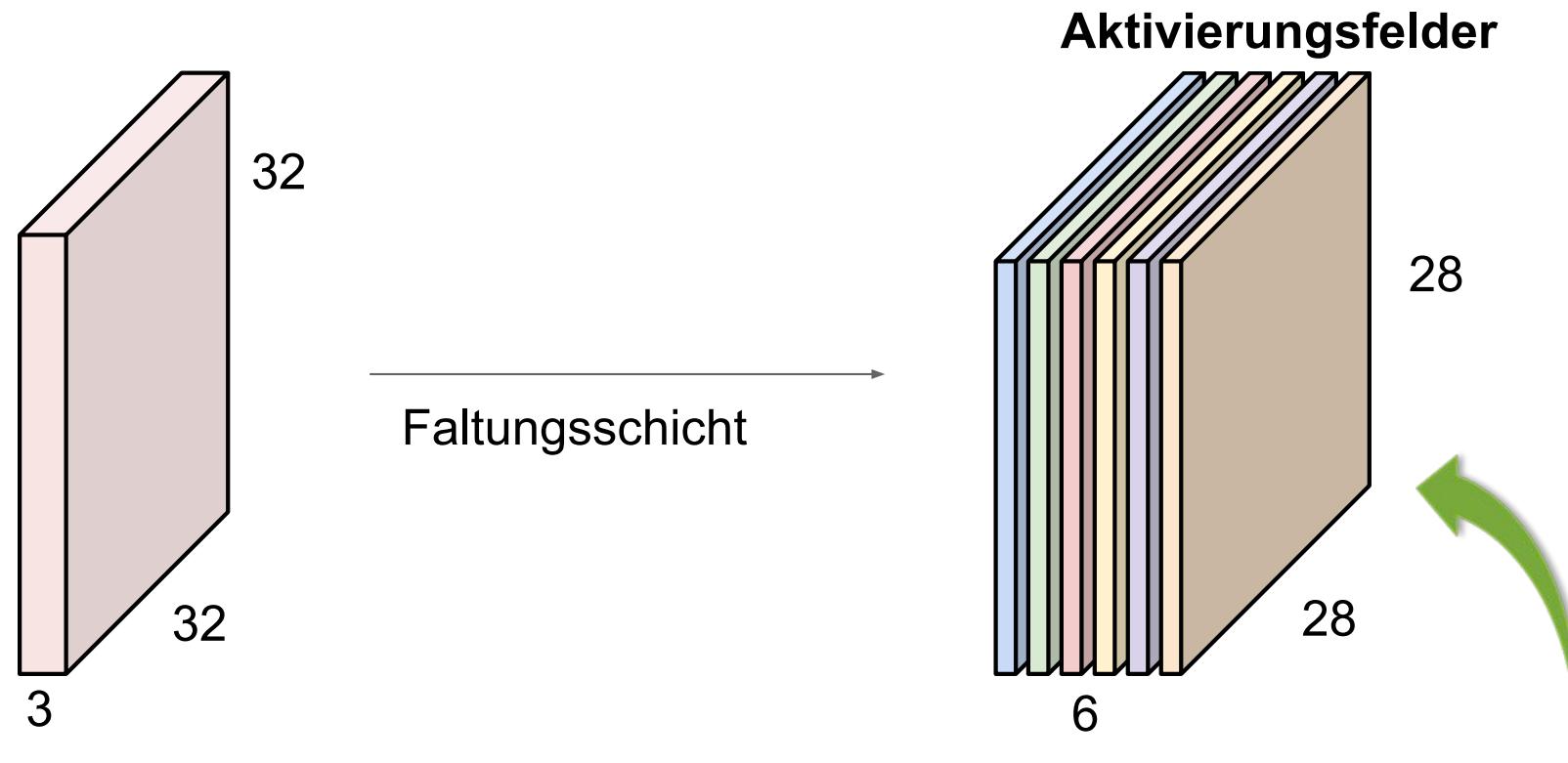
**FxFxd** filter w

**I** stride S

**padding P**

## Zurück zu Faltungsschichten

Wenn wir z.B. 6 5x5x3 Filter benutzen, bekommen wir 6 Aktivierungsfelder



Diese legen wir übereinander und erhalten ein “neues Bild“ der Größe 28x28x6

$$\text{Räumliche Auflösung: } \frac{32 - 5 + 2 \cdot 0}{1} + 1 = 28$$

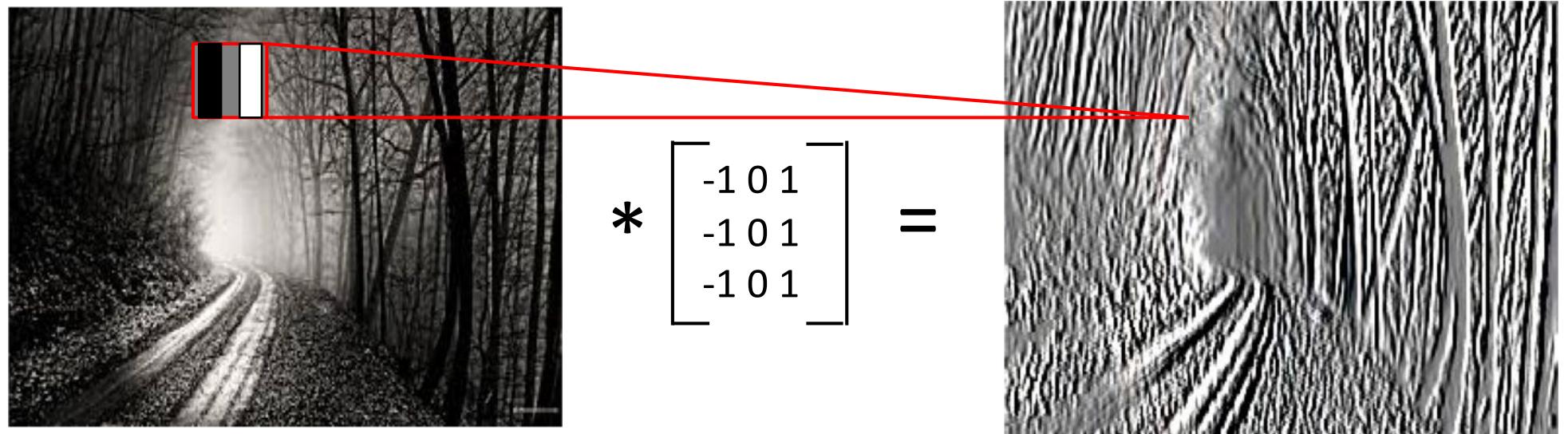


# Warum wird das Faltungsschicht genannt?

Weil es mit der Faltung von zwei Signalen zu tun hat:

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

elementweise Multiplikation and Summe eines Filter und eines Signals (Bild)





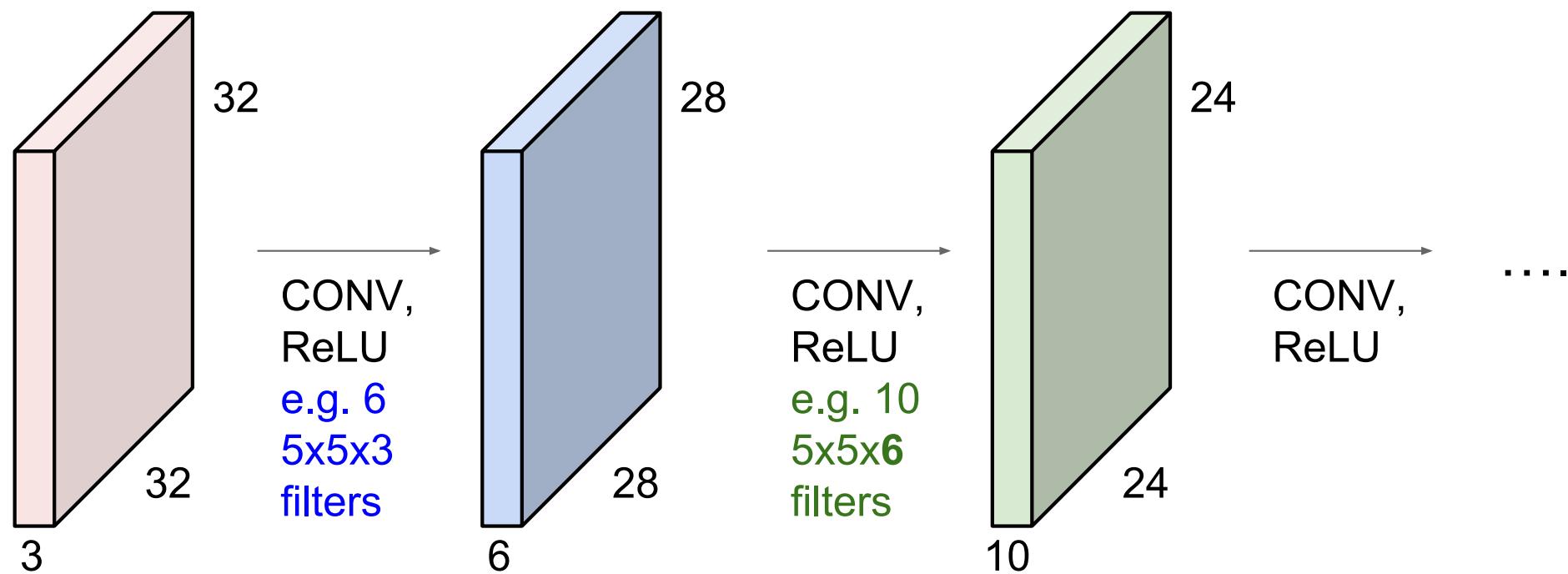
Einer der ersten Durchbrüche von DCNs war die

## BILDKLASSIFIKATION

- Faltungsschichten
- Nicht-lineare Aktivierungsfunktion ReLU
- Max pooling Schicht
- Vollständig verbundene Schicht

# Convolutional Network (ConvNet)

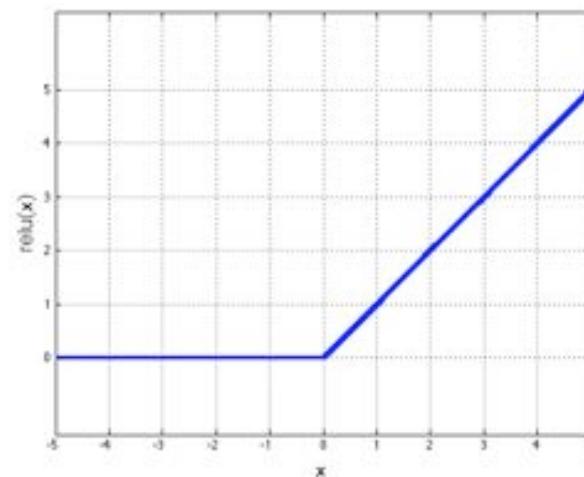
Ein ConvNet ist eine Abfolge von Faltungsschichten, die durch Aktivierungsfunktionen miteinander verbunden werden



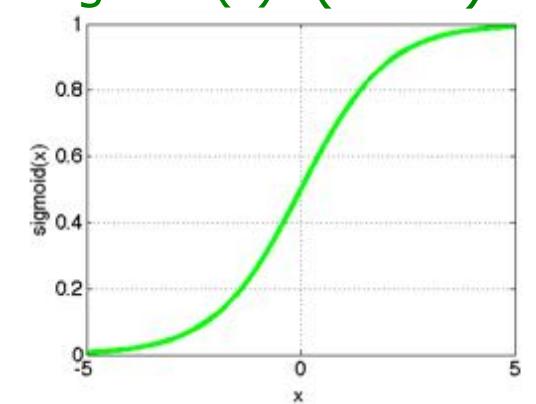
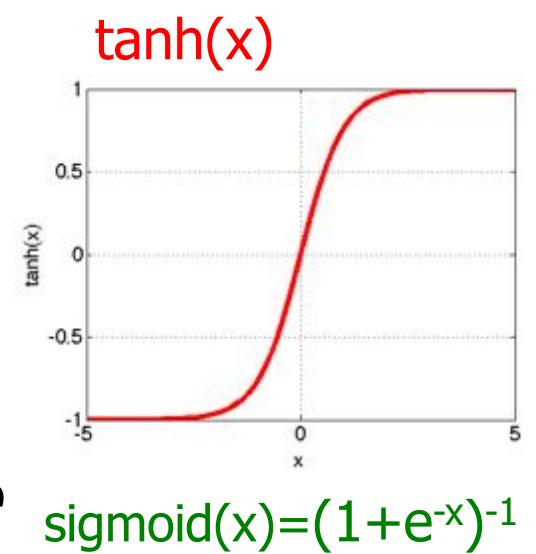
# Rectified Linear Unit (ReLU, Gleichrichter)

- Nicht-lineare Aktivierungsfktn. werden per Element angewendet
- Rectified linear unit (**ReLU**):

- $\max(0, x)$
- Beschleunigt das Lernen (empirisch  $\times 6$ )
- Vereinfacht das Lernen mittels Backprop.
- Vermeidet "Sättigungsproblem"  
(im Gegensatz zu sigmoid, tanh)
- preferred option (works well)



Andere Aktivierungsfktn.



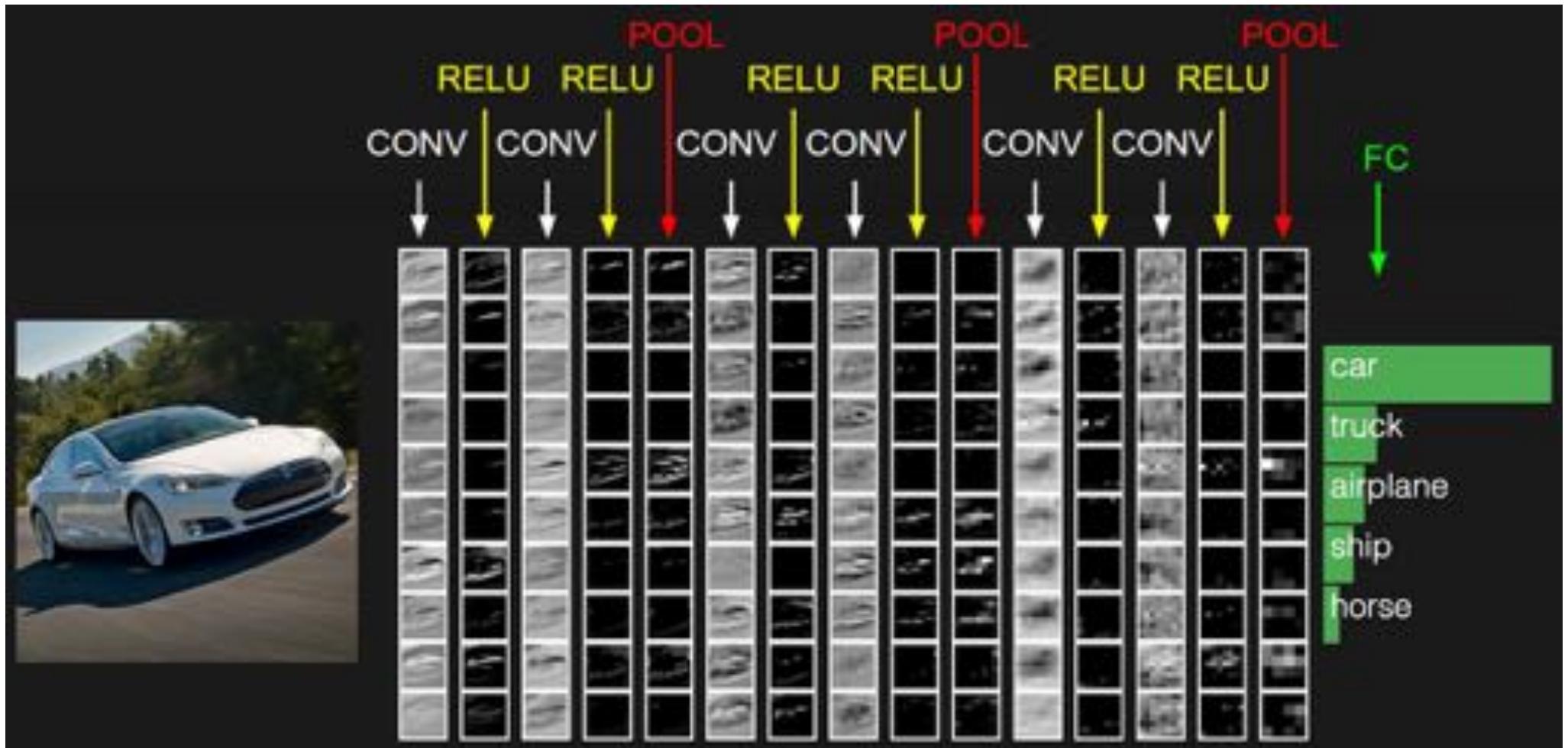


Einer der ersten Durchbrüche von DCNs war die

## BILDKLASSIFIKATION

- Faltungsschichten
- Nicht-lineare Aktivierungsfunktion ReLU
- Max pooling Schicht
- Vollständig verbundene Schicht

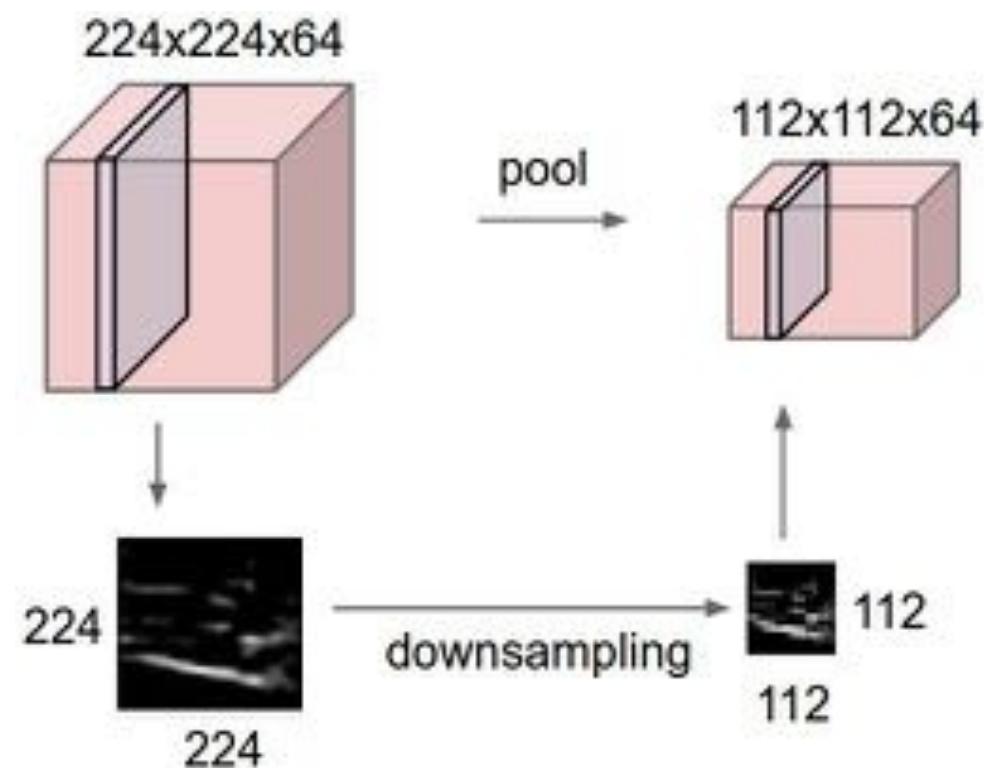
# Wo finden Zusammenfassungen (Pooling) statt?



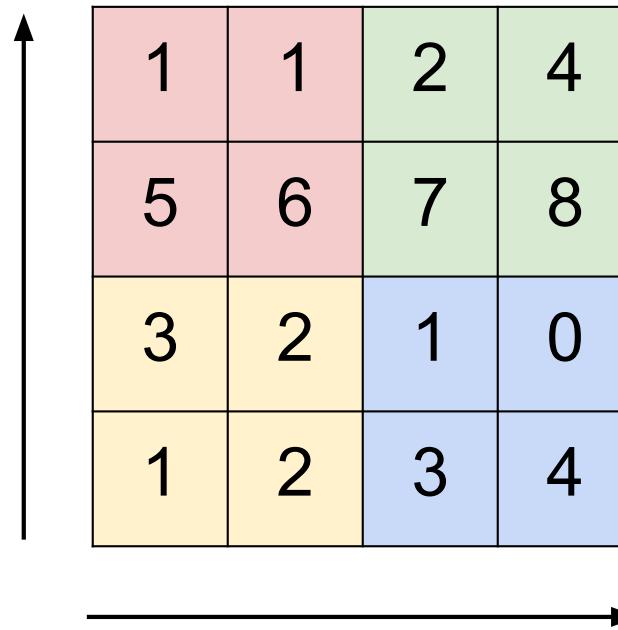
# Räumliche Zusammenfassung (spatial pooling)



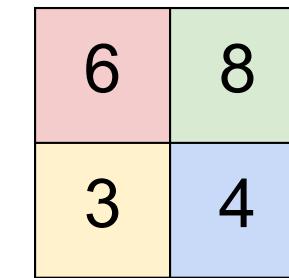
Pooling Schichten komprimieren die Repräsentation (downsampling/  
Unterabtastung). Sie werden auf jedes Aktivierungsfeld unabhängig  
angewendet und sollen das Netzwerk invariant zu kleineren  
Transformationen machen



# Max Pooling



Max pooling mit 2x2  
Filter und stride 2



An output matrix resulting from max pooling with a 2x2 filter and stride 2:

6	8
3	4

The output matrix has two pink cells (top-left and top-right) and two yellow cells (bottom-left and bottom-right).

Alternativen: Sum Pooling, überlappendes Pooling

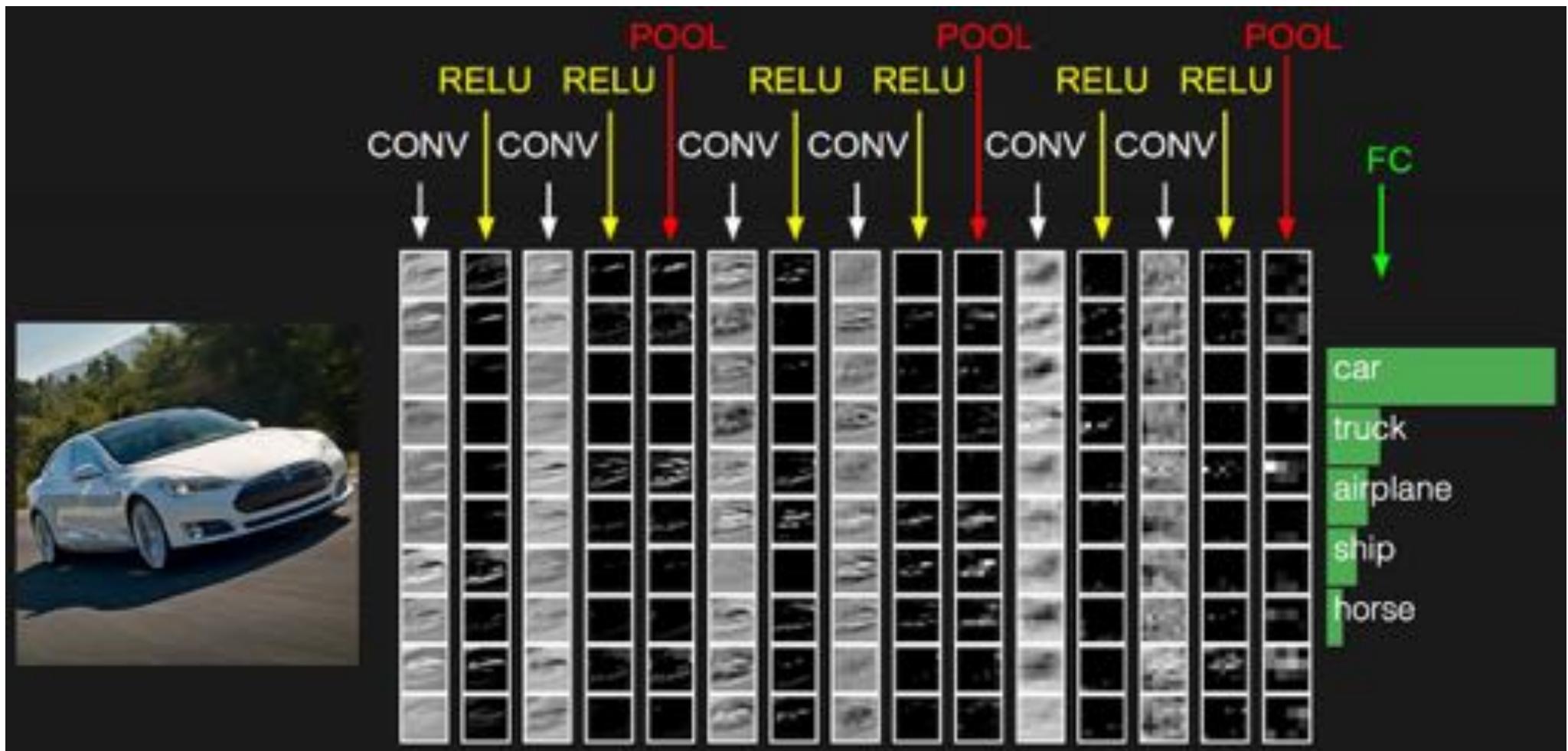


Einer der ersten Durchbrüche von DCNs war die

## BILDKLASSIFIKATION

- Faltungsschichten
- Nicht-lineare Aktivierungsfunktion ReLU
- Max pooling Schicht
- Vollständig verbundene Schicht

# Zum Schluss kommt ein vollständig verbundenes Mehrschichtennetzwerk zur Klassifikation

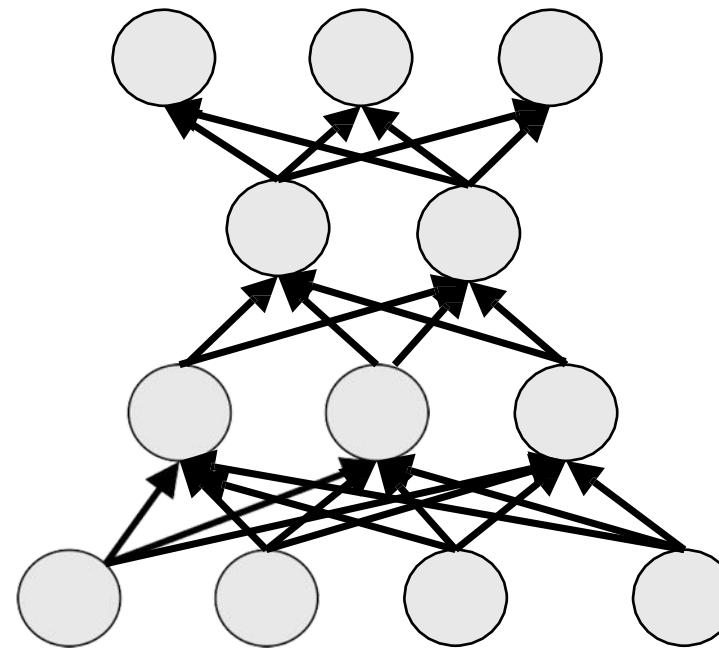


# Vollständig verbundes Mehrschichtnetzwerk

Ausgabeschicht

Unbeobachtete Schicht

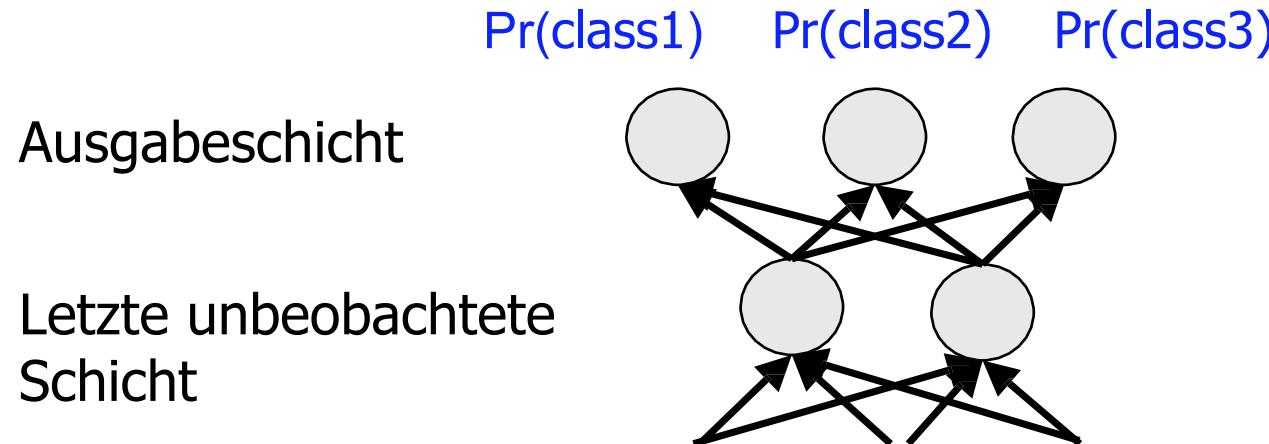
Unbeobachtete Schicht



**Alle Neuronen zweier aufeinander folgender Schichten sind paarweise verbunden. Neuronen innerhalb einer Schicht sind nicht verbunden**

# Vollständig verbundes Mehrschichtnetzwerk

Zur Klassifikation ist die letzte unbeobachtete Schicht vollständig mit einer **Ausgabeschicht** verbunden, die für jedes Klassenlabel ein Neuron hat. Die Aktivierungsfunktion ist Soft-Max



Demo: <https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>



## ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)

- 1K Kategorien
- 1.2M Trainingsbilder (~1000 pro Kategorie)
- 50,000 Validierungsbilder
- 150,000 Testbilder

Einer der ersten Durchbrüche von DCNs war die

# BILDKLASSIFIKATION

- Faltungsschichten
- Nicht-lineare Aktivierungsfunktion ReLU
- Max pooling Schicht
- Vollständig verbundene Schicht

# Zum Trainieren: (Mini-batch) Stochastic gradient descent (SGD)

Initialisiere die Parameter zufällig (auf intelligente Art und Weise)

Gehe über die Trainingsdaten (mehrmals):

- Ziehe zufällig** einen Datenpunkt (oder eine kleine Stichprobe)
- Propagiert Aktivierungen **vorwärts** von der Eingabeschicht zur Ausgabeschicht und berechne den Klassifikationsfehler, z.B.

$$E = \frac{1}{2} (y_{predicted} - y_{true})^2$$

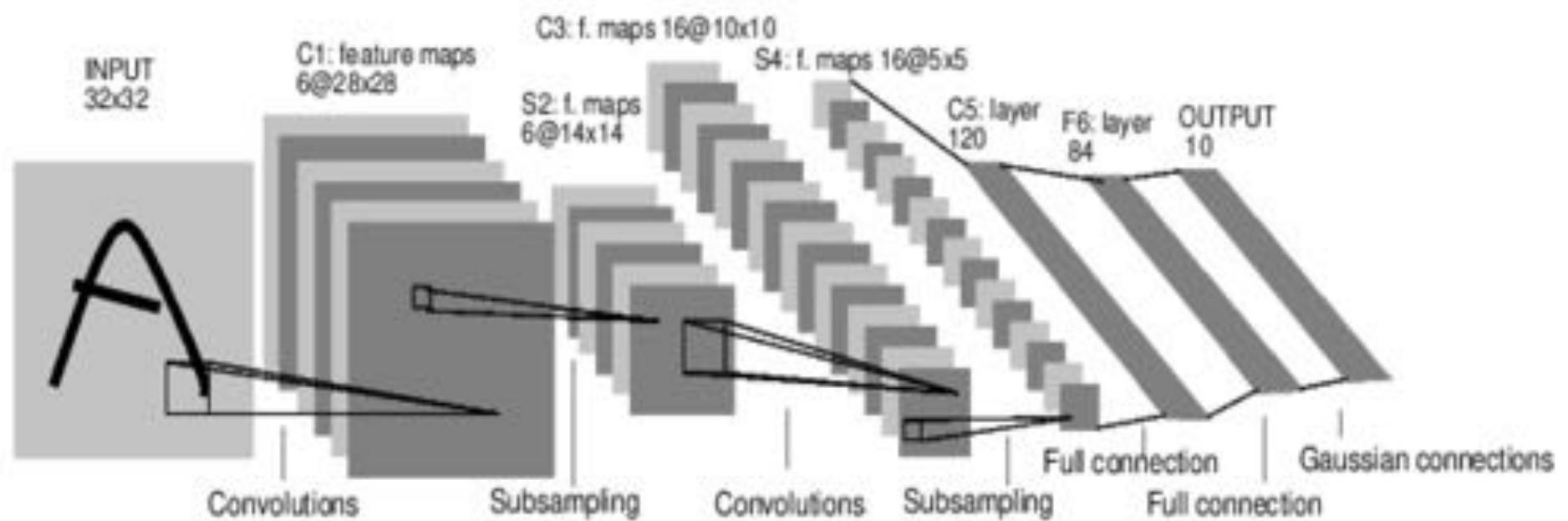
- Melde den Fehler **zurück**, d.h., berechne den Gradient der Fehlerfunktion für die Parameter mittels Backprop
- Update** die Parameter entsprechend des Gradientens, **SGD**:

$$w^{t+1} = w^t - \alpha \cdot \frac{dE}{dw}(w^t)$$

Und viele, viele Tricks, die wir hier nicht weiter besprechen

# Beispiel: LeNet-5

[LeCun et al., 1998]



Conv Filter mit 5x5 und stride 1

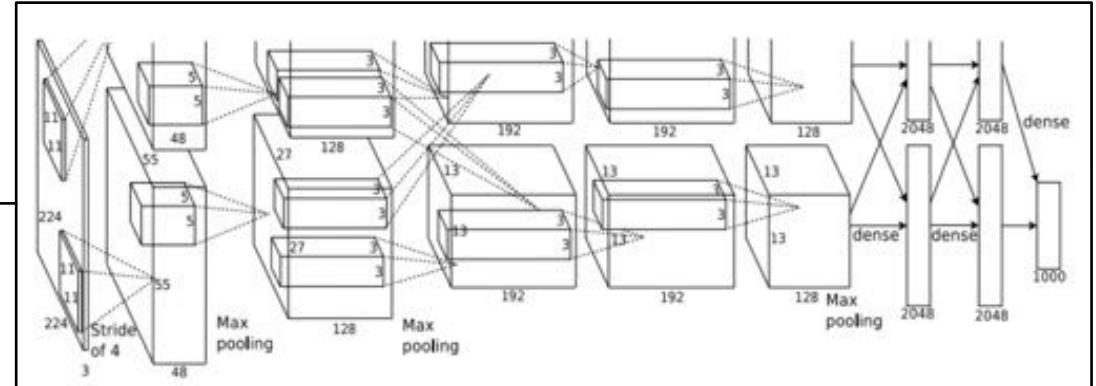
2x2 Pooling Schichten mit stride 2

Tanh Nicht-Linearitäten angewendet auf die Filter-Ausgaben, d.h. die Architektur ist

[CONV-POOL-CONV-POOL-CONV-FC]

# Beispiel: AlexNet

[Krizhevsky et al., 2012]



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

Input: 227x227x3 images

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

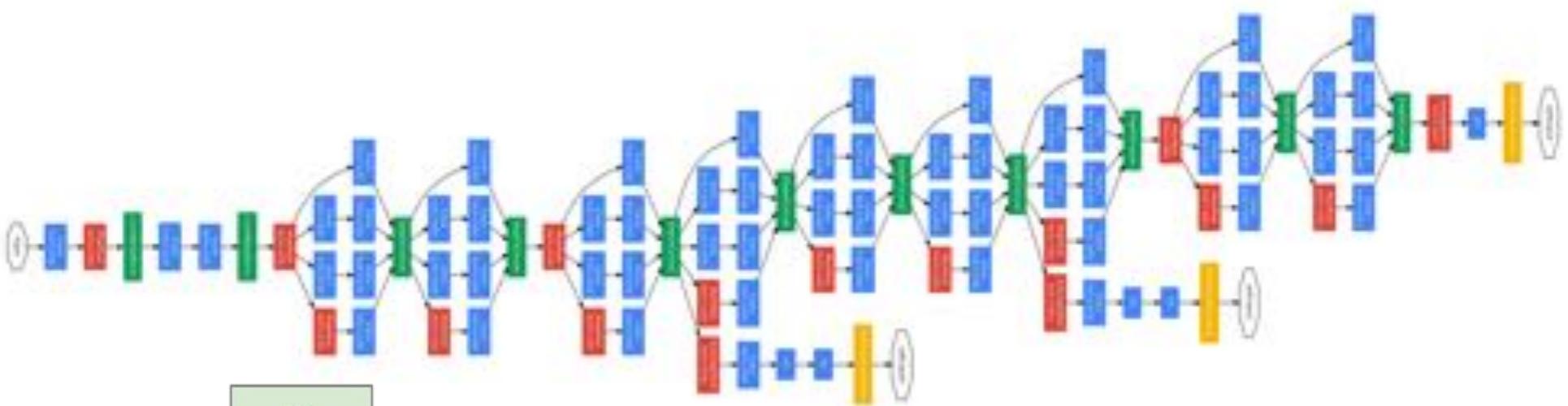
[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

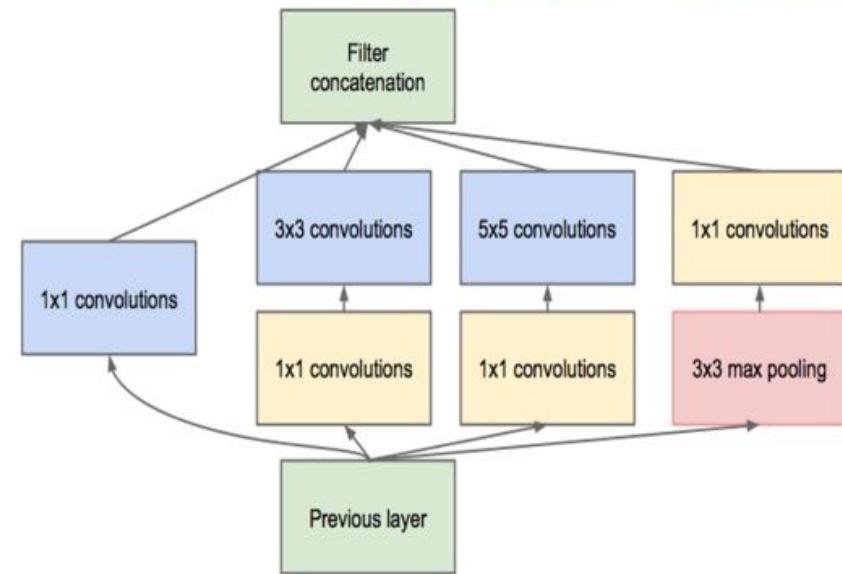
[1000] FC8: 1000 neurons (class scores)

# Beispiel: GoogLeNet

[Szegedy et al., 2014]



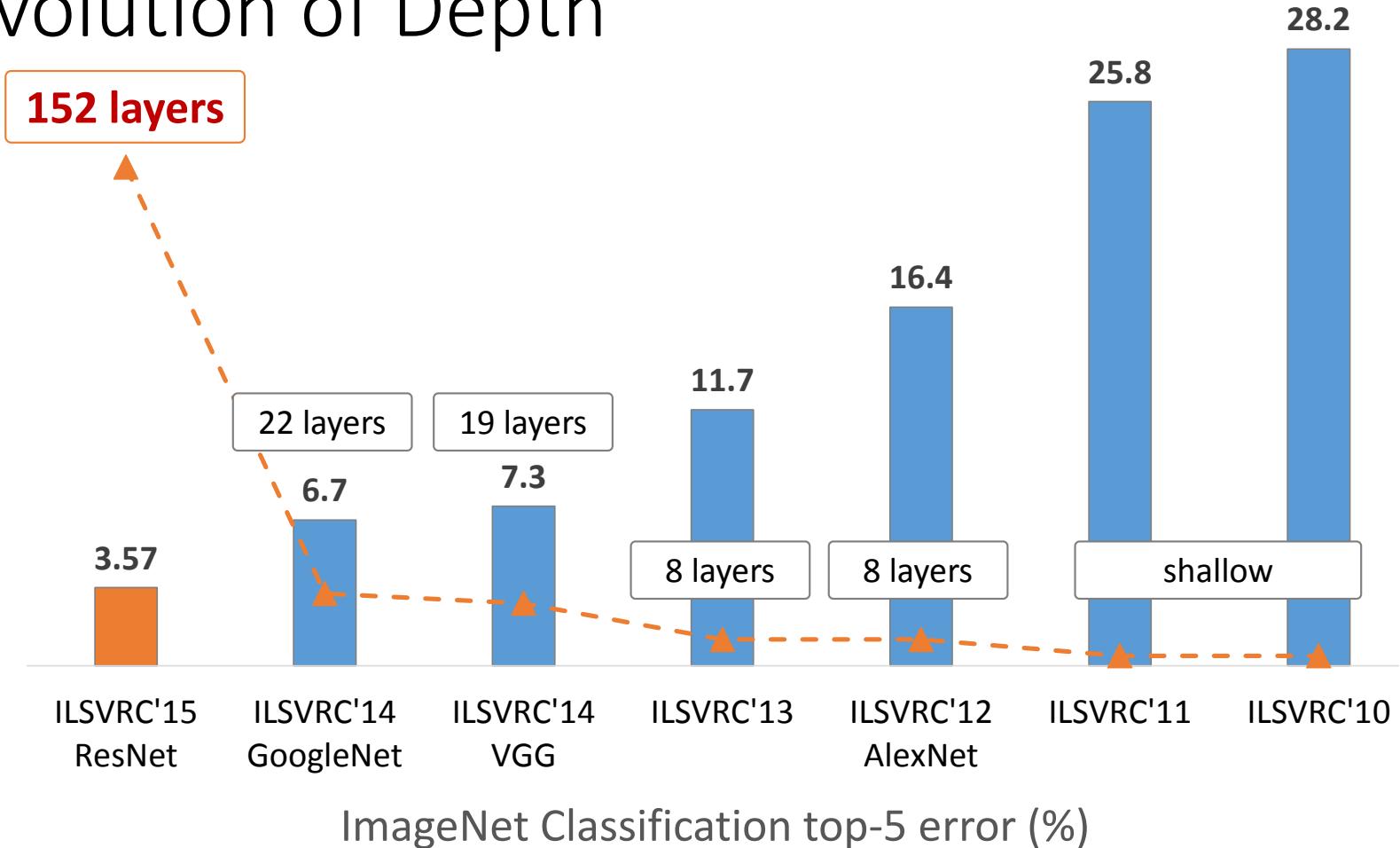
Inception Modul



ILSVRC 2014 Gewinner (6.7% top 5 error)

# Die Deep Learning Revolution

## Revolution of Depth





# Was haben wir gelernt?

- Tiefen neuronale Netzwerke lernen Merkmals hierarchien
- Wir haben eine der wichtigsten Standardarchitekturen kennengelernt:  
Faltende neuronale Netzwerke:
  - Faltungs schicht, ReLU, Max Pooling, Vollständig verbundene Schicht
  - Wir haben auch gesehen, wie man die räumliche Auflösung berechnen kann und damit auch die Zahl der Parameter
  - Tiefe neuronale Netzwerke werden mittels stochastischem Gradient abstieg trainiert
  - Tiefe neuronale Netzwerke haben zwar Millionen von Parameter, zeigen aber oft State-of-the-art Performanz

# Wie lernt man mit tiefen Netzwerken umzugehen?

Yann Lecun : "DNNs require an interplay between intuitive insights, theoretical modeling, practical implementations, empirical studies, and scientific analyses"



Das heist, es gibt aktuell noch keine Kern-Prinzipien, die alles erklären. Tiefes Lernen ist in diesem Sinne eine Kunst.

# Typische Deep Learning Frameworks



OpenAI

