Healthcare Data Analytics

Deep Convolutional Neural Networks

Dr. Michael Strobel

13.06.2022

Inhalt

Letzte Vorlesung

- Neuronale Netze (Fortsetzung)
- Unüberwachtes Lernen
- Semi-Überwachtes Lernen

Diese Woche

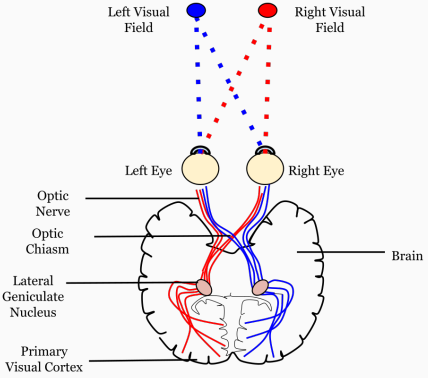
- Motivation durch visuellen Cortex
- Convolution (Faltung)
- Pooling
- Deep Convolution Neural Networks

Deep Learning

- Neuronalen Netze bestanden für uns bis jetzt aus wenigen und sehr ähnlichen Layern
- Deep Learning ändert den Aufbau von neuronalen Netzen so, dass
  - viel mehr Layer verwendet werden
  - Layer nicht mehr notwendigerweise vollständig mit dem vorherigen bzw. nachfolgenden verbunden sind
- Deep Learning hat in vielen Bereichen zu starken Sprüngen in der Leistungsfähigkeit geführt
  - Bildverarbeitung (z.B. Convolutional Neural Networks)
  - Spracherkennung (z.B. Recurrent Neural Network)
  - Unsupervised Learning (z.B. Autoencoder)
  - Strategie u.a. in Spielen (z.B. Reinforced Learning)
- Wir schauen uns heute Convolutional Neural Networks an

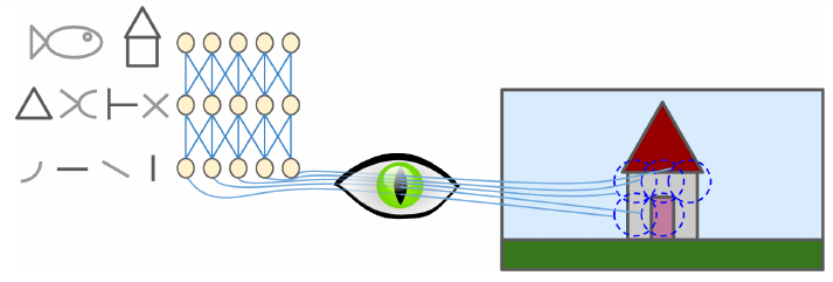
Der Visuelle Cortex

- Die Verarbeitung von Bildern im menschlichen Gehirn findet im *Visuellen Cortex*, einem Bereich des Gehirn statt
- Eine bedeutende Erkenntnis in den 50er Jahren war, dass viele Neuronen im Visuellen Cortex nur auf Ausschnitte des Sehfeldes reagieren
- Diese Ausschnitte nennt man *Lokales Rezeptives Feld* (engl. local receptive field)



Der Visuelle Cortex, cont'd

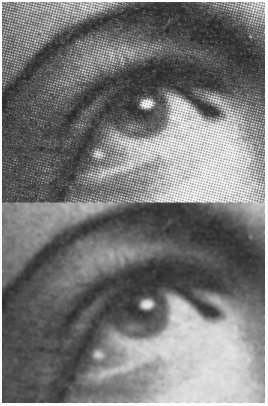
- Desweiteren reagieren manche Neuronen nur auf horizontale bzw. vertikale Linien oder andere geometrische Formen
- Zudem gibt es verschieden große receptive fields und somit reagieren manche Neuronen auf *low level* Details (Linien, Kurven ...) und andere auf komplexere *high level* Formen
- Diese Beobachtungen haben zur Vermutung geführt, dass die high level Neuronen auf dem Input der low level Neuronen basieren
- Schlussendlich hat dies zur Idee von *convolutional neural networks* geführt



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Convolution (Faltung)

- Faltungen werden in der Bildverarbeitung als *Filter* eingesetzt
- Es können zahlreiche Effekte erreicht werden wie z.B. Glättung, Schärfen, Kantenerkennung, ...
- Dies kann auch in verschiedene Richtungen erreicht werden wie z.B. erhalte mir nur Kanten die horizontal laufen



Gaußfilter zur Glättung

Convolution (Faltung), Definition

- Eine Faltung besteht aus einem Input Signal I und einem Faltungskern k
- Das gefaltete Signal I^* mit dem Faltungskern k definiert sich über

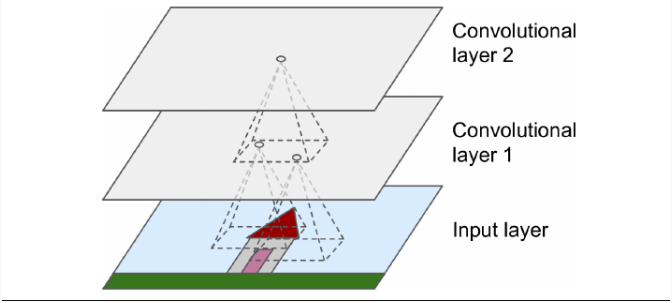
$$I^*(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} I(x-i+a,\ y-j+a)k(i,j)$$

- Ein 3×3 Gaußfilterkern stellt im Output-Bild I^* ein Pixel als Mittelwert der Umgebung dar

$$k(x,y) = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

CNN Layer

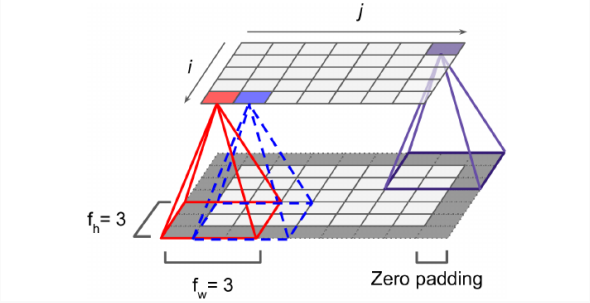
- Diese Faltungen können als Layer in ein NN eingebaut werden
- Neuronen in diesem Netzwerk sind dann nur mit Teilen des vorhergehenden Convolution Layer verbunden (local receptive field)
- Das Netz lernt im ersten Layer dann die Low Level Details
- Diese Low Level Features werden im nächsten Layer zu komplexeren Strukturen zusammen gesetzt und diese wiederum im nächsten Layer in noch komplexere Strukturen usw. . .



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Randbehandlung mit Padding

- An den Rändern des Bildes kann es zu Problemen kommen, da die Bildgröße eventuell kein Vielfaches der Faltungskerngröße
- Hier benutzt man das sogenannte *Padding*:
  - *zero padding* (TensorFlow: same padding): hier wird mit 0 aufgefüllt
  - *no padding* (TensorFlow: valid padding): hier wird einfach abgeschnitten

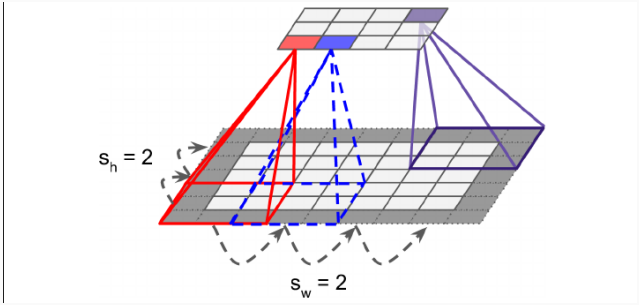


Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Dimensionsreduktion mit Strides

- Um die Dimension des Bildes zu verkleinern kann ein Convolutional Layer einen Versatz (*Stride*) größer als 1 haben
- Hierbei wird ein größerer Abstand zwischen den receptive fields angesetzt

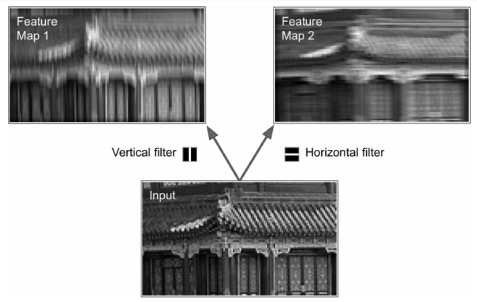
Beispiel: Input Layer $5 \times 7 \rightarrow 3 \times 4$ Layer mit einem 3×3 receptive field und einem Stride von 2



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Feature Maps

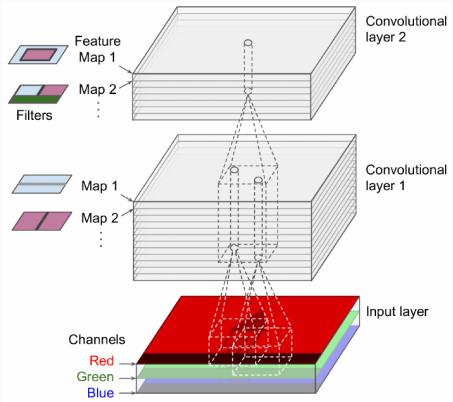
- Die Anwendung eines Faltungskern (= Filter) auf eine Eingabe nennen wir das *Feature Map*
- Faltungskerne werden **nicht** vordefiniert, sondern das CNN lernt die besten Filter anhand der Trainingsdaten



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

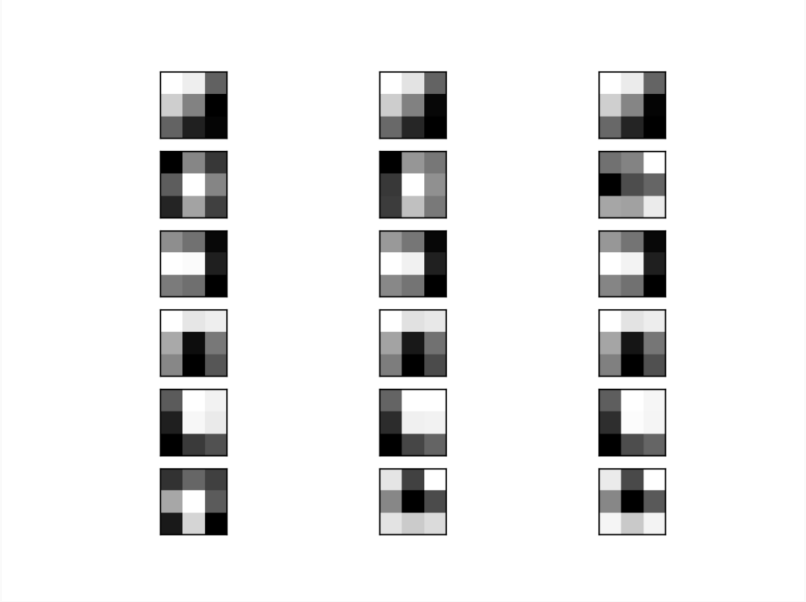
Stapel von Feature Maps

- Ein Convolutional Layer kann viele Faltungskerne und somit auch viele Feature Maps beinhalten
- Jede Feature Map kann unterschiedliche Faltungskerne lernen
- Die Gewichte sind aber für alle Feature Maps gleich (Layer Gewicht)



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

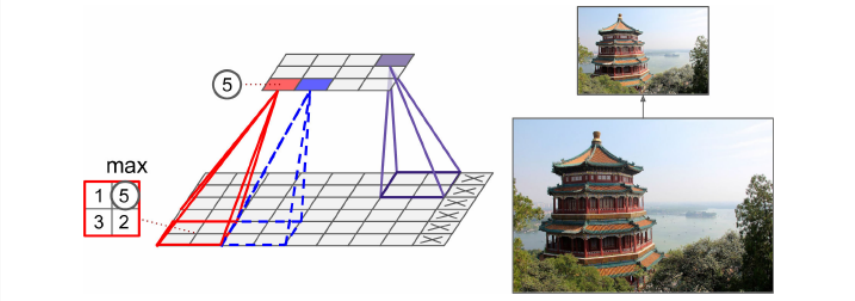
Stapel von Feature Maps, Visualisierung der Gewichte von VGG Net



https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

Pooling

- Ein *Pooling Layer* funktioniert ähnlich wie ein Convolutional Layer
- Im Gegensatz zum Convolutional Layer hat er aber **keine** Gewichte die trainiert werden können
- Auch er hat Parameter wie größe, stride und padding
- Die Ausgabe eines Pooling Layer hängt ab vom Typ
  - **Max Pooling**: gibt maximum des receptive field zurück
  - **Average Pooling**: gibt Mittelwert des receptive field zurück

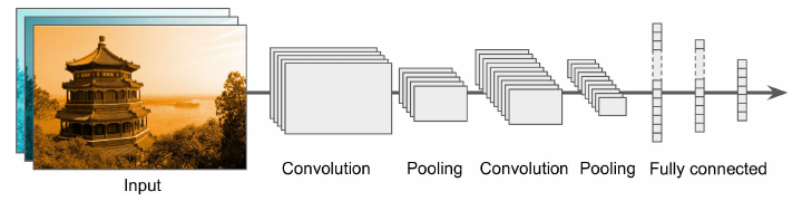


Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Architektur von Convolutional Neural Networks

Ein typisches CNN hat die folgende Form

- Input
- Convolution
- Pooling
- Convolution
- Pooling
- ...
- Fully Connected Neural Network (klassische Architektur) mit Regularisierung (Dropout)
- Output



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Implementierung in TensorFlow mit Keras

- TensorFlow wird von Google entwickelt und stellt eine Python Schnittstelle für die GPU bereit
- Keras basiert auf TensorFlow und implementiert typische Machine Learning Elemente

Keras / TF Implementierung einer CNN Architektur

```
from functools import partial

DefaultConv2D = partial(keras.layers.Conv2D,
                        kernel_size=3, activation='relu', padding="SAME")

model = keras.models.Sequential([
    DefaultConv2D(filters=64, kernel_size=7, input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=128),
    DefaultConv2D(filters=128),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=256),
    DefaultConv2D(filters=256),
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=10, activation='softmax'),
])
```

Géron, Aurélien. “Hands-on machine learning with scikit-learn and tensorflow”

LeNet-5

- Eines der ersten CNN Netze war *LeNet-5* von Yann LeCun (1988!)
- Es wurde u.a. erfolgreich genutzt um handgeschriebene Zahlen zu erkennen (MNIST)

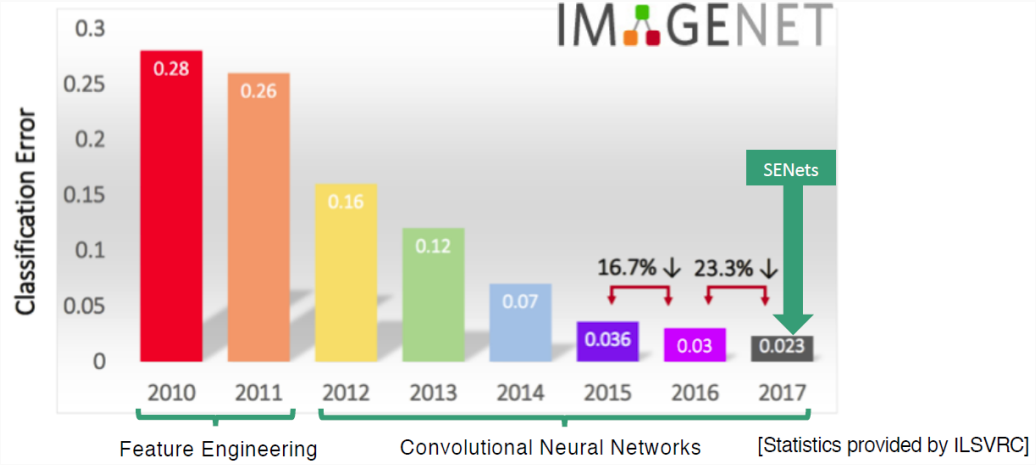
Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Anwendung: ImageNet Large Scale Visual Recognition Challenge

- 2012: 150.000 Bilder und 1000 Kategorien
- Ziel: möglichst wenig Fehler bei der Kategorisierung
- CNN haben die Fehlerrate von 26% auf 17% gedrückt
- Die nächsten Jahre ging die Fehlerrate auf unter 0.1% zurück

ImageNet Statistik



AlexNet

- AlexNet gewann das 2012 ImaeNet Challenge
- Als Regularisierung wurde Dropout verwendet aber auch Data Augmentation

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	–
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	–
C1	Convolution	96	55×55	11×11	4	VALID	ReLU
In	Input	3 (RGB)	227×227	–	–	–	–

Géron, Aurélien. “Hands-on machine learning with scikit-learn and tensorflow”

Weitere Netze und Transfer Learning

Es gibt noch viele weitere Netztopologien

- GoogLeNet
- VGGNet
- ResNet
- Xception
- SENet

Diese sind meist frei verfügbar und können für eigene Experimente verwendet werden

Transfer Learning

- Statt komplett neue CNN Netze zu trainieren verwendet man meistens *Transfer Learning*
- Bei Transfer Learning wird ein bestehendes Netz mit bereits trainierten Gewichten als Basis genommen (ohne Output Layer)
- Diesem Netz werden dann weitere Schichten hinzugefügt und die neuen Schichten trainiert, wobei das Basisnetz nicht verändert wird
- Hiermit kann ausgenutzt werden das viele Strukturen eventuell schon gelernt wurden
- Gegebenenfalls kann auch das Basisnetz mit trainiert werden