Medizinische Bildanalyse
Wintersemester 2024/25

UNIVERSITÄT BONN

Kapitel 6b: Bildanalyse mit CNNs

Prof. Dr.-Ing. Thomas Schultz

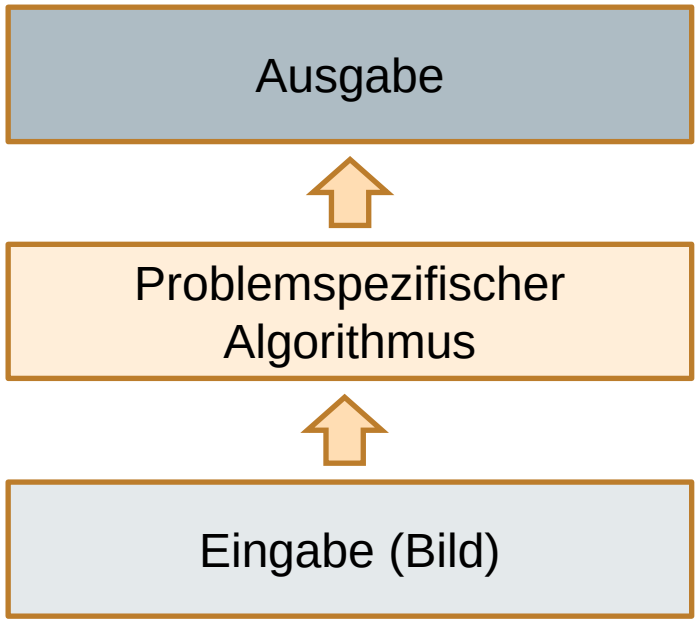
URL: http://cg.cs.uni-bonn.de/schultz/
E-Mail: schultz@cs.uni-bonn.de
Büro: Friedrich-Hirzebruch-Allee 6, Raum 2.117

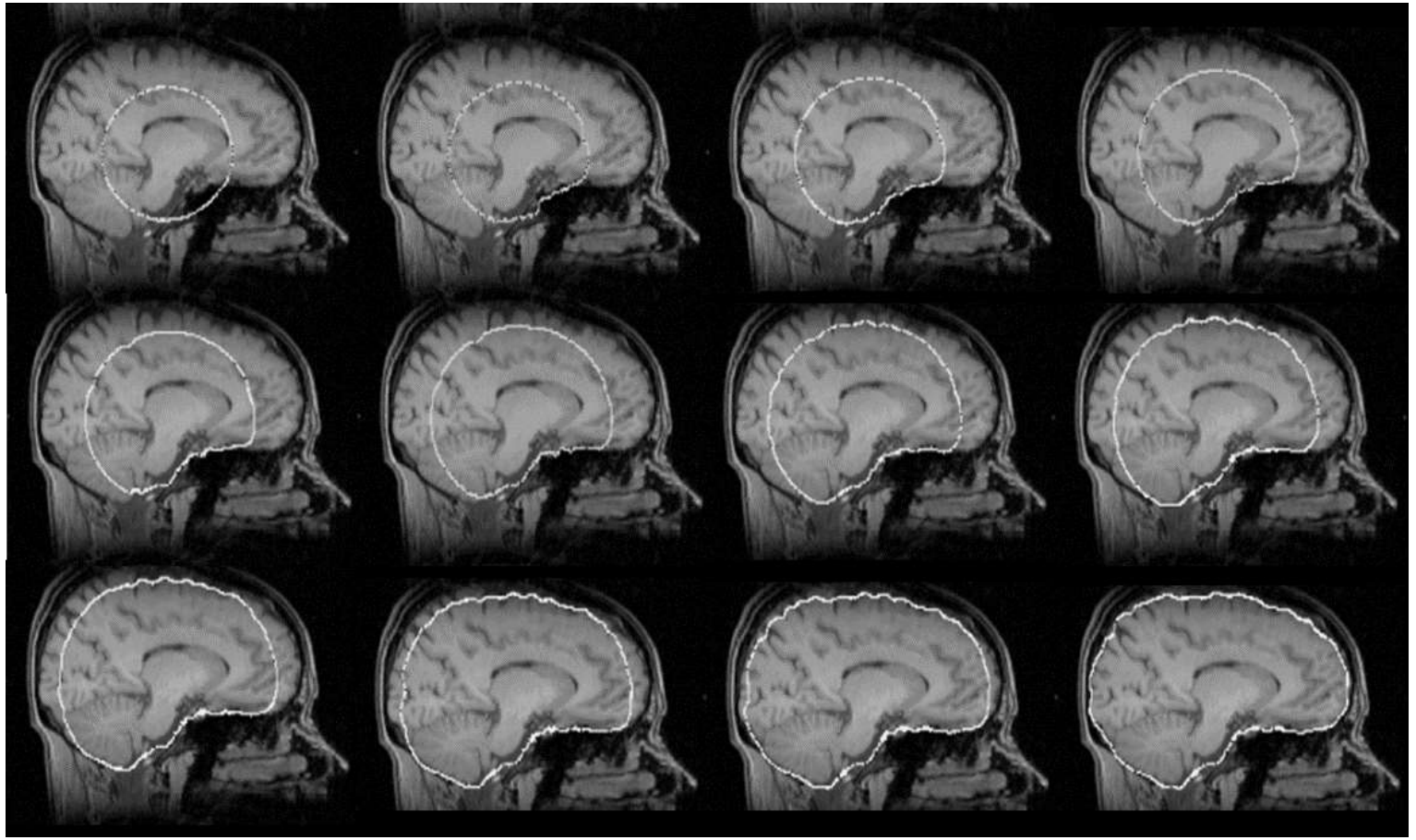
13./20./27. Januar 2025

b-it Bonn-Aachen International Center for Information Technology

6b.1 Einführung

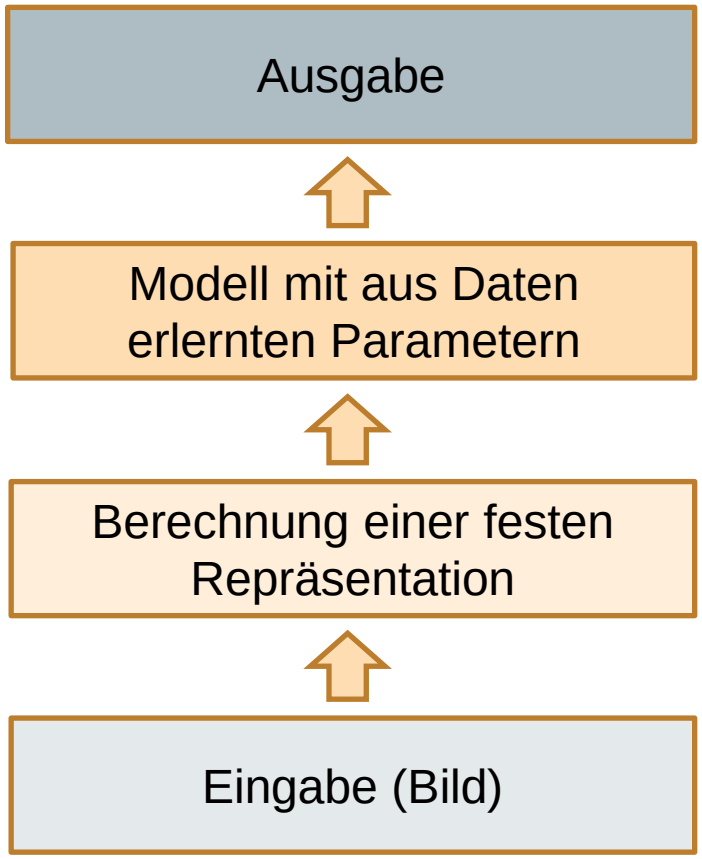
Regelbasierte Bildanalyse

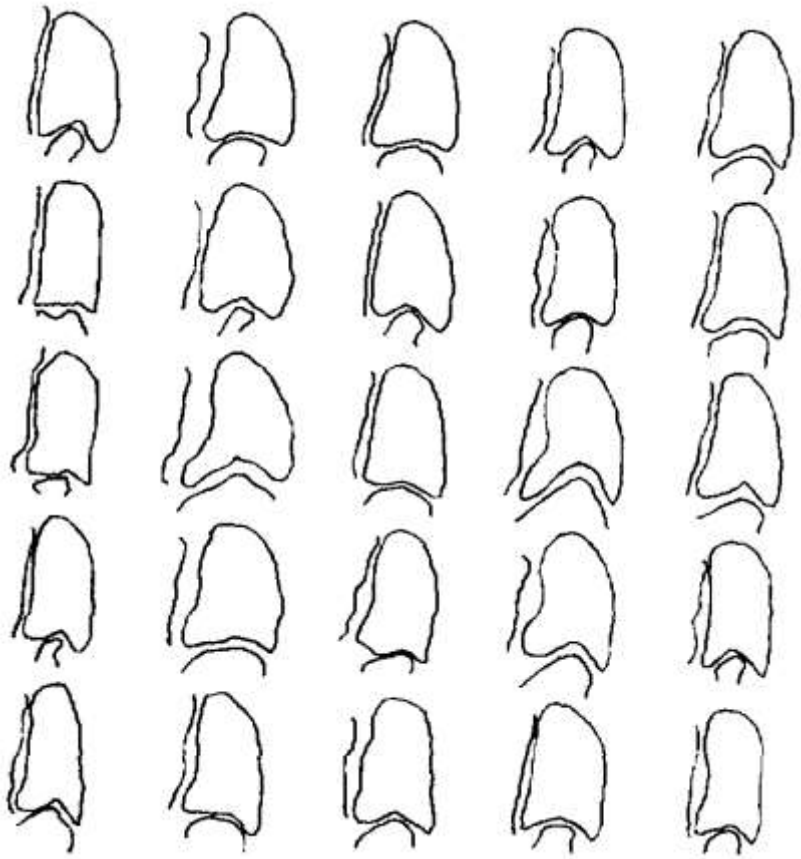


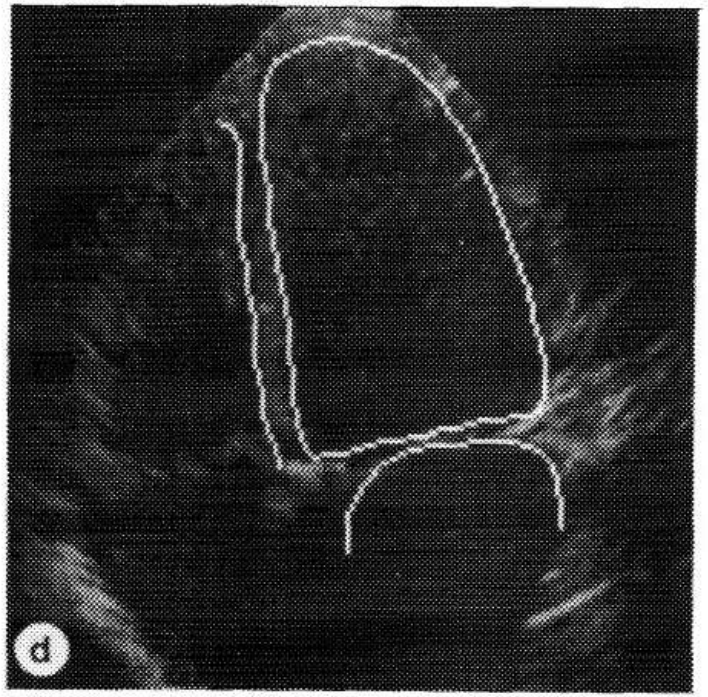


Bildquelle: S. Smith: *Fast Robust Automated Brain Extraction.* Human Brain Mapping 17:143-155, 2002

Lernbasierte Bildanalyse

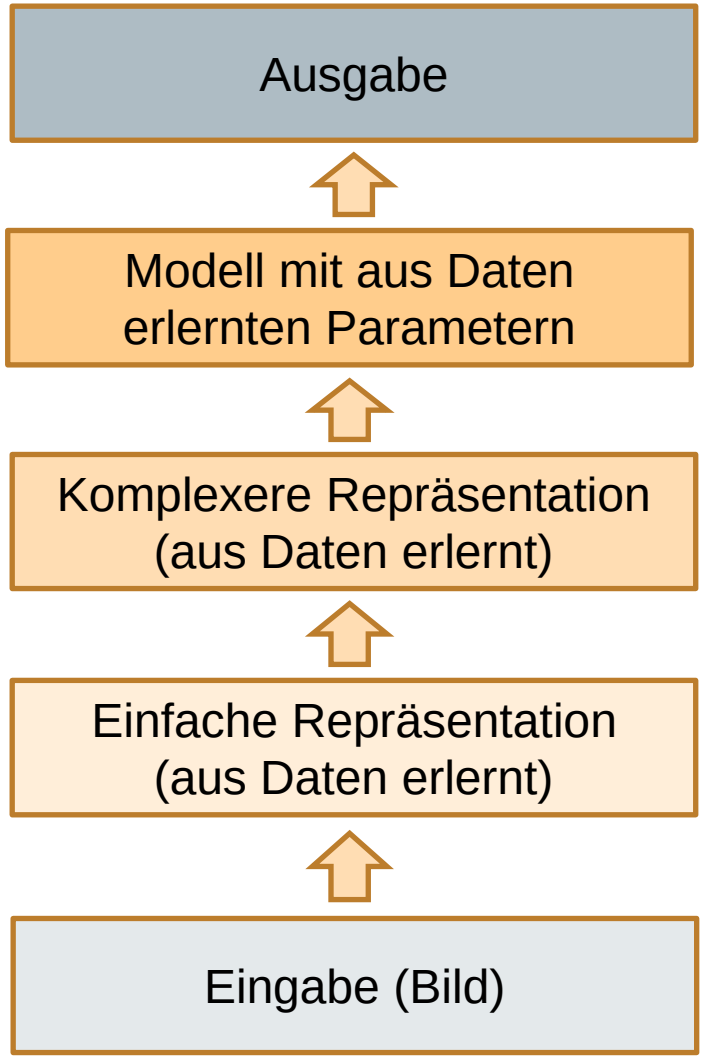


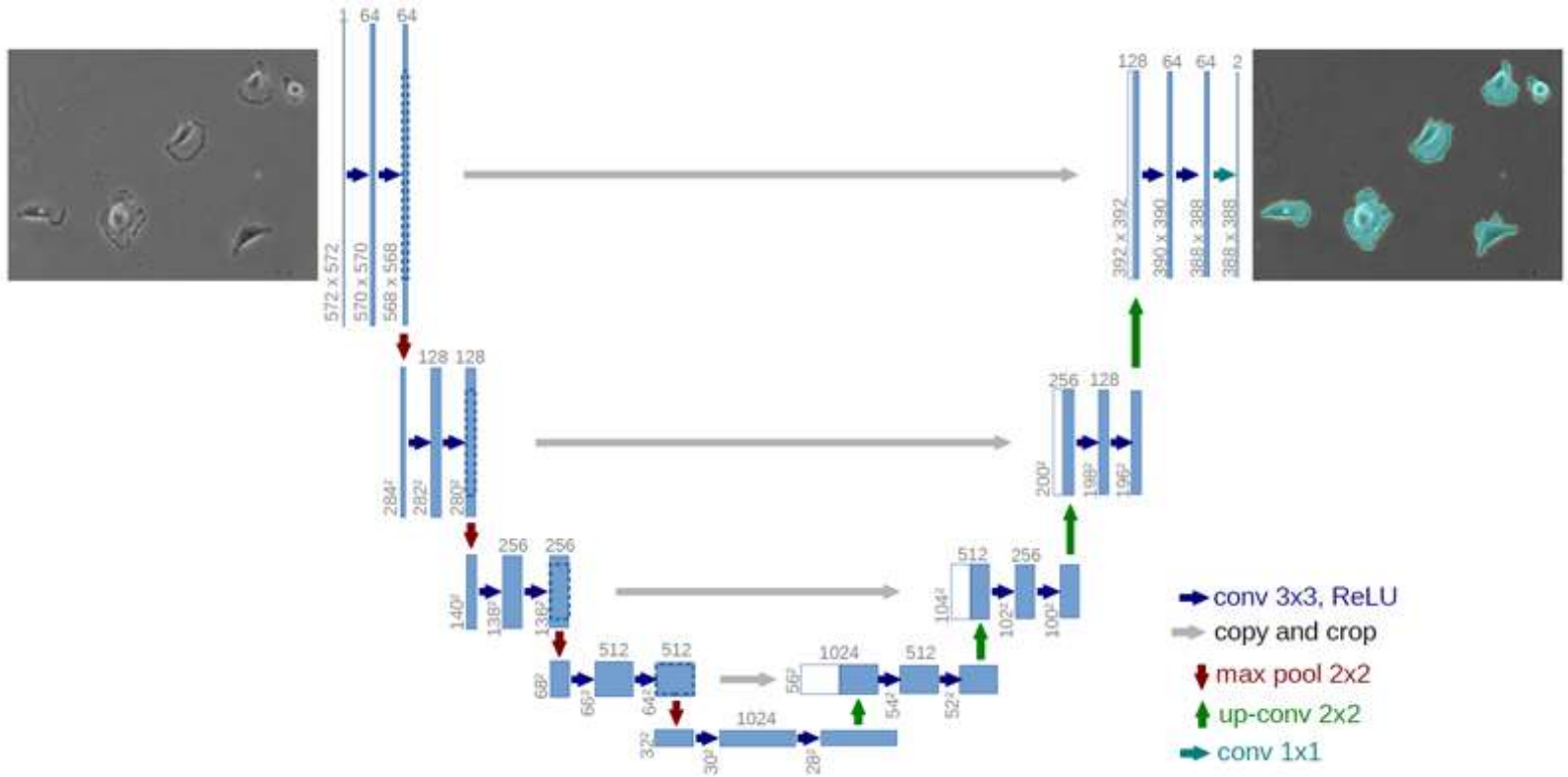




Bildquelle: Cootes et al., *Active Shape Models – Their Training and Application*. Computer Vision and Image Understanding, 1995

Bildanalyse mit Tiefem Lernen

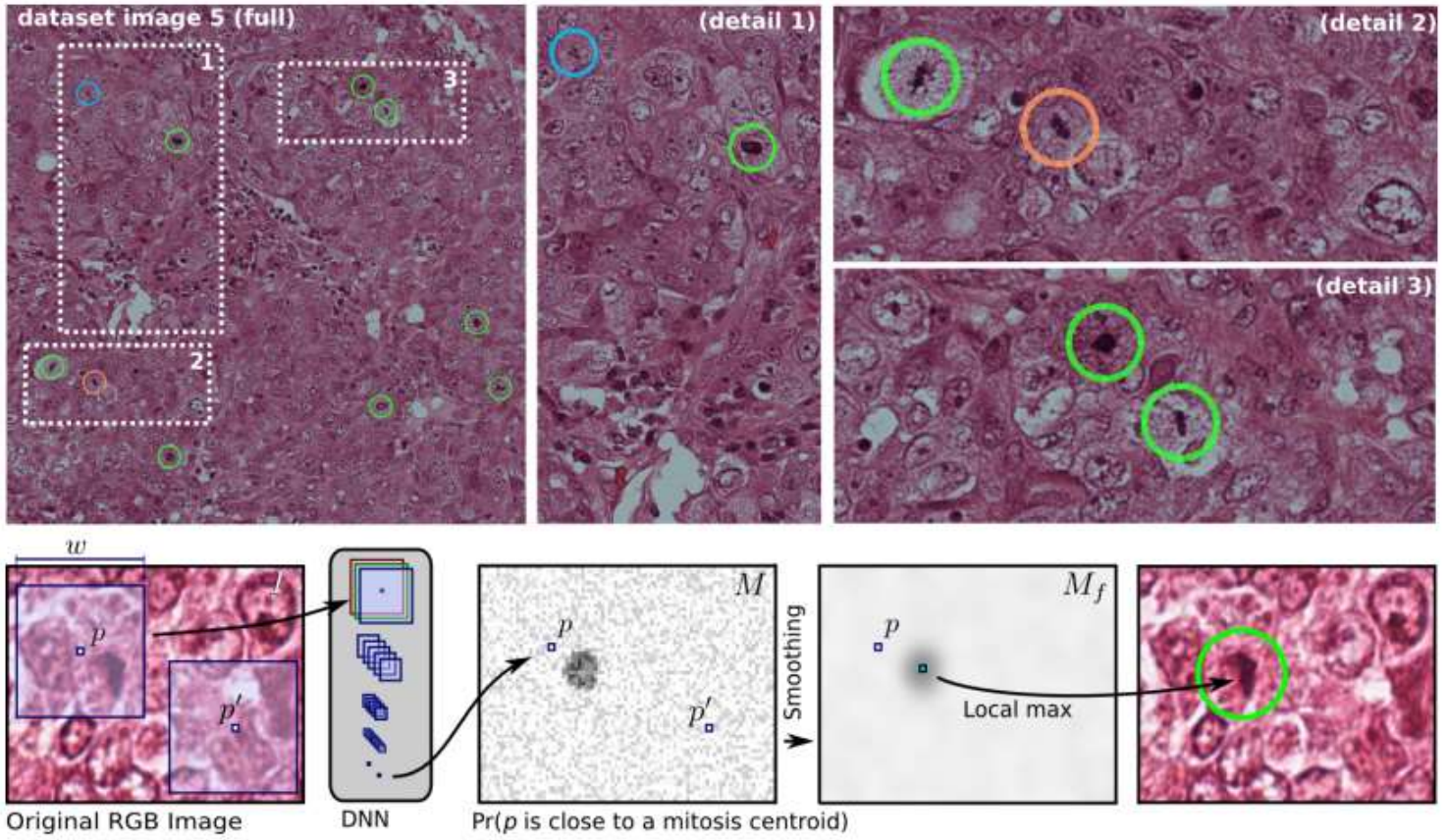




Bildquelle: O. Ronneberger, P. Fischer, T. Brox: *U-net: Convolutional networks for biomedical image segmentation.* MICCAI 2015.

Beispiel: Erkennung von Mitosen

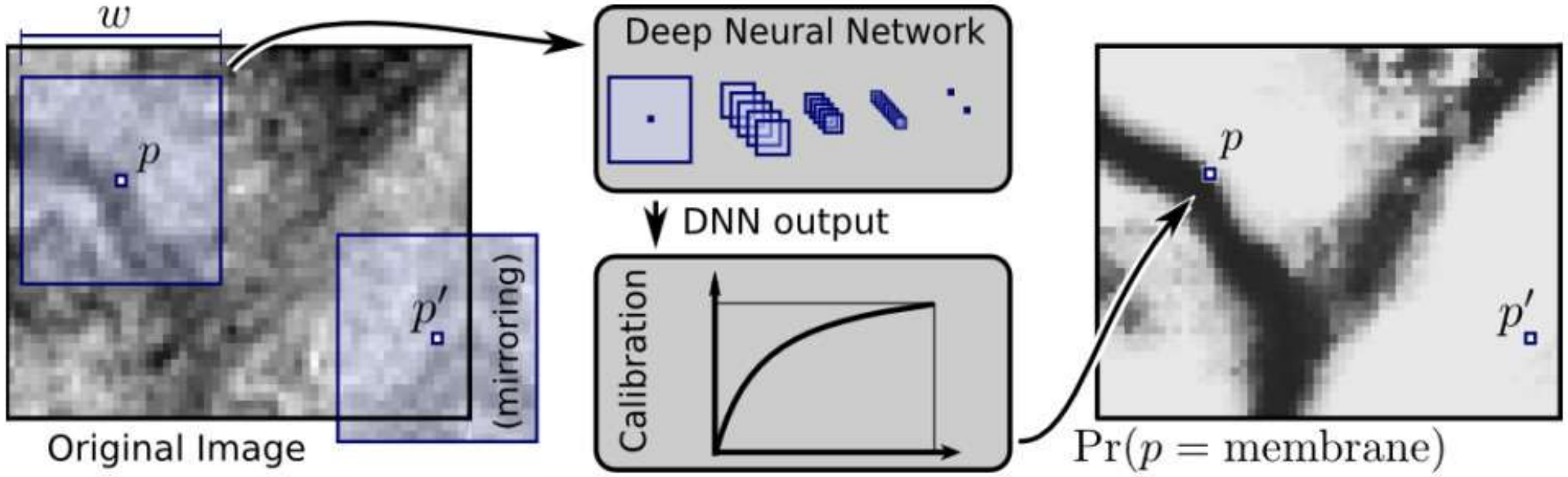
Meilenstein 2012a: Erstmals schlägt ein tiefes neuronales Netz in einem großen Wettbewerb zur *Objekterkennung* (Mitosen in histologischen Schnitten) mit Abstand alle traditionellen Verfahren



Bildquelle: Dan C. Cireşan et al., „Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks“ Medical Image Computing and Computer Assisted Intervention 2013, pp. 411-418

Beispiel: Segmentierung von Zellmembranen

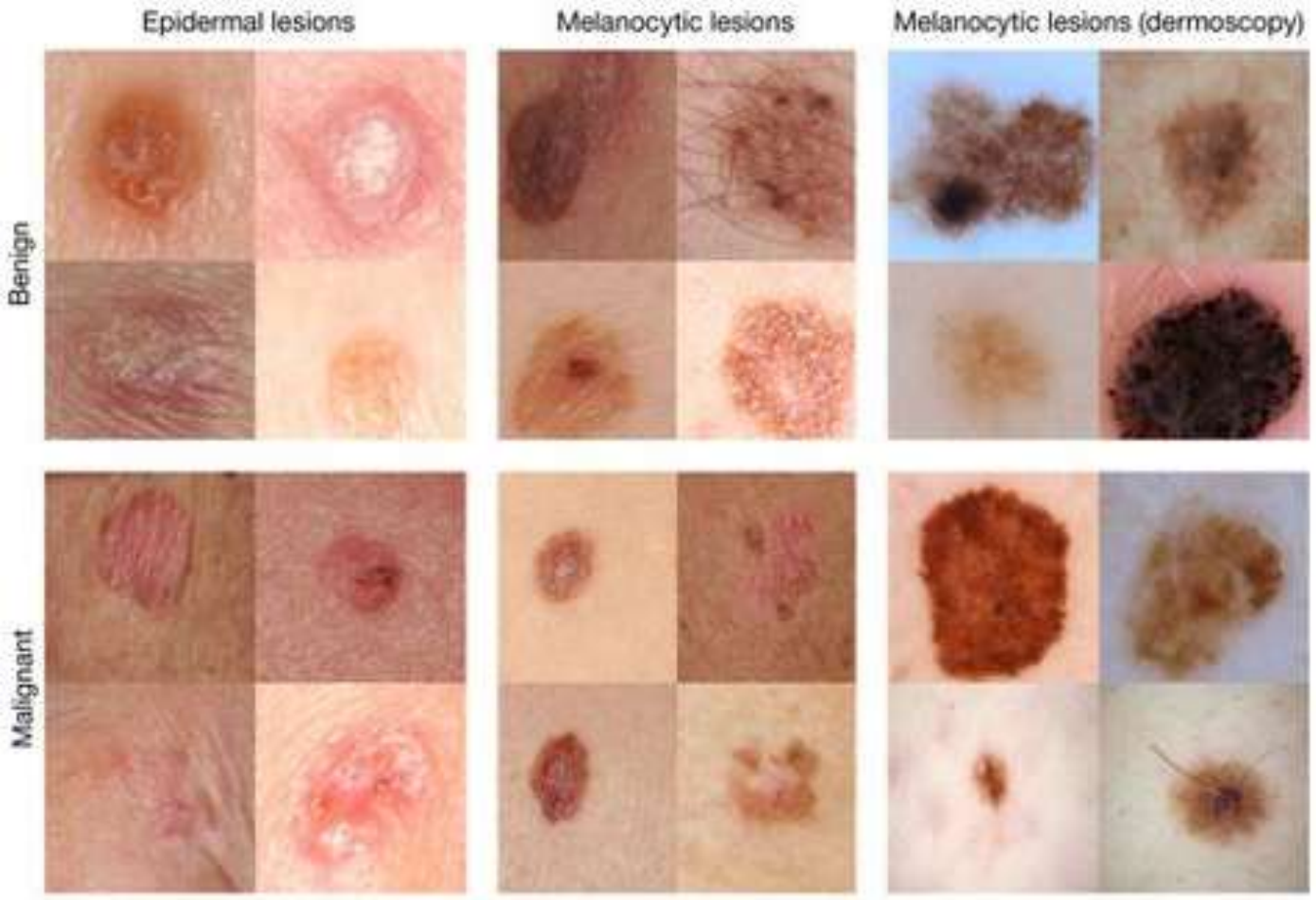
Meilenstein 2012b: Erstmals schlägt ein tiefes neuronales Netz in einem großen Wettbewerb zur *Bildsegmentierung* (Zellmembranen in elektronenmikroskopischen Bildern) alle traditionellen Verfahren



Bildquelle: Dan C. Cireşan et al., „Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images“ Proc. NeurIPS 2012, pp. 2852-2860

Beispiel: Erkennung von Hautkrebs

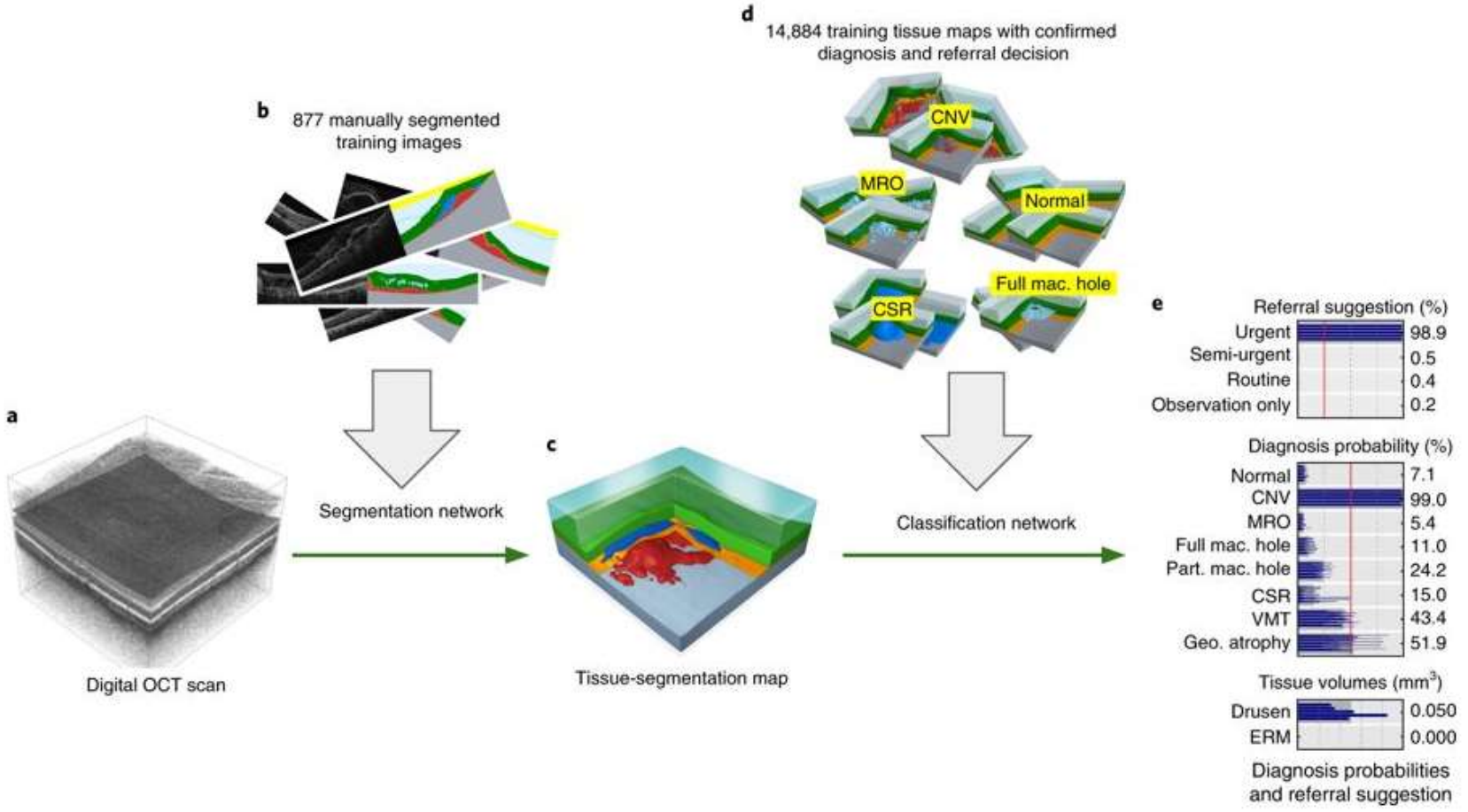
[Esteva et al. 2017] berichten mittels ≈130.000 Fotos ein neuronales Netz trainiert zu haben, das zwei Typen von Hautkrebs so zuverlässig erkenne wie 21 Fachärzte für Dermatologie



Bildquelle: Andre Esteva et al., „Dermatologist-level classification of skin cancer with deep neural networks“ Nature 542:115-118, 2017

Beispiel: Ersteinschätzung und Diagnose

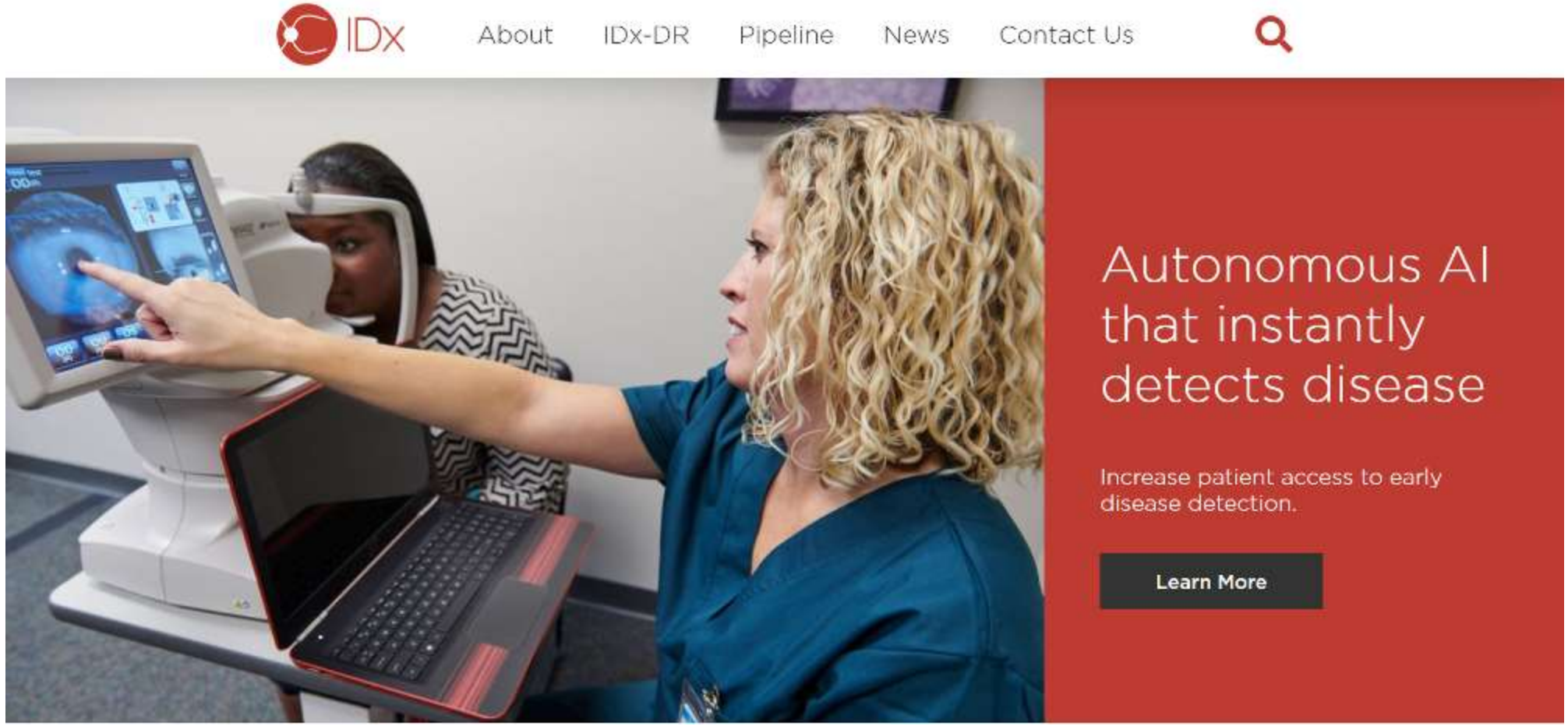
[De Fauw et al. 2018] berichten, dass ein neuronales Netz aus nur ≈14.000 OCT-Scans gelernt habe, die Dringlichkeit einer Behandlung so zuverlässig einzuschätzen wie klinische Experten



Bildquelle: Jeffrey De Fauw et al., „Clinically Applicable Deep Learning for Diagnosis and Referral in Retinal Disease“ Nature Medicine 24:1342-1350, 2018

Beispiel: Praktischer Einsatz

Meilenstein 2018: US-amerikanische FDA lässt mit IDx-DR erstmals ein autonomes diagnostisches System zu

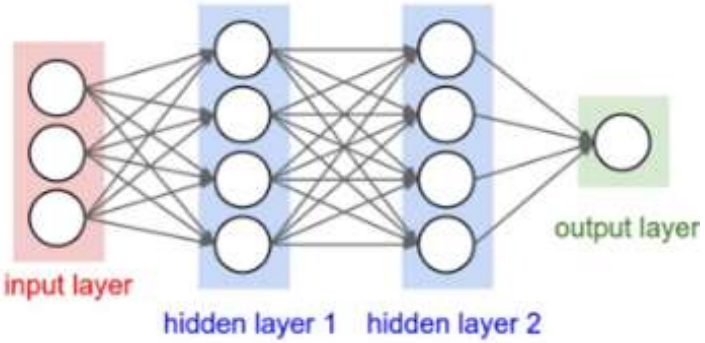


https://www.eyediagnosis.co/

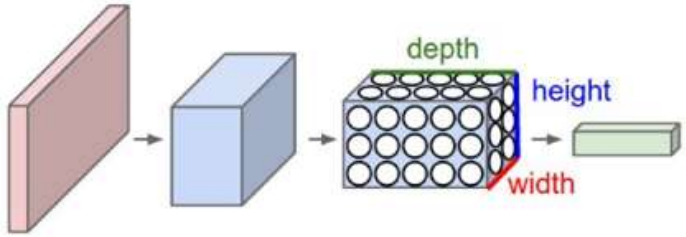
6b.2 CNNs – Faltende Netzwerke

Convolutional Neural Networks

- *Problem*: Analyse von Bildern realistischer Größe (nur) mit vollständig verbundenen Schichten ist nicht sinnvoll
  - Hohe Zahl von Gewichten erzeugt enormen Speicher- und Rechenaufwand und benötigt sehr große Trainingsdatensätze



- **Faltende neuronale Netze** (*engl*. convolutional neural networks, CNNs) verarbeiten Bilder u.a. durch Faltung mit erlernten Kernen
  - Verringert die Zahl der Modellparameter drastisch
  - Äquivariant bzgl. Verschiebungen: Verschiebung der Eingabe führt zu entsprechender Verschiebung der Ausgabe
    - Objekte werden unabhängig davon erkannt, wo im Bild sie sich befinden



Bildquelle: http://cs231n.github.io/convolutional-networks/

Klarstellung: Faltung vs. Kreuzkorrelation

Aus Kapitel 1: Faltung

$$f(i,j) = (h * g)(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) \cdot g(i-u, j-v)$$

und Kreuzkorrelation

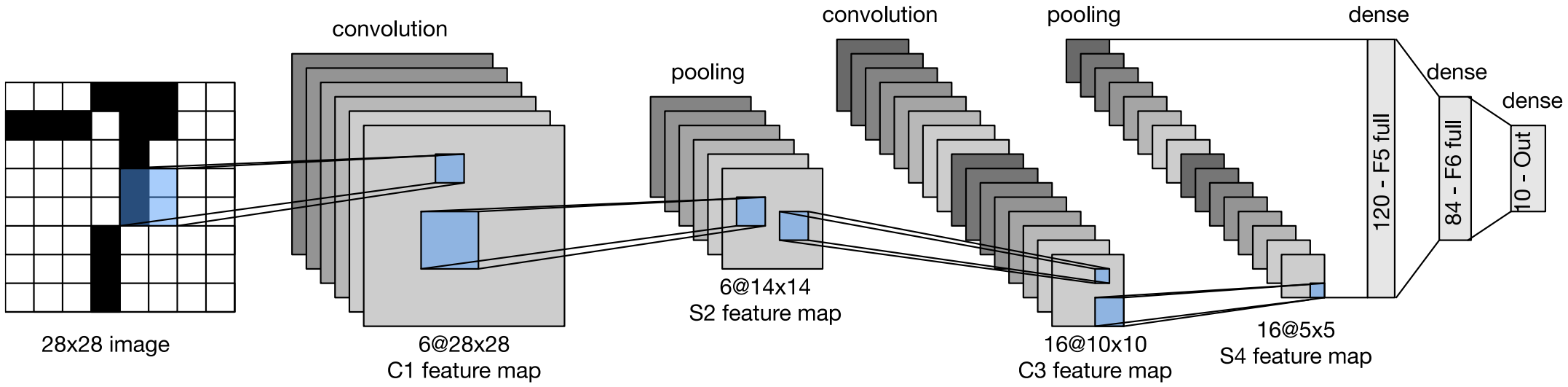
$$f(i,j) = (h \otimes g)(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) \cdot g(i+u, j+v)$$

unterscheiden sich durch die Vorzeichen

- Entspricht Spiegelung des Kerns in beiden Richtungen
- CNNs nutzen häufig Kreuzkorrelation, sprechen aber von „Faltung“
  - Streng genommen ist das Missbrauch der Terminologie
  - Solange die Kerne mit derselben Architektur gelernt werden, wird die Spiegelung „mitgelernt“, Unterschied daher praktisch nicht relevant

Bausteine von CNNs

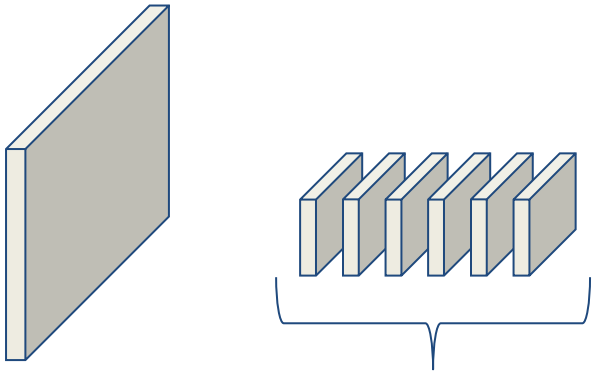
- LeNet, das erste CNN zur Unterscheidung handgeschriebener Ziffern [LeCun et al. 1998], enthält bereits wesentliche Bausteine:
  - Faltungsschichten
  - Pooling zum Downsampling
  - Vollständig verbundene Schichten am Ende des Netzwerks



Bildquelle: http://d2l.ai/chapter_convolutional-neural-networks/lenet.html

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



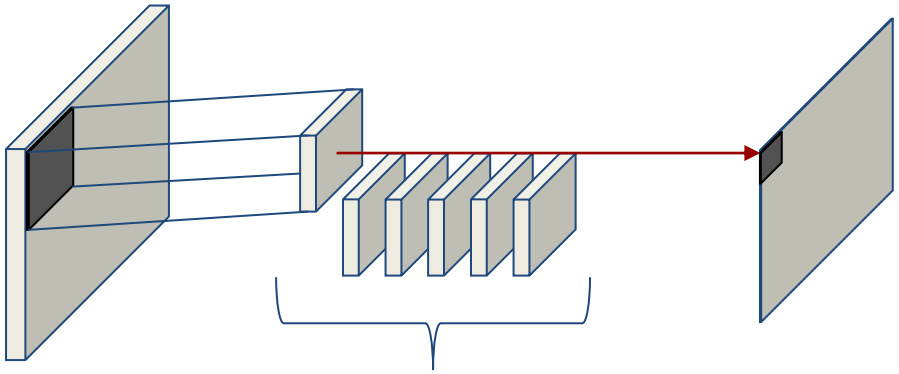
Eingabe
32x32x3

Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Die Größe von Faltungskernen wird üblicherweise in 2D angegeben (hier: 5x5)
Die dritte Dimension deckt implizit alle Kanäle der Eingabe ab.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

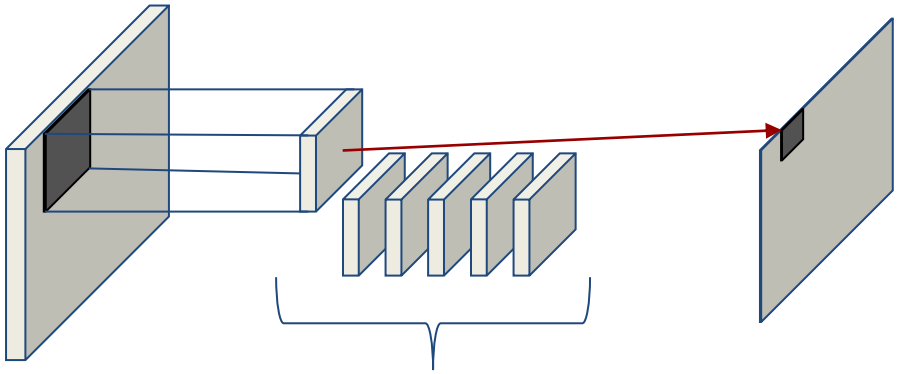
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

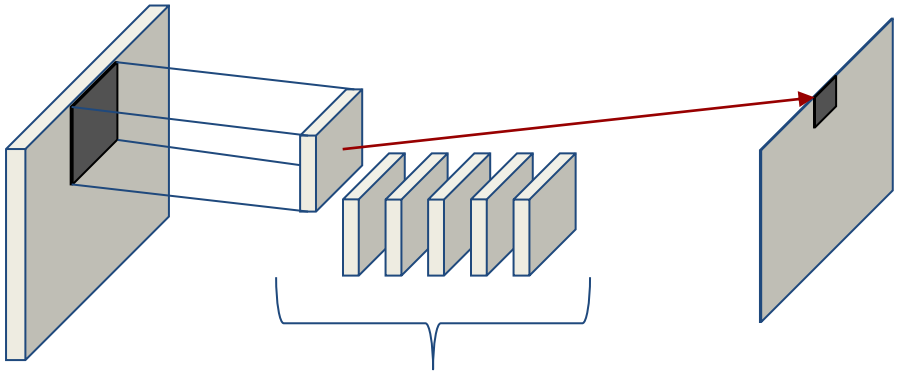
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

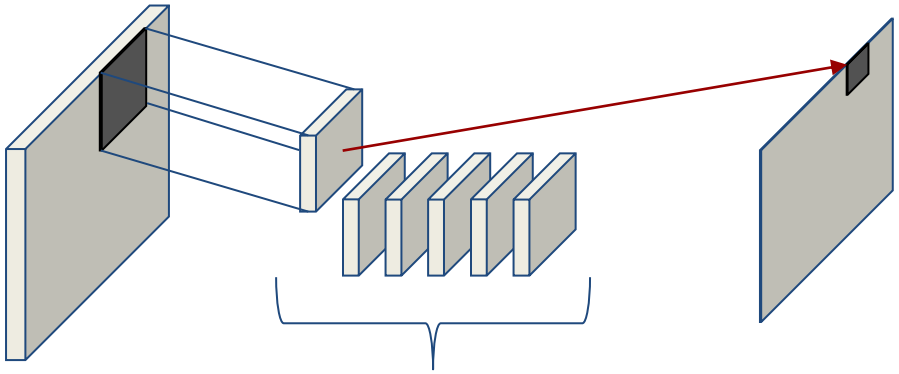
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

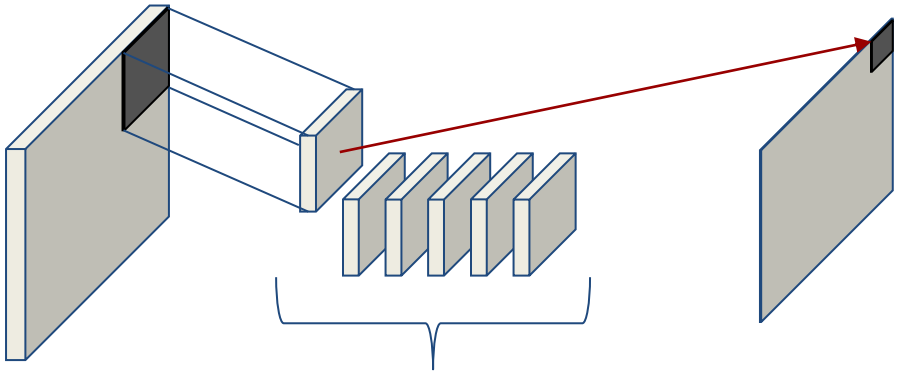
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

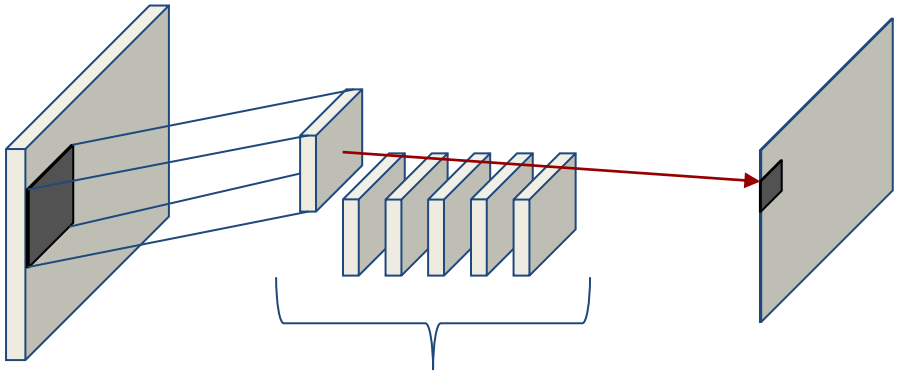
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe 32x32x3

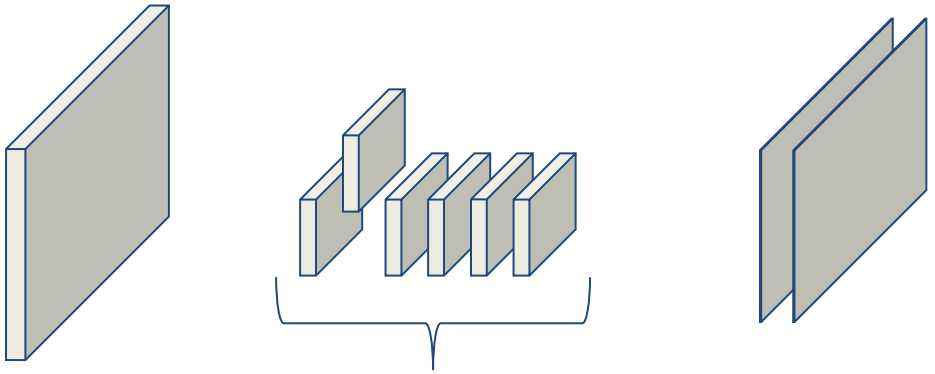
Faltungskerne Filtergröße:5x5x3 Zahl der Filter: 6

Aktivierungskarte 28x28x1

Die Faltung der Eingabe mit einem der Faltungskerne ergibt eine Aktivierungskarte. Ohne weitere Behandlung der Ränder ist diese kleiner als die Eingabe.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

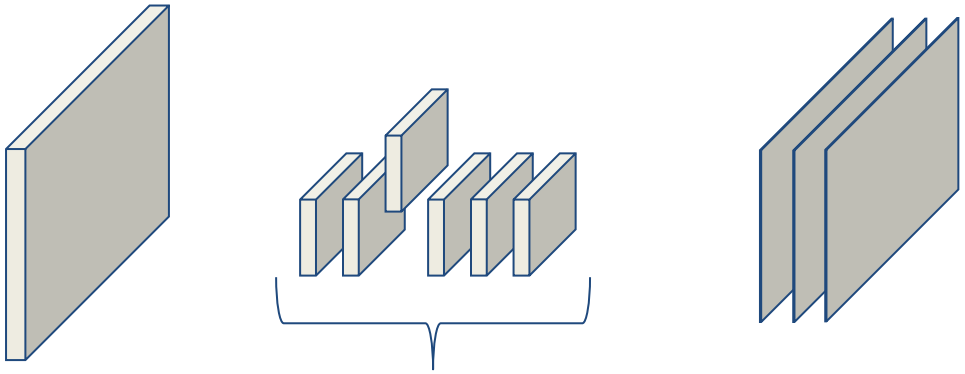
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Jeder Faltungskern erzeugt eine eigene Aktivierungskarte.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

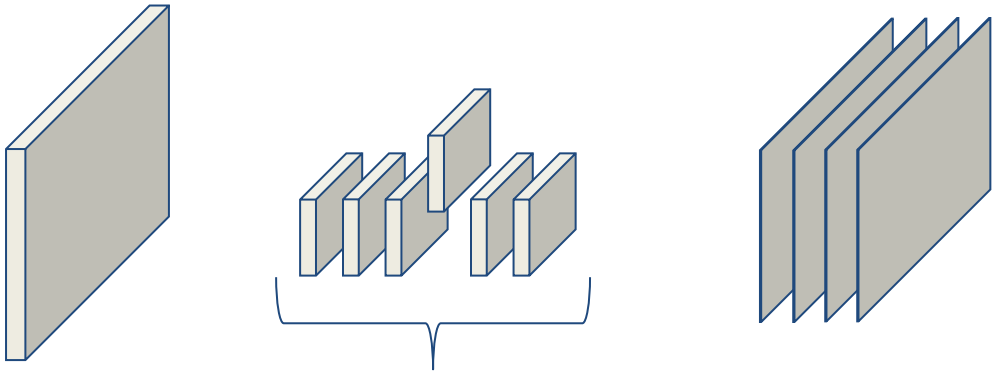
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Jeder Faltungskern erzeugt eine eigene Aktivierungskarte.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

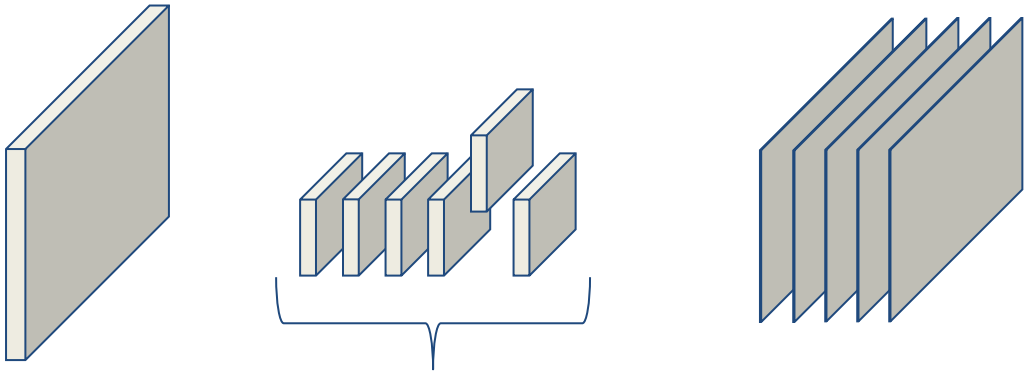
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Jeder Faltungskern erzeugt eine eigene Aktivierungskarte.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

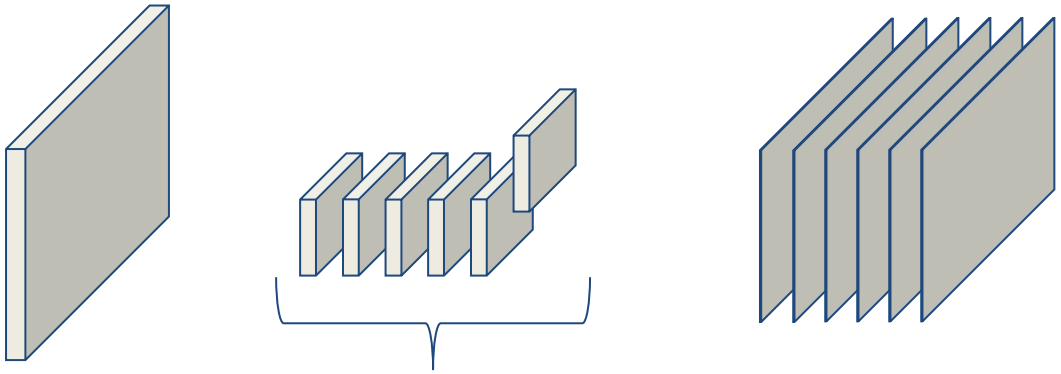
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Jeder Faltungskern erzeugt eine eigene Aktivierungskarte.

Faltungsschichten

- Eine Faltungsschicht ist gegeben durch
  - Zahl der Faltungskerne
  - Größe der Faltungskerne
  - Padding
  - Schrittweite



Eingabe
32x32x3

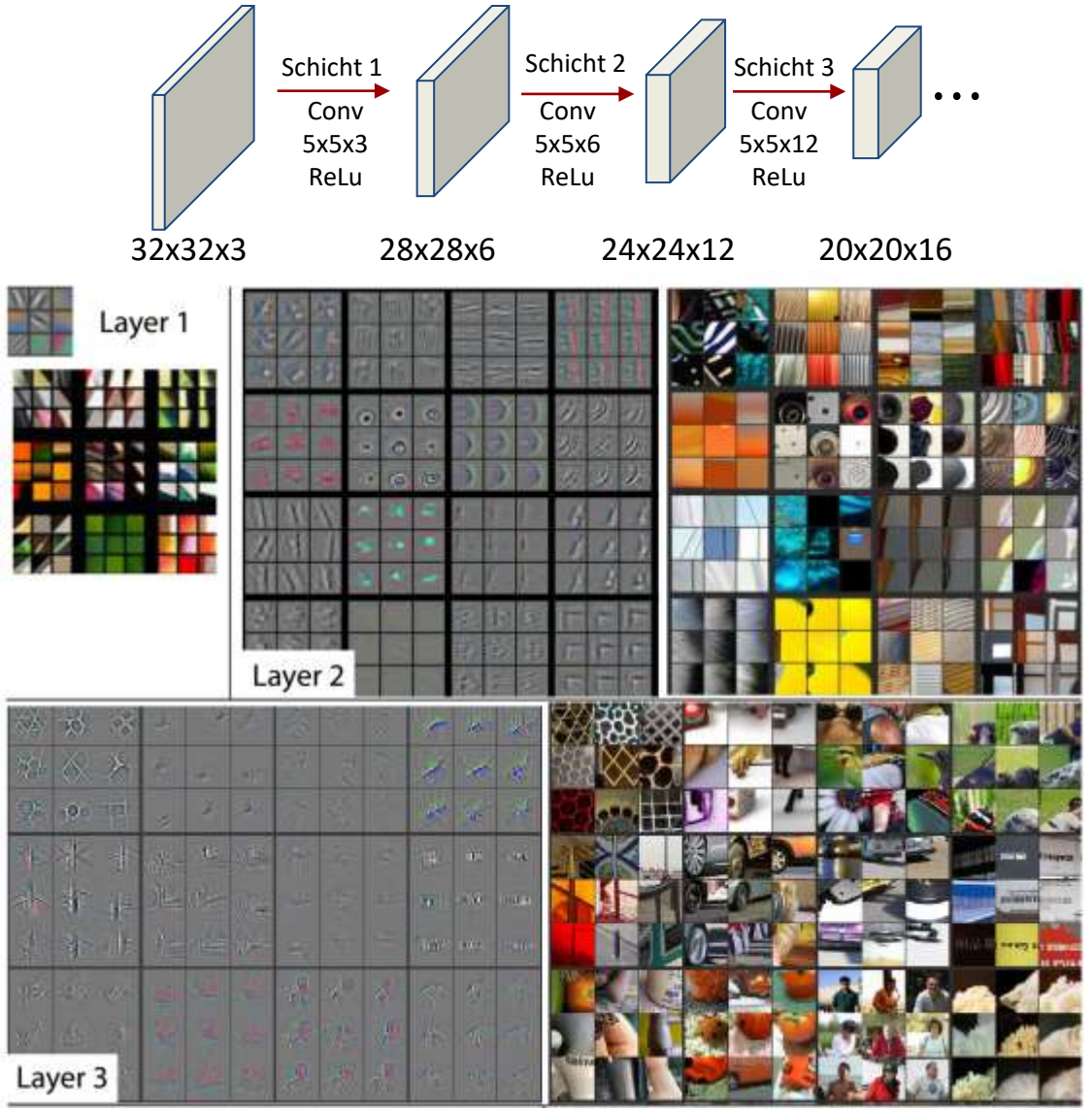
Faltungskerne
Filtergröße:5x5x3
Zahl der Filter: 6

Aktivierungskarte
28x28x1

Das aus allen Aktivierungskarten bestehende 3D-Array („Tensor“) wird zur Eingabe der nächsten Schicht.

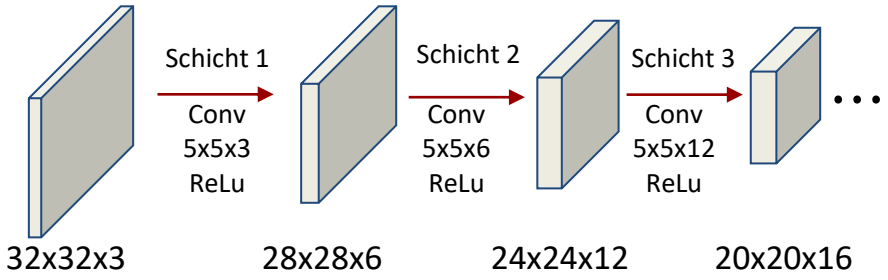
Merkmalshierarchie

- Die Faltungskerne sind **Parameter** des Netzwerks
  - Zufällige Initialisierung
  - Optimierung via Backpropagation
- Durch die **hierarchische Anordnung** erkennen tiefere Schichten komplexere Merkmale
  - Das rezeptive Feld tieferer Neurone wird immer größer



Bildquelle: Zeiler et al. "Visualizing and understanding convolutional networks." *ECCV* 2014

Das rezeptive Feld



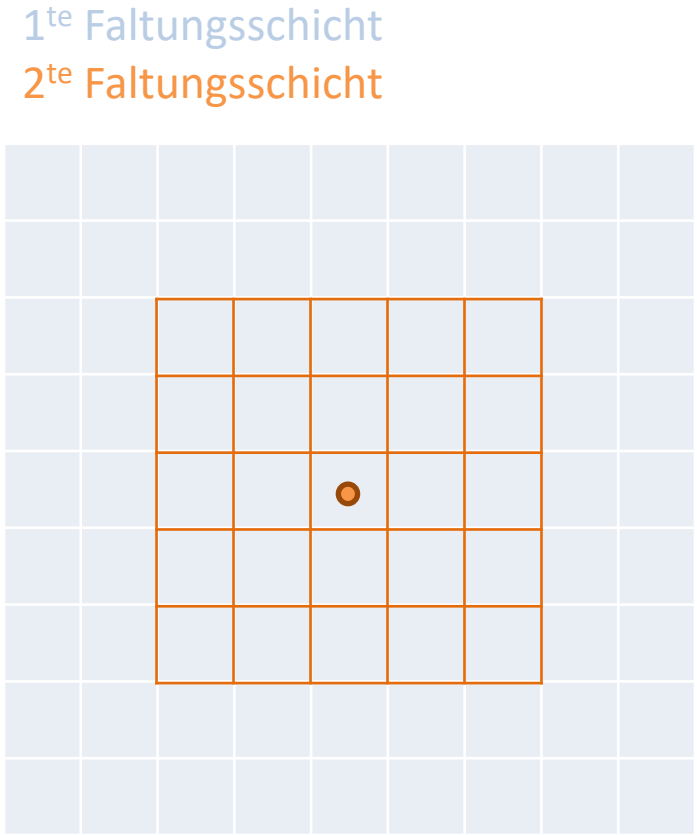
Das **rezeptive Feld** eines Neurons ist der Ausschnitt seiner Eingabe, der einen Einfluss auf seine Ausgabe hat.

Das rezeptive Feld der ersten Schicht ist 5x5

Quiz: Wie ist es mit Schicht 2 und Schicht 3?

Antwort: 9x9 bzw. 13x13

- Können wir es schneller vergrößern?
- …ohne dabei so viele Pixel am Rand zu verlieren?



Faltung: Verlust des Bildrandes

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

Quiz: Welche Dimensionen hat die Ausgabe dieser Faltung?
Antwort: 5x5

Padding: Faltung mit Auffüllen der Ränder

0	0	0	0	0	0	0	0	0
0	1	0	2	1	1	2	2	0
0	1	0	1	0	1	0	1	0
0	2	1	2	1	1	1	1	0
0	-1	0	1	2	1	2	2	0
0	1	2	0	-1	0	1	-1	0
0	6	1	7	-1	0	1	0	0
0	1	1	1	2	1	0	2	0
0	0	0	0	0	0	0	0	0

*

1	-1	0
-1	0	0
1	0	1

3x3

Quiz: Welche Dimensionen hat die Ausgabe dieser Faltung?

Antwort: 5x5

Quiz: Wie ist die Dimensionen, wenn wir den Rand einen Pixel breit mit Nullen auffüllen (padding=1)?

Antwort: 7x7

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

4		

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

4	3	

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

4	3	0

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

4	3	0
3		

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Faltung mit größeren Schrittweiten

1	0	2	1	1	2	2
1	0	1	0	1	0	1
2	1	2	1	1	1	1
-1	0	1	2	1	2	2
1	2	0	-1	0	1	-1
6	1	7	-1	0	1	0
1	1	1	2	1	0	2

7x7

*

1	-1	0
-1	0	0
1	0	1

3x3

=

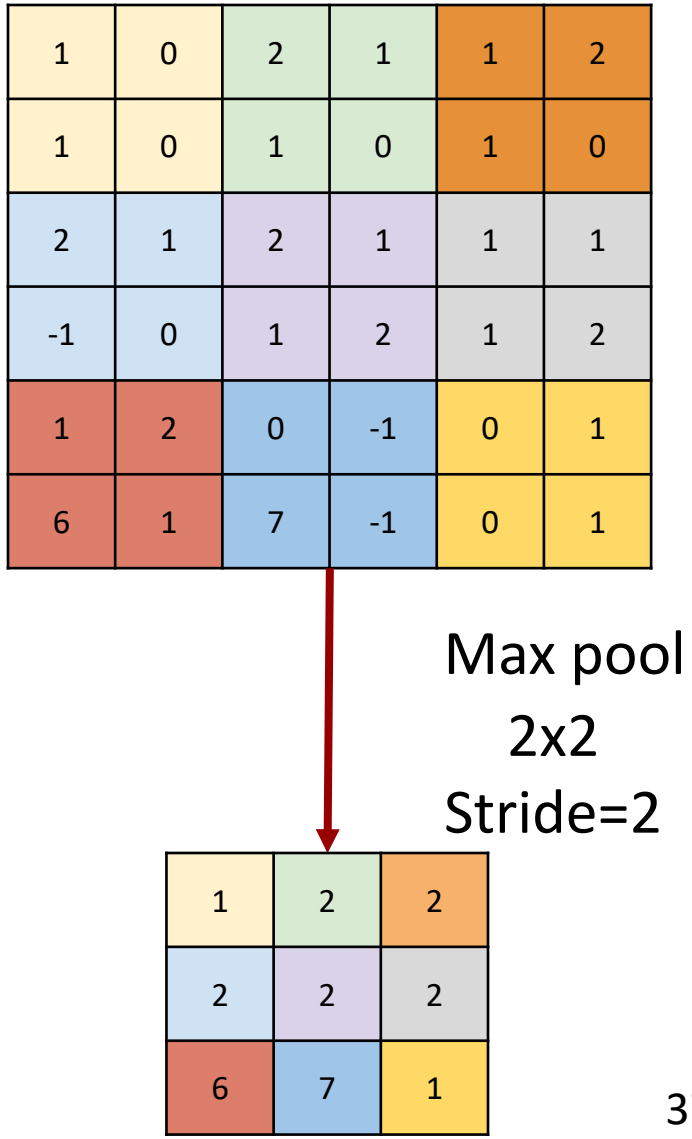
4	3	0
3	0	-2
-5	-4	2

Stride=2
Padding=0

Faltungen mit größeren Schrittweiten (engl. stride) vergrößern das rezeptive Feld in den folgenden Schichten

Pooling: Aggregation über Nachbarschaften

- **Pooling-Schichten** aggregieren Aktivierungen über eine räumliche Nachbarschaft. Sie sind definiert über
  1. Größe (z.B. 2x2)
  2. Art des Poolings (z.B. Maximum, Mittelwert)
  3. Schrittweite (stride)
- Pooling ist eine beliebte Alternative zum **Downsampling** innerhalb von CNNs
  - Am üblichsten ist Max-Pooling
  - Hierbei benötigt die Backpropagation die Positionen der Maxima („switches“)



Zusammenfassung

- **Convolutional Neural Networks** (CNNs) nutzen Faltungsschichten, um auf Gittern definierte hochdimensionale Eingabedaten (insb. Bilder) effizienter zu verarbeiten
  - Reduzierte Zahl der **Parameter**, Äquivarianz bzgl. **Verschiebungen**
  - **Hierarchische Anordnung** der Faltungsschichten repräsentiert zu Beginn einfache, lokale Merkmale, erkennt tiefer im Netz mit größerem rezeptiven Feld komplexere Konzepte
- Hinreichend große **rezeptive Felder** mit praktikabler Zahl von Parametern werden ermöglicht durch
  - Faltungen mit größerer Schrittweite (*engl*. strided convolution)
  - Pooling-Schichten

6b.3 Bildklassifikation mit CNNs

Überblick

- Grundlagen der CNN-basierten Bildklassifikation sind seit 1989 (erste Version des LeNet) bekannt
- Fest etabliert hat sich diese Idee 2012 und wurde seither rapide weiterentwickelt
  - Wesentliche Faktoren für den Erfolg waren:
    - Ideen zum erfolgreichen Training sehr **tiefer Netze**
    - Verfügbarkeit hinreichender und kostengünstiger **Rechenkapazität** durch GPUs (hochgradig parallele Koprozessoren, urspr. zur Bildsynthese)
    - Verfügbarkeit sehr umfangreicher **Trainingsdatensätze**

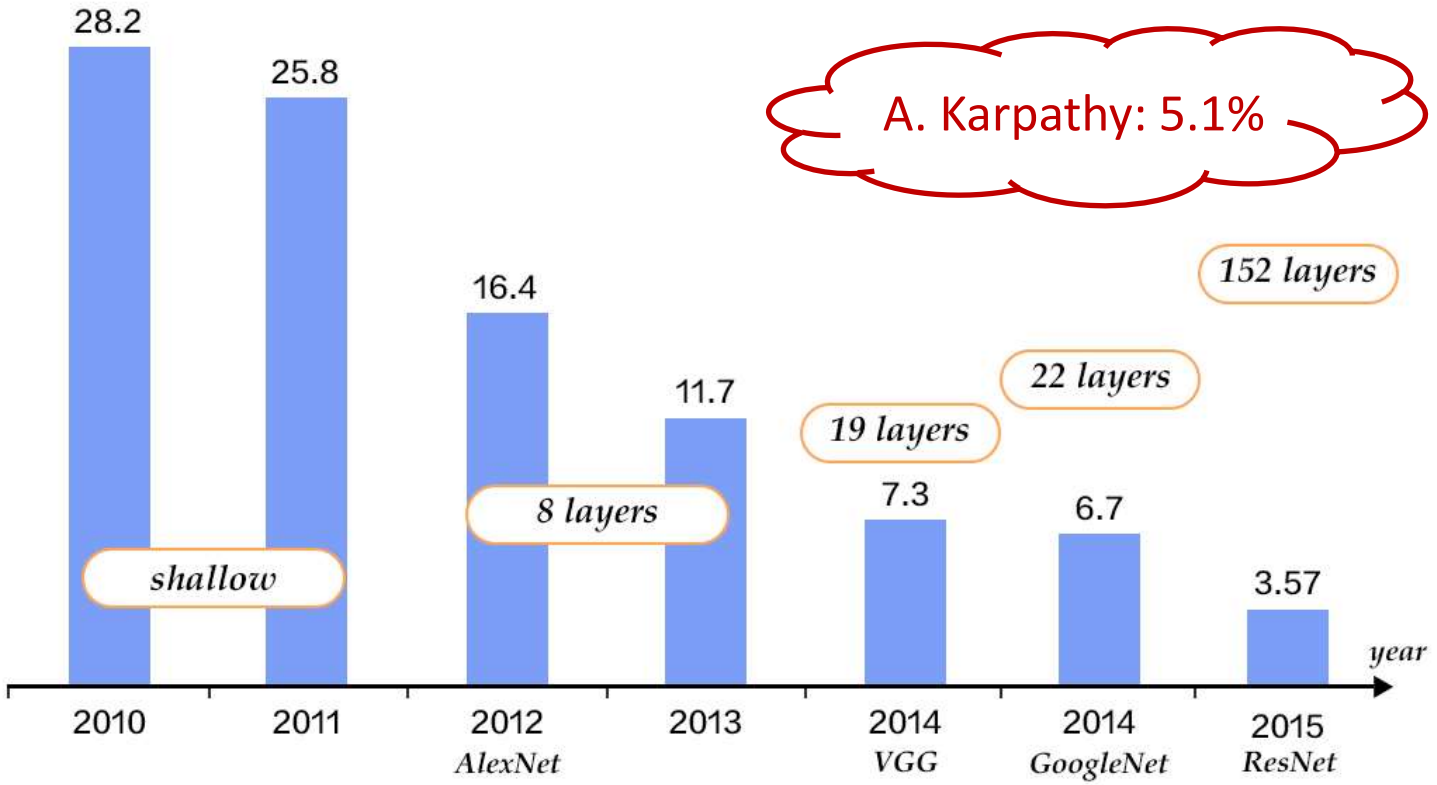
ImageNet-Wettbewerb

- **ImageNet** ist eine Datenbank von mehr als 14 Mio Bildern, die jeweils einer von 20.000 Kategorien zugeordnet wurden
- Mit einer Teilmenge von ca. 1,5 Mio Bildern aus 1000 Klassen wurde 2010-2017 die **ImageNet Large Scale Visual Recognition Challenge** (ILSVRC) ausgerichtet
  - Unterschiede zum Teil sehr fein, allein 120 Klassen sind verschiedene Hunderassen
  - Top-5-Fehler wertet es als Erfolg, wenn die korrekte Klasse unter den 5 als am wahrscheinlichsten vorhergesagten ist

Bildquelle: Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge” IJCV 115(3):211-252, 2015

Thema dieses Kapitels

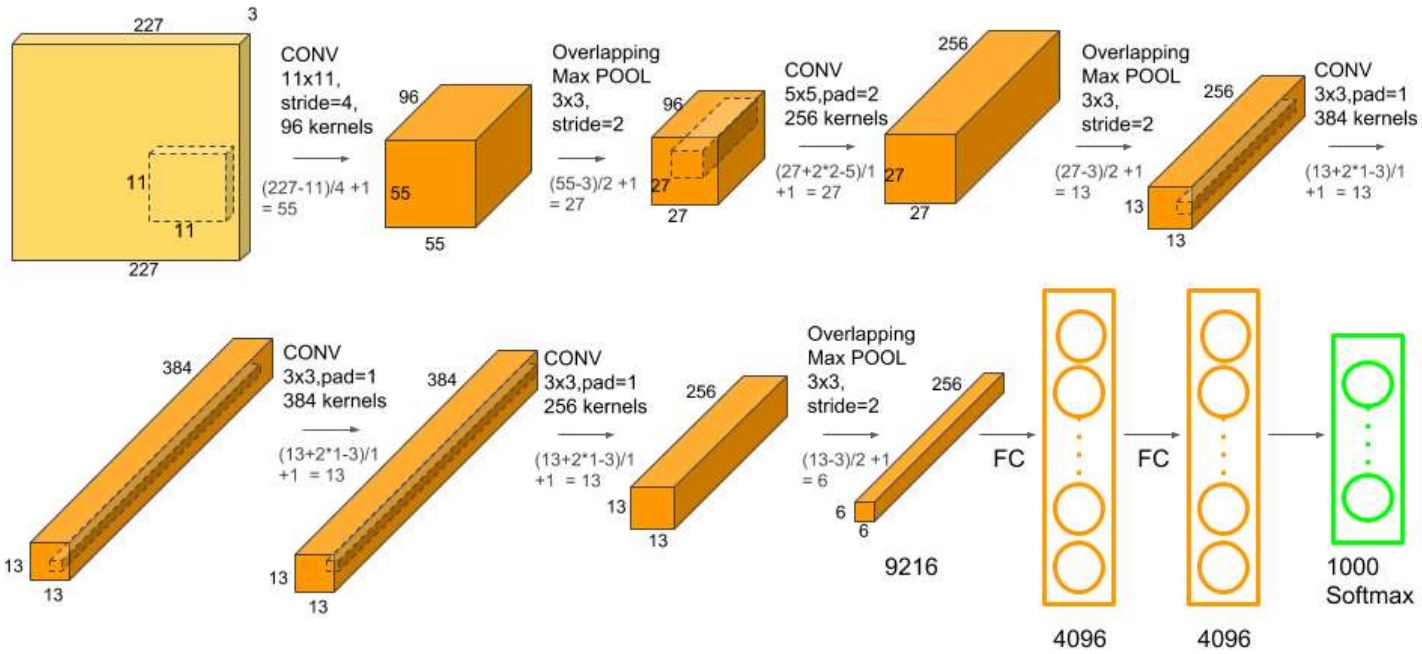
Wir betrachten im Folgenden die CNN-Architekturen, die zu wesentlichen Fortschritten auf der ILSVRC geführt haben



Bildquelle: http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html

AlexNet

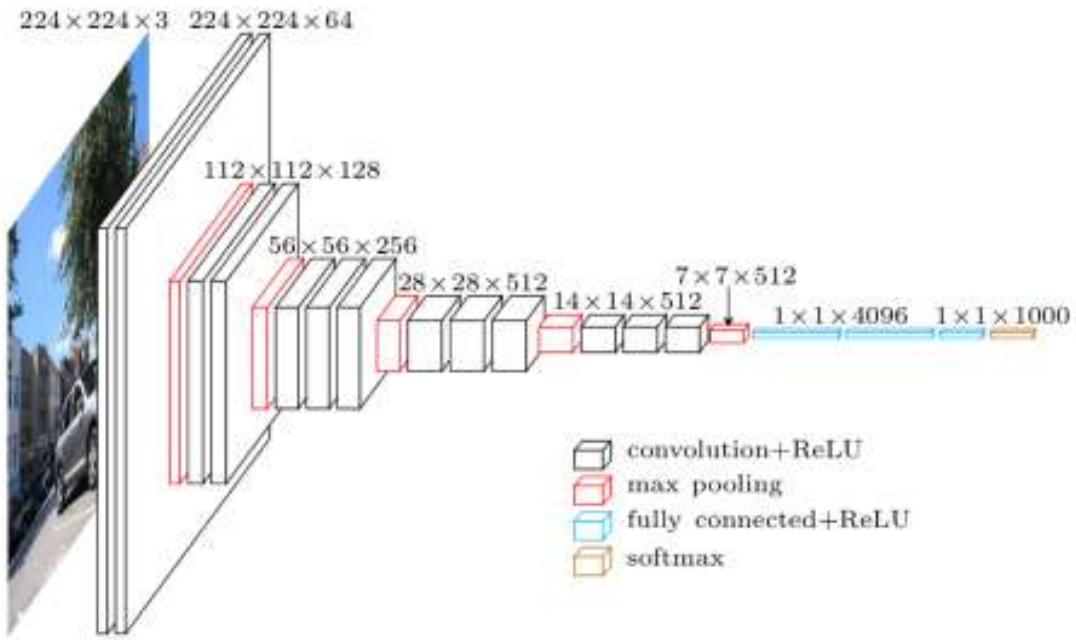
- **AlexNet** [Krizhevsky et al.] gewann 2012 mit 16,4% Top-5-Fehler als erste CNN-Architektur die ILSVRC
  - Training auf zwei GPUs dauerte 5-6 Tage
- Grundidee und Bausteine ähnlich wie im LeNet. Neu waren:
  - Mehr Schichten
  - Viel mehr Parameter
  - Direkt hintereinander ausgeführte Faltungen
  - ReLU-Aktivierungen
  - Augmentierung
  - Dropout



Bildquelle: https://www.learnopencv.com/understanding-alexnet/

VGG

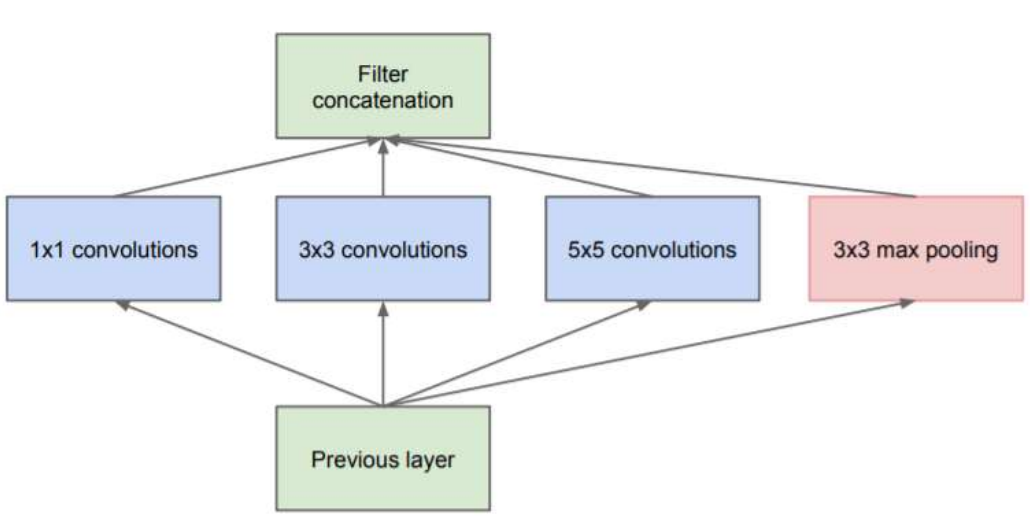
- **VGG** [Simonyan/Zisserman] gewann bei der ILSVRC 2014 mit 7,3% Top-5-Fehler den zweiten Platz
  - Training auf 4 GPUs dauerte 2-3 Wochen
- **Hauptidee**: Tiefer ist besser!
  - Mehr, dafür kleinere Kerne (3x3)
  - Einheitlichere Architektur
  - Empfehlung nach Experimenten mit verschiedenen Tiefen: 16-19 Schichten
    - 138/144 Mio Parameter statt 60 Mio im AlexNet



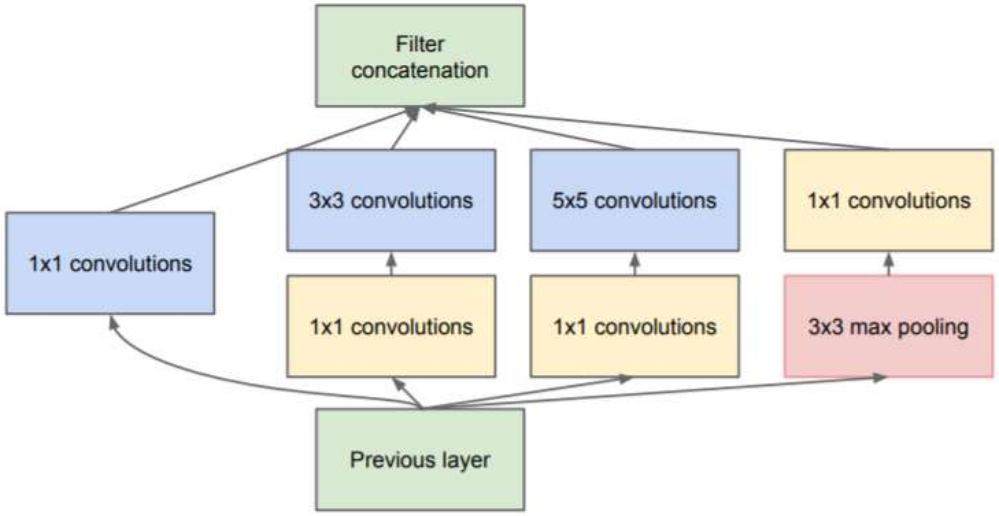
Bildquelle: https://www.cs.toronto.edu/~frossard/post/vgg16/

Inception-Modul

- Das **Inception-Modul** ermöglicht die Konstruktion tiefer Netze mit relativ wenig Parametern
  - **Erste Idee:** Kombination von Faltungskernen verschiedener Größe (Mehr-Skalen-Repräsentation) sowie max-Pooling mit Schrittweite 1
  - **Zweite Idee:** Dimensionsreduzierung mit 1x1-Faltungen reduziert die Zahl der Parameter in den Faltungskernen



(a) Inception module, naïve version

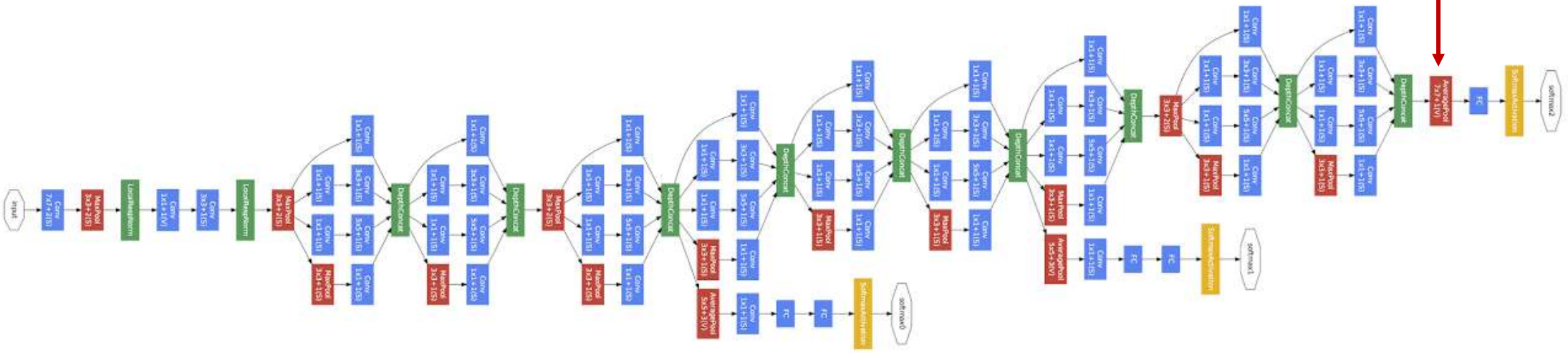


(b) Inception module with dimension reductions

Bildquelle: [Szegedy et al. 2014]

GoogLeNet

- **GoogLeNet** [Syegedy et al.] gewann mit 6,7% Top-5-Fehler die ILSVRC 2014
  - Enthält 9 Inception-Module, insgesamt 22 Schichten
  - Hat dennoch viel weniger Parameter als AlexNet (5 Mio statt 60 Mio)
    - Wichtiger Beitrag hierzu: 7x7 Mittelwert-Pooling nach letzter Faltungsschicht



Bildquelle: [Szegedy et al. 2014]

Faltung / Pooling / Softmax / Sonstige

GoogLeNet: Der Teufel im Detail (Teil 1)

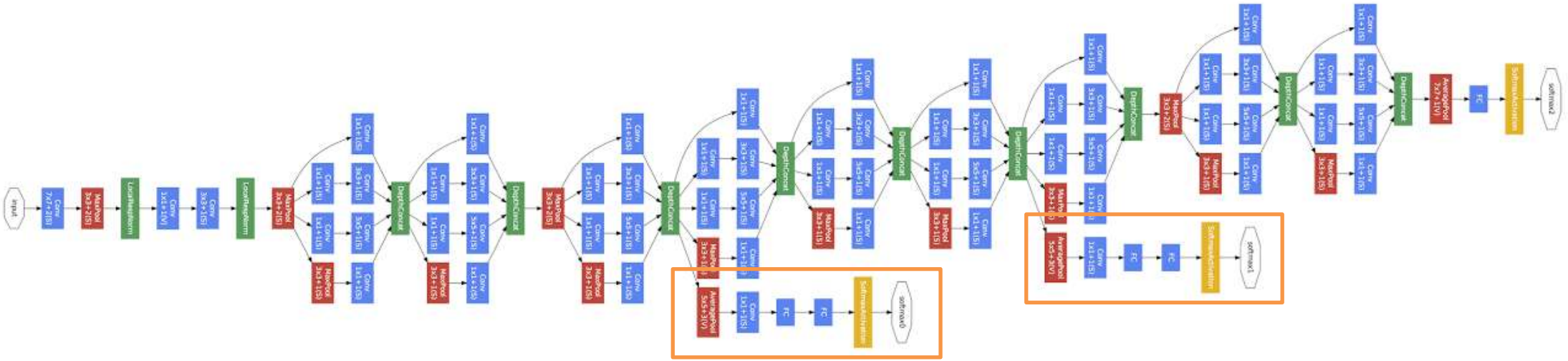
- Den knappen Vorsprung hatte GoogLeNet u.a. einer sorgfältigen **Optimierung der Hyperparameter** zu verdanken
  - Betrifft u.a. genaue Zahl der Aktivierungskarten jeder Schicht

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Bildquelle: [Szegedy et al. 2014]

GoogLeNet: Der Teufel im Detail (Teil 2)

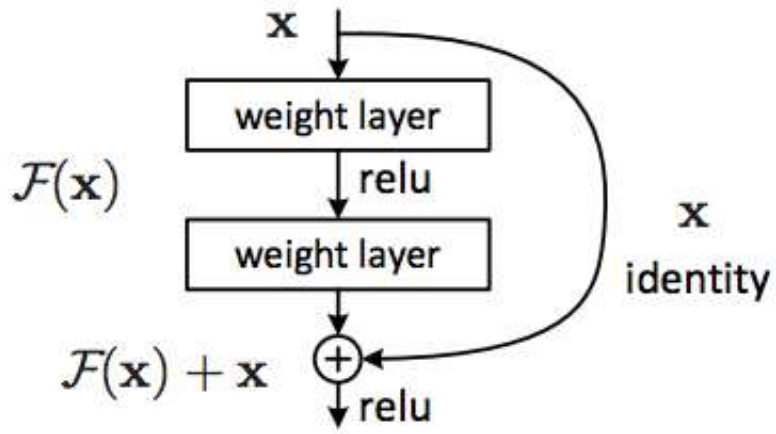
- Der **genaue Trainingsprozess** ist wichtig um gute Ergebnisse zu erhalten, wird in der Publikation aber nicht beschrieben
  - *Zitat:* “Es ist schwer eine klare Richtschnur anzugeben, wie man diese Netzwerke am effektivsten trainiert.”
  - Keine Angaben zu Hardware oder Dauer des Trainings
  - Um frühe Schichten zu trainieren kamen zusätzliche Klassifikatoren zum Einsatz, die nach dem Training wieder entfernt wurden



Bildquelle: [Szegedy et al. 2014]

ResNet: Grundidee

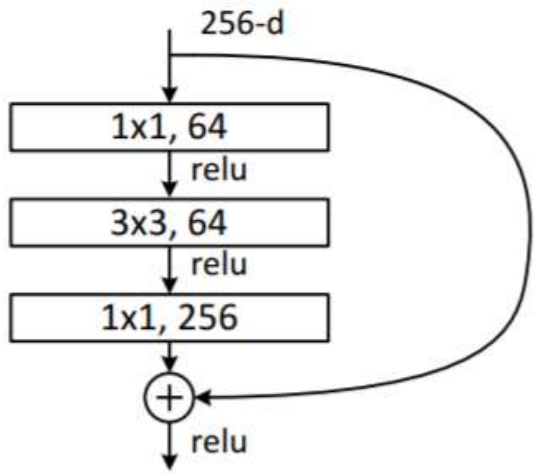
- [He et al. 2015] beschreiben ein **Degradierungs-Problem:**
  - Hinzufügen weiterer Schichten erhöht ab einem bestimmten Punkt den Trainingsfehler
  - Einzige Erklärung: Probleme beim Training solcher Architekturen
- **Grundidee**: Schichten sollten statt einer Funktion $f(\mathbf{x})$ ein additives Residual $f_{res}(\mathbf{x})$ ausgeben, so dass $f(\mathbf{x}) = f_{res}(\mathbf{x}) + \mathbf{x}$
  - *Begründung*: Identität $f(\mathbf{x})=\mathbf{x}$ ist eine nützlichere „Grundeinstellung“ als die Nullfunktion $f(\mathbf{x})=\mathbf{0}$
  - *Implementierung*: Shortcut-Verbindung, erfordert keine weiteren Parameter



Bildquelle: [He et al. 2015]

ResNet: Flaschenhals-Architektur

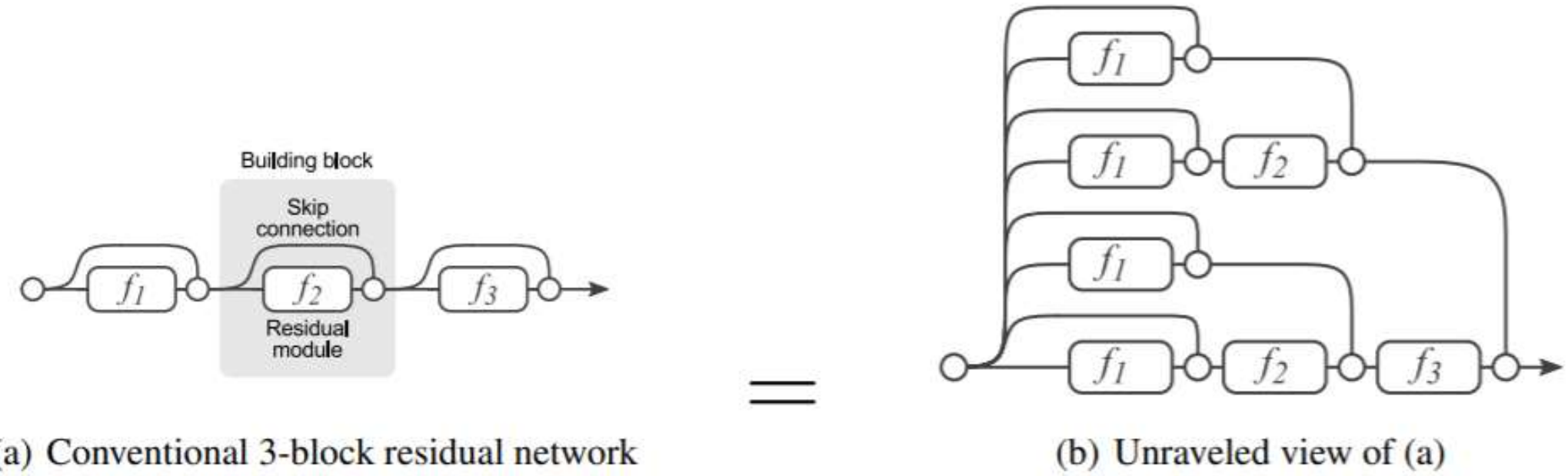
- Eine **Flaschenhals-Architektur** mit 1x1-Faltungen ermöglicht die Konstruktion tieferer Netze
  - Ähnliche Idee wie bei Inception
- **ResNet-152** [He et al.] gewann mit 3,6% Top-5-Fehler die ILSVRC 2015
  - Besteht aus vielen dieser Blöcke
  - Ergibt insgesamt 152 Schichten
- Training: 2-3 Wochen auf 8 GPUs
  - Einzelner Vorwärts-Durchlauf schneller als bei VGG
    - 11,3 Mrd FLOPs vs. 15,3/19,6 Mrd bei VGG-16/19
    - Viele der Schichten im ResNet-152 sind günstige 1x1-Faltungen



Bildquelle: [He et al. 2015]

ResNet: Sehr tief oder ein Ensemble?

- [Veit et al. 2016] schlagen vor, dass man sich ResNets besser als **Ensembles flacherer Netze** (mit 10-34 Schichten) vorstellen solle denn als sehr tiefe Netze



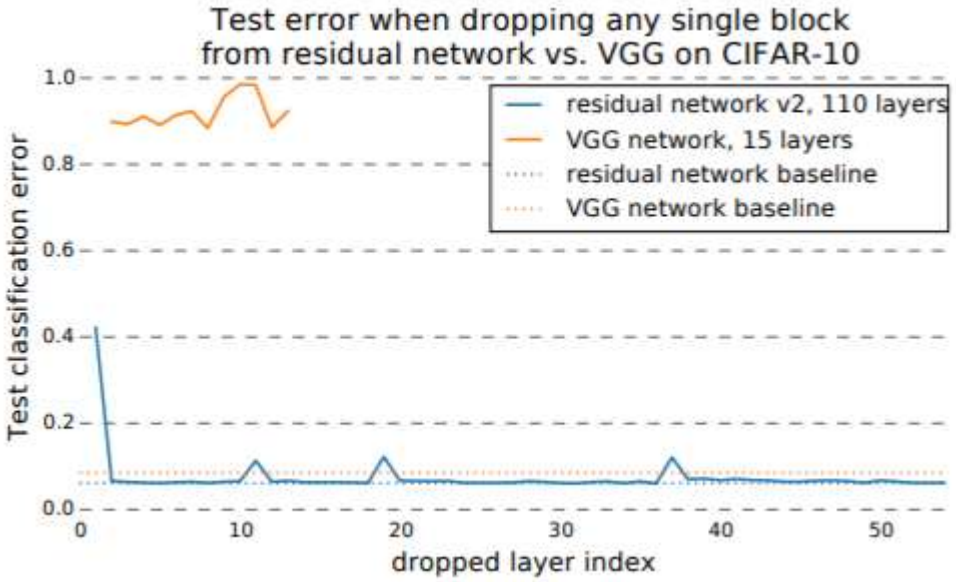
(a) Conventional 3-block residual network

(b) Unraveled view of (a)

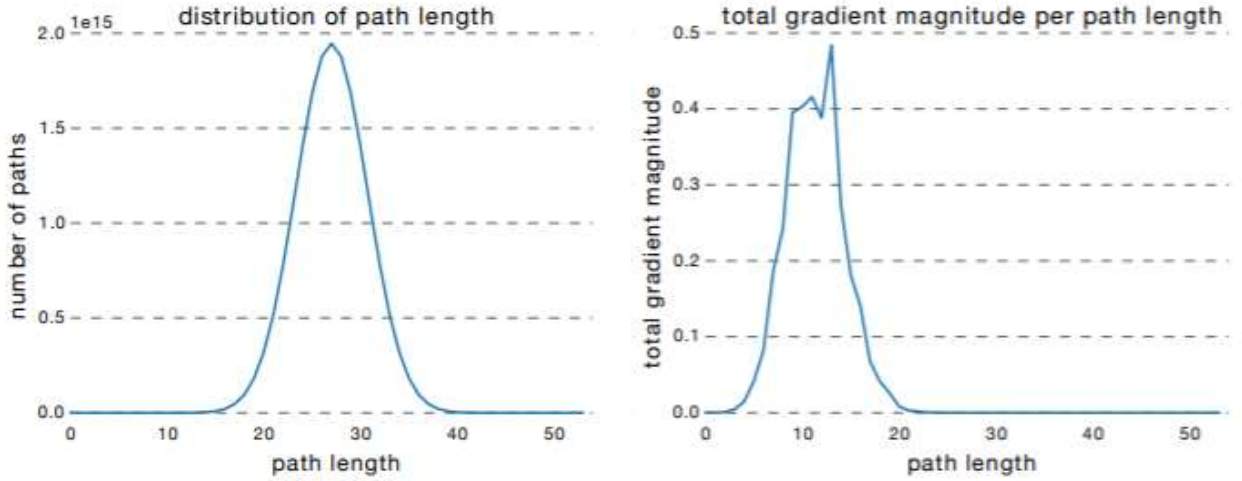
Bildquelle: [Veit et al. 2016]

ResNet: Gründe für die Interpretation als Ensemble

- **Läsionsstudie**: Entfernen einzelner ResNet-Module hat kaum einen Effekt
  - Ausnahme: Downsampling



- Untersuchung von **Pfadlängen**:
  - Die meisten Pfade sind etwa halb so lang wie die nominelle Tiefe
  - Gradienten entlang kurzer Pfade sind stärker

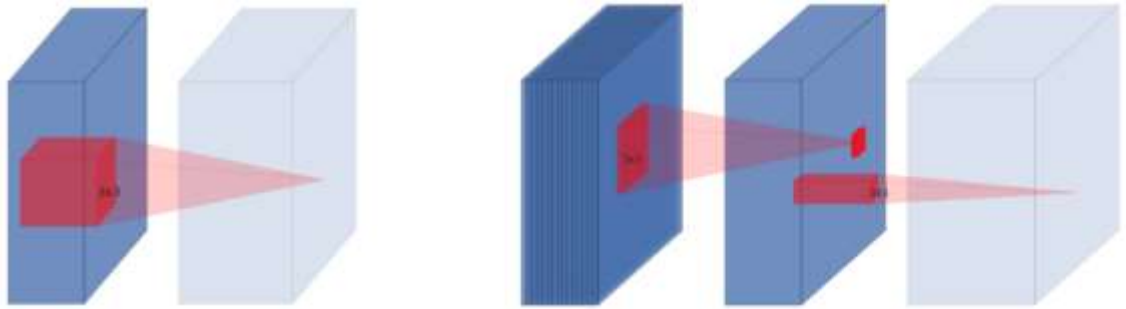


Binomialverteilung

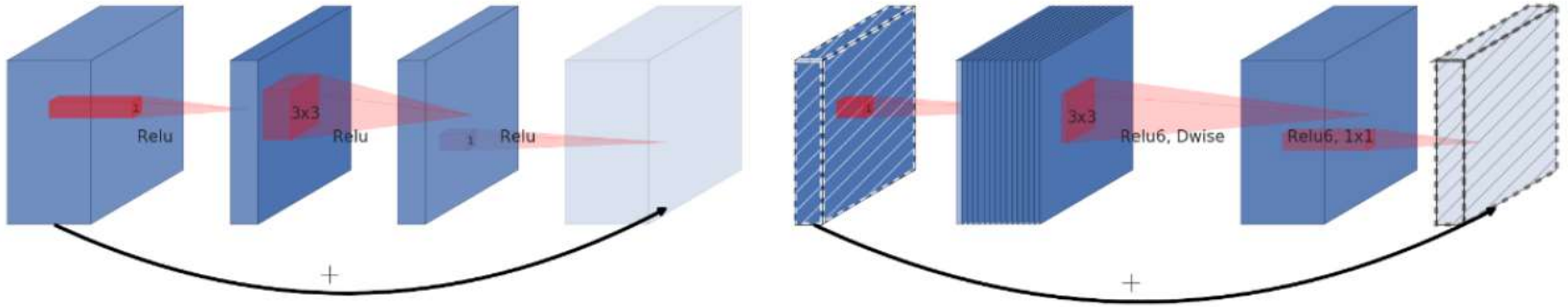
Bildquelle: [Veit et al. 2016]

EfficientNet: Grundarchitektur

- **EfficientNet** [Tan et al. 2019] optimiert die Effizienz
  - Maximierung der Klassifikationsrate *bei festem Rechenbudget*
- Grundarchitektur nutzt Ideen des **MobileNet** [Sandler et al. 2018]
  - **In Tiefenrichtung separierte Faltungen** („depthwise convolution“) sparen Parameter und Rechenoperationen

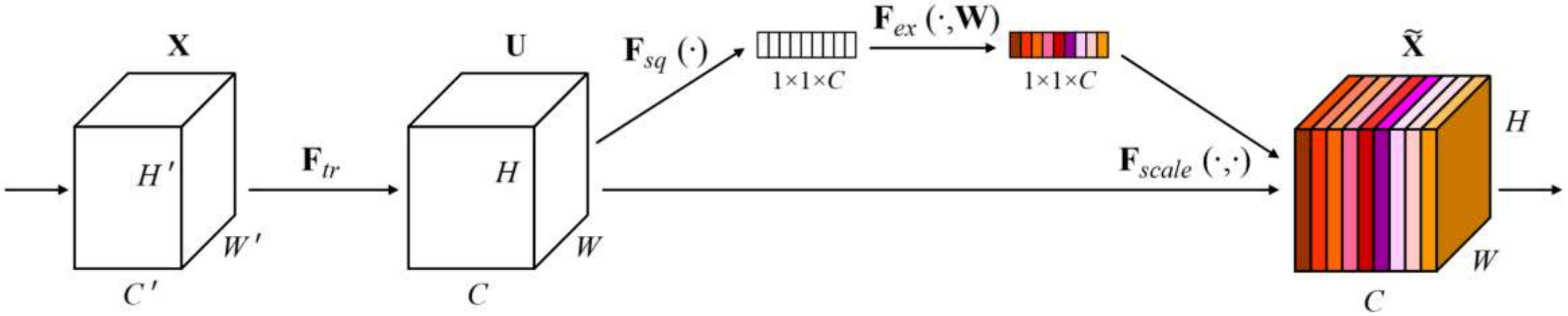


  - **Invertierte Residual-Blöcke** kommen mit weniger Additionen aus



EfficientNet: Squeeze and Excitation

- EfficientNet nutzt außerdem den **squeeze-and-excitation**-Mechanismus [Hu et al. 2018], um wichtige Aktivierungen zu erkennen und höher zu gewichten



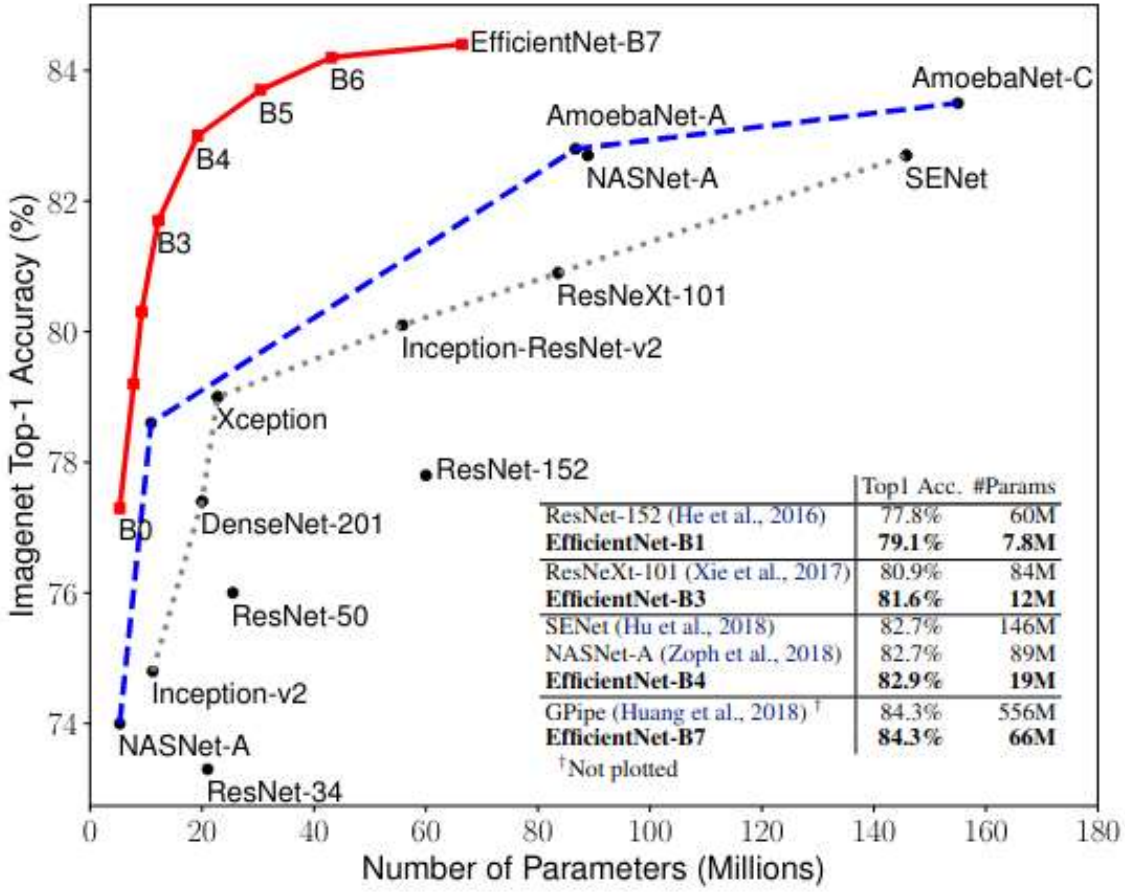
  - **„Squeeze“** durch globales Mittelwert-Pooling über Höhe und Breite
  - **„Excitation“** durch $\sigma\left(\mathbf{W}_2 \mathrm{ReLU}(\mathbf{W}_1 \mathbf{z})\right)$
    - $\mathbf{W}_1$ reduziert C Kanäle um den Faktor r, $\mathbf{W}_2$ stellt Dimension C wieder her

EfficientNet: Gemeinsame Skalierung

- Die wesentliche neue Idee von EfficientNet ist es, die Tiefe *d* und Breite *w* der Grundarchitektur sowie die Auflösung *r* des Eingabebilds mit einem **gemeinsamen Exponenten** ϕ zu skalieren:

$$d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi}$$

  - $\alpha \geq 1, \beta \geq 1, \gamma \geq 1$ so gewählt, dass $\alpha \times \beta^2 \times \gamma^2 \approx 2$

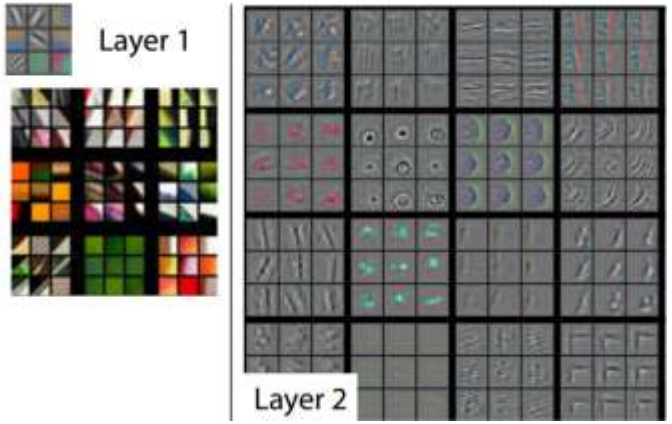


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

† Not plotted

Transferlernen: Grundidee

- Zur Klassifikation medizinischer Bilder nutzt man meist etablierte CNN-Architekturen
  - Diese wurden aufwändig optimiert und sind schwer zu schlagen
  - Wenn relativ wenige Bilder zum Training zur Verfügung stehen, nutzt man häufig auf ImageNet vortrainierte Gewichte
    - *Idee*: Viele Merkmale, insbesondere in den ersten Schichten, sollten hinreichend generell sein um einen Transfer auf andere Aufgaben zu ermöglichen
    - Bibliotheken (z.B. torchvision) stellen fertig implementierte und vortrainierte „Modell-Zoos“ zur Verfügung



Bildquelle: Zeiler et al. "Visualizing and understanding convolutional networks." *ECCV* 2014

Transferlernen: Umsetzung

Zwei übliche Strategien für Transferlernen sind

1. Verwendung des CNNs zur **Merkmalsextraktion**
   - Entfernen des finalen Klassifikators
     - i.d.R. vollständig verbundene Schichten am Ende
     - Bei sehr unterschiedlichen Aufgaben z.T. auch spätere Faltungsschichten
   - Ersetzen durch beliebiges überwachtes Lernverfahren
2. **Verfeinerung** (*engl*. fine tuning) des CNN
   - Wenn die Klassifikation wieder durch ein neuronales Netz erfolgt ist es möglich auch die Faltungsschichten weiter zu trainieren
   - Viele Varianten sind denkbar, z.B.
     - Einfrieren des Merkmalsextraktors, bis der Klassifikator trainiert wurde
     - Einfrieren früherer Schichten, die vermutlich generelle Merkmale enthalten
     - Optimierung früherer Schichten mit geringeren Lernraten

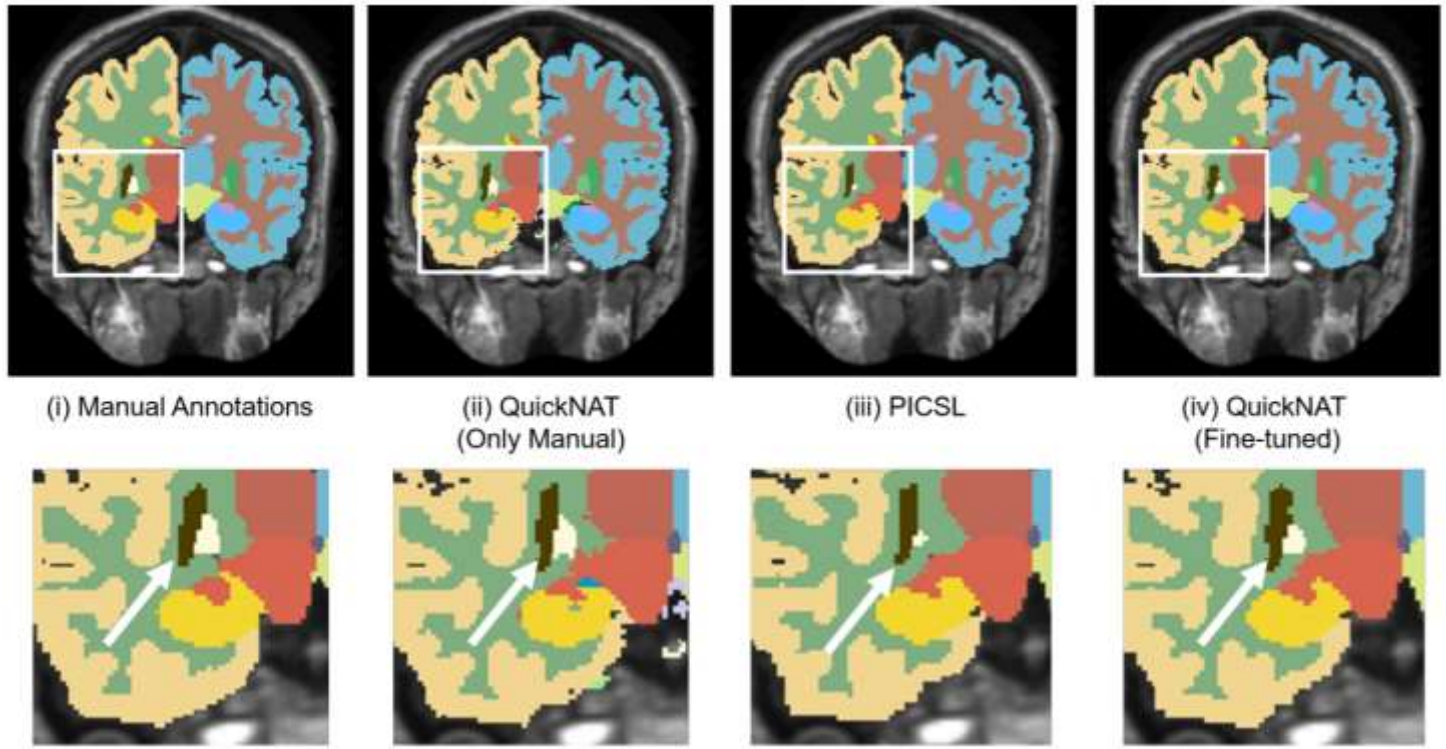
Zusammenfassung

- **CNNs zur Bildklassifikation** wurden mit der Zeit immer tiefer:
  - **AlexNet** führte zur Etablierung von CNNs zur Bildklassifikation
  - **VGG-16/19** arbeiten einheitlich mit 3x3-Faltungen
  - Die Inception-Module in **GoogLeNet** sparen massiv Parameter ein
  - Shortcut-Verbindungen im **ResNet** erlauben viel tiefere Netzwerke
    - Alternative Interpretation als Ensemble flacherer Netze
  - **EfficientNet** ist zusätzlich im Hinblick auf Recheneffizienz optimiert
- In der Praxis nutzt man meist eine Variante dieser etablierten Architekturen, häufig in Verbindung mit **Transferlernen**

6b.4 Bildsegmentierung mit CNNs

Motivation: Segmentierung von Hirnstrukturen

- *Beispiel*: Die **Segmentierung anatomischer Strukturen** in Hirn-MRT-Scans ist ein intensiv beforschtes Problem
  - Klassische Verfahren haben einen hohen **Rechenaufwand**
    - PICSL ca. 30h pro Hirn, Freesurfer ca. 4h
  - **CNN-basierte Verfahren** benötigen weniger als eine Minute, bei höherer Genauigkeit

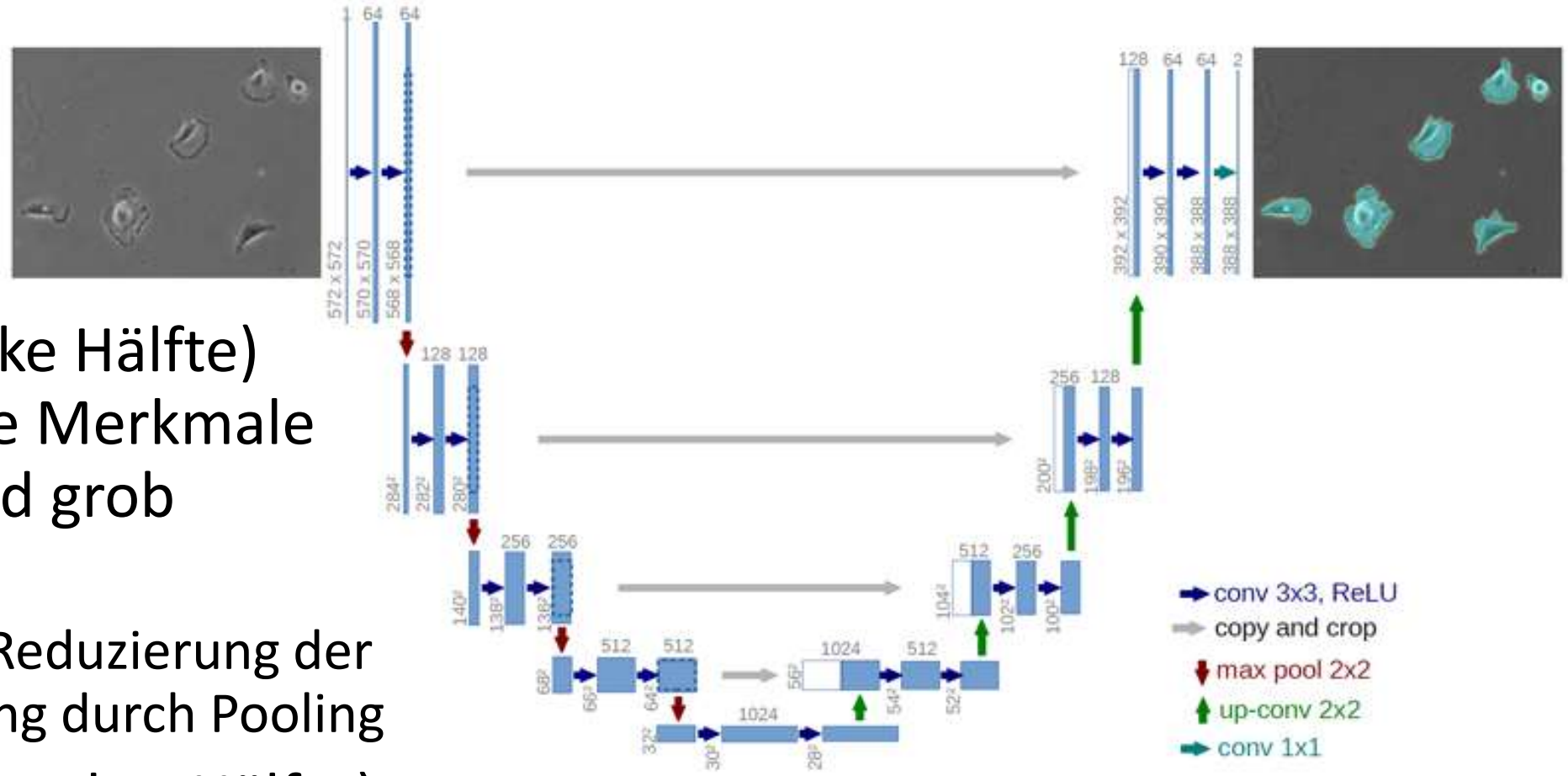


Bildquelle: Abhijit Guha Roy et al., „QuickNAT: A fully convolutional network for quick and accurate segmentation of neuroanatomy“ NeuroImage 2019

Bildsegmentierung mit CNNs

- **Bildsegmentierung** erfordert Klassifikation jeden Pixels/Voxels
- Erste Ansätze nutzten ein **gleitendes Fenster** zur Klassifikation des zentralen Pixels
  - Rechenaufwändig, hohe Redundanz zwischen den Durchläufen
- **Vollständig faltungsbasierte** (*engl*. fully convolutional) **Ansätze** klassifizieren einen größeren Bildausschnitt in einem einzigen Durchlauf
  - Die in der medizinischen Bildanalyse mit Abstand beliebteste Architektur ist das **U-Net** [Ronneberger et al. 2015]
    - Bis Januar 2025: ≈99.500 Zitate

Grundstruktur des U-Net

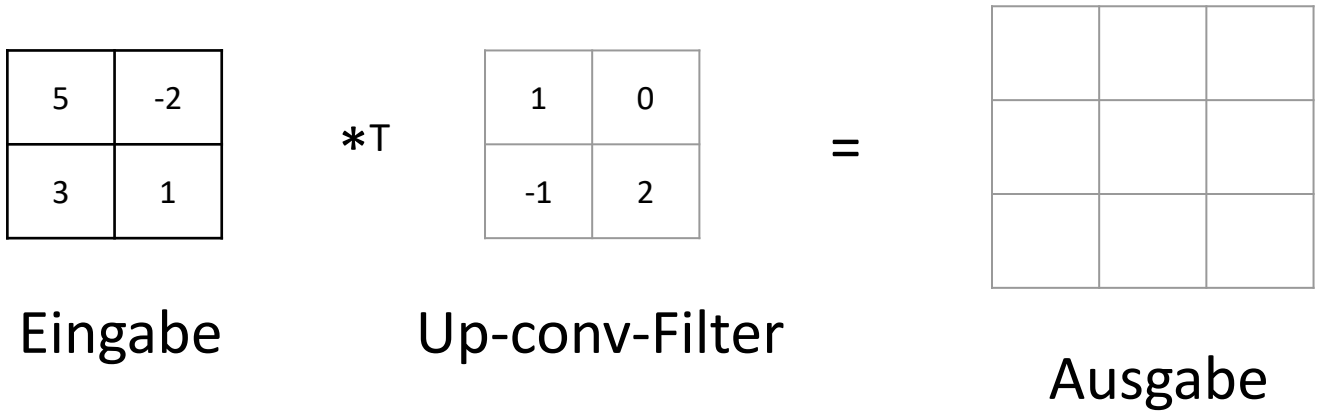


- **Kodierer** (linke Hälfte)
  soll relevante Merkmale erkennen und grob lokalisieren
  - Drastische Reduzierung der Bildauflösung durch Pooling
- **Dekodierer** (rechte Hälfte)
  soll die Klassen erkennen und pixelgenau abgrenzen
  - Transponierte Faltung stellt die Bildauflösung wieder her

Bildquelle: [Ronneberger et al. 2015]

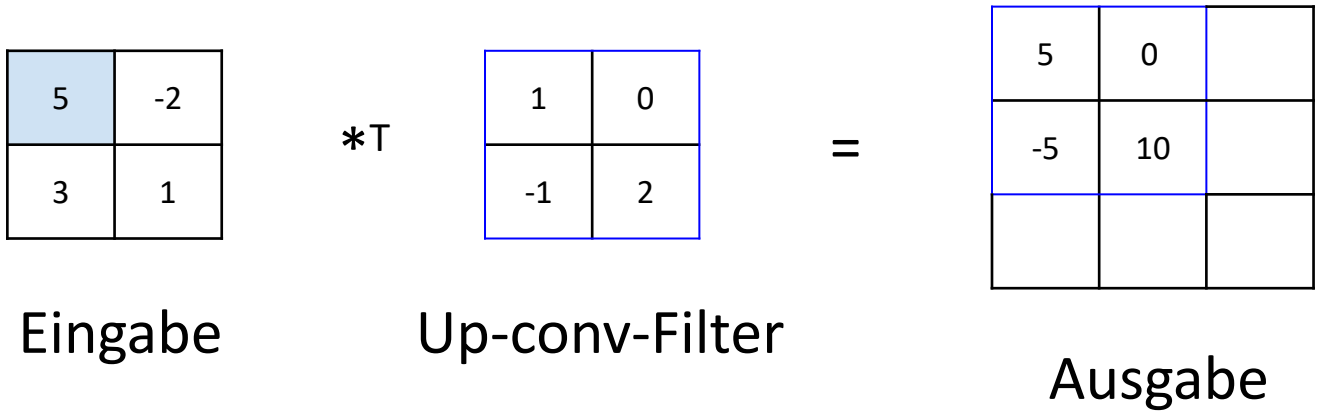
Transponierte Faltung

- Die **transponierte Faltung** (*engl*. auch „up convolution“) führt ein Upsampling mit lernbaren Gewichten durch
  - Pixel des Eingabe-Bilds skalieren den Filter
  - Ergebnisse werden mit einer vorgegebenen Schrittweite akkumuliert
  - *Beispiel*: 2x2-Filter mit Schrittweite 1
    - *Hinweis*: Der U-Net-Dekoder nutzt Schrittweite 2



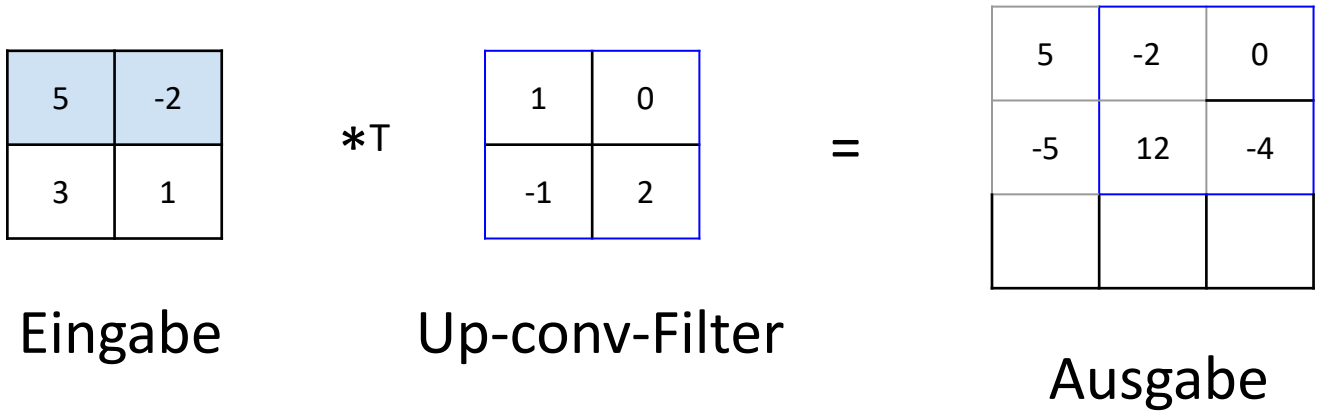
Transponierte Faltung

- Die **transponierte Faltung** (*engl*. auch „up convolution“) führt ein Upsampling mit lernbaren Gewichten durch
  - Pixel des Eingabe-Bilds skalieren den Filter
  - Ergebnisse werden mit einer vorgegebenen Schrittweite akkumuliert
  - *Beispiel*: 2x2-Filter mit Schrittweite 1
    - *Hinweis*: Der U-Net-Dekoder nutzt Schrittweite 2



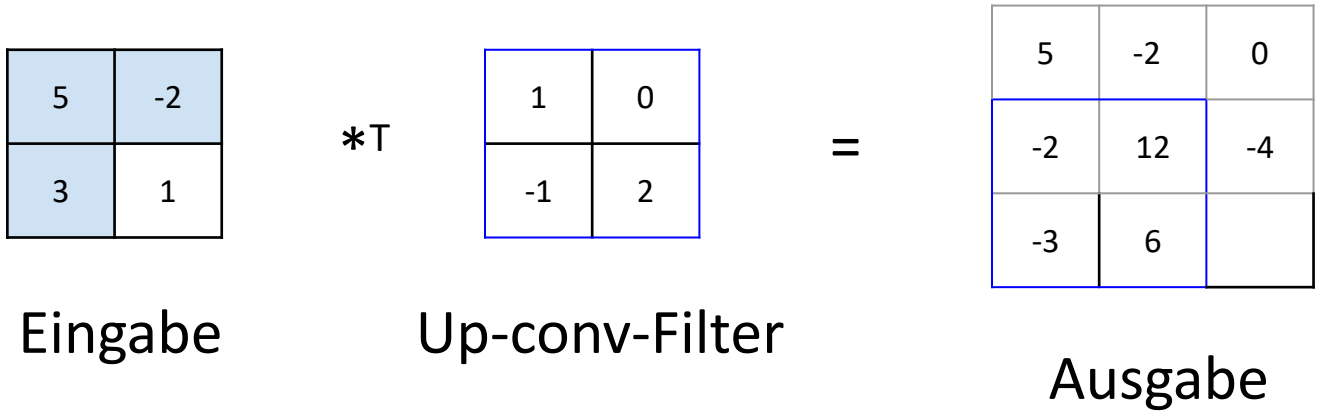
Transponierte Faltung

- Die **transponierte Faltung** (*engl*. auch „up convolution“) führt ein Upsampling mit lernbaren Gewichten durch
  - Pixel des Eingabe-Bilds skalieren den Filter
  - Ergebnisse werden mit einer vorgegebenen Schrittweite akkumuliert
  - *Beispiel*: 2x2-Filter mit Schrittweite 1
    - *Hinweis*: Der U-Net-Dekoder nutzt Schrittweite 2



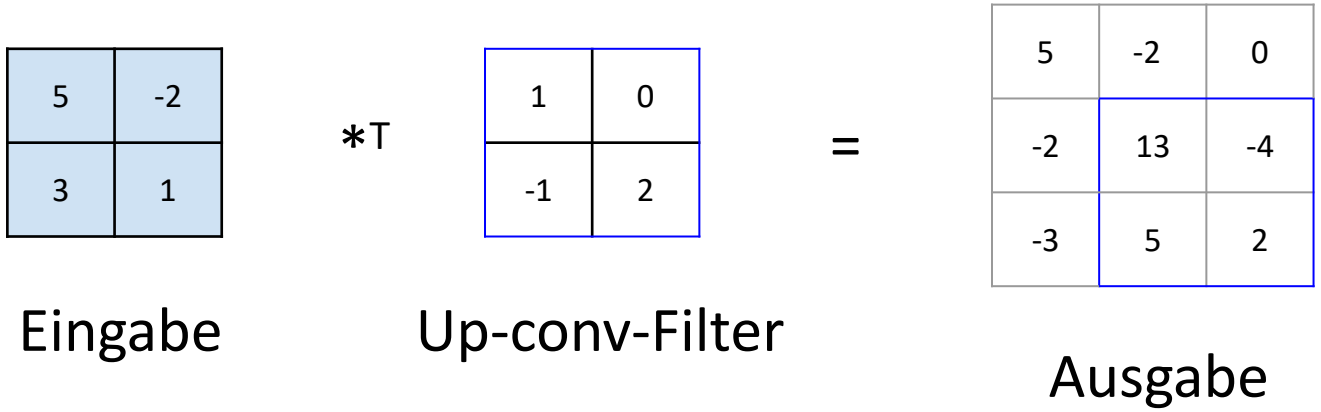
Transponierte Faltung

- Die **transponierte Faltung** (*engl*. auch „up convolution“) führt ein Upsampling mit lernbaren Gewichten durch
  - Pixel des Eingabe-Bilds skalieren den Filter
  - Ergebnisse werden mit einer vorgegebenen Schrittweite akkumuliert
  - *Beispiel*: 2x2-Filter mit Schrittweite 1
    - *Hinweis*: Der U-Net-Dekoder nutzt Schrittweite 2

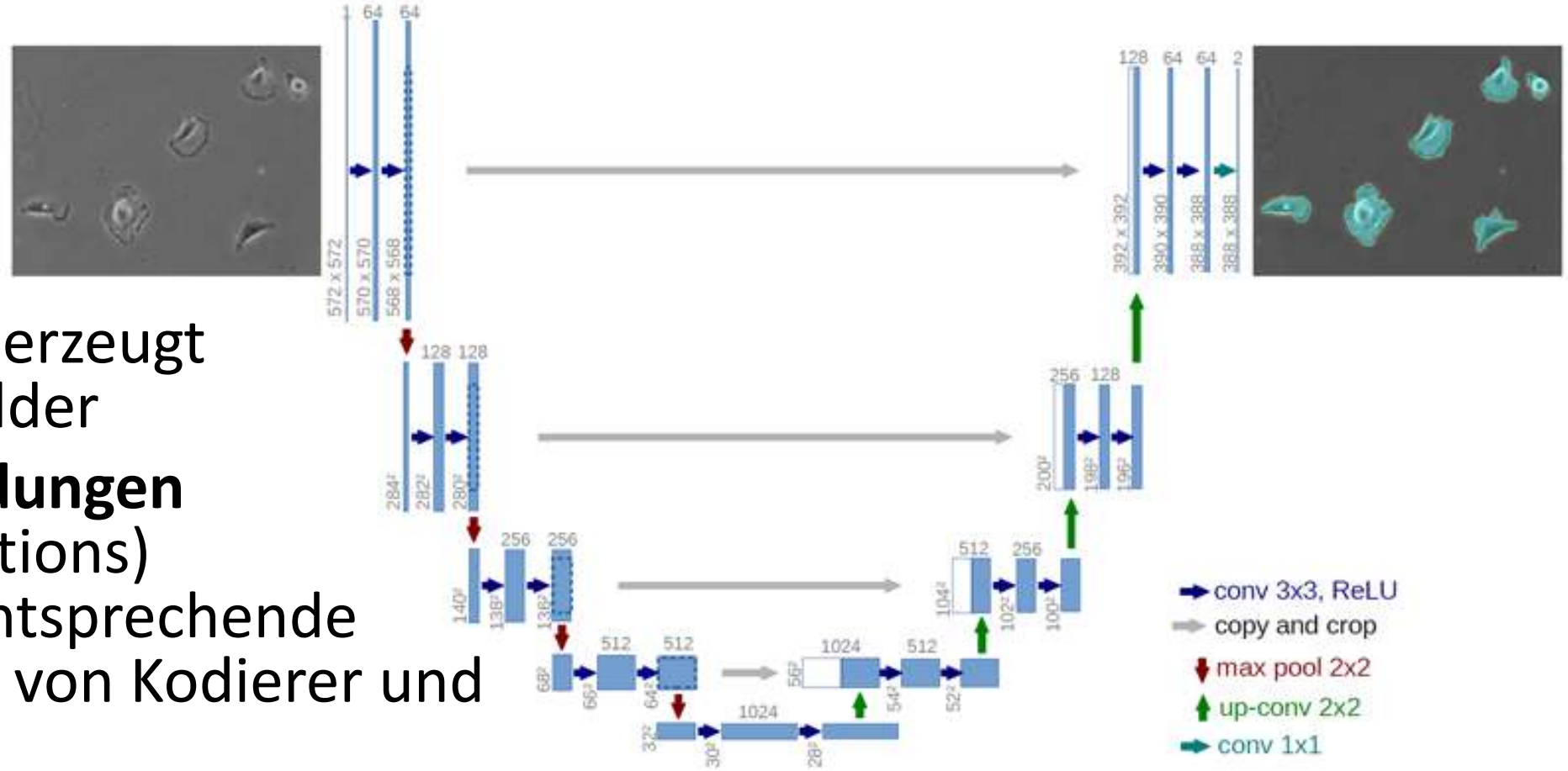


Transponierte Faltung

- Die **transponierte Faltung** (*engl*. auch „up convolution“) führt ein Upsampling mit lernbaren Gewichten durch
  - Pixel des Eingabe-Bilds skalieren den Filter
  - Ergebnisse werden mit einer vorgegebenen Schrittweite akkumuliert
  - *Beispiel*: 2x2-Filter mit Schrittweite 1
    - *Hinweis*: Der U-Net-Dekoder nutzt Schrittweite 2



Skip-Verbindungen

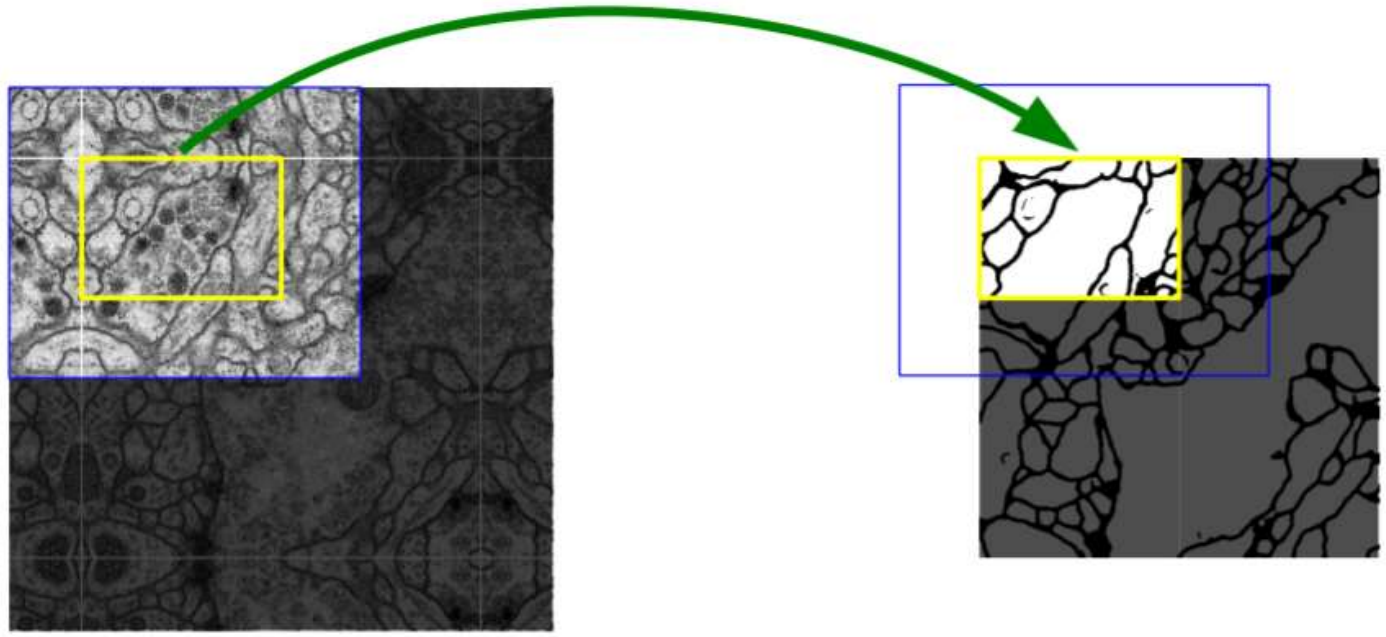


- **Upsampling** erzeugt unscharfe Bilder
- **Skip-Verbindungen** (skip connections) verbinden entsprechende Auflösungen von Kodierer und Dekodierer
  - Helfen bei der Dekodierung scharfer Segmentierungsmasken
  - Schneiden den bei „übersprungenen“ Faltungen verlorenen Rand ab

Bildquelle: [Ronneberger et al. 2015]

Verarbeitung überlappender Bildausschnitte

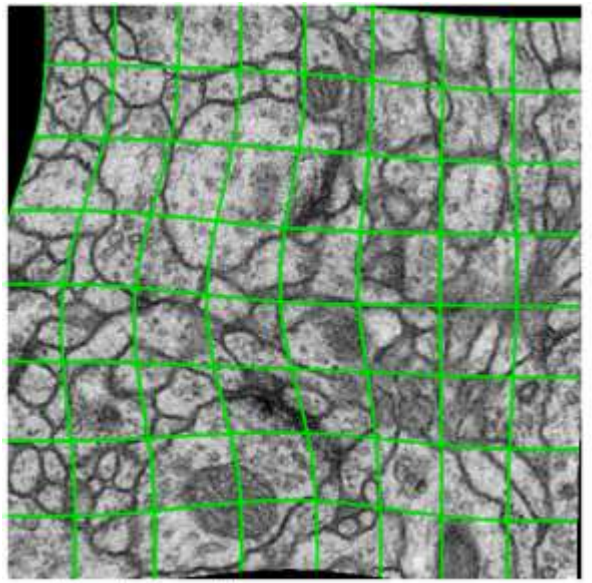
- Das U-Net nutzt nur den gültigen Teil der Faltungen
  - Kein Padding innerhalb des Netzes
  - Padding des Eingabebilds durch Spiegelung
- Größere Bilder werden stückweise verarbeitet
  - Ausschnitte überlappen so, dass die Ausgaben nahtlos aneinanderpassen

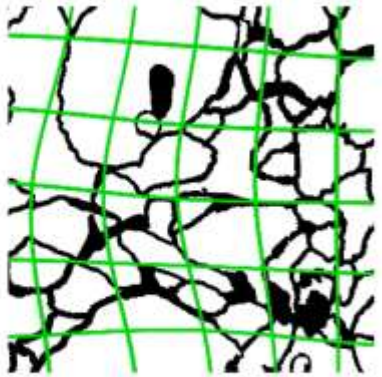


Bildquelle: [Ronneberger et al. 2015]

Augmentierung mit elastischen Deformationen

- Manuelle Annotationen von **Trainingsdaten** für Segmentierungsaufgaben sind besonders arbeitsintensiv
- [Ronneberger et al. 2015] nutzen zufällige **elastische Deformationen** als mächtige Augmentierung
  - Sinnvoll für die Segmentierung von Mikroskopiebildern von Zellen
- Augmentierungen müssen
  - sowohl auf die Bilder als auch auf die Segmentierungskarten angewandt werden
  - zur konkreten Anwendung passen


resulting deformed image


correspondingly deformed manual labels

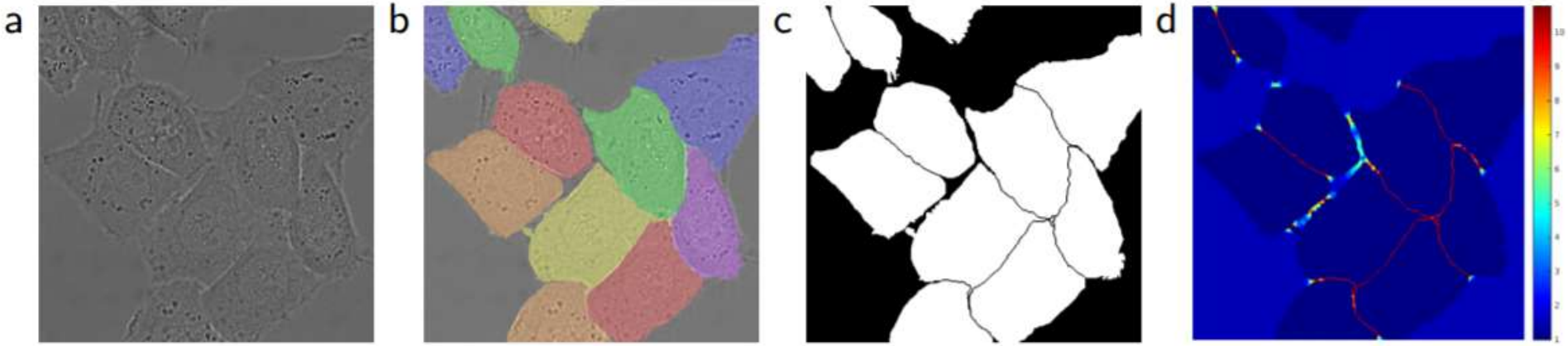
Bildquelle: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

Gewichtete Verlustfunktion

- **Mittelung der Kreuzentropie** über Pixel ist eine naheliegende Verlustfunktion für Segmentierungsaufgaben
  - *Aber*: Vernachlässigt kleine Klassen und feine Strukturen
- Zur Zellsegmentierung nutzen [Ronneberger et al. 2015] daher **Pixelgewichte**

$$w(x) = w_c(x) + w_0 \exp\left(-\frac{\left(d_1(x) + d_2(x)\right)^2}{2\sigma^2}\right)$$

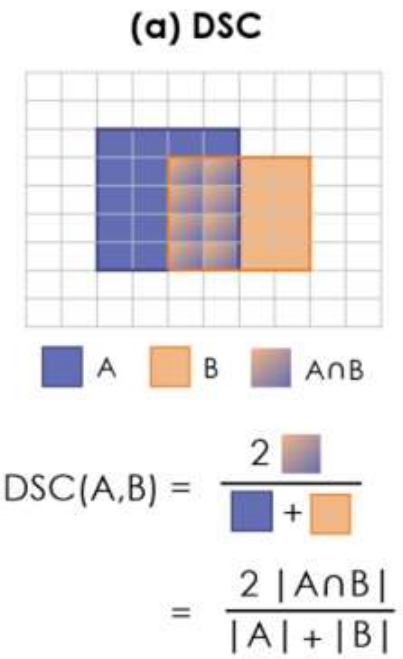
  - w_c berücksichtigt Häufigkeit der Klassen, w_0 Grenzen zwischen Zellen
  - d_1/d_2 sind Abstand zur nächsten und zweitnächsten Zelle



Bildquelle: [Ronneberger et al. 2015]

Alternative: Dice als Verlustfunktion

- *Erinnerung*: In Kapitel 4 haben wir den Dice-Score zur Quantifizierung des Segmentierungsüberlapps eingeführt
- [Milletari et al. 2016] leiten davon eine differenzierbare Verlustfunktion ab, mit der Netzwerke zur Segmentierung trainiert werden können:



$$\mathrm{DL}(a,b) = 1 - \frac{2\sum_i a_i b_i + \epsilon}{\sum_i a_i^2 + \sum_i b_i^2 + \epsilon}$$

  - Summen laufen über Pixel *i*, kleines $\epsilon > 0$ zur Regularisierung
  - Bestraft Fehlen von Vordergrund-Pixeln auch dann, wenn der Hintergrund dominiert

Abbildung von Reinke et al. 2021

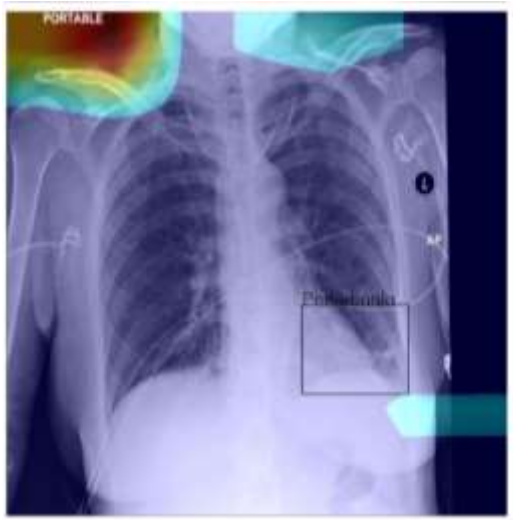
Zusammenfassung

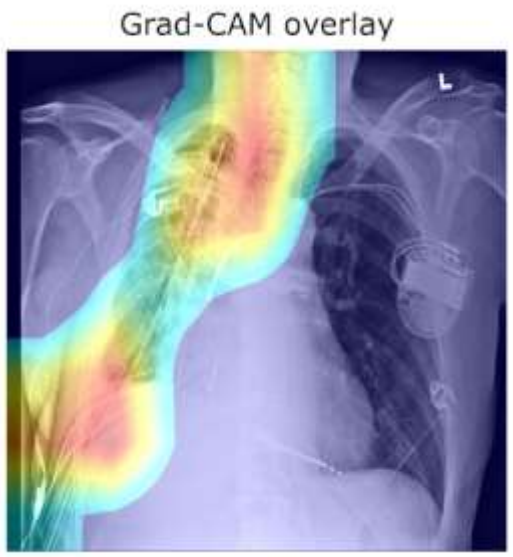
- **Vollständig faltungsbasierte Ansätze** ermöglichen eine effiziente Bildsegmentierung
- Das **U-Net** ist ein besonders beliebtes Beispiel
  - Basiert auf einer **Kodierer-Dekodierer**-Grundstruktur
  - **Transponierte Faltungen** lernen optimales Upsampling
  - **Skip-Verbindungen** unterstützen Dekodierung scharfer Karten
- **Erfolgreiches Training** wird ermöglicht durch
  - Geeignete **Augmentierungen**, z.B. elastische Deformationen
  - **Wichtung** der pixelweisen Kreuzentropie oder **Dice-Verlustfunktion**

6b.5 Einschränkungen Neuronaler Netze

Der „Kluger Hans“-Effekt

- Statt **unmittelbaren Anzeichen** einer Krankheit lernen neuronale Netze bisweilen, deren Behandlung oder Auswirkungen auf die Wahl der Bildgebung zu erkennen
- *Beispiele*:
  - Schwerere Fälle werden häufiger mit portablen Röntgengeräten untersucht
  - Drainageschlauch ist ein Indiz für einen behandelten Pneumothorax, aber keine sinnvolle Grundlage einer Diagnose

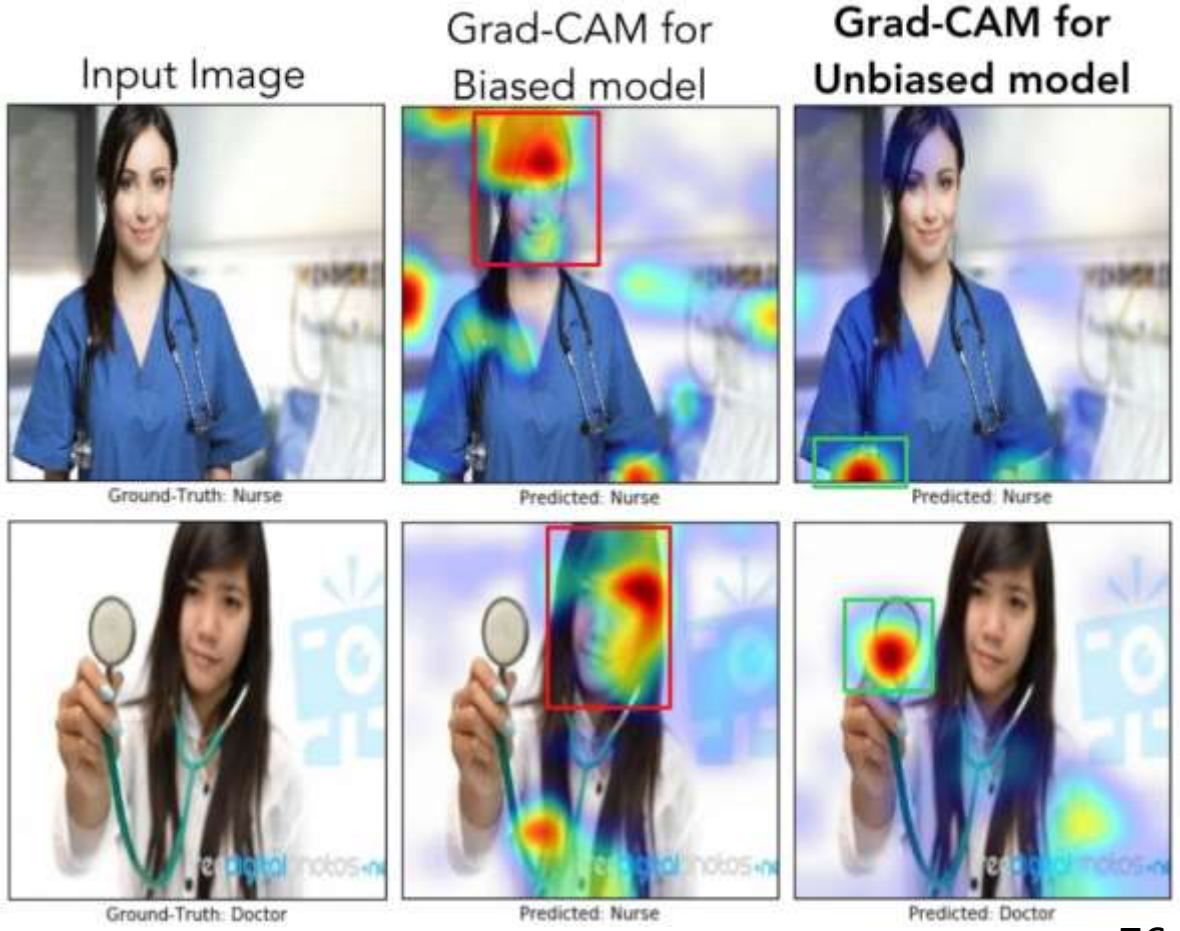




Bildquelle: Valsson/Arandjelovic, Sci 2022

MangelnDe Fairness

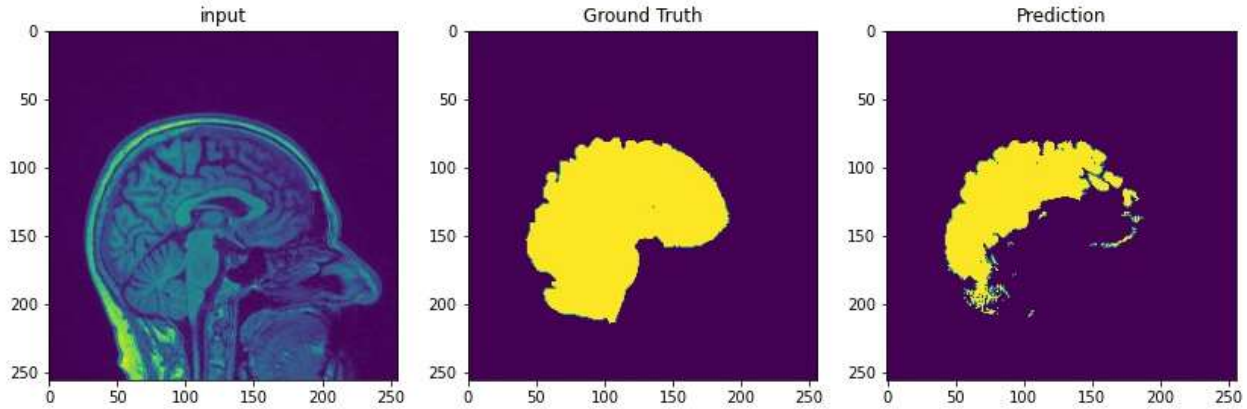
- Neuronale Netze **benachteiligen** häufig Personen aufgrund von Faktoren wie Geschlecht oder Ethnizität
  - Im Beispiel rechts wurde dieser Effekt durch sorgfältigere Auswahl der Trainingsdaten reduziert
- **Visualisierung** kann dabei helfen, solche Effekte aufzudecken
  - Visual Data Analysis, SoSe 2025

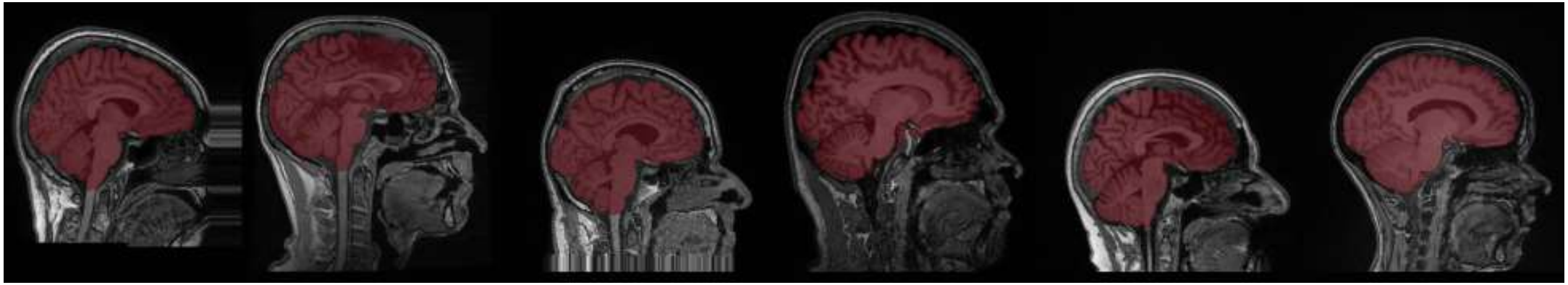


Bildquelle: Selvaraju et al. 2019

Domain Shift

- Genauigkeit neuronaler Netze bricht häufig ein, wenn die Bildcharakteristika sich von den Trainingsdaten unterscheiden **(“domain shift”)**
  - *Beispiel*: Wechsel / Upgrade des Scanners
  - Zusätzliches Problem: „Stilles“ Versagen

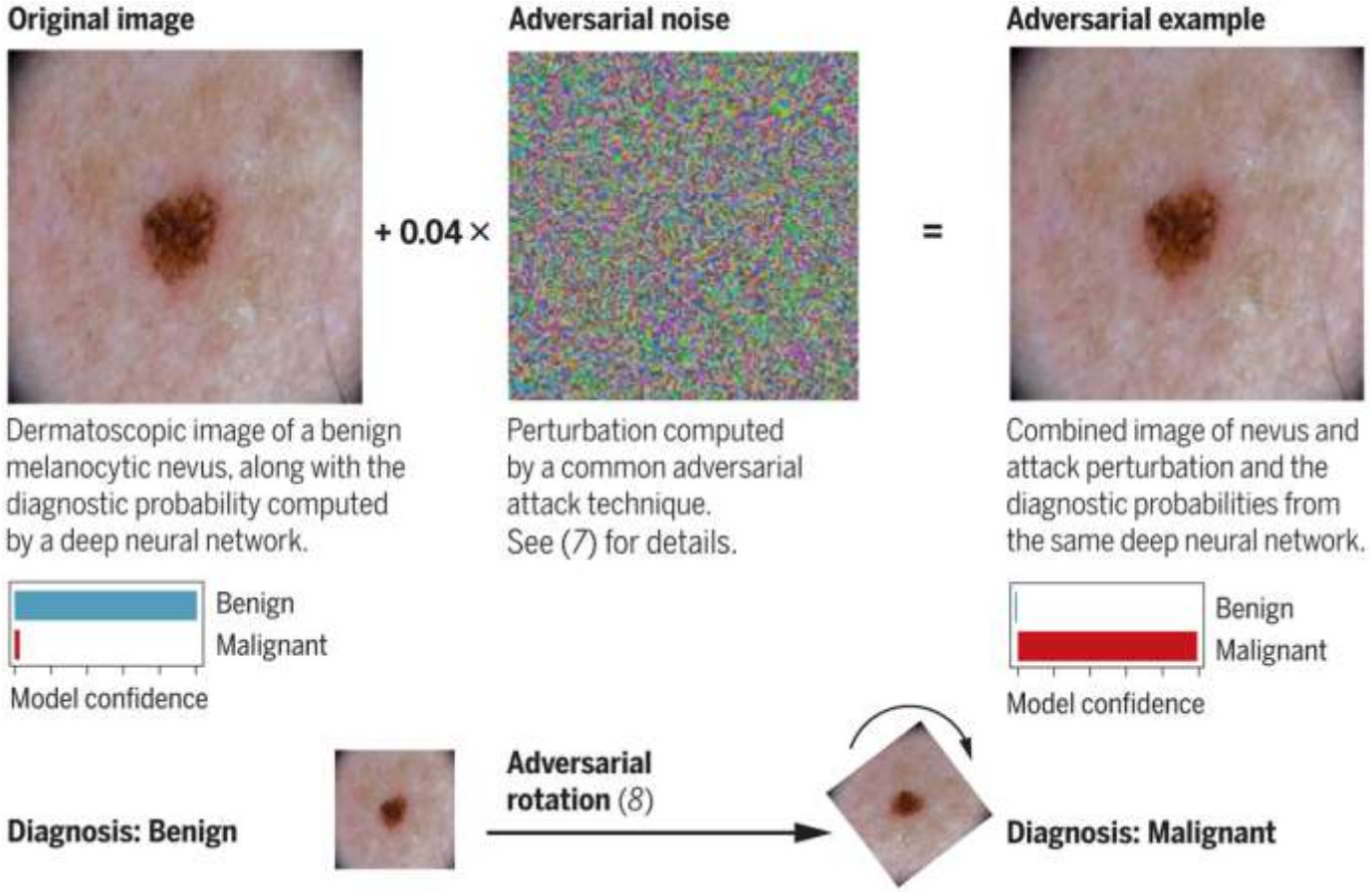




Bildquelle: Jonathan Lennartz

Angreifbarkeit mit manipulierten Bildern

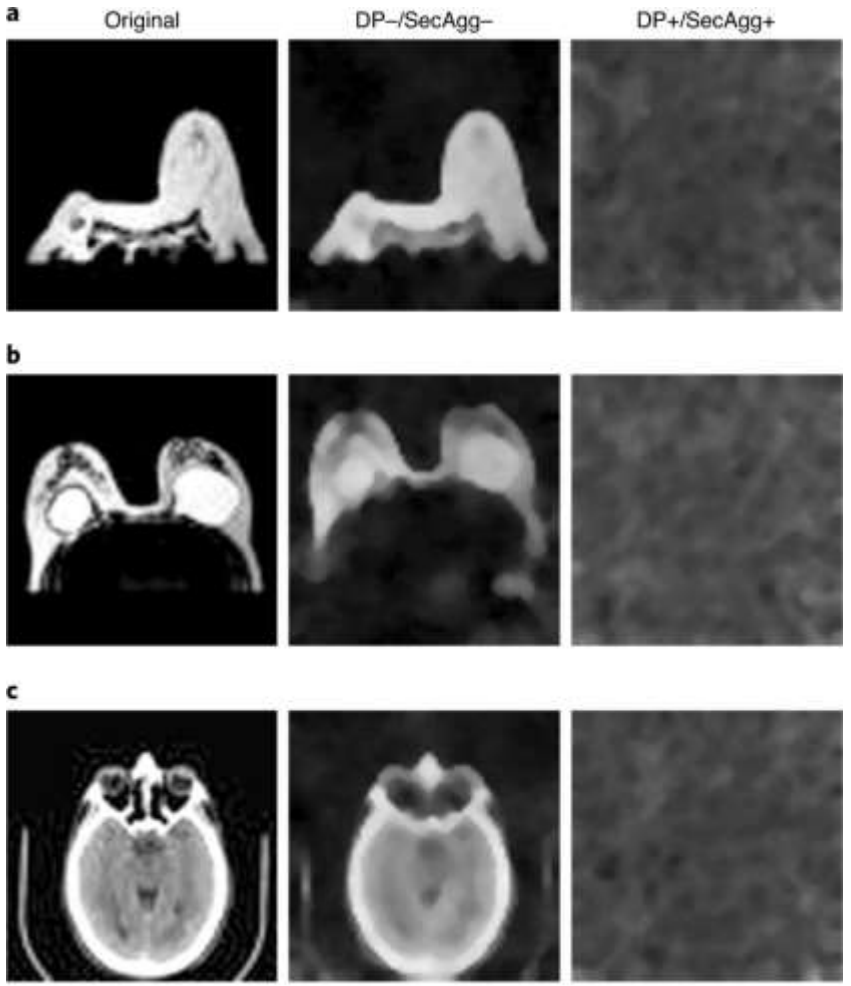
- Durch gezielte, für das menschliche Auge nicht erkennbare Manipulationen (*engl*. **adversarial attacks**) lassen sich die Entscheidungen neuronaler Netzwerke umkehren



Bildquelle: Finlayson et al., Science 2019

Rekonstruierbarkeit vertraulicher Trainingsdaten

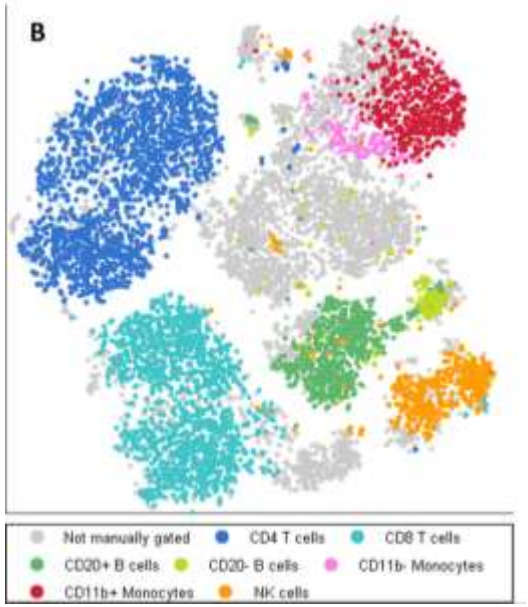
- Training neuronaler Netze zur medizinischen Bildanalyse erfordert große Mengen **besonders sensibler personenbezogener Daten**
- Ziel des **föderierten Lernens** ist es, die Bilder selbst vertraulich zu halten und nur (Updates der) Modellparameter auszutauschen
- Ohne weitere Sicherheitsvorkehrungen ist es jedoch möglich, erkennbare Bilder aus den Updates zu **rekonstruieren**

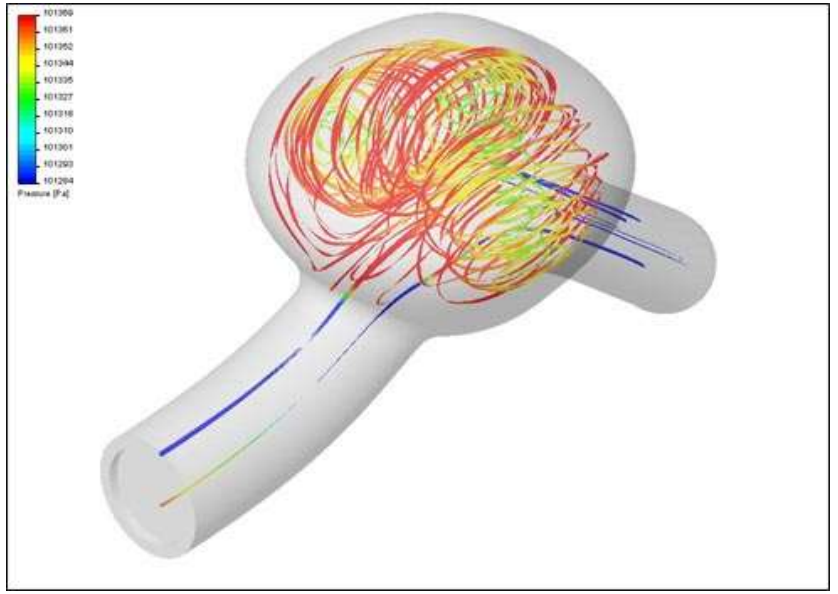


Bildquelle: Kaissis et al., Nature Machine Intelligence 2021

Angebote im SoSe 2025

- Unsere **Lehr-/Lernangebote im SoSe 2025:**
  - BSc-Projektgruppen und BSc-Arbeiten zum Thema „Medizinische Bildanalyse“
  - Vorlesung „Visual Data Analysis“ (4+2 SWS)
  - MSc-Lab „Visualization and Medical Image Analysis“





Zum Nach- und Weiterlesen

- Ian Goodfellow, Yoshua Bengio, Aaron Courville: “Deep Learning.” MIT Press, 2016 https://www.deeplearningbook.org/
- Christopher Bishop with Hugh Bishop: “Deep Learning. Foundations and Concepts.” Springer, 2024 https://www.bishopbook.com/
- Y. LeCun, L. Bottou, Y. Bengio, P. Haffner: Gradient-based learning applied to document recognition. Proc. of the IEEE 86(11):2278-2324, 1998
- A. Krizhevsky, I. Sutskever, G.E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012
- K. Simonyan, A. Zisserman: Very Deep Convolutional Networks for Large-Scale Visual Recognition. ICLR 2015

Zum Nach- und Weiterlesen

- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich: Going Deeper with Convolutions. CVPR 2015
- K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. CVPR 2016
- A. Veit, M. Wilber, S. Belongie: Residual Networks Behave Like Ensembles of Relatively Shallow Networks. NIPS 2016
- O. Ronneberger, P. Fischer, T. Brox: U-net: Convolutional networks for biomedical image segmentation. MICCAI 2015.