

Convolutional Neural Networks (Faltungsnetze)¹

Vorlesung 9, Artificial Intelligence

Dozenten:

Prof. Dr. M. O. Franz, Prof. Dr. O. Bittel, Prof. Dr. O. Dürr

HTWG Konstanz, Fakultät für Informatik

¹Folien basieren z.T. auf B. Sick, O. Dürr, DL Course.

Übersicht

1 Faltungsschichten

2 Deep Learning mit Faltungsnetzen

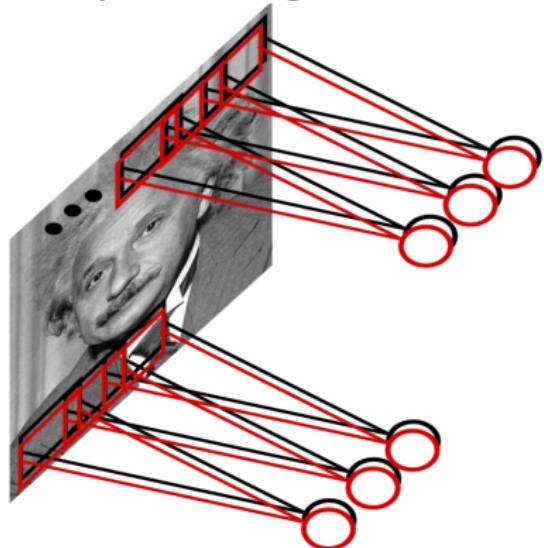
Übersicht

1 Faltungsschichten

2 Deep Learning mit Faltungsnetzen

Mehrere Filter und Eingabekanäle

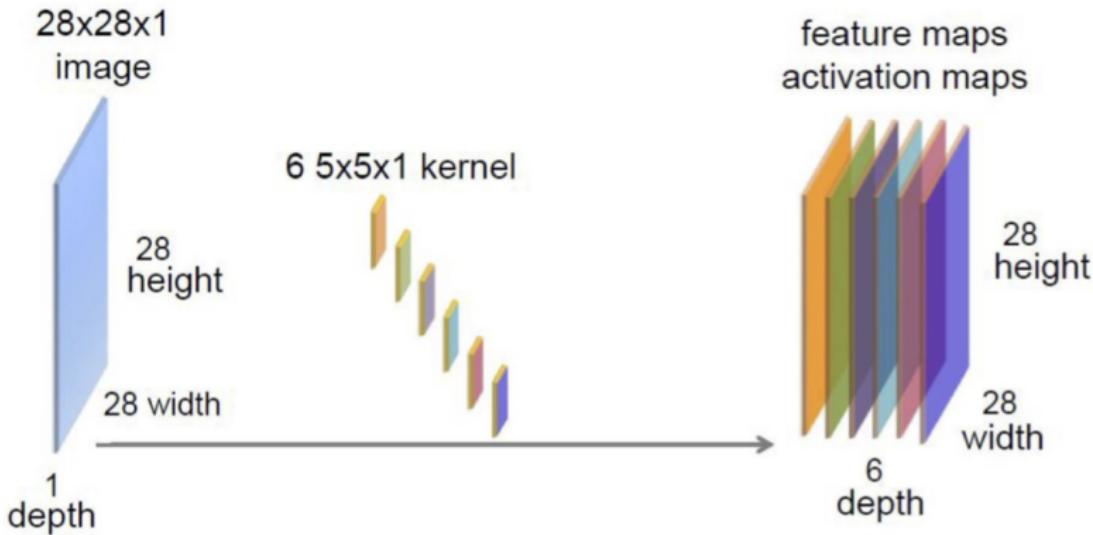
Meist werden mehrere gelernte Filter parallel eingesetzt:



[Ranzato, 2014]

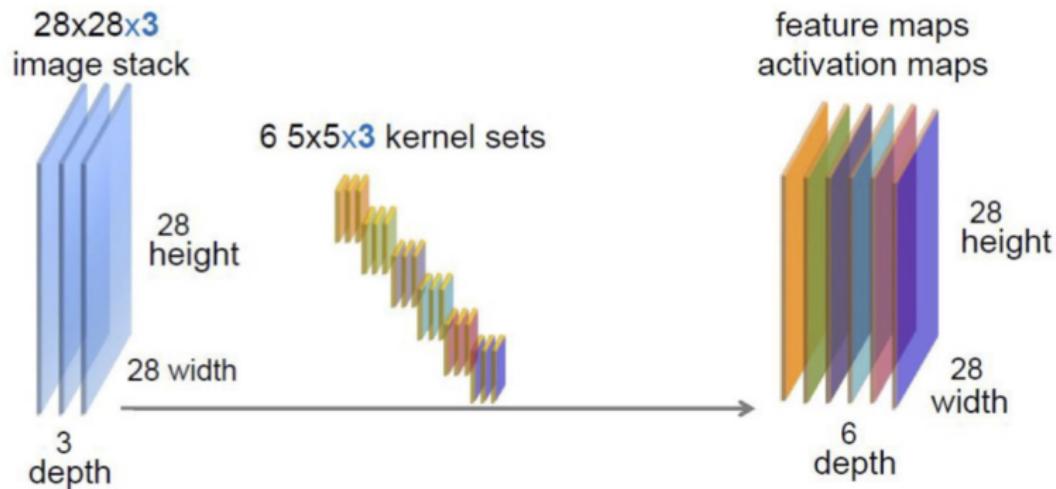
- Jede Faltung erzeugt aus einem Eingangsbild wieder ein Ausgangsbild.
- Bei n Filtern ist der Output der Faltungsschicht ein Block aus n Aktivierungskarten.
- Besteht der Input aus mehreren Kanälen (z.B. Farben oder Filterausgaben der vorigen Schicht), haben die Fenster in allen Kanälen die gleiche Größe und Position, aber eigene Gewichte.

Beispiel: einkanaliger Input, 6 Filterkerne



Die Faltung des Eingangsbildes mit 6 unterschiedlichen Filterkernen erzeugt 6 Aktivierungskarten. Wenn das Eingangsbild nur einen Kanal hat, dann sind auch alle Filterkerne einkanalig.

Beispiel: dreikanaliger Input, 6 Filterkerne

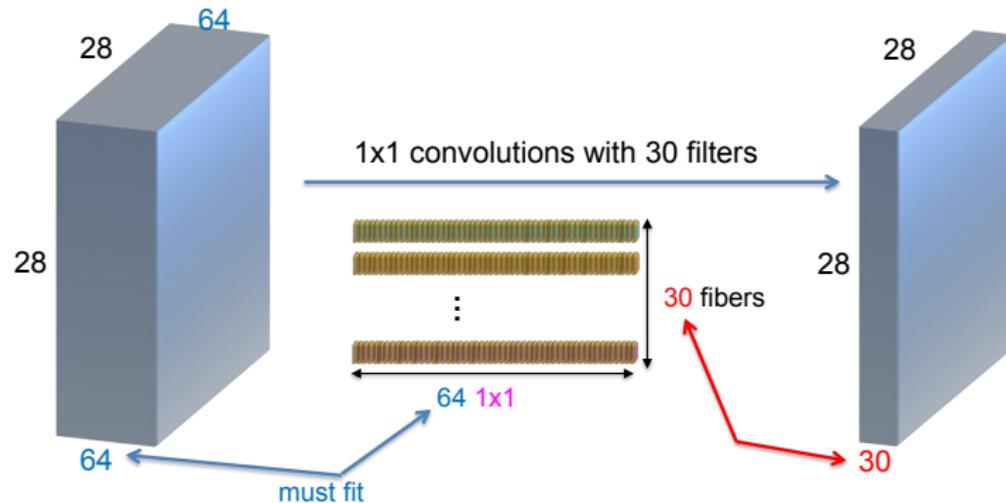


Die Faltung des Eingangsbildes mit 6 unterschiedlichen Filterkernen erzeugt 6 Aktivierungskarten. Wenn das Eingangsbild drei Kanäle hat, dann sind auch alle Filterkerne dreikanalig.

Animation: Faltung eines mehrkanaligen Bildes

[M.Gorner, TensorFlow, Keras and deep learning, without a PhD]

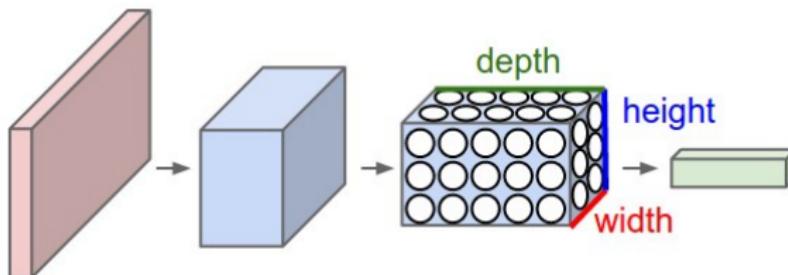
Filtergröße 1: 1x1-Faltungen wirken nur in die Tiefe



- 1x1-Faltungen wirken nur in der Tiefendimension entlang der verschiedenen Kanäle.
- Auf diese Weise kann die Anzahl der Kanäle verändert werden, ohne dass sich das Bild verändert.
- Führt zu einem verstärkt nichtlinearen Verhalten des Netzes.

Tensordarstellung

I.A. transformiert eine Faltungsschicht also einen dreidimensionalen Pixelblock in einen neuen dreidimensionalen Pixelblock.

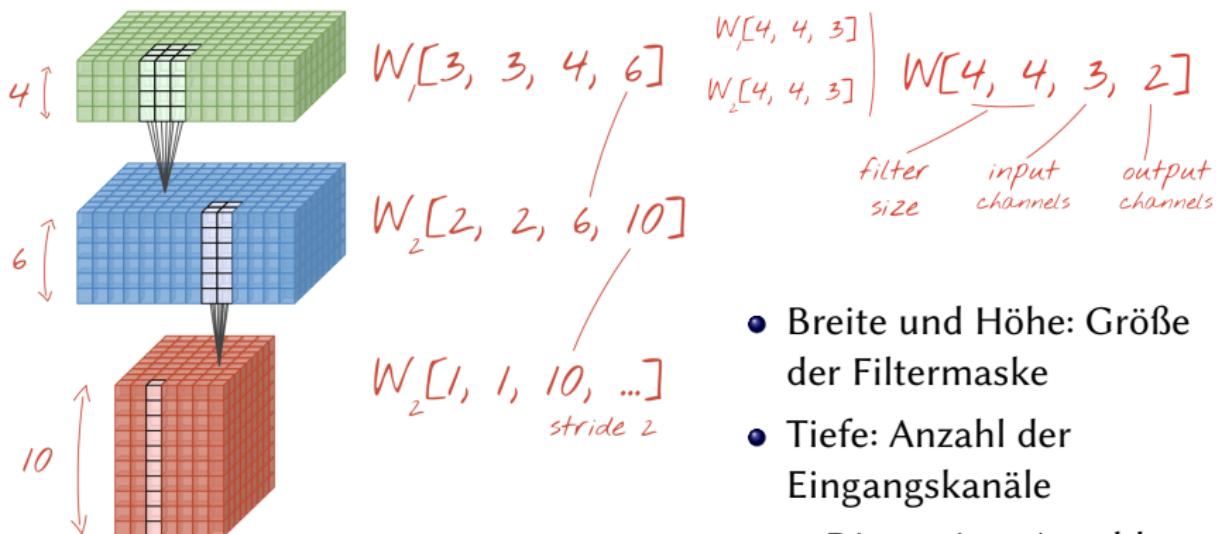


[<http://cs231n.github.io/convolutional-networks/>]

Die dreidimensionalen Arrays $w_{x',y',z}^l$ der Gewichte in den Faltungsschichten entsprechen einer Erweiterung des Matrixbegriffs auf mehr als zwei Dimensionen. Diese werden oft (mathematisch unpräzise) **Tensoren** genannt.

Filterkerne einer Faltungsschicht in Keras

Bei mehreren Filterkernen innerhalb einer Faltungsschicht ist der Gewichtstensor 4-dimensional:



[M.Gorner, TensorFlow, Keras and deep learning, without a PhD]

- Breite und Höhe: Größe der Filtermaske
- Tiefe: Anzahl der Eingangskanäle
- 4. Dimension: Anzahl Filterkerne

Bei mehreren gestapelten Schichten muss gelten: Anzahl Filterkerne = Anzahl Inputkanäle nächste Schicht

Berechnung einer Faltungsschicht

Im Netzwerk wird eine einzelne Faltungsschicht l mit einem Inputkanal mithilfe von 2D-Indizes x, y, x', y' als

$$a_{x,y}^l = \sigma \left(\sum_{x'} \sum_{y'} w_{x',y'}^l a_{x+x',y+y'}^{l-1} + b^l \right)$$

geschrieben, bzw. als schichtenweise Korrelation mit 3D-Indizes x, y, z, x', y', z' bei mehreren In- und Outputkanälen

$$a_{x,y,z}^l = \sigma \left(\sum_{x'} \sum_{y'} \sum_{z'} w_{x',y',z',z}^l a_{x+x',y+y',z'}^{l-1} + b_z^l \right).$$

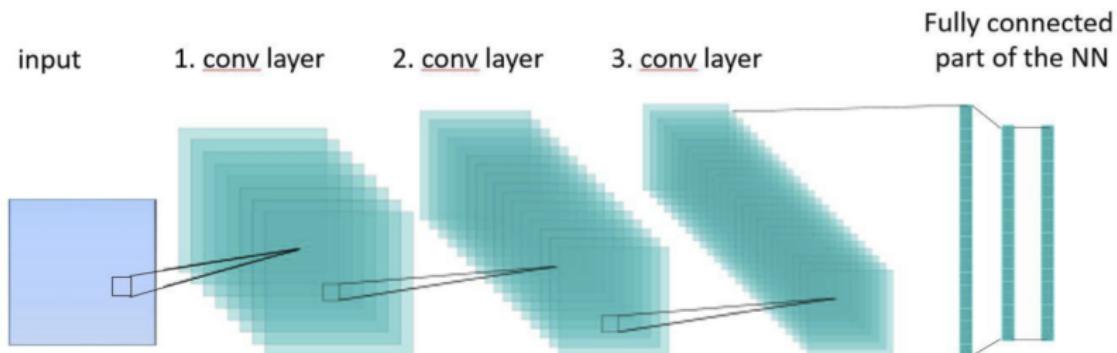
Ähnlich wie bei voll verbundenen Schichten werden die Gewichtstensoren mithilfe des Backpropagation-Algorithmus trainiert.

Übersicht

1 Faltungsschichten

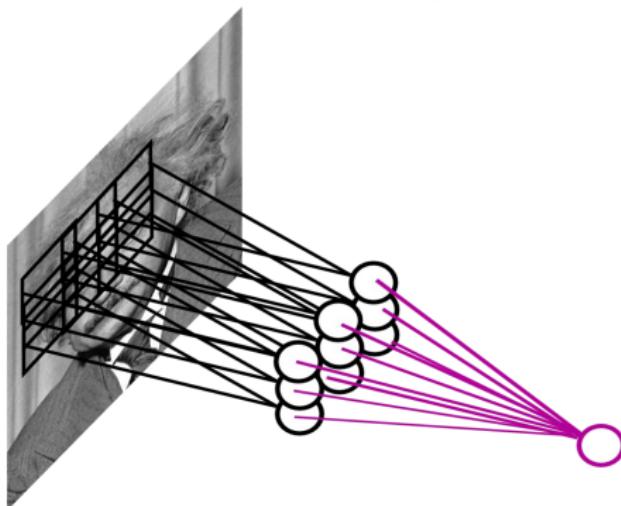
2 Deep Learning mit Faltungsnetzen

Faltungsnetze



Durch die Stapelung mehrerer Faltungsschichten erhält man ein **Faltungsnetz (Convolutional Neuronal Network, CNN)**. Nach dem Training haben wir **gelernte Deskriptoren**, die den Bildinhalt beschreiben. Hinter die Deskriptoren wird meist wieder ein voll verbundenes Netzwerk als Klassifikator oder Regressor geschaltet.

Ein weiterer CNN-Baustein: Pooling-Schichten



[Ranzato, 2014]

- Bei *Pooling* nimmt man z.B. das Maximum oder den Durchschnitt des Filteroutputs in einer lokalen Nachbarschaft zum Erreichen einer gewissen Translationsinvarianz.
- Durch eine Pooling-Schicht verkleinert sich die Größe des Outputs je nach Schrittweite zwischen den Pools.
- Pooling-Schichten haben keine trainierbaren Gewichte.

G. Hinton: "*The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.*"

Unterschiedliche Poolingtypen

Der Output wird um den Faktor m in x-Richtung und um n in y-Richtung verkleinert, die Anzahl der *feature maps* bleibt gleich.
 $\mathcal{N}_{x/m, y/n}$ ist der Pool des Neurons.

- Max-Pooling:

$$a_{x/m, y/n}^l = \max \left\{ a_{x,y}^{l-1} \mid x, y \in \mathcal{N}_{x/m, y/n} \right\}$$

- Durchschnittspooling:

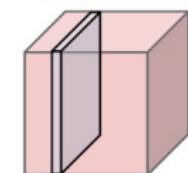
$$a_{x/m, y/n}^l = \frac{1}{mn} \sum_{x,y \in \mathcal{N}_{x/m, y/n}} a_{x,y}^{l-1}$$

- L2-Pooling:

$$a_{x/m, y/n}^l = \sqrt{\sum_{x,y \in \mathcal{N}_{x/m, y/n}} (a_{x,y}^{l-1})^2}$$

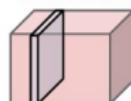
Beispiel Max-Pooling

224x224x64



pool

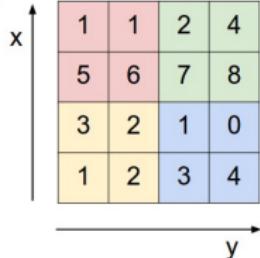
112x112x64



224
224

downsampling

112
112

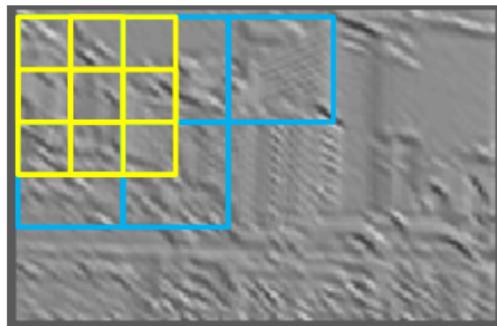


max pool with 2x2 filters
and stride 2

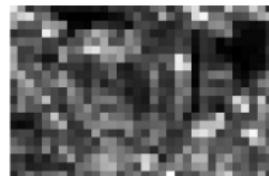
A 2x2 grid showing the result of max pooling. The values are:

6	8
3	4

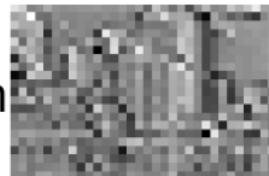
[CS231]



Max

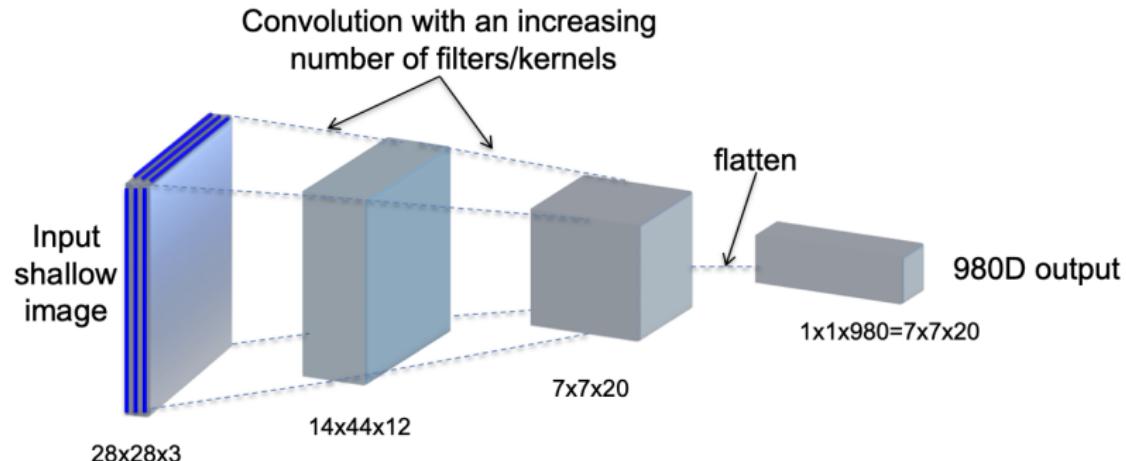


Sum



[Fergus, 2014]

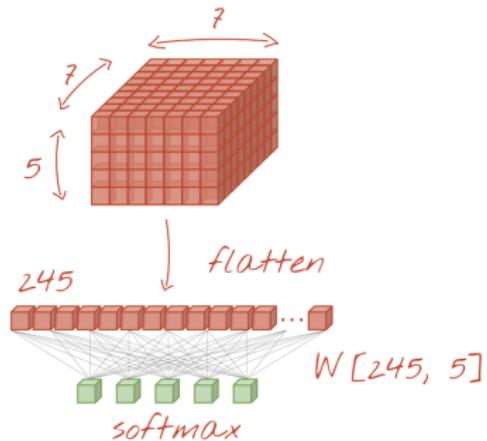
Faltungsnetz: typische Architektur



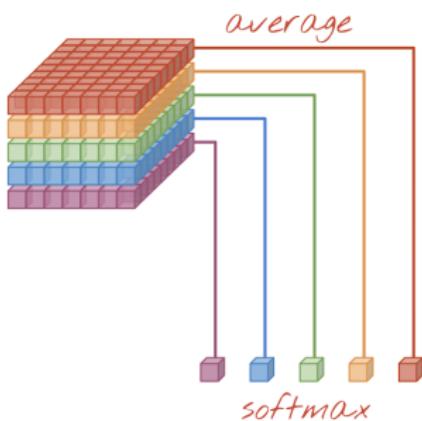
Die räumliche Auflösung wird durch Pooling oder Strided Convolution reduziert. Jede Stufe extrahiert komplexere bzw. abstraktere Merkmale als ihre Vorgänger. Nach einigen Stufen bekommen die Neuronen Informationen aus dem gesamten Bild. Pooling führt dabei zu einer deutlich verkleinerten Anzahl von Gewichten, was wiederum weniger Trainingsadaten erfordert.

Globales Average Pooling statt voll verbundener Schichten

Fully connected layer



Global average pooling



1225 weights

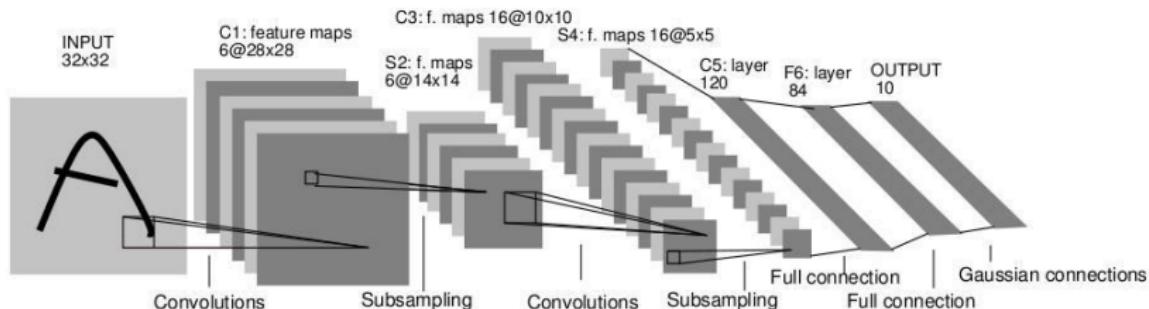
cheaper

0 weights



[M.Gorner, TensorFlow, Keras and deep learning, without a PhD]

Urvater aller CNNs: LeNet 5 (LeCun, 1998)



- 0.8% Fehler auf MNIST-Datensatz
- 7 Schichten, davon 2 Faltungs- und 2 Pooling-Schichten, sigmoide Aktivierungsfunktionen.
- 8.094 Neuronen, 344.068 Verknüpfungen, 11.880 frei trainierbare Parameter.
- 20 Iterationen über 65.000 Trainingsbeispiele, Gradientenabstieg mit stochastischem Levenberg-Marquart-Algorithmus

LeNet Demo 1993

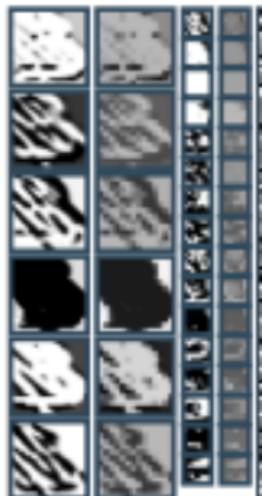


Läuft auf einem 486er-PC mit AT&T DSP32C add-on board (20 Mflops!)

Beispiel: LeNet Aktivierungskarten



4

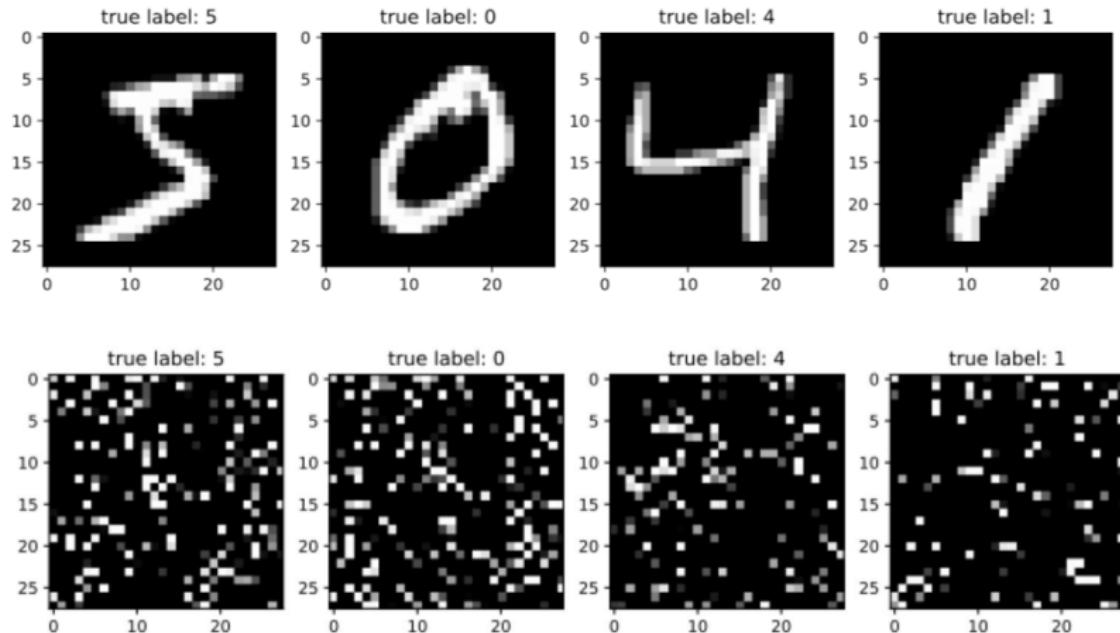


3



[LeCun et al., 1998]

Experiment: MNIST und Pixelpermutationen mit CNNs



[Demo]

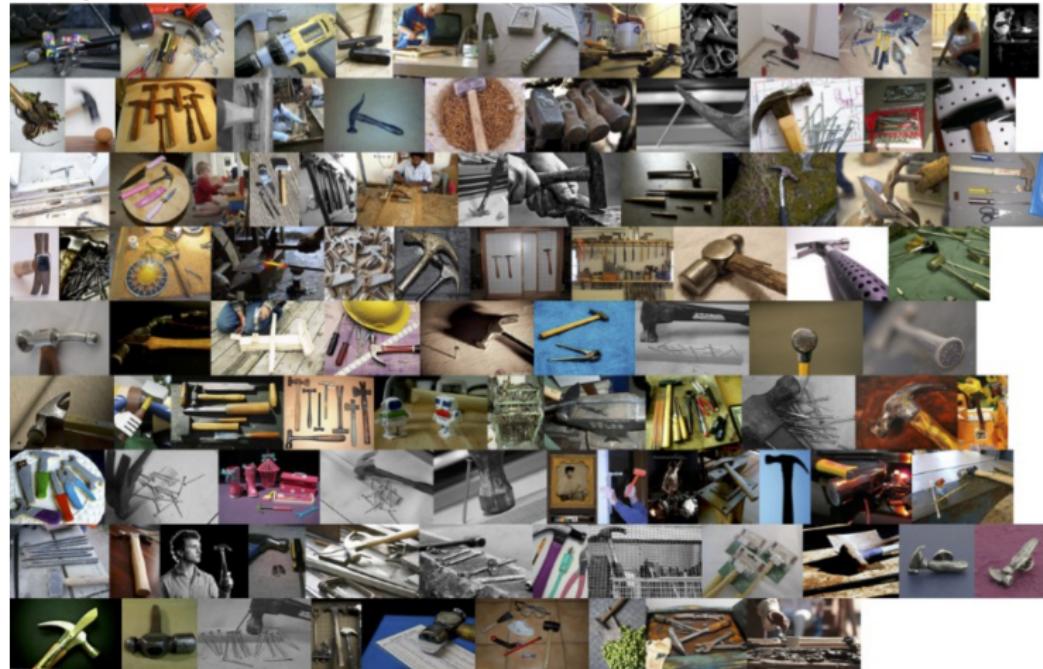
Vergleich: Voll verbundene Netze vs. CNNs

- Bei einem voll verbundenen Netz sind die Nachbarschaftsbeziehungen der Eingangsvariablen egal, bei einem CNN stören Pixelpermutationen erheblich.
- CNNs haben daher Bias-Effekt eingebaut: sie gehen von Daten aus, bei denen die Einzelvariablen eine räumliche Nachbarschaftsbeziehung haben.
- CNNs eignen sich für geordnete Daten, z.B. Bilder, Audiofiles oder 3D-Volumina, voll verbundene Netze eher für tabellarische Daten.
- Ein Neuron eines voll verbundenen Netzwerkes entspricht einem faltenden Neuron und damit einer Aktivierungskarte.
- In jeder voll verbundenen Schicht werden p Eingangsneuronen mit q Ausgangsneuronen verbunden, somit lernt man für jedes Neuron q verschiedene Linearkombinationen der p Eingangsneuronen. Im CNN werden p Eingangskanäle mit q Ausgangskanälen verbunden, somit werden q p -kanalige Filter gelernt.

ImageNet

200 Klassen, 450.000 Bilder der Größe 482×415

Beispiel Hammer:



CNN-Architekturen der ImageNet-Gewinner

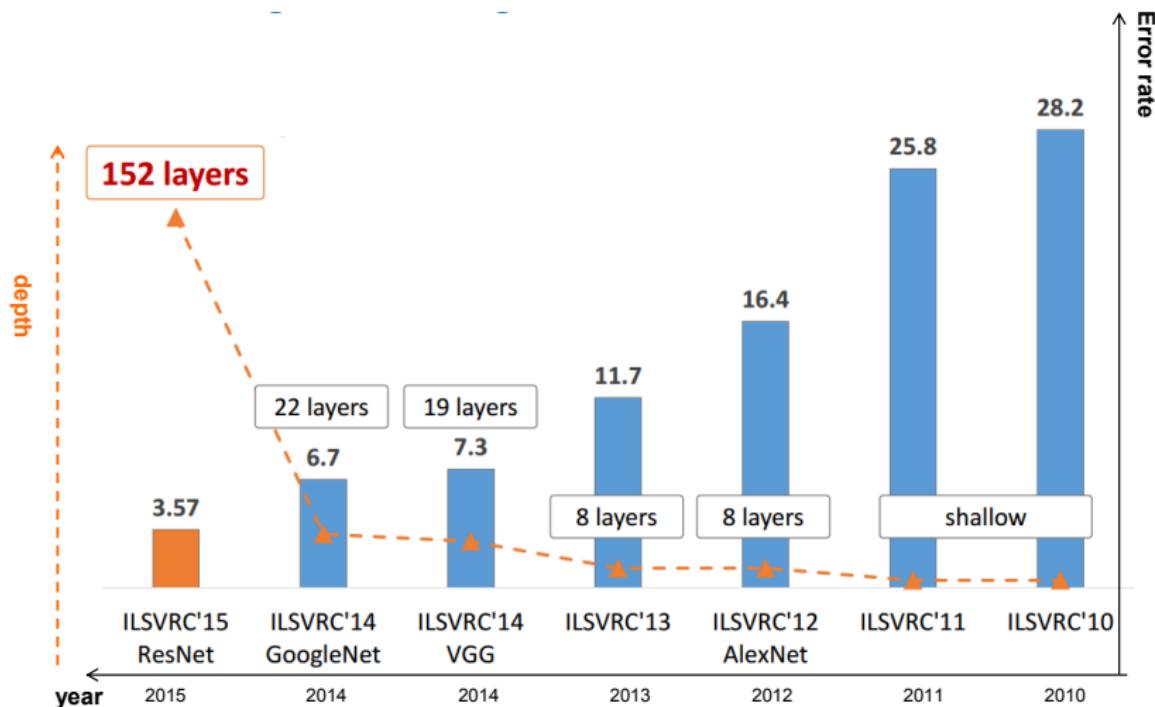
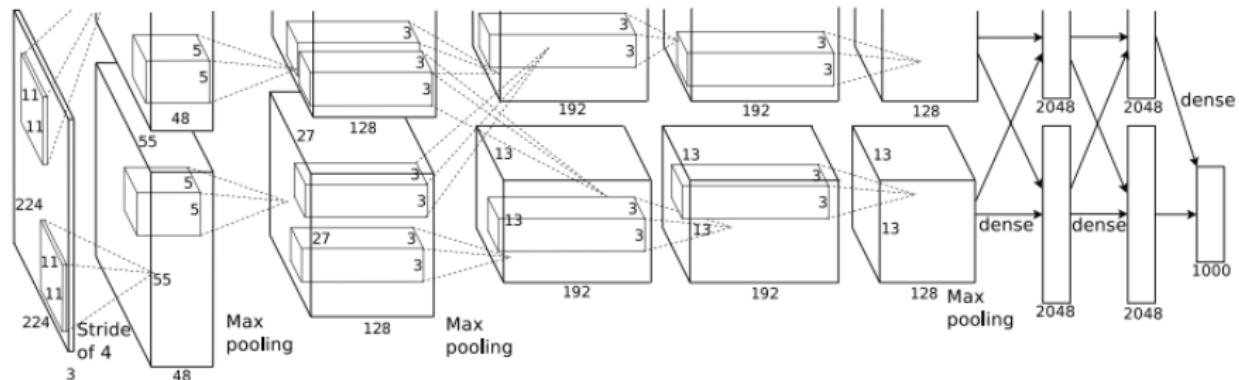


Image credits (modified): K.He, X.Zhang, S.Ren, J.Sun. "Deep Residual Learning for Image Recognition". arXiv 2015

AlexNet (Krishevsky et al., 2012)



- 8 Schichten, 650.000 Neuronen, 60 Mio. Verknüpfungen
- Trainiert eine Woche lang auf 2 GPU-Karten.
- ReLUs statt sigmoider Aktivierungsfunktionen.

AlexNet: Durchbruch der Deep-Learning-Architekturen



Gewinner der ImageNet 2012: AlexNet verbesserte den Fehler von 25.8% auf 16.4%. Danach wurden i.W. nur noch Deep-Learning-Architekturen verwendet.