Vorabfragen

- Ein klassisches neuronales Netz ist vollständig verbunden. Welche Probleme könnten auftreten, wenn man damit direkt Bilder (z. B. 256×256 Pixel) verarbeitet?
- Im menschlichen visuellen System reagieren Neuronen nur auf kleine Bildbereiche. Warum könnte dieses Prinzip für die automatische Bilderkennung nützlich sein?
- Stellen Sie sich vor, Sie möchten in Bildern Kanten erkennen. Wie könnte ein Algorithmus erkennen, ob in einem Bild eher horizontale oder vertikale Strukturen vorkommen?

Deep Learning, Machine Learning und Künstliche Intelligenz – SS 26

Gelesen von Prof. Dr. Daniel Gaida

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Lernziele von heute und Fragen zur Überprüfung der Lernziele

Die Studierenden können ein Deep Learning Projekt zur Bildklassifikation selbstständig durchführen, indem sie

- für eine gegebene Anwendung ein geeignetes convolutional neuronales Netz in keras implementieren,
- trainieren, validieren und testen
- und bei schlechten Validierungsergebnissen Lösungsstrategien zur Verbesserung der Ergebnisse anwenden können,

um später eigene Deep Learning Projekte zur Bilderkennung umsetzen zu können.

Überprüfung des Lernziels: S. fortlaufende Übung heute.

Lernraum II: Deep Learning

- Künstliche neuronale Netze
  - Grundlagen
  - Training
  - Implementierung
  - Lösungsstrategien, um gute Ergebnisse zu erzielen
-
  -
-
  -
-
  -
-
  -
  -

Lernraum II: Deep Learning

- Künstliche neuronale Netze
  - Grundlagen
  - Training
  - Implementierung
  - Lösungsstrategien, um gute Ergebnisse zu erzielen
- **Convolutional neuronale Netze**
  - **Bilderkennung**
-
  -
-
  -
-
  -
  -

Lernraum II: Deep Learning

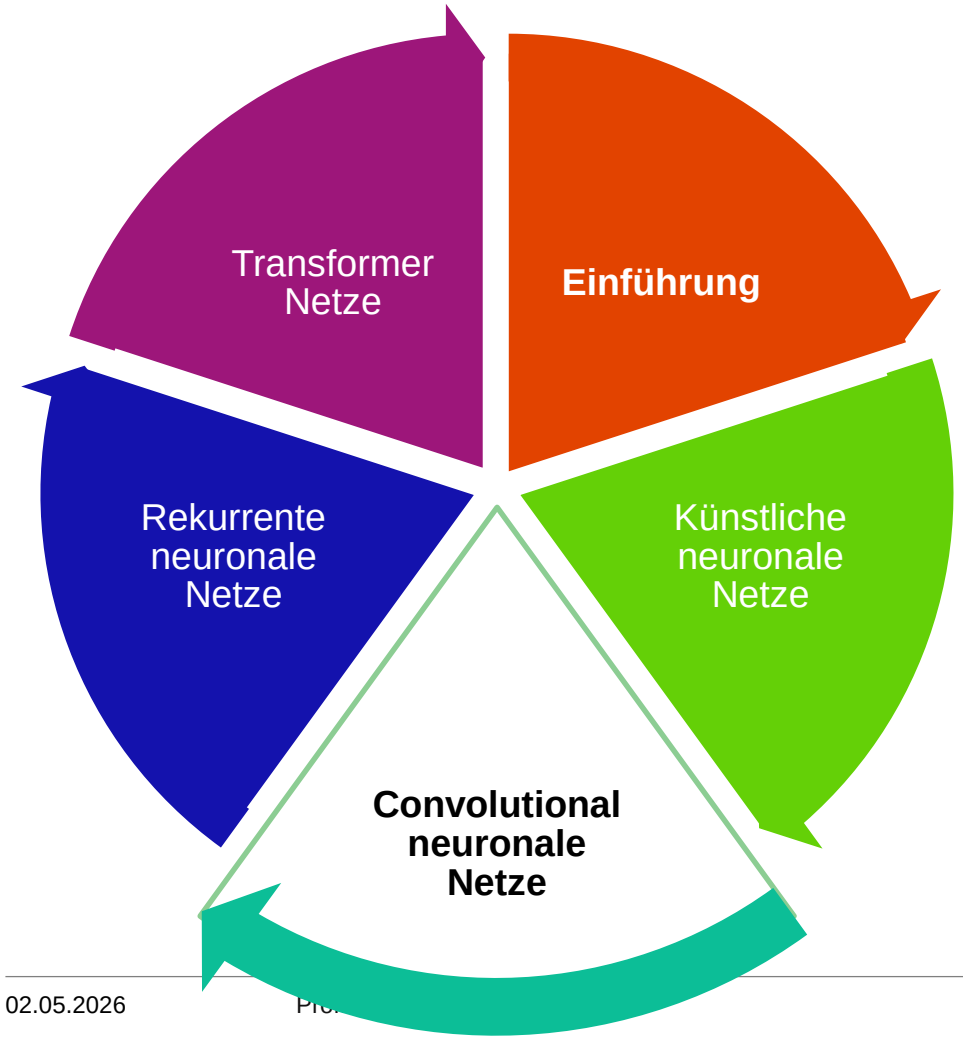
- Künstliche neuronale Netze
  - Grundlagen
  - Training
  - Implementierung
  - Lösungsstrategien, um gute Ergebnisse zu erzielen
- **Convolutional neuronale Netze**
  - **Bilderkennung**
- Rekurrente neuronale Netze
  - Zeitreihen, Text
- Transformer Netze
  - Text, Bilderkennung
-
  -
  -

Lernraum II: Deep Learning

- Künstliche neuronale Netze
  - Grundlagen
  - Training
  - Implementierung
  - Lösungsstrategien, um gute Ergebnisse zu erzielen
- **Convolutional neuronale Netze**
  - **Bilderkennung**
- Rekurrente neuronale Netze
  - Zeitreihen, Text
- Transformer Netze
  - Text, Bilderkennung
- Anwendungen
  - Entwickeln eines Chatbots
  - Objektdetektion und -segmentierung

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Deep Learning
Ausblick

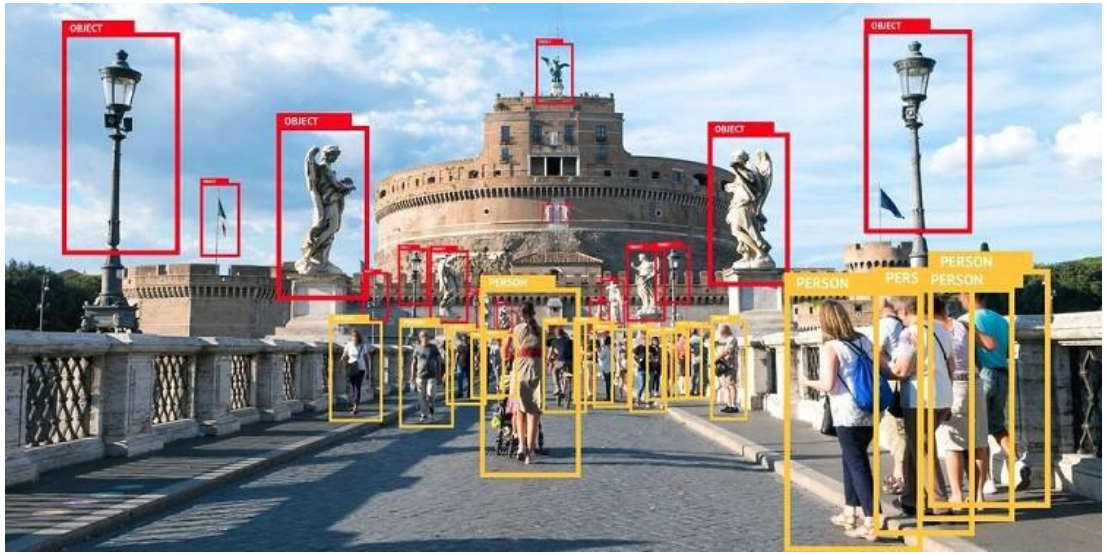


Convolutional neuronale Netze (CNN)

- Der visuelle Cortex
- Aufbau eines CNNs
  - Rezeptive Felder
  - Filter
  - Feature Maps
  - Poolingschicht
- Beispielnetz: LeNet-5
- Keras Code

Convolutional Neural Network - CNN

Einführung

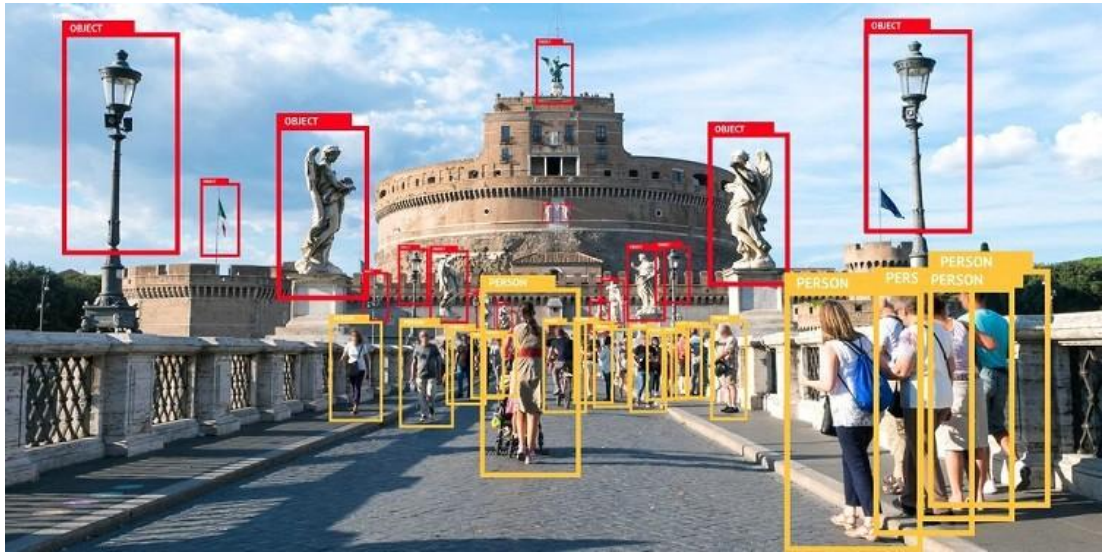


CNNs in der optischen Verarbeitung

- Unterscheidung von Objekten
- Autonomes Fahren
- Augmented Reality
- Gesichtserkennung

Convolutional Neural Network - CNN

Einführung



CNNs in der optischen Verarbeitung

- Unterscheidung von Objekten
- Autonomes Fahren
- Augmented Reality
- Gesichtserkennung

Außerdem

- Stimmenerkennung
- Natural Language Processing
- Synthetische Bild- und Textgenerierung

Convolutional Neural Network - CNN

Leitfragen:

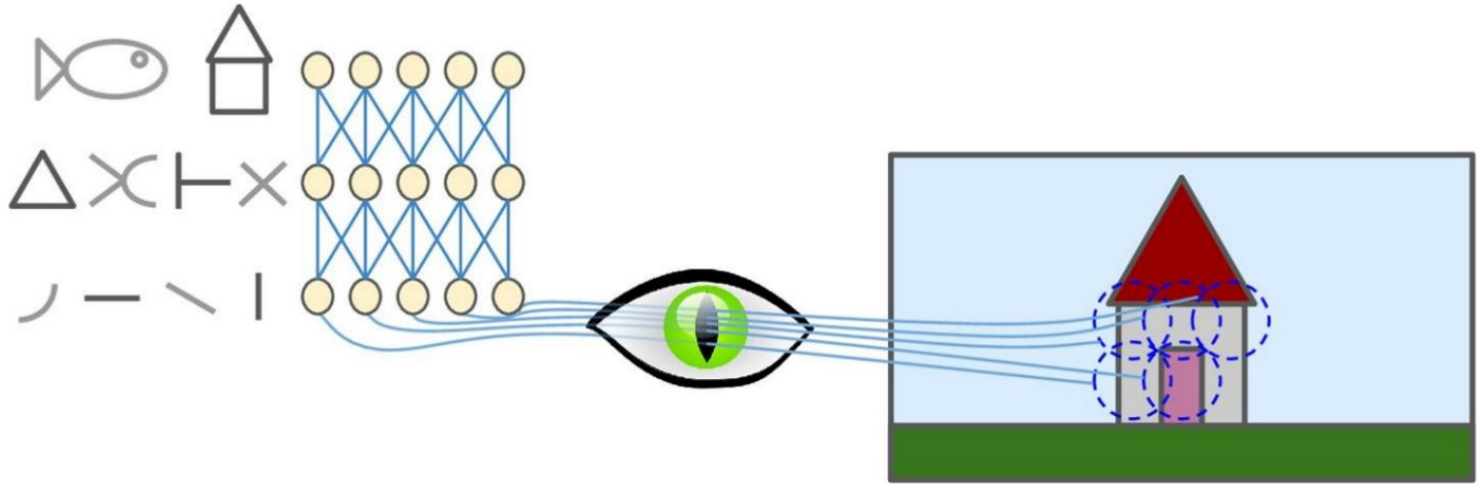
- Wie funktioniert der visuelle Kortex?
- Wie wird das biologische Vorbild in einem Convolutional neural network umgesetzt?

Convolutional Neural Network - CNN

Rezeptive Felder im visuellen Cortex

Visueller Cortex

- Neuronen verfügen über ein kleines rezeptives Feld
- Visuelle Reize werden nur in dieser begrenzten Region verarbeitet
- Können überlappen und decken zusammen das vollständige Bild ab
-
-



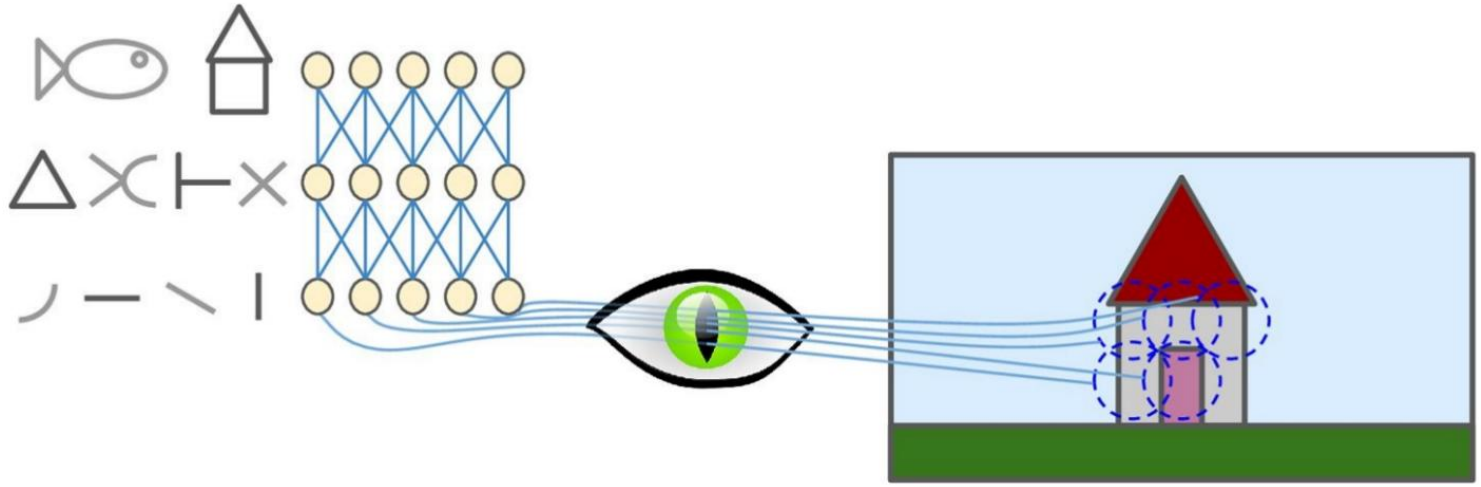
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Rezeptive Felder im visuellen Cortex

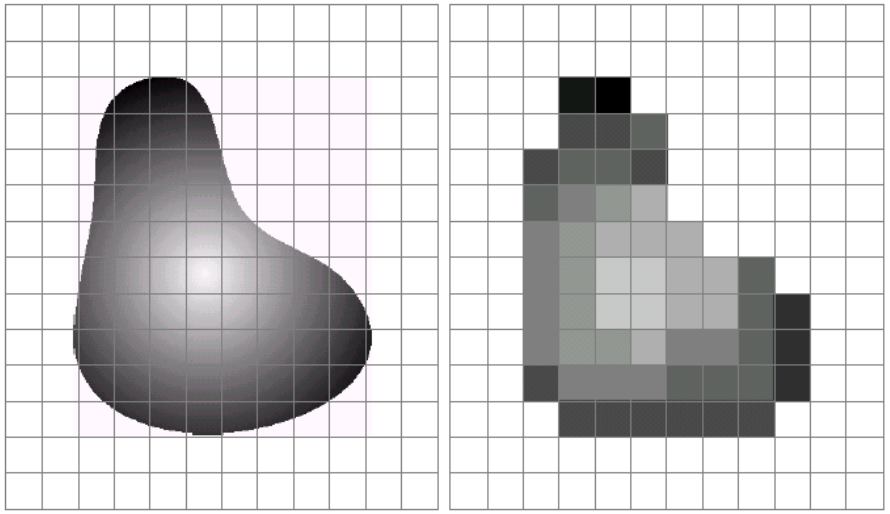
Visueller Cortex

- Neuronen verfügen über ein kleines rezeptives Feld
- Visuelle Reize werden nur in dieser begrenzten Region verarbeitet
- Können überlappen und decken zusammen das vollständige Bild ab
- Verschiedene Neuronen reagieren auf verschiedene Muster / Linien
- Higher-level Neuronen reagieren auf die Ausgänge der lower-level Neuronen



Wie wird ein Grauwertbild im Computer abgebildet?

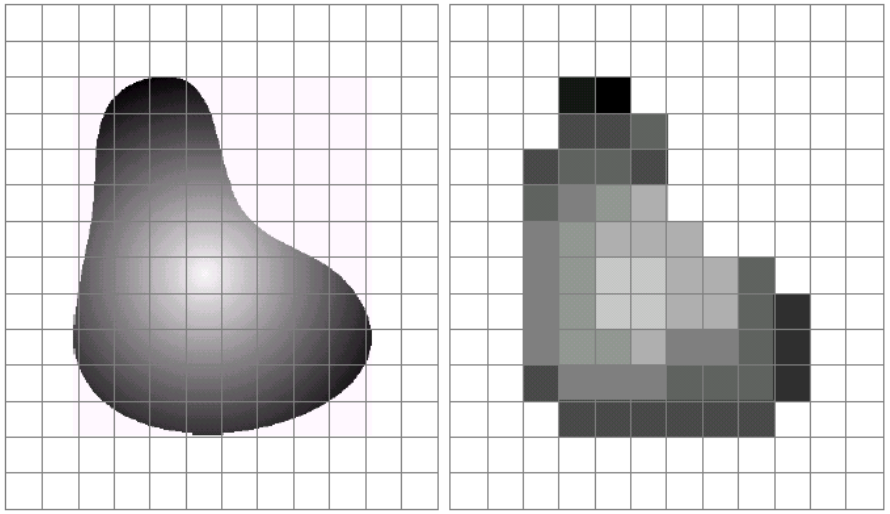
Eine Matrix von Intensitätswerten



Wie wird ein Grauwertbild im Computer abgebildet?

Eine Matrix von Intensitätswerten

- Üblicherweise wird ein Byte pro Intensitätswert verwendet:
  - 0 = schwarz, 255 = weiß



=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

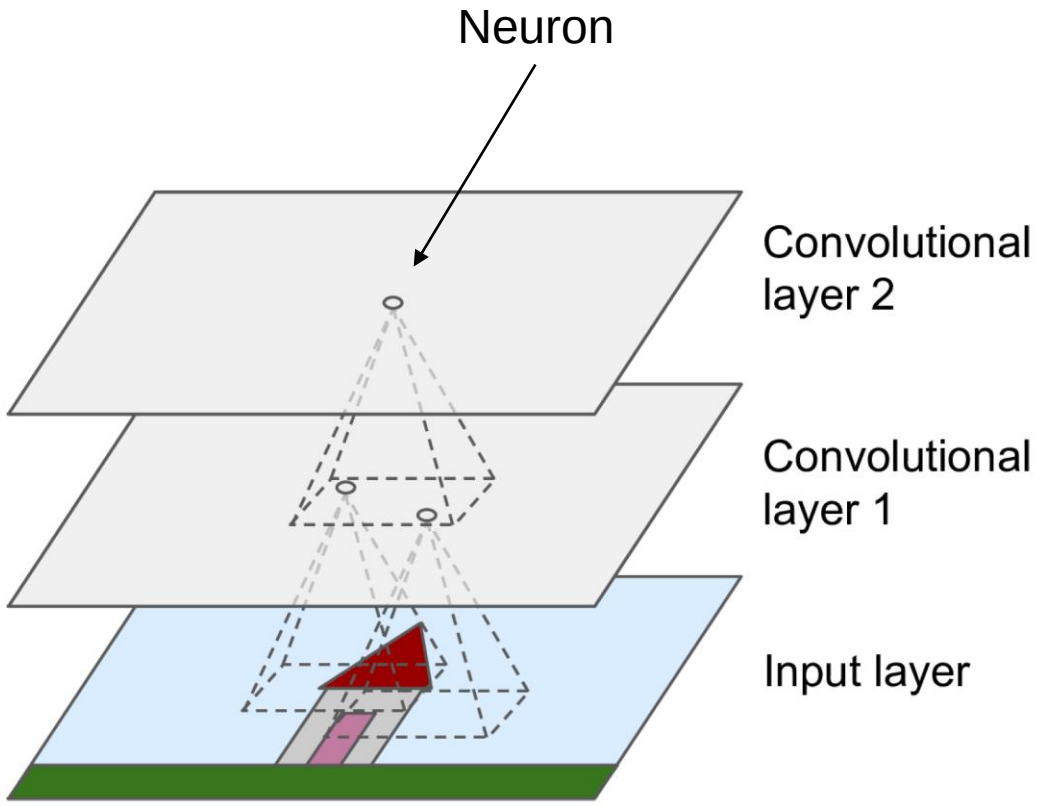
Quelle: Intro to Computer Vision and Computational Photography (CS 194/294-26), UC Berkeley, Fall 2020

Convolutional Neural Network - CNN

Aufbau eines CNN

Die Eingabeschicht

- Neuronen der Eingabeschicht
  - Verbunden mit Bild
  - Es gibt so viele Neuronen wie es Pixel gibt
  - Neuronen liefern nur das Eingangssignal weiter
-
-



Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

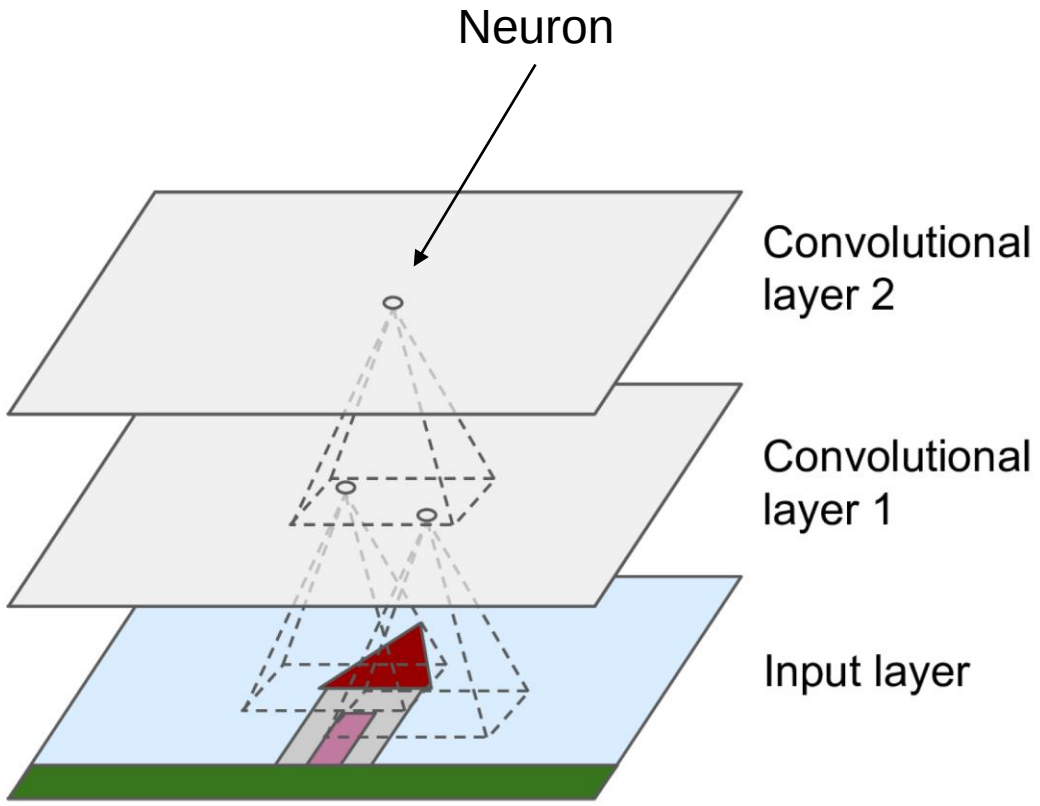
Convolutional Neural Network - CNN

Aufbau eines CNN

Die Eingabeschicht

- Neuronen der Eingabeschicht
  - Verbunden mit Bild
  - Es gibt so viele Neuronen wie es Pixel gibt
  - Neuronen liefern nur das Eingangssignal weiter

- Rezeptives Feld eines Neurons in Schicht 1 wird durch Quadrat in Eingangsschicht visualisiert
-



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

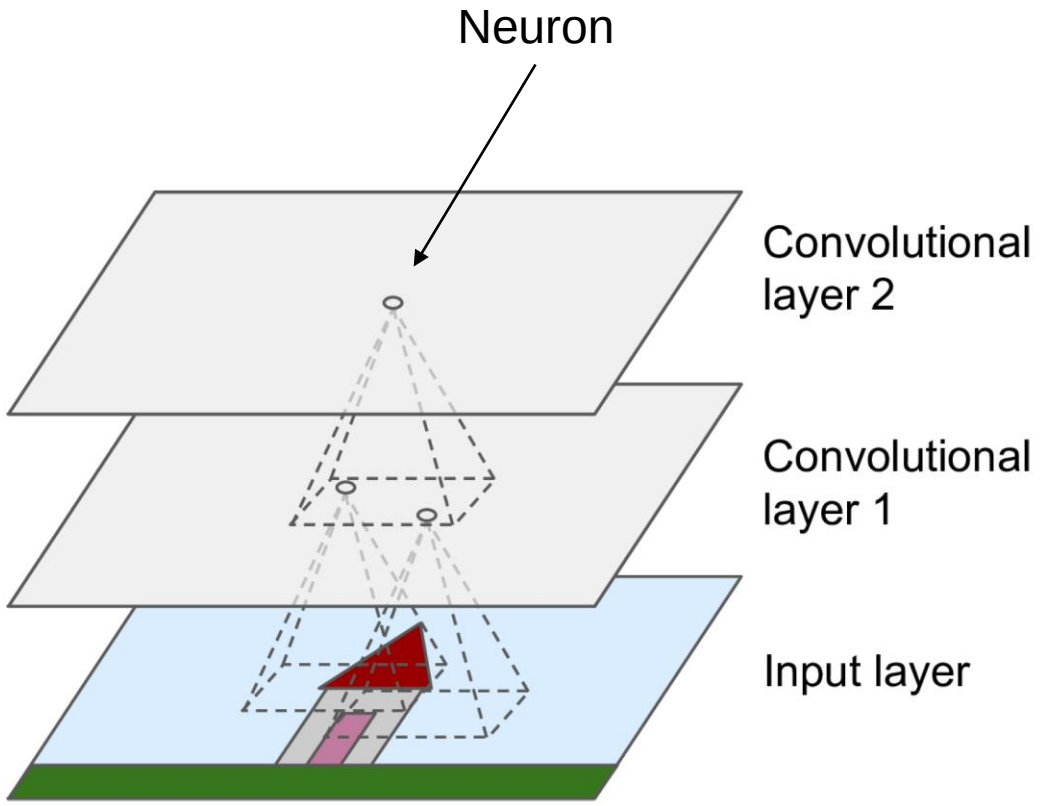
Convolutional Neural Network - CNN

Aufbau eines CNN

Die Eingabeschicht

- Neuronen der Eingabeschicht
  - Verbunden mit Bild
  - Es gibt so viele Neuronen wie es Pixel gibt
  - Neuronen liefern nur das Eingangssignal weiter

- Rezeptives Feld eines Neurons in Schicht 1 wird durch Quadrat in Eingangsschicht visualisiert
- Neu: bisher war ein Neuron immer mit allen Neuronen der vorherigen Schicht verbunden (vollvernetzt). Jetzt nur verbunden mit Neuronen im rezeptiven Feld



Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

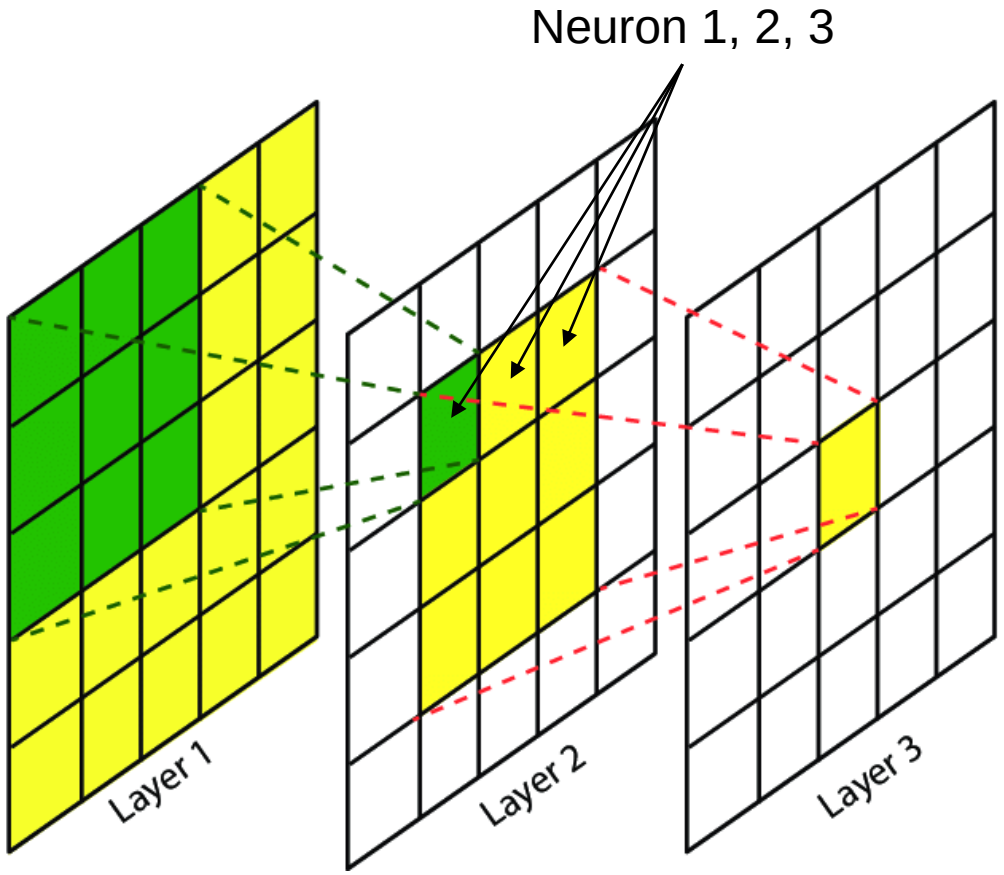
Convolutional Neural Network - CNN

Aufbau eines CNN

Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Die Convolutional Schicht

- Neuronen der ersten CNN Schicht
  - Verbunden mit Eingangsneuronen aus ihrem rezeptiven Feld (Viereck)
  - Erkennung von **Low-level Features** (Linien, Kurven,…)



Eingabeschicht

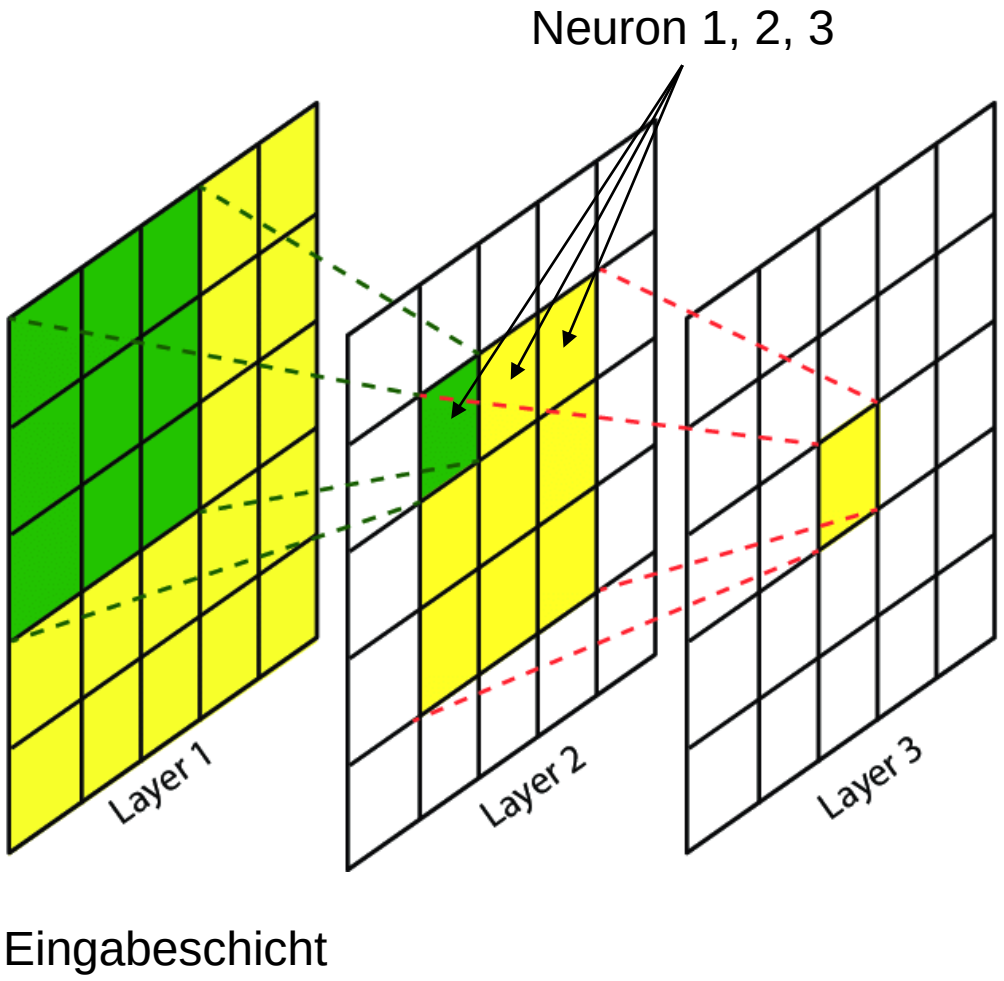
Convolutional Neural Network - CNN

Aufbau eines CNN

Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Die Convolutional Schicht

- Neuronen der ersten CNN Schicht
  - Verbunden mit Eingangsneuronen aus ihrem rezeptiven Feld (Viereck)
  - Erkennung von **Low-level Features** (Linien, Kurven,…)
- Wie bisher, berechnet jedes Neuron eine gewichtete Summe seiner Eingänge
-
-
-



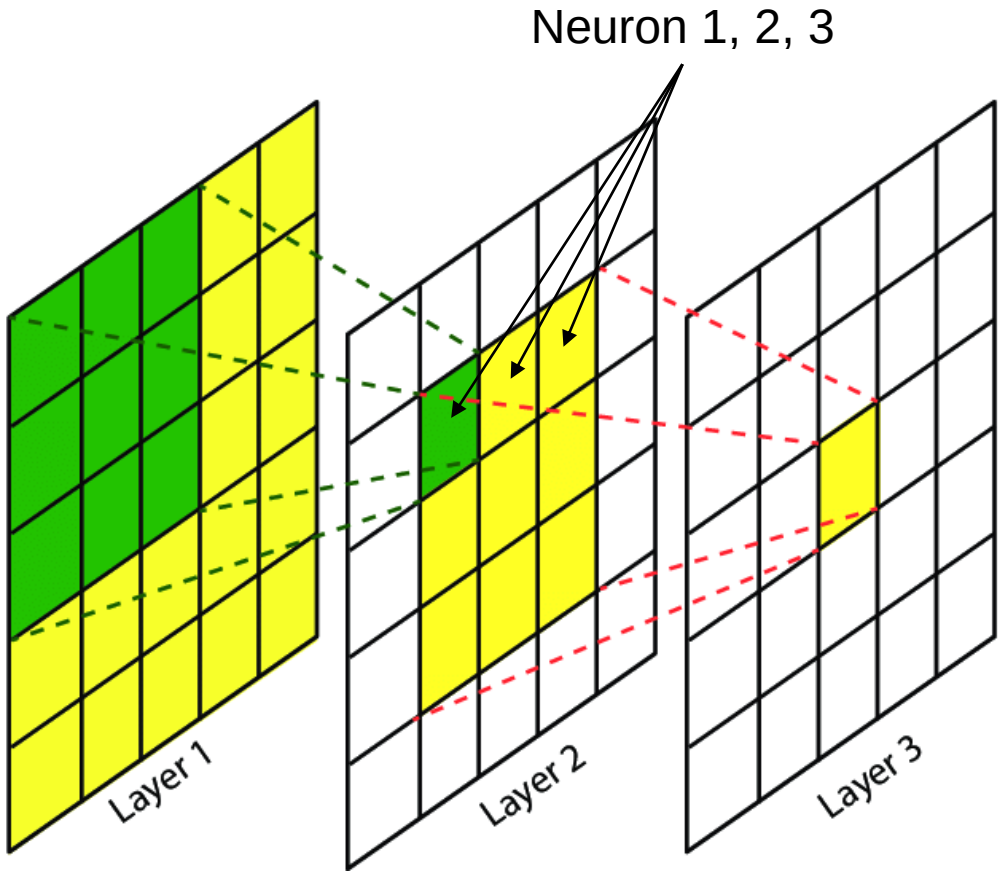
Convolutional Neural Network - CNN

Aufbau eines CNN

Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Die Convolutional Schicht

- Neuronen der ersten CNN Schicht
  - Verbunden mit Eingangsneuronen aus ihrem rezeptiven Feld (Viereck)
  - Erkennung von **Low-level Features** (Linien, Kurven,…)
- Wie bisher, berechnet jedes Neuron eine gewichtete Summe seiner Eingänge
- Neu: Alle Neuronen in einer Schicht teilen sich die Gewichte!
-
-



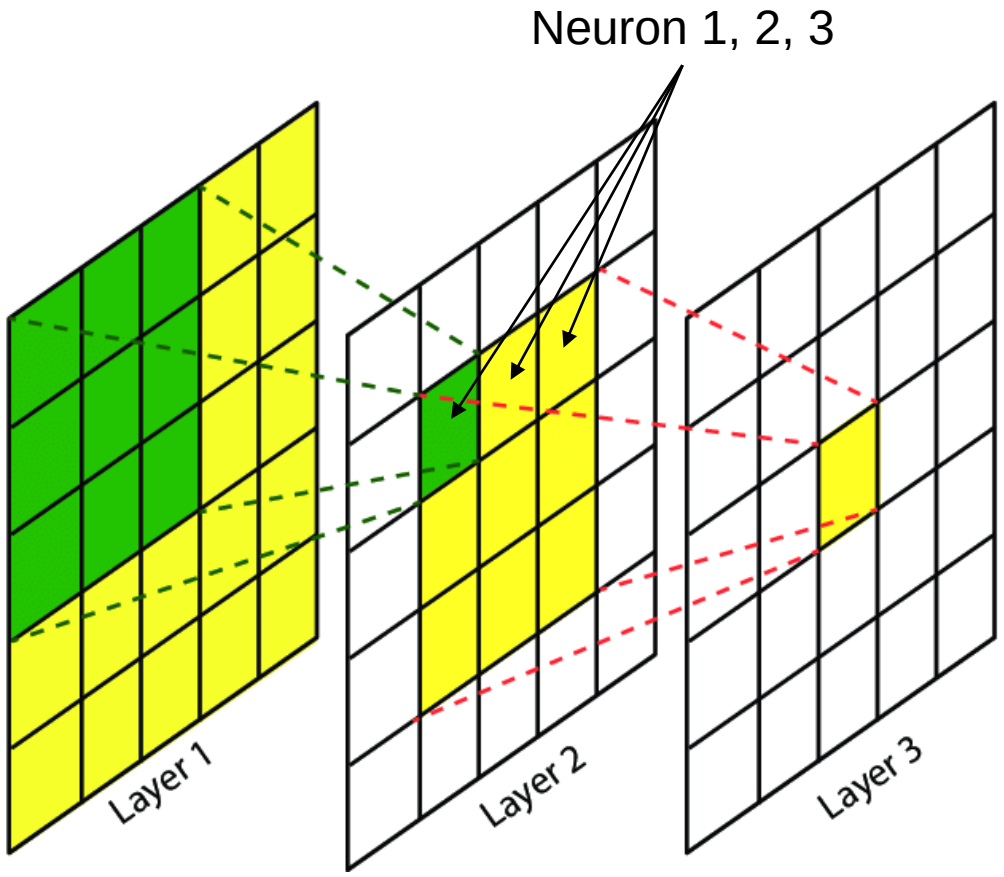
Convolutional Neural Network - CNN

Aufbau eines CNN

Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Die Convolutional Schicht

- Neuronen der ersten CNN Schicht
  - Verbunden mit Eingangsneuronen aus ihrem rezeptiven Feld (Viereck)
  - Erkennung von **Low-level Features** (Linien, Kurven,…)
- Wie bisher, berechnet jedes Neuron eine gewichtete Summe seiner Eingänge
- Neu: Alle Neuronen in einer Schicht teilen sich die Gewichte!
- Gewichte definieren welche Muster ein Neuron erkennen kann (warum, s. später)
-



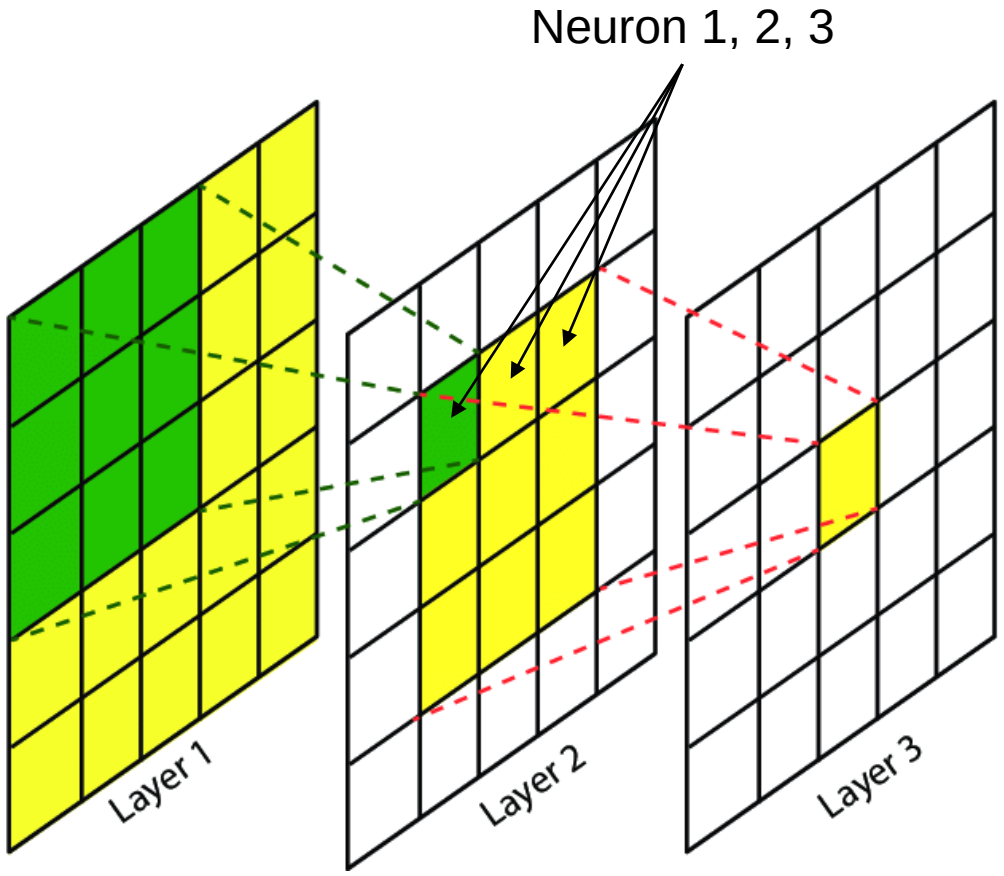
Convolutional Neural Network - CNN

Aufbau eines CNN

Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Die Convolutional Schicht

- Neuronen der ersten CNN Schicht
  - Verbunden mit Eingangsneuronen aus ihrem rezeptiven Feld (Viereck)
  - Erkennung von **Low-level Features** (Linien, Kurven,…)
- Wie bisher, berechnet jedes Neuron eine gewichtete Summe seiner Eingänge
- Neu: Alle Neuronen in einer Schicht teilen sich die Gewichte!
- Gewichte definieren welche Muster ein Neuron erkennen kann (warum, s. später)
- → Eine Convolutional Schicht erkennt nur eine Art von Muster (bspw. horizontale Kanten)

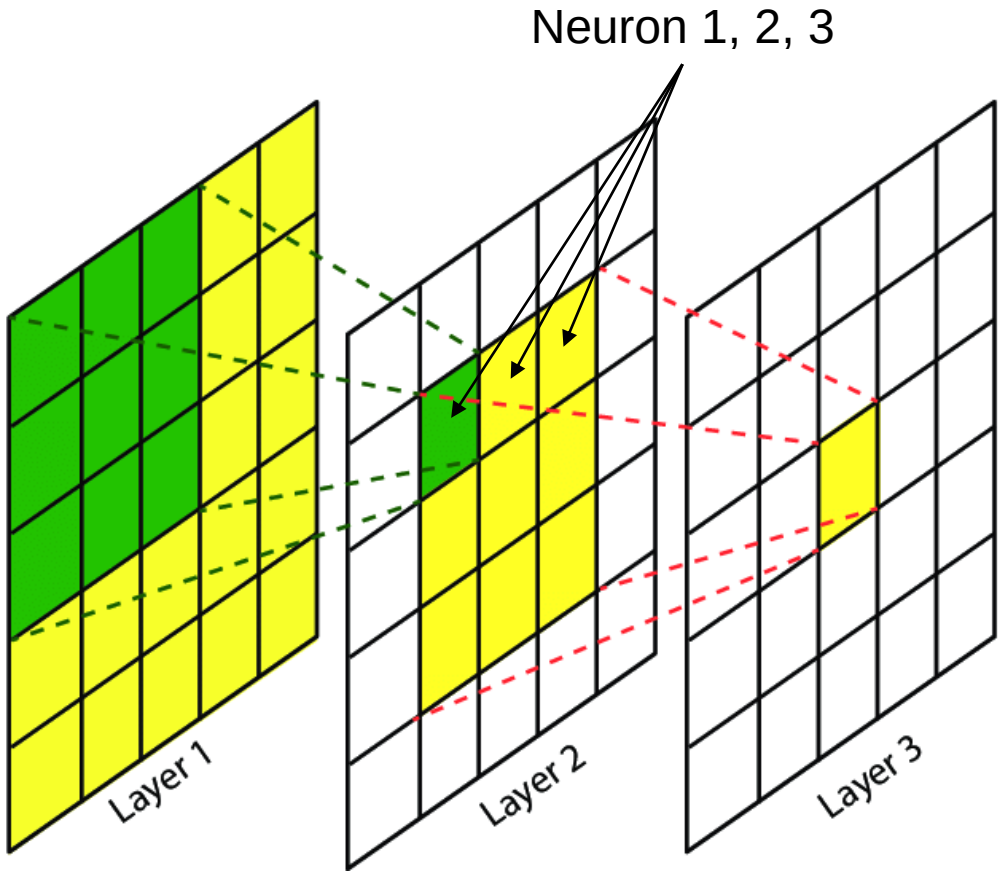


Convolutional Neural Network - CNN

Aufbau eines CNN

Die Convolutional Schicht

- Ausgabewerte der Neuronen einer Convolutional Schicht ergeben ein Bild
- In der Abbildung rechts werden die Schichten als diese Bilder dargestellt



Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

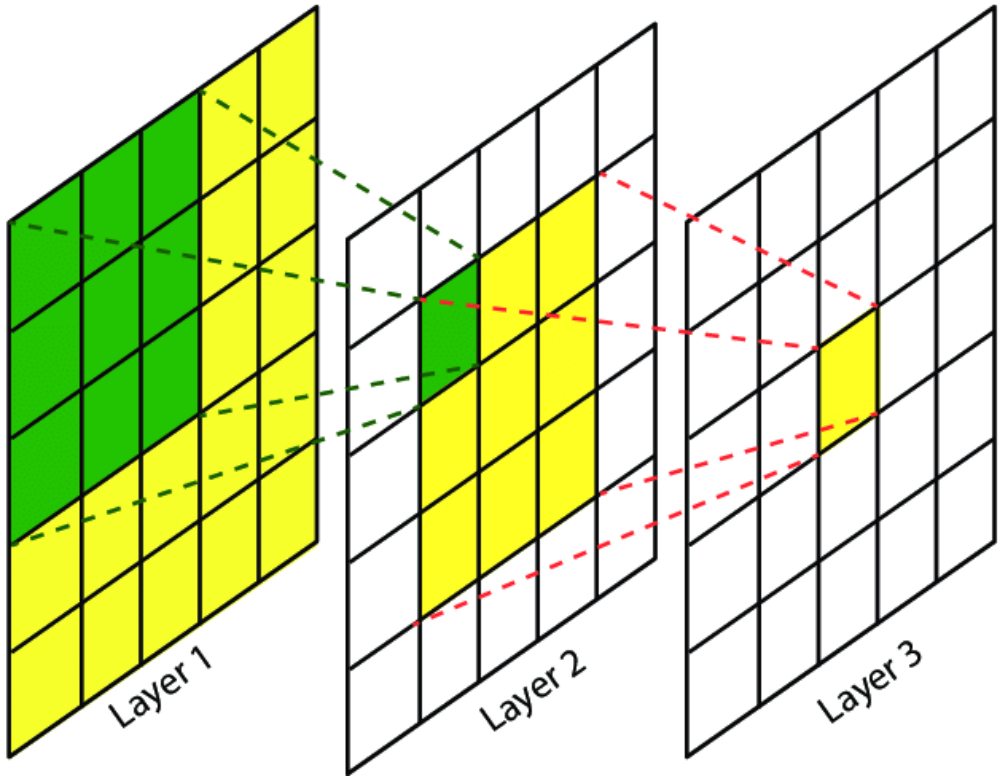
Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Convolutional Neural Network - CNN

Aufbau eines CNN

Die Convolutional Schicht

- Neuronen aus der zweiten CNN Schicht
  - Verbunden mit Neuronen aus der ersten Conv. Schicht aus ihrem rezeptiven Feld
  - Erkennung von Features zusammengesetzt aus der vorherigen Schicht.
  - **High-level Features** (Augen, Ohren, Mund, Pfoten,…)
-
  -



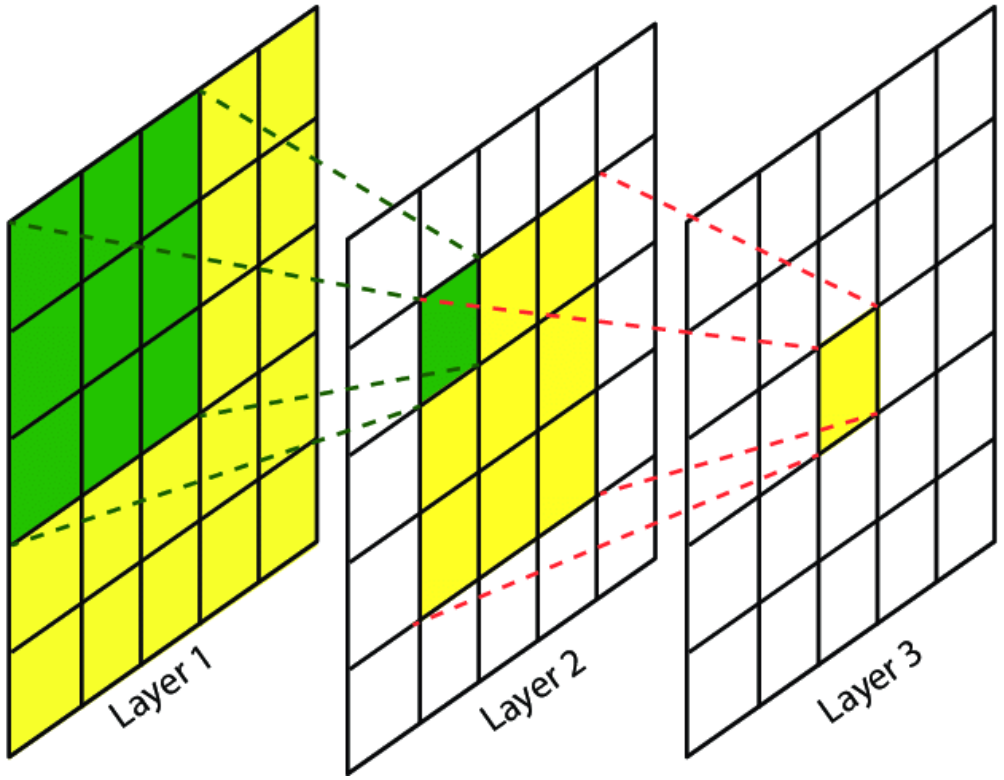
Bildquelle: Lin, H., Shi, Z., & Zou, Z. (2017). Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, *9*(5), 480.

Convolutional Neural Network - CNN

Aufbau eines CNN

Die Convolutional Schicht

- Neuronen aus der zweiten CNN Schicht
  - Verbunden mit Neuronen aus der ersten Conv. Schicht aus ihrem rezeptiven Feld
  - Erkennung von Features zusammengesetzt aus der vorherigen Schicht.
  - **High-level Features** (Augen, Ohren, Mund, Pfoten,…)
- Schema kann beliebig oft wiederholt werden
  - Erkennung von sehr komplexen Mustern



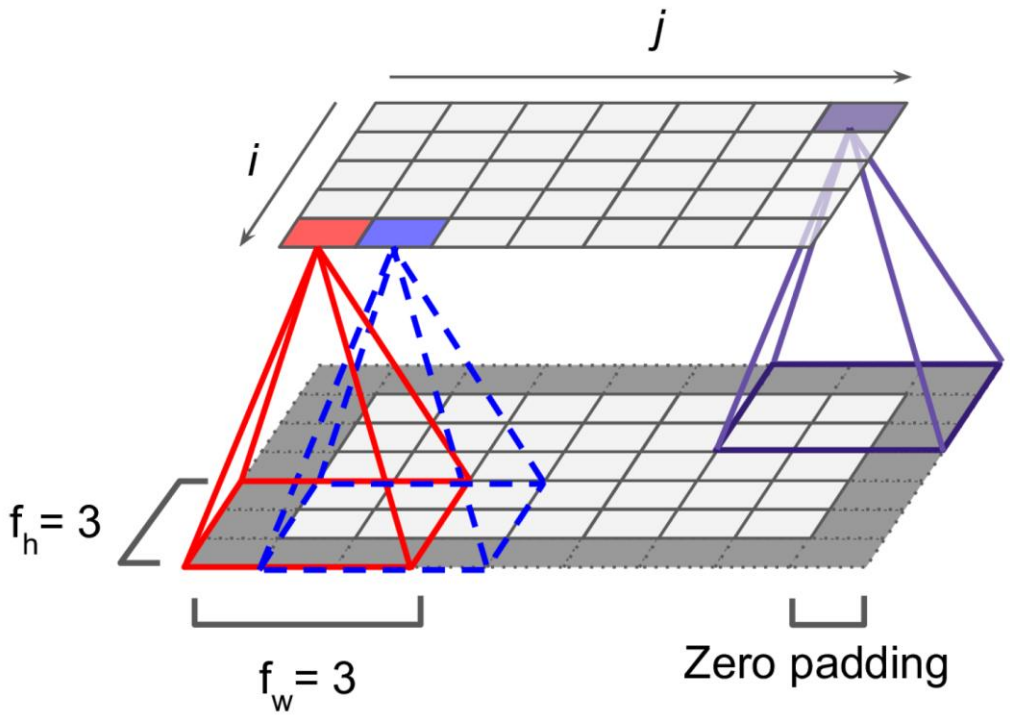
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Aufbau eines CNN

Parameter der Convolutional Schicht

- Breite (f_w) und Höhe (f_h) des rezeptiven Feldes ist 3
- Stride: Verschiebung der rezeptiven Felder gegeneinander
- Zero Padding: einfügen von Nullen um die Eingänge, damit Ausgangsbild so groß ist wie Eingangsbild



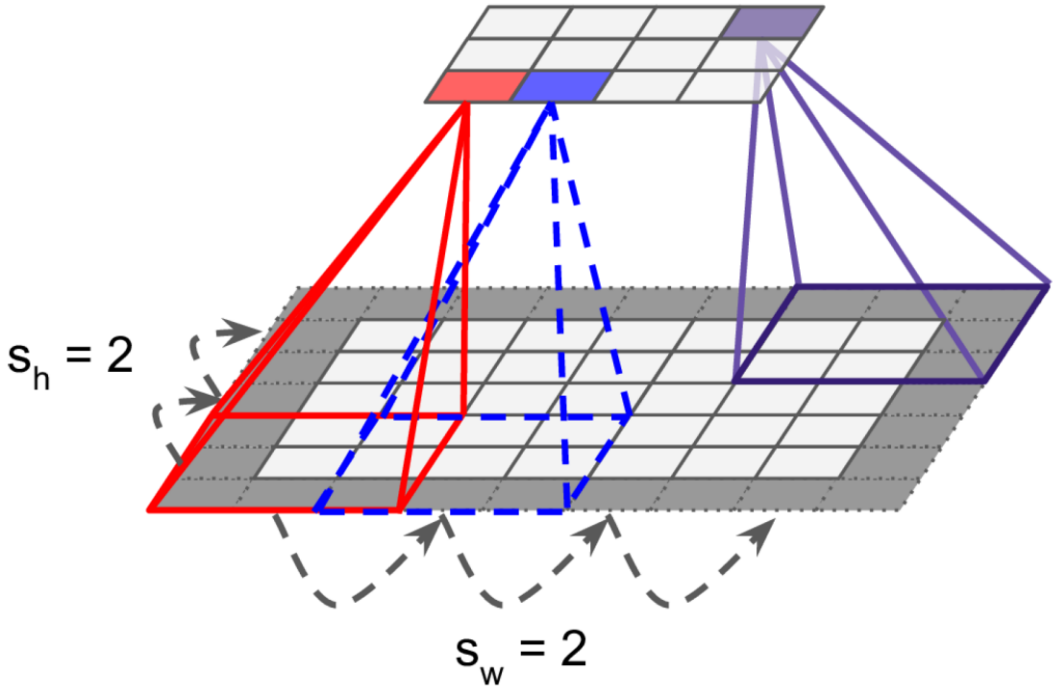
Quelle: Patrick S., Hahn-Schickard
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

Convolutional Neural Network - CNN

Aufbau eines CNN

Parameter der Convolutional Schicht

- Breite (f_w) und Höhe (f_h) des rezeptiven Feldes ist 3
- Stride: Verschiebung der rezeptiven Felder gegeneinander (hier: $s_h = s_w = 2$)
  - Verkleinert das Ausgangsbild



Quelle: Patrick S., Hahn-Schickard
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

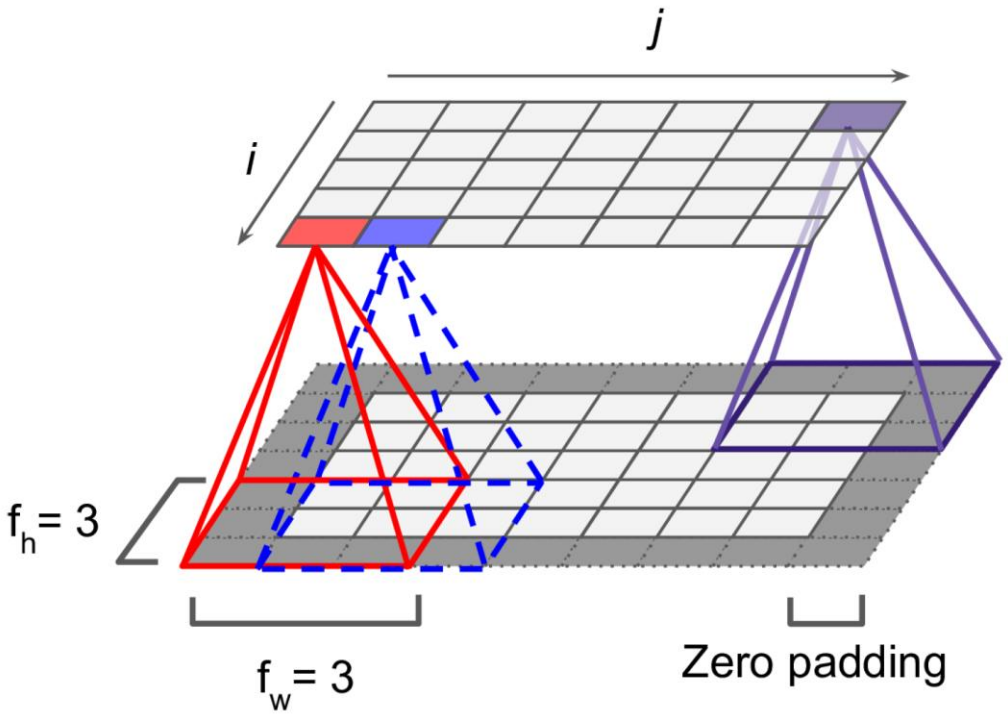
Convolutional Neural Network - CNN

Aufbau eines CNN

Filter (bzw. Kernel):

- Vektor von Gewichten, visualisiert als Matrix in der Dimension des rezeptiven Felds
- Speichert das Muster bzw. Template
-
  -

0	1	0
0	1	0
0	1	0



Quelle: Patrick S., Hahn-Schickard
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

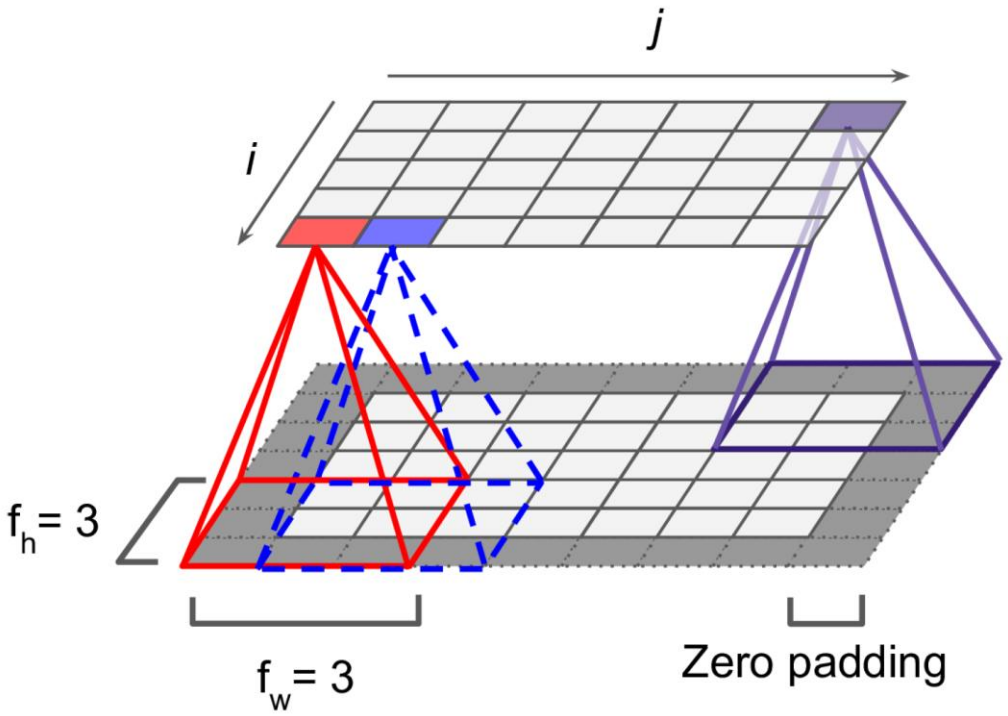
Convolutional Neural Network - CNN

Aufbau eines CNN

Filter (bzw. Kernel):

- Vektor von Gewichten, visualisiert als Matrix in der Dimension des rezeptiven Felds
- Speichert das Muster bzw. Template
- Eingabe wird mit den Filtern gefaltet
  - Bestimmung der Ähnlichkeit zwischen Eingabe und gespeichertem Template

0	1	0
0	1	0
0	1	0

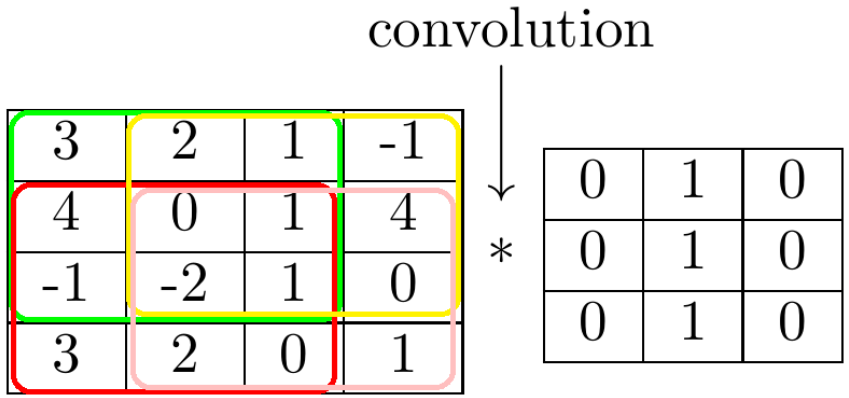


Quelle: Patrick S., Hahn-Schickard
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

Convolutional Neural Network - CNN

Beispiel: Faltung mit einem Filter

Beispiel: Anwendung eines vertikalen Filters



Prof. Dr. Daniel Gaida

Professor für Cyber-Physische Systeme

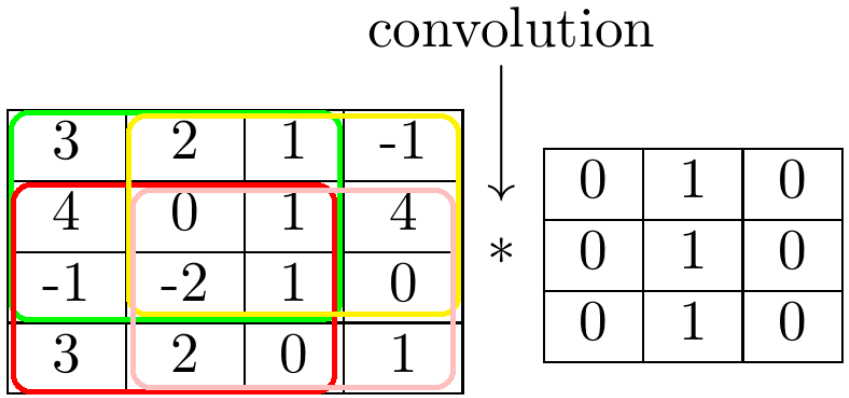
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Beispiel: Faltung mit einem Filter

Beispiel: Anwendung eines vertikalen Filters



Berechnung der **0**:

$$(3 \cdot 0 + 4 \cdot 0 + (-1) \cdot 0) +$$

$$(2 \cdot 1 + 0 \cdot 1 + (-2) \cdot 1) +$$

$$(1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0) = 0$$

Prof. Dr. Daniel Gaida

Professor für Cyber-Physische Systeme

Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Beispiel: Faltung mit einem Filter

Beispiel: Anwendung eines vertikalen Filters

convolution

3	2	1	-1
4	0	1	4
-1	-2	1	0
3	2	0	1

$*$

0	1	0
0	1	0
0	1	0

$=$

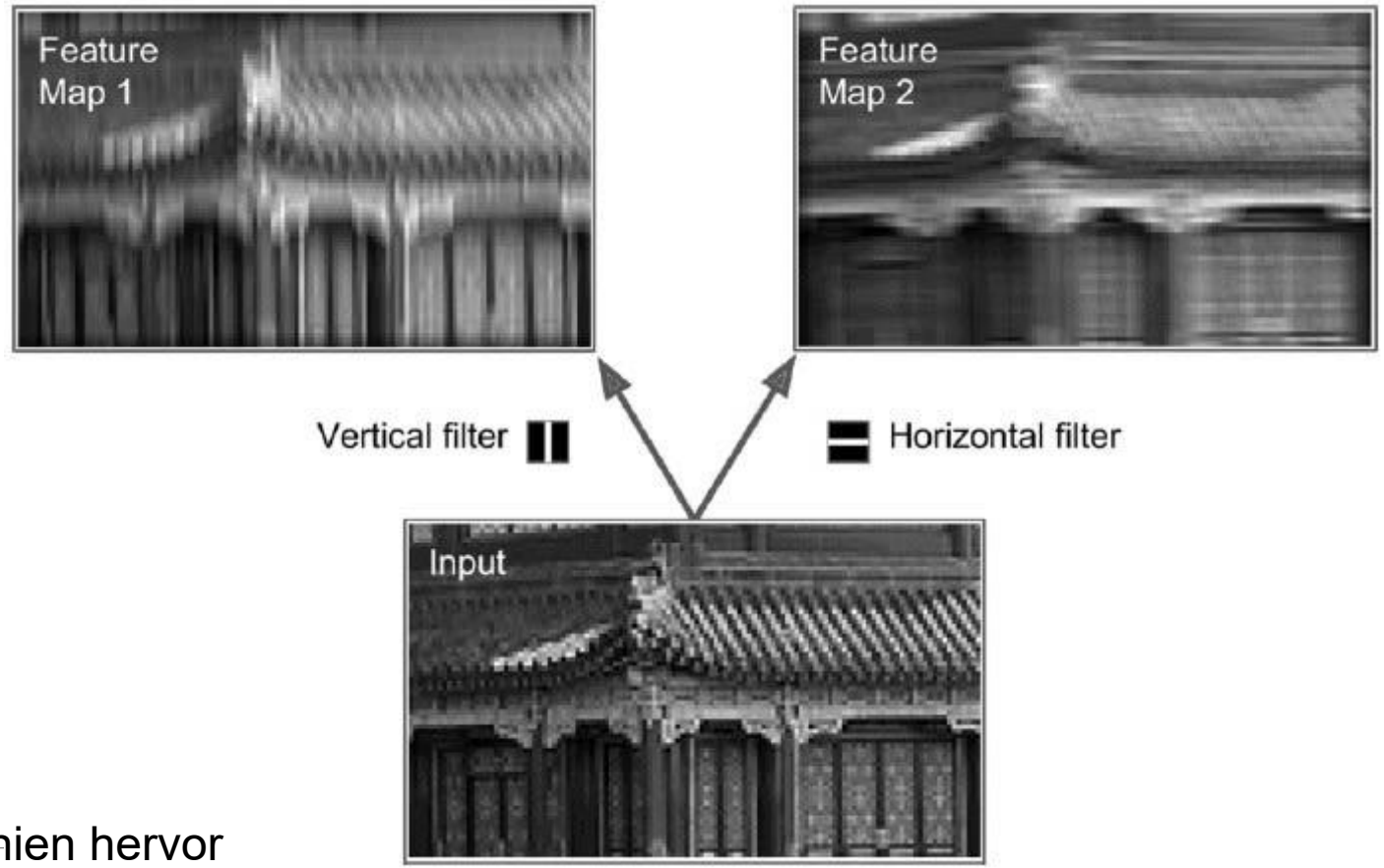
0	3
0	2

Berechnung der **0**:

$$(3 \cdot 0 + 4 \cdot 0 + (-1) \cdot 0) +$$

$$(2 \cdot 1 + 0 \cdot 1 + (-2) \cdot 1) +$$

$$(1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0) = 0$$



→ Vertikaler Filter: hebt Bereiche mit vertikalen Linien hervor

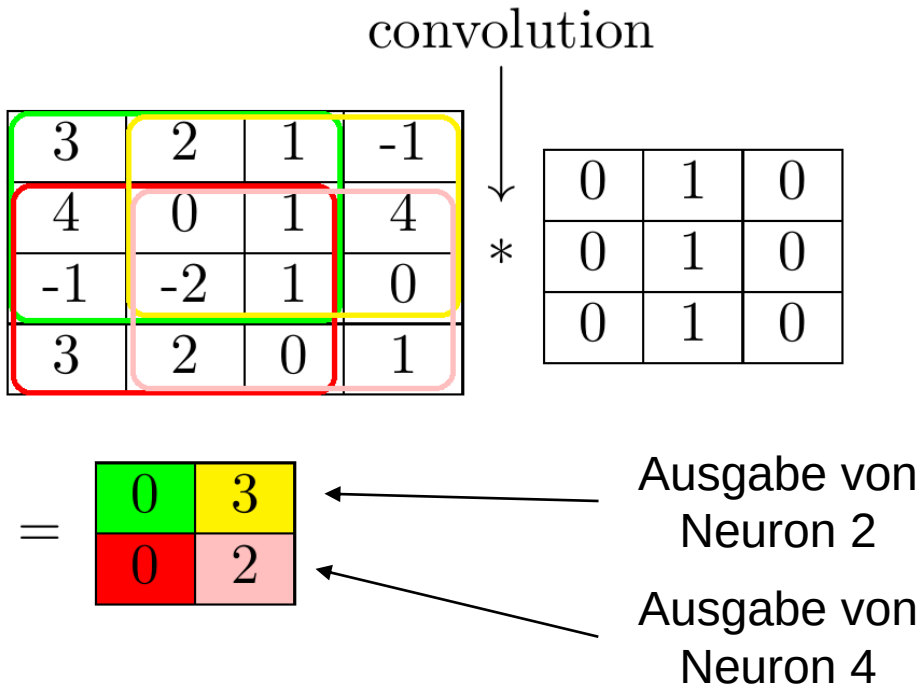
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

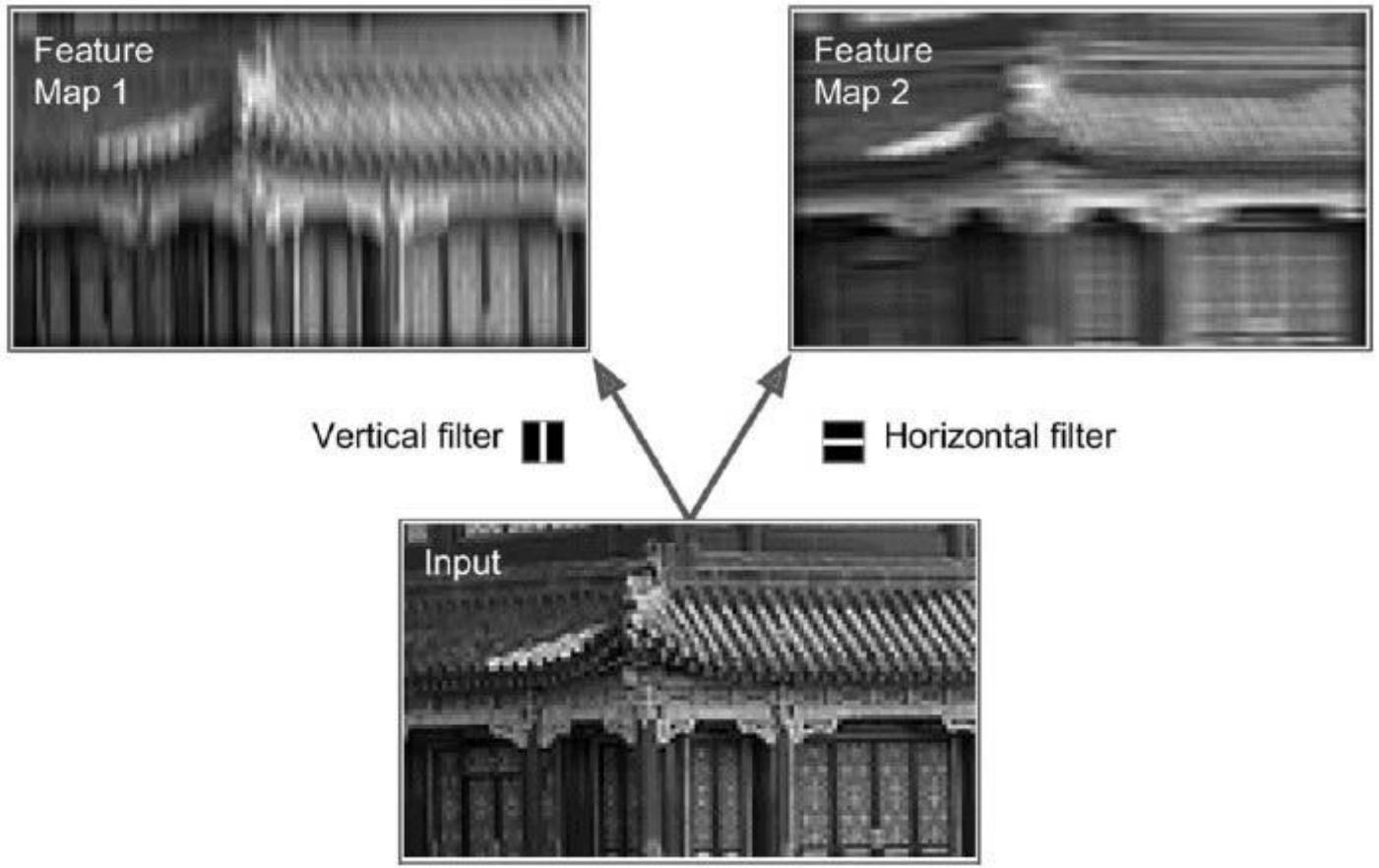
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Beispiel: Faltung mit einem Filter

Beispiel: Anwendung eines vertikalen Filters

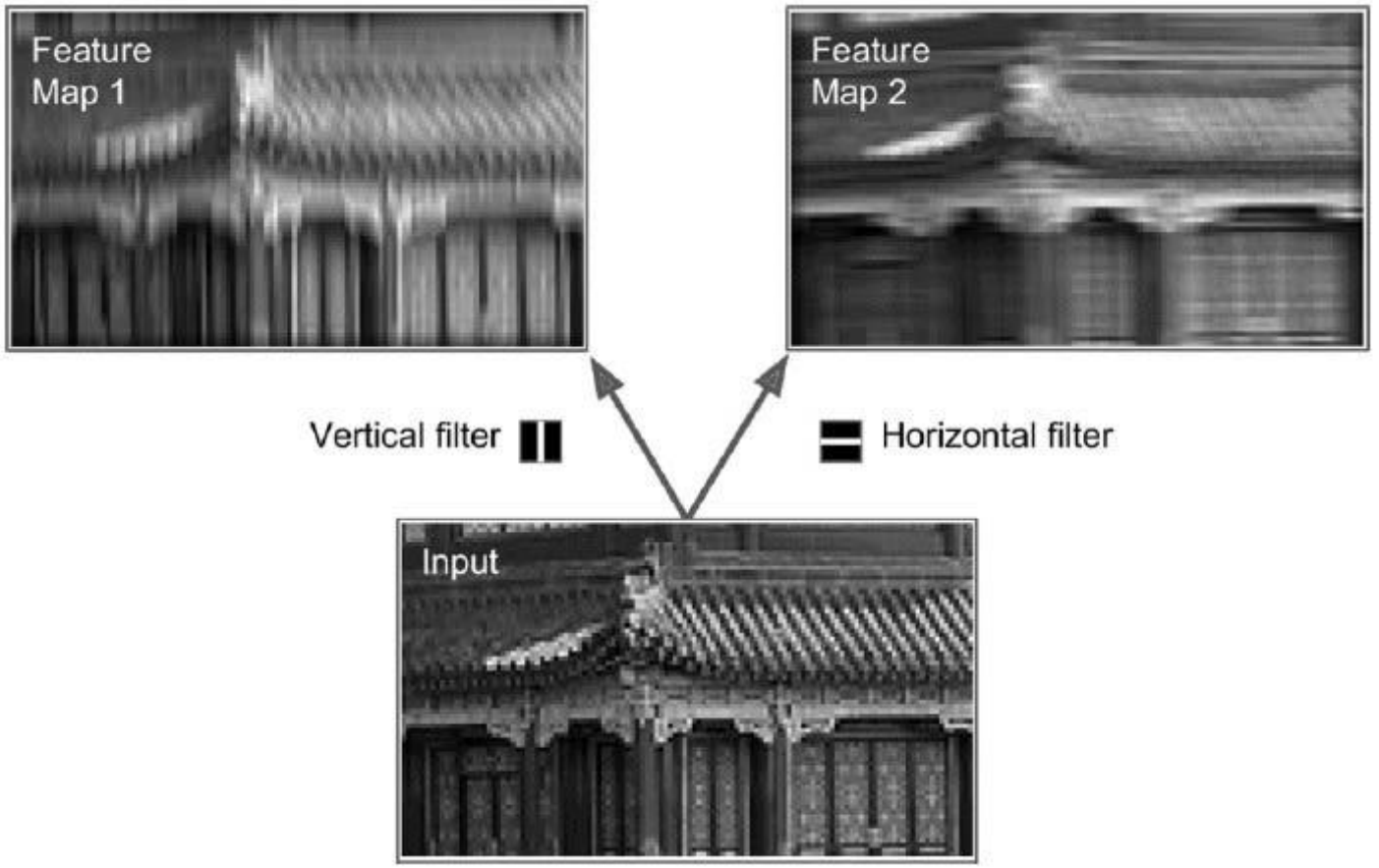




Convolutional Neural Network - CNN

Filter

- Filter heißen so, weil sie nur bestimmte Bildinformationen durchlassen
- Vertikale Filter lassen nur vertikale Kanten sichtbar zurück
- Horizontale Filter heben horizontale Kanten hervor

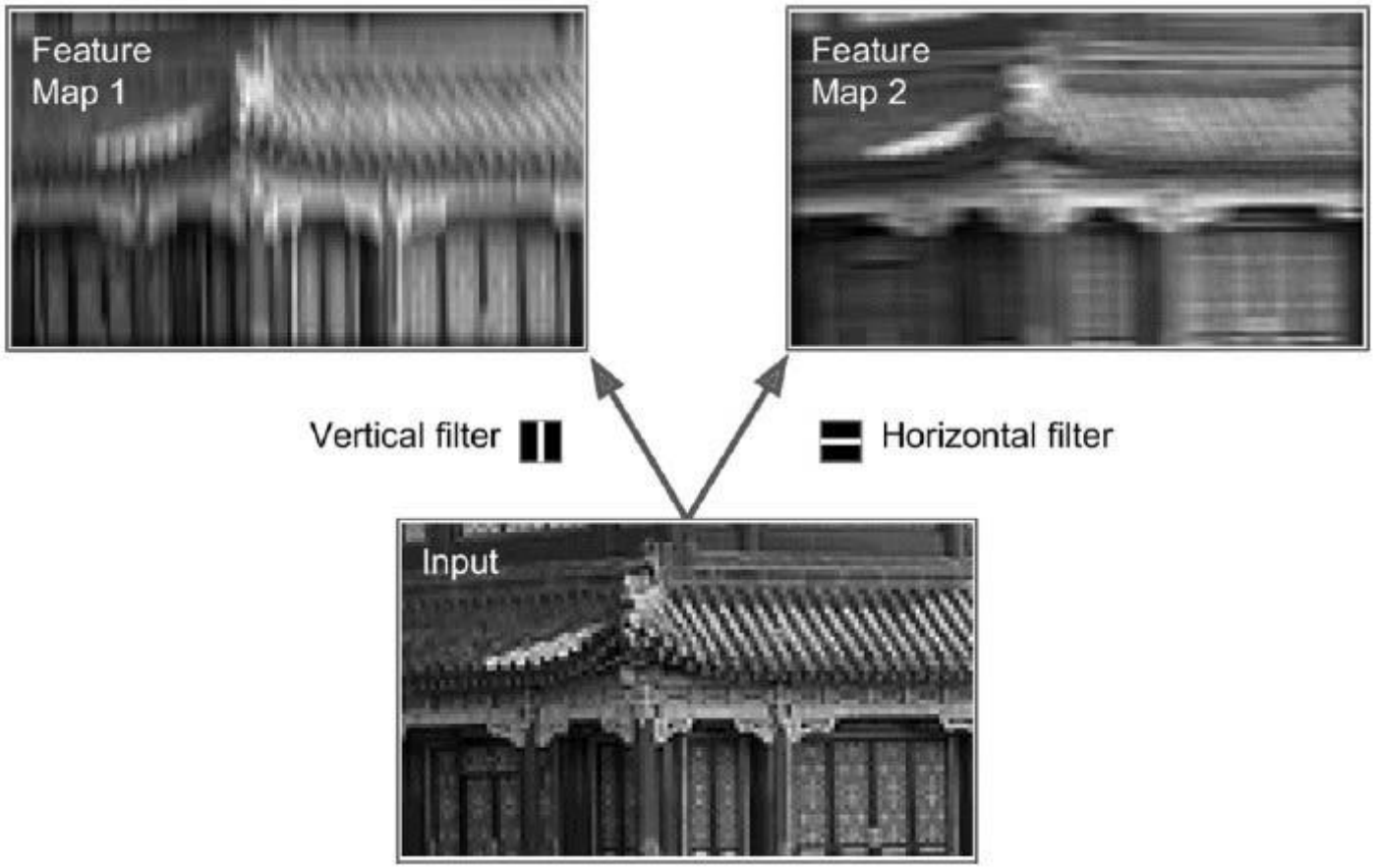


Convolutional Neural Network - CNN

Filter

- Filter heißen so, weil sie nur bestimmte Bildinformationen durchlassen
- Vertikale Filter lassen nur vertikale Kanten sichtbar zurück
- Horizontale Filter heben horizontale Kanten hervor

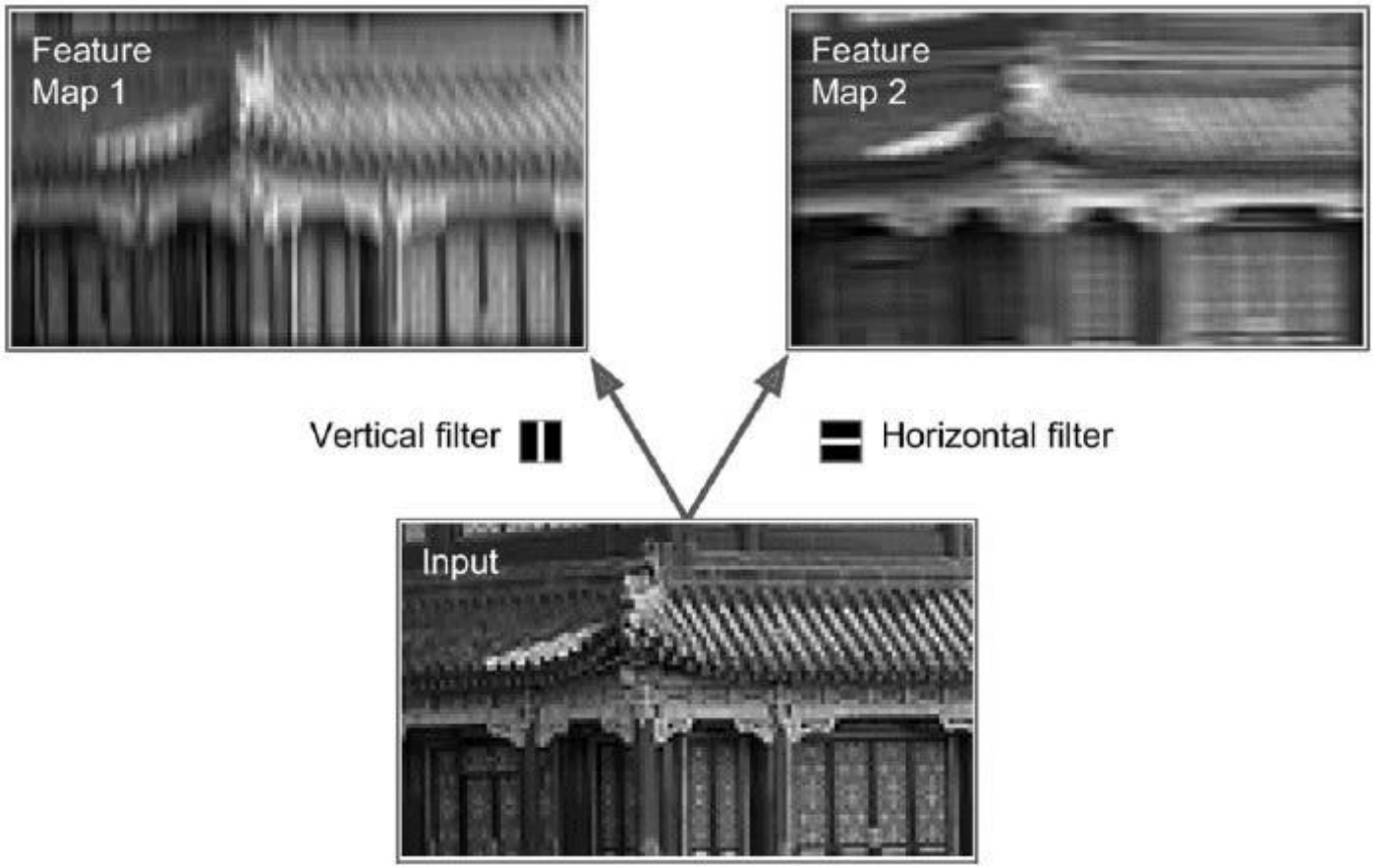
0	0	0
1	1	1
0	0	0



Convolutional Neural Network - CNN

Feature Map

- Feature Map = Ausgangsbild einer Convolutional Schicht, die als Eingangsbild der nächsten Convolutional Schicht genutzt wird
- Da wir unterschiedliche Muster detektieren möchten, benötigen wir mehrere Filter pro Schicht



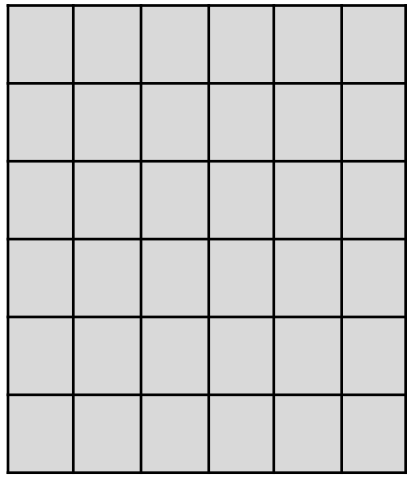
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Feature Map

Hier im Beispiel:
Stride = 1
Kein Padding

Eingangsbild
(Graustufenbild)



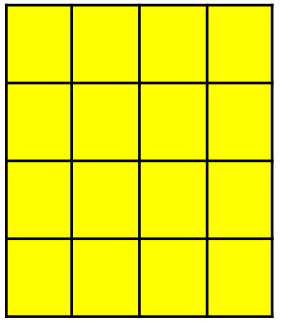
6x6 Pixel

*

2 Filter (vertikale und
horizontale Kanten)

0	1	0
0	1	0
0	1	0

=

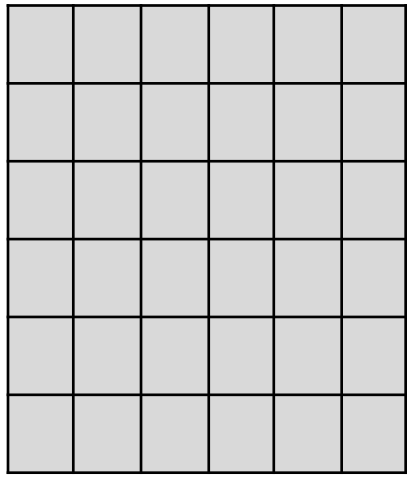


Convolutional Neural Network - CNN

Feature Map

Hier im Beispiel:
Stride = 1
Kein Padding

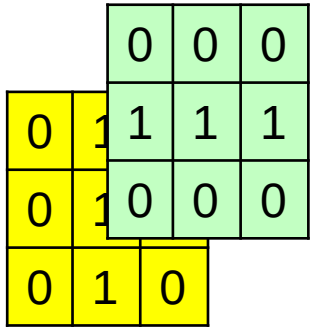
Eingangsbild
(Graustufenbild)



6x6 Pixel

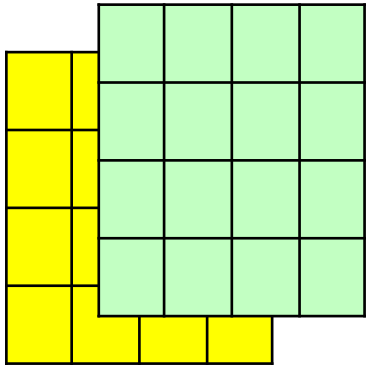
*

2 Filter (vertikale und
horizontale Kanten)



=

2 Feature Maps

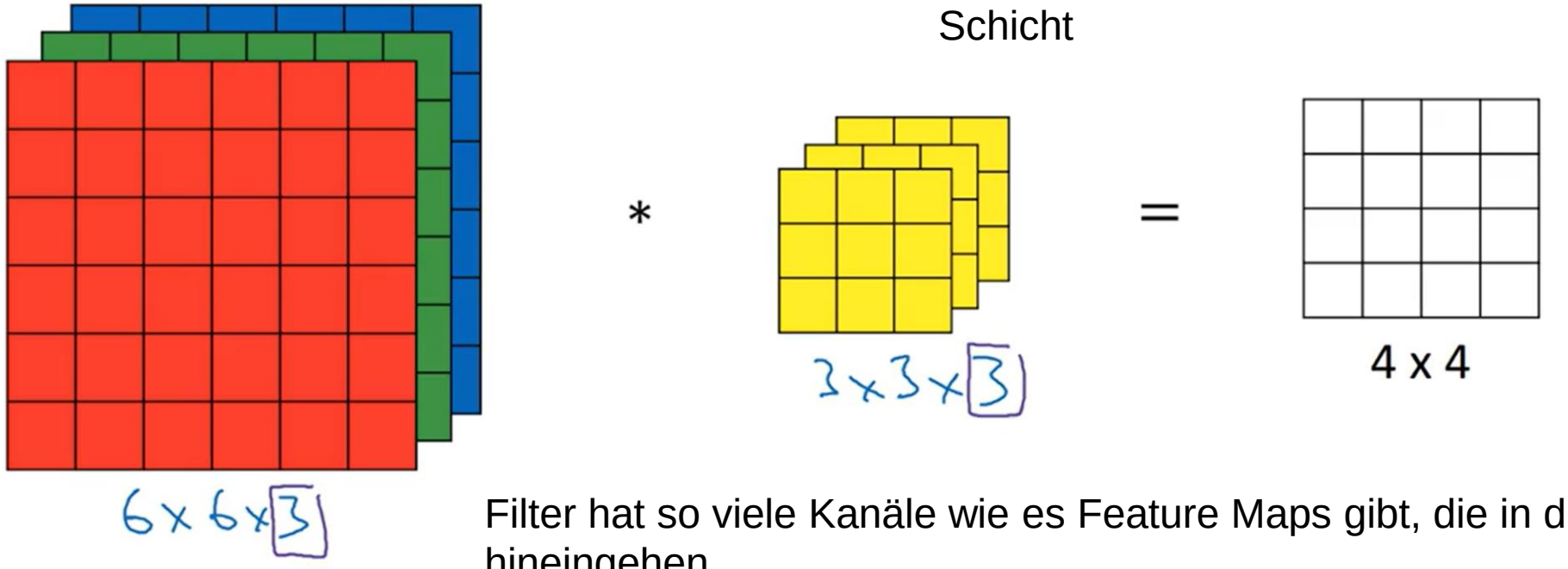


4x4x2 Pixel

Convolutional Neural Network - CNN

Multiple Feature Maps

Feature Map berechnet sich als gewichtete Summe aller Feature Maps der vorherigen Schicht

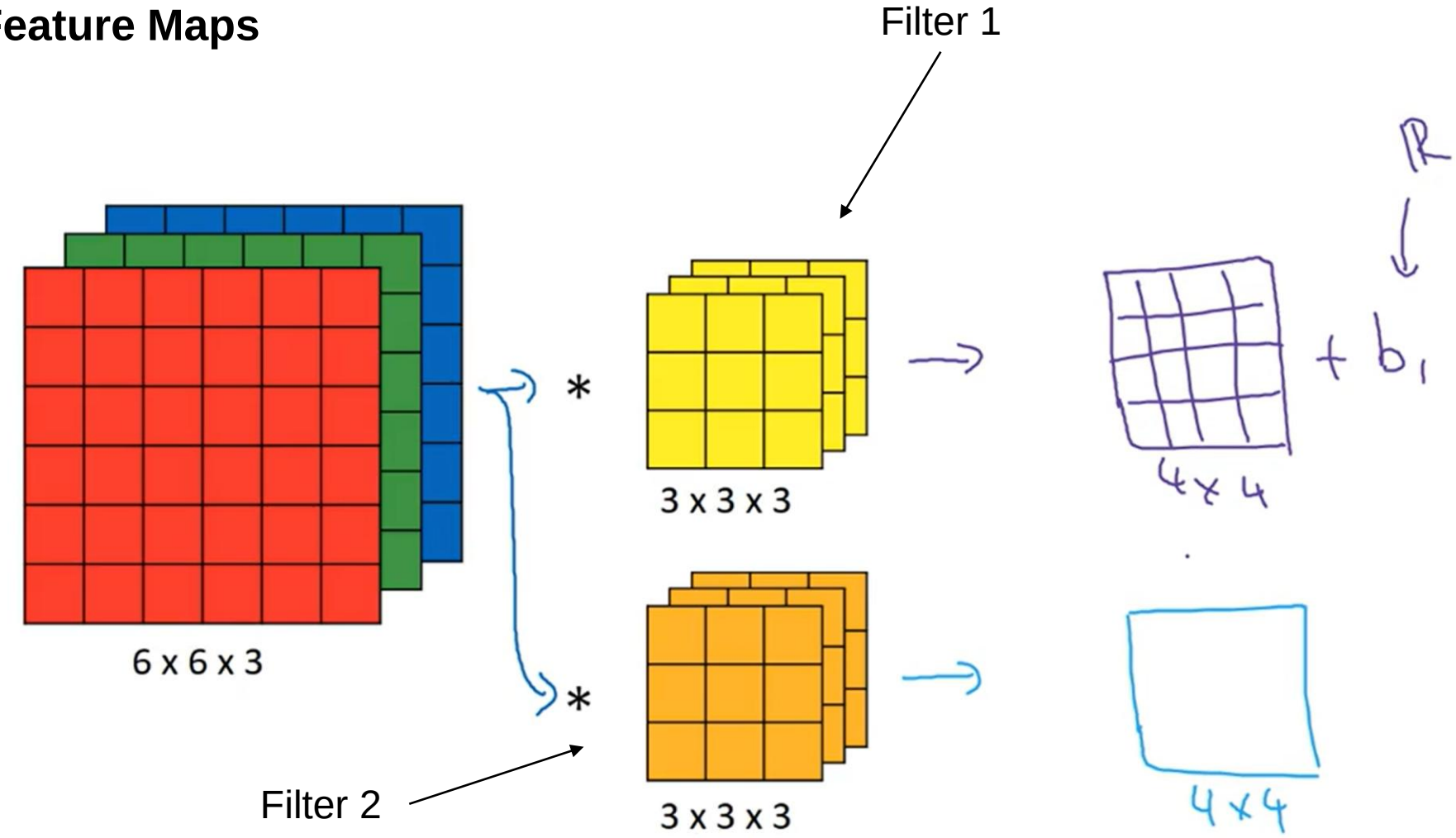


Filter hat so viele Kanäle wie es Feature Maps gibt, die in die Schicht hineingehen.
Hier ist nur 1 Filter dargestellt, dieser hat 9*3=27 Gewichte.
Kanal 1 des Filters könnte nach horizontalen Kanten in Feature Map 1 schauen, Kanal 2 nach vertikalen Kanten in Feature Map 2, …

Quelle: Andrew Ng, Convolutional Neural Networks, Coursera, https://www.coursera.org/learn/convolutional-neural-networks/lecture/ctQZz/convolutions-over-volume

Convolutional Neural Network - CNN

Multiple Feature Maps



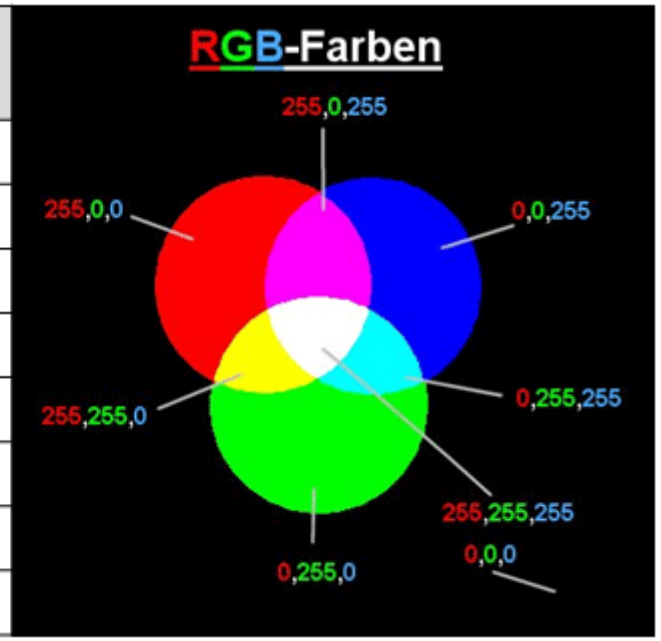
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

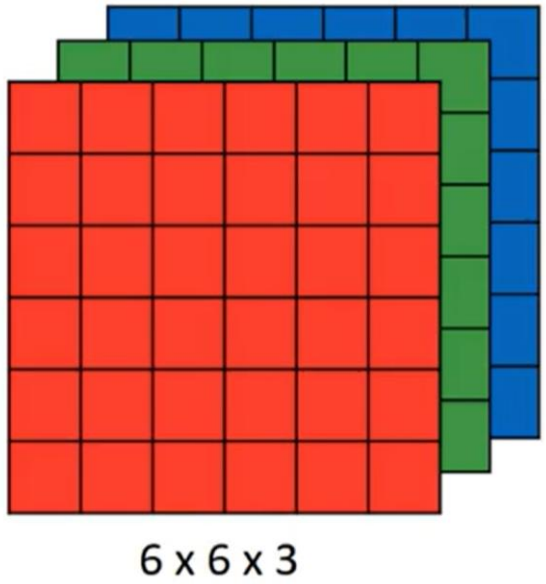
Quelle: Andrew Ng, Convolutional Neural Networks, Coursera, https://www.coursera.org/learn/convolutional-neural-networks/lecture/ctQZz/convolutions-over-volume

Farbbilder

- Was ist der RGB Farbraum? (Additive Farbmischung)

Farbnamen		Hex-Wert	R	G	B
	black	#000000	0	0	0
	blue	#0000FF	0	0	255
	lime	#00FF00	0	255	0
	cyan	#00FFFF	0	255	255
	red	#FF0000	255	0	0
	magenta	#FF00FF	255	0	255
	yellow	#FFFF00	255	255	0
	white	#FFFFFF	255	255	255





- D.h., Farbbilder werden durch die drei Kanäle rot, grün, blau dargestellt und werden als RGB-Bilder bezeichnet

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

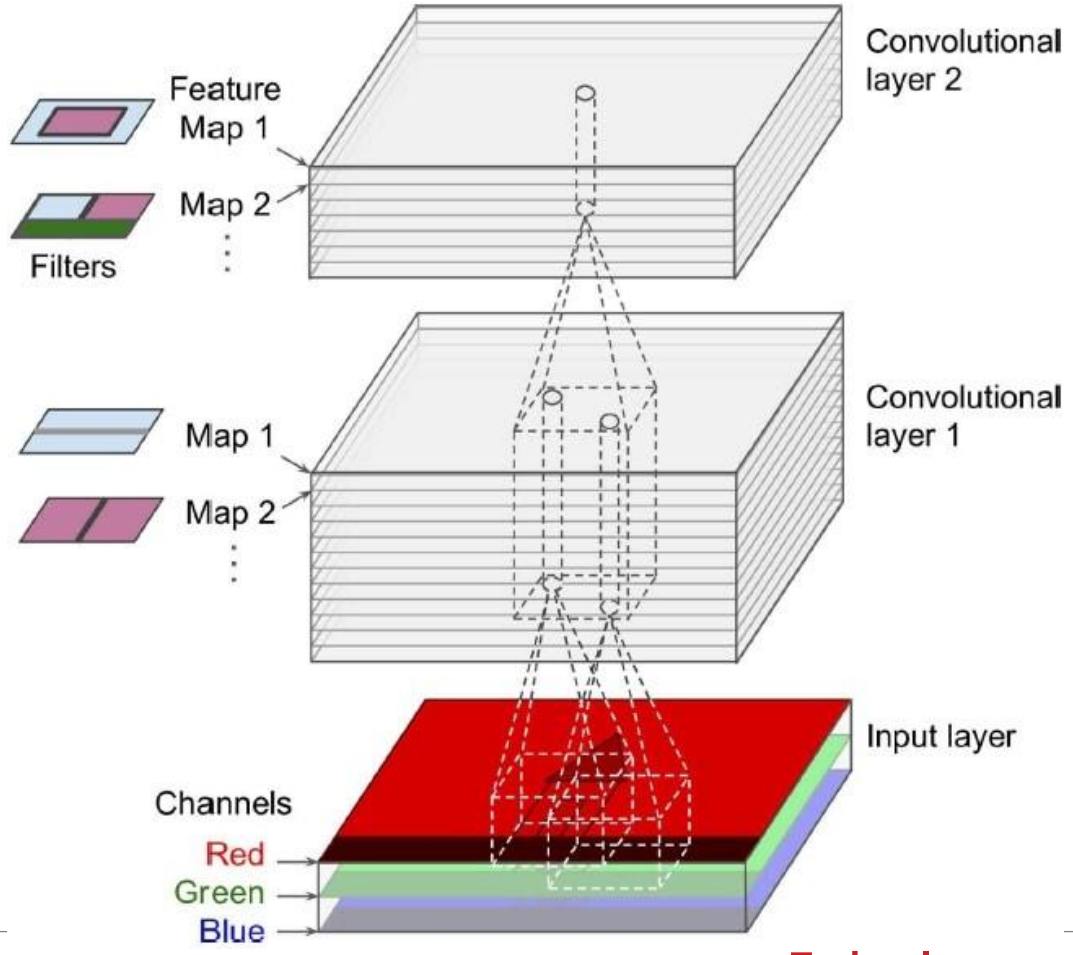
Bild links: https://www.html-seminar.de/css-einsetzen.htm
Bild rechts: Andrew Ng, Convolutional Neural Networks, Coursera, https://www.coursera.org/learn/convolutional-neural-networks/lecture/ctQZz/convolutions-over-volume

Convolutional Neural Network - CNN

Multiple Feature Maps

Eingabe

- Farbbild (3 Kanäle: RGB), Graustufenbild (1 Kanal)



Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

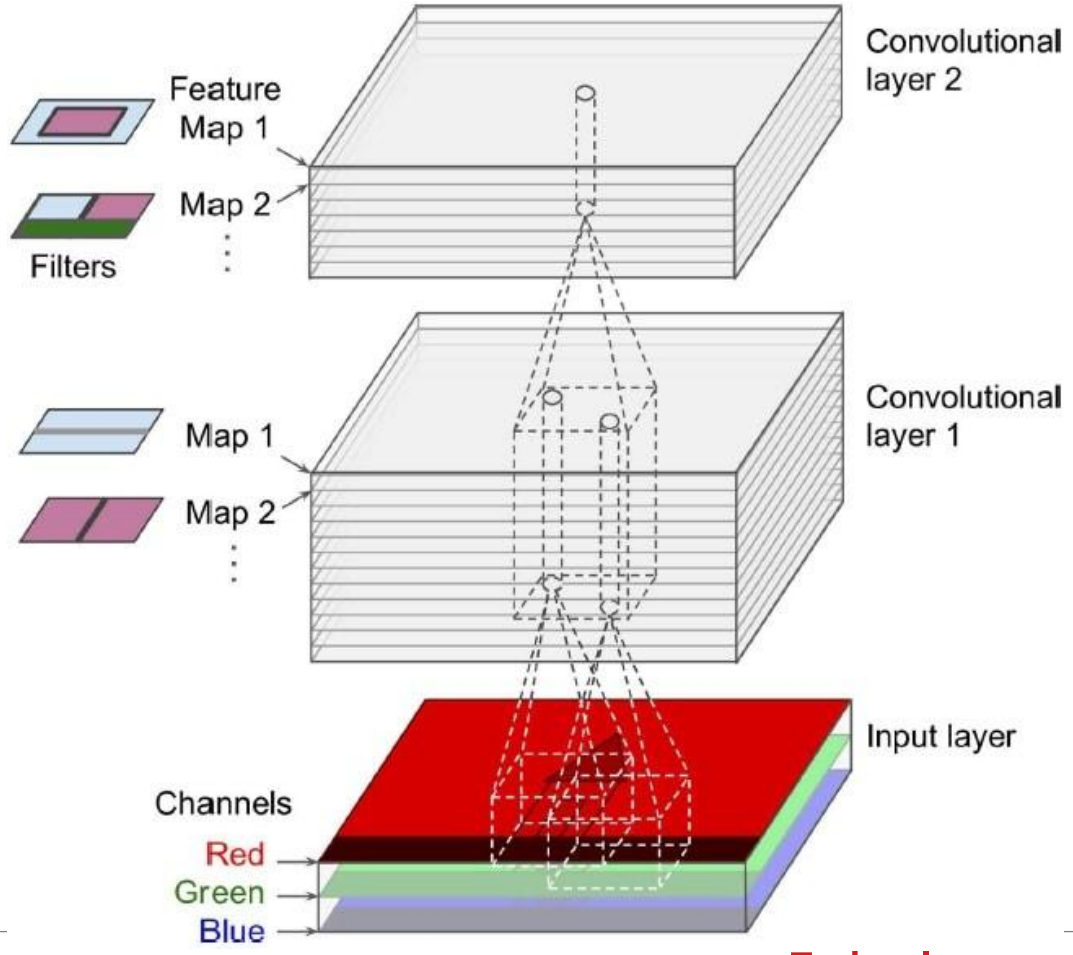
Multiple Feature Maps

Eingabe

- Farbbild (3 Kanäle: RGB), Graustufenbild (1 Kanal)

Neuronen

- einer Feature Map besitzen gleiche Parameter
- verschiedener Feature Maps besitzen unterschiedliche Parameter
- gleiches rezeptives Feld, durchdringt alle Feature Maps der vorherigen Schicht



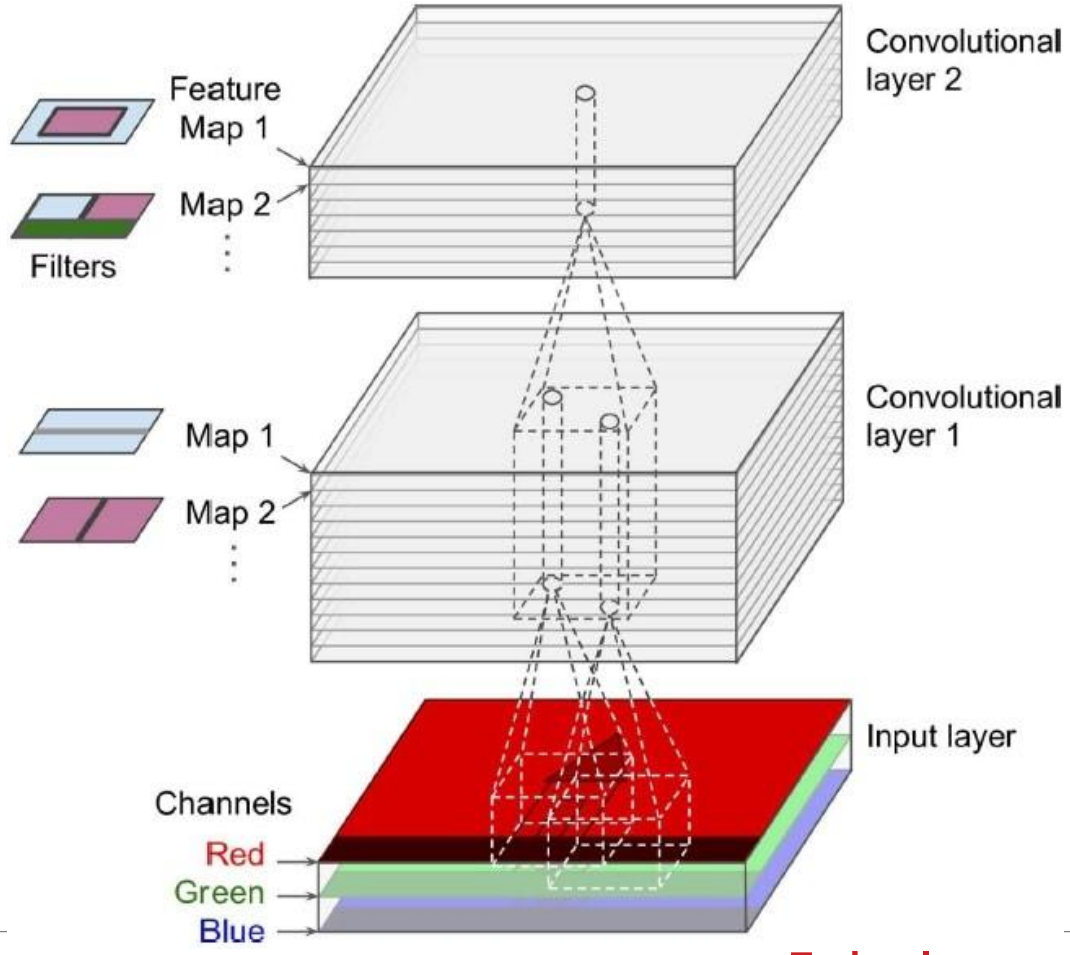
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Multiple Feature Maps

Filtermatrix

- speichert alle Filter einer Schicht in einem Array
- jede Convolution Schicht verfügt über eine Filtermatrix



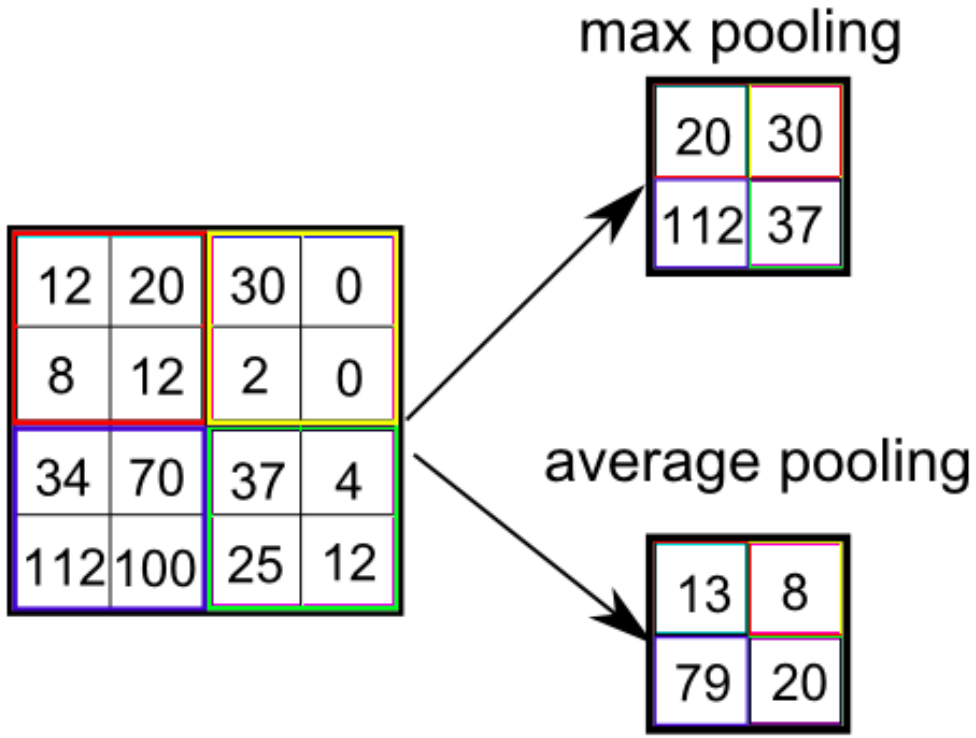
Quelle: Patrick S., Hahn-Schickard

Convolutional Neural Network - CNN

Pooling-Schicht

Pooling-Schicht

- Aggregiert die Ausgabe nach einer Convolutional Schicht (nach der Aktivierungsfunktion)
  - Max oder Mean
-
-
  -
  -
  -
-

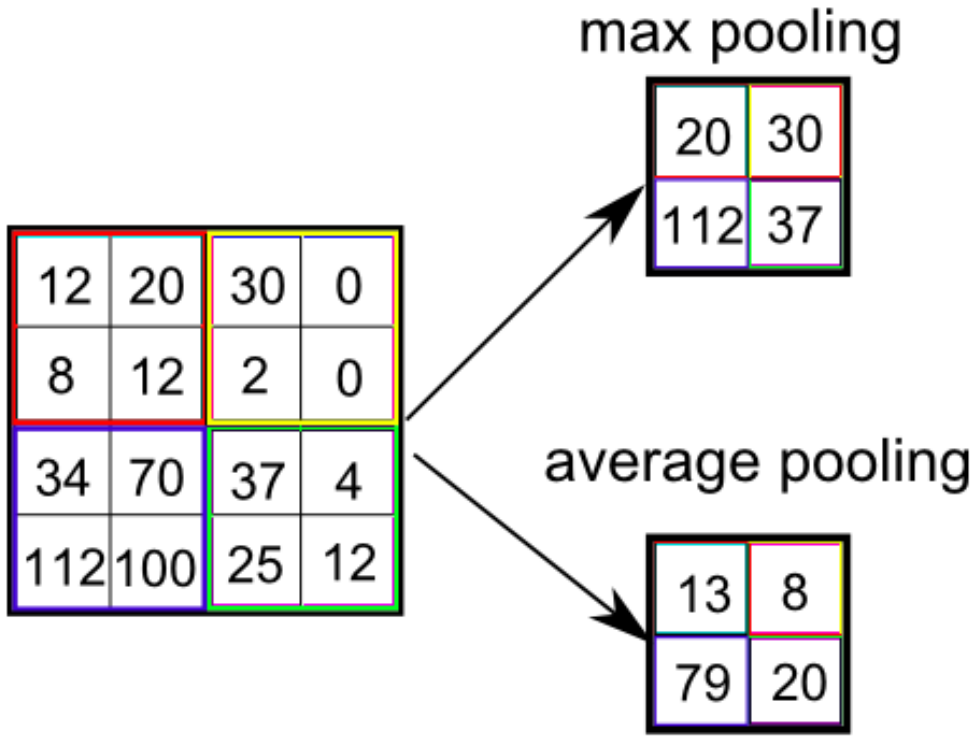


Convolutional Neural Network - CNN

Pooling-Schicht

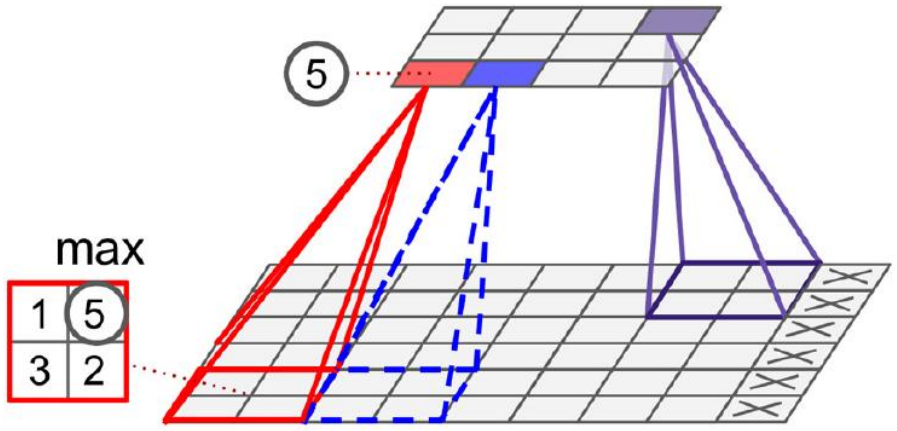
Pooling-Schicht

- Aggregiert die Ausgabe nach einer Convolutional Schicht (nach der Aktivierungsfunktion)
  - Max oder Mean
- Schrumpft das Eingabebild
- Einstellungen
  - Pooling-Kernel (z.B. 2x2)
  - Stride
  - Paddingtyp
- Tiefe der Schicht bleibt erhalten

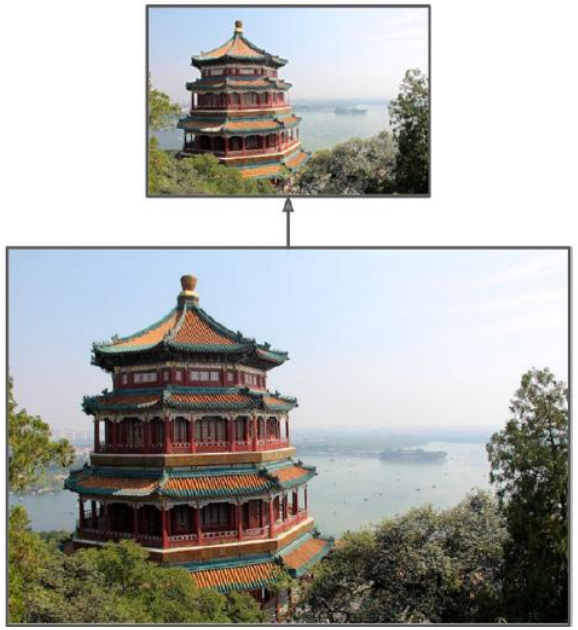


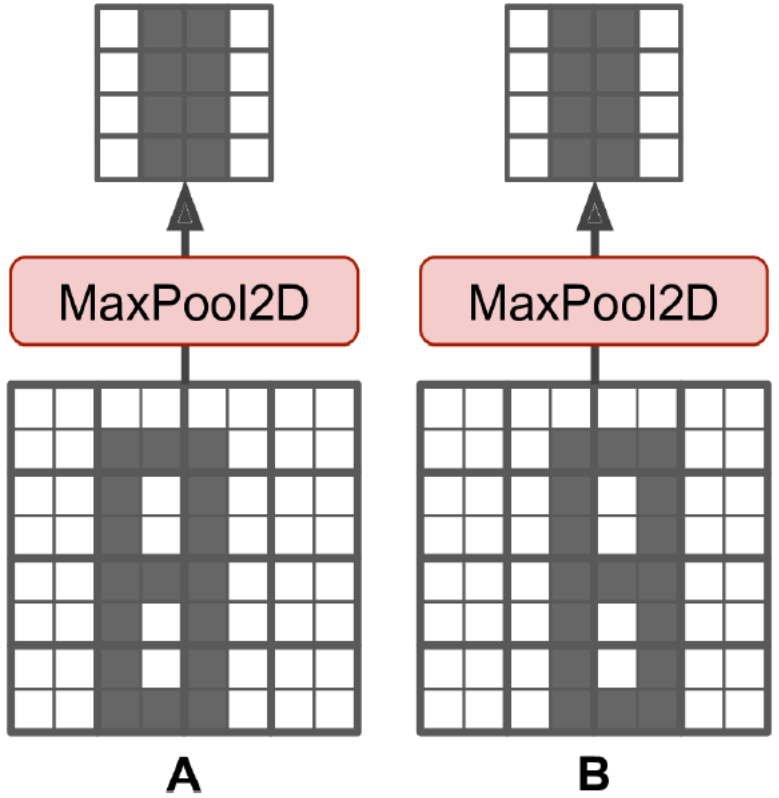
Convolutional Neural Network - CNN
Pooling-Schicht

- Reduziert Speicherbedarf
- Gewährleistet kleine Translations- und Rotationsinvarianz



Max Pooling, filter: 2x2, stride = 2, kein Padding

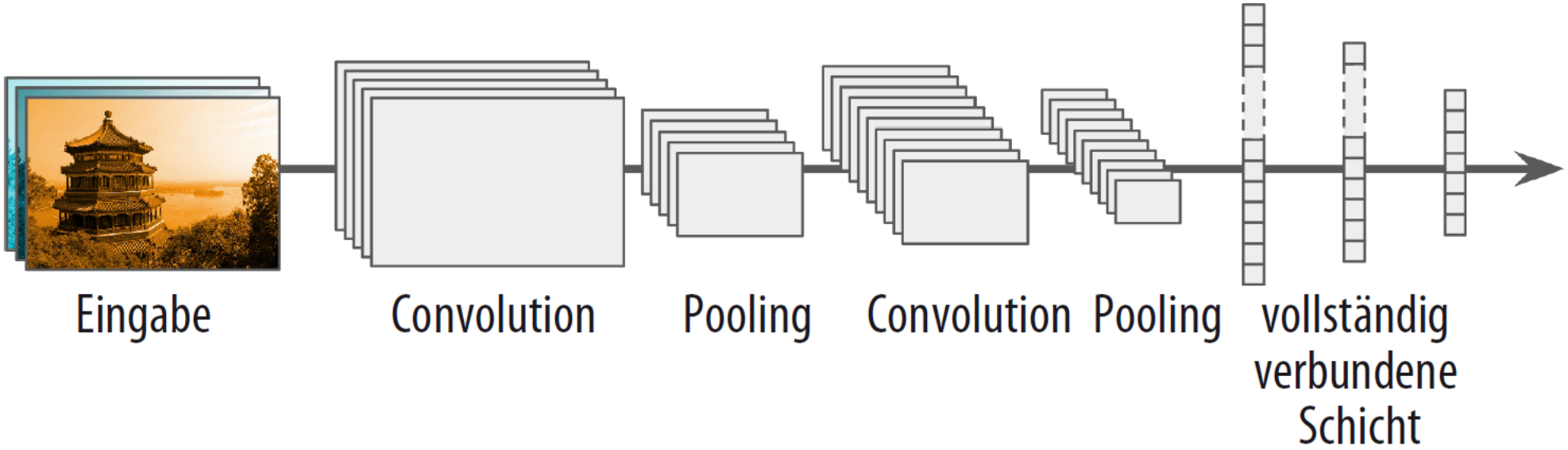




Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

Convolutional Neural Network - CNN

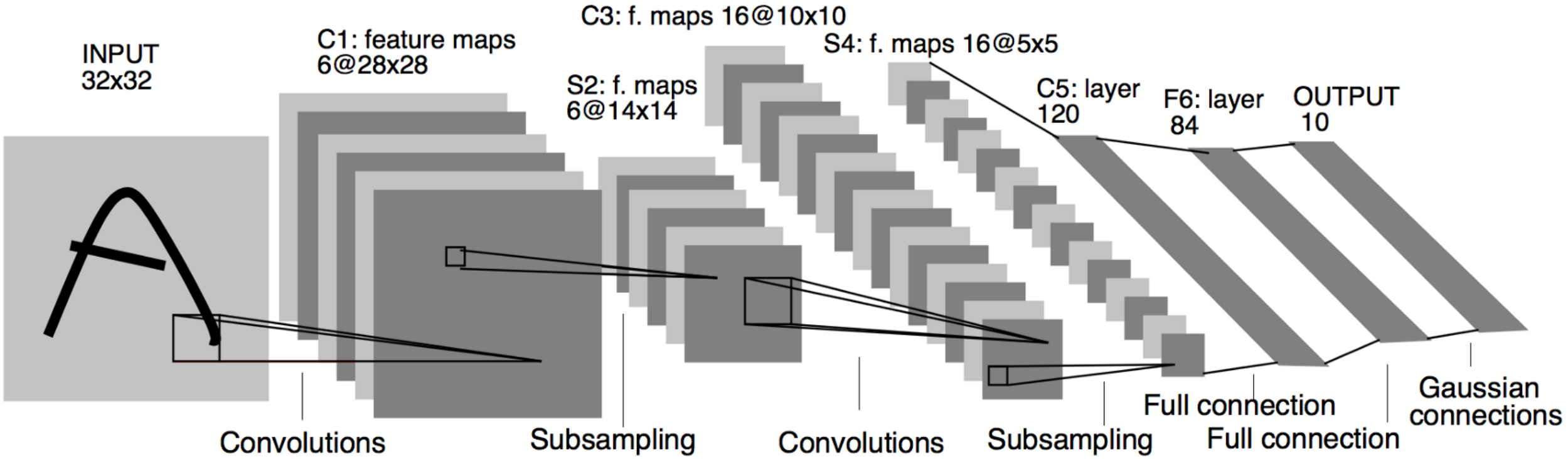
Typische Architektur



Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 2nd edition, 2019.

Convolutional Neural Network - CNN

LeNet-5 von Yann LeCun (1989)



- Drei CNN-Schichten: C1 (6 Filter), C3 (16 Filter) & C5 (120 Filter) (5x5 Filtergröße, Stride=1, kein padding)
- Zwei Pooling-Schichten: S2 & S4 (Poolingkernel: 2x2, Stride=2)
- Eine Fully-Connected-Schicht: F6 (84 Neuronen)
- Ausgabeschicht mit 10 Neuronen (Erkennung von handgeschriebenen Ziffern: 0 bis 9)

Convolutional Neural Network - CNN

Keras Implementierung des LeNet-5 CNNs

01_fashion_mnist_lenet.ipynb

Eingabeschicht wird nicht als eigener Layer modelliert, sondern über input_shape definiert

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, AveragePooling2D, Flatten, Dense

model = Sequential([
    Conv2D(filters=6, kernel_size=(5,5), activation='tanh', input_shape=(32,32,1), padding='valid'),
    AveragePooling2D(pool_size=(2,2), strides=2),
    Conv2D(filters=16, kernel_size=(5,5), activation='tanh', padding='valid'),
    AveragePooling2D(pool_size=(2,2), strides=2),
    Conv2D(filters=120, kernel_size=(5,5), activation='tanh', padding='valid'),
    Flatten(),
    Dense(units=84, activation='tanh'),
    Dense(units=10, activation='softmax')])
```

Convolutional Neural Network - CNN

Keras Implementierung des LeNet-5 CNNs

01_fashion_mnist_lenet.ipynb

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, AveragePooling2D, Flatten, Dense


model = Sequential([
    Conv2D(filters=6, kernel_size=(5,5), activation='tanh', input_shape=(32,32,1), padding='valid'),
    AveragePooling2D(pool_size=(2,2), strides=2),
    Conv2D(filters=16, kernel_size=(5,5), activation='tanh', padding='valid'),
    AveragePooling2D(pool_size=(2,2), strides=2),
    Conv2D(filters=120, kernel_size=(5,5), activation='tanh', padding='valid'),
    Flatten(),
    Dense(units=84, activation='tanh'),
    Dense(units=10, activation='softmax')])
```

Eingabeschicht wird nicht als eigener Layer modelliert, sondern über input_shape definiert

Konkateniert die 120 Feature Maps zu einem Vektor; hier ebenfalls 120 dimensional, da Feature Maps des 3. Conv-Layers nur noch 1-dimensional, s. vorherige Folie

Convolutional Neural Network - CNN

Flatten Layer

```
from tensorflow.keras.layers import Conv2D, AveragePooling2D, Flatten, Dense


model = Sequential([
    …
    Conv2D(filters=120, kernel_size=(5,5), activation='tanh', padding='valid'),
    Flatten(),
    Dense(units=84, activation='tanh'),
    Dense(units=10, activation='softmax')])
```

Konkateniert die 120 Feature Maps zu einem Vektor; hier ebenfalls 120 dimensional, da Feature Maps des 3. Conv-Layers nur noch 1-dimensional, s. vorherige Folie

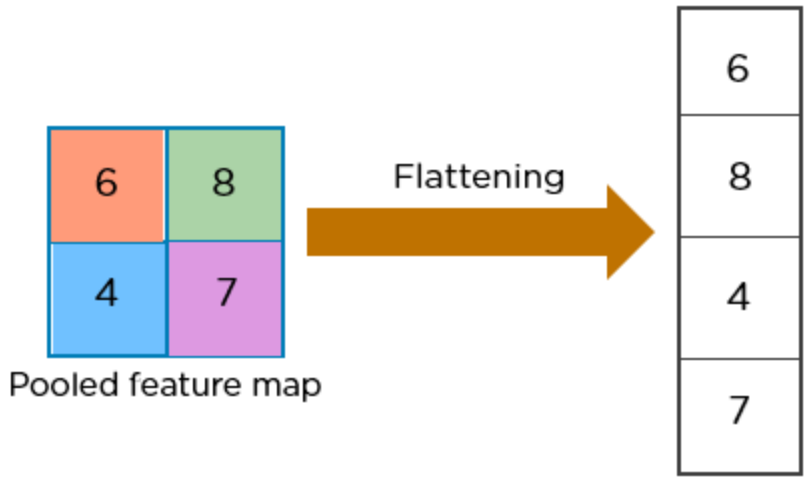


Abbildung: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network

Convolutional Neural Network - CNN

Keras Implementierung eines CNNs

02_fashion_mnist_cnn.ipynb

Implementierung einer CNN-Architektur für das MNIST-Fashion Klassifizierungsproblem

```
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'),
])
```

Convolutional Neural Network - CNN

Keras Implementierung eines CNNs

Zusammenfassung des Modells

Wie berechnen sich die Anzahl Parameter?

Layer 1 (kernel_size = 7, s. vorherige Folie):

- (7*7*1 + 1)*64 = 3200
- „*1“ da Grauwertbild mit 1 Kanal

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        3200
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               295040
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650
=================================================================
Total params: 1,413,834
Trainable params: 1,413,834
Non-trainable params: 0
```

Convolutional Neural Network - CNN

Keras Implementierung eines CNNs

Zusammenfassung des Modells

Wie berechnen sich die Anzahl Parameter?

Layer 1 (kernel_size = 7, s. vorherige Folie):

- (7*7*1 + 1)*64 = 3200
- „*1“ da Grauwertbild mit 1 Kanal

Layer 2:

- (3*3*64+1)*128 = 73856

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        3200
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               295040
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650
=================================================================
Total params: 1,413,834
Trainable params: 1,413,834
Non-trainable params: 0
```

Künstliche Neuronale Netze - KNN

Keras Implementierung eines CNNs

1. Kompilieren und trainieren des Modells

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="nadam", metrics=["accuracy"])
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_valid, y_valid))
```

2. Auswerten des Modells auf den Testdaten

```
model.evaluate(X_test, y_test)
```

```
10000/10000 - loss: 0.3467 - accuracy: 0.8816
```

Künstliche Neuronale Netze - KNN

Keras Implementierung eines CNNs

1. Kompilieren und trainieren des Modells

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="nadam", metrics=["accuracy"])
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_valid, y_valid))
```

2. Auswerten des Modells auf den Testdaten

```
model.evaluate(X_test, y_test)
```

```
10000/10000 - loss: 0.3467 - accuracy: 0.8816
```

3. Vorhersage auf „neue“ Daten

```
y_proba = model.predict(X_test[:3])
y_proba.round(2)
```

```
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.01, 0.  , 0.99],
       [0.  , 0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ]],
      dtype=float32)
```

Künstliche Neuronale Netze - KNN

Keras Implementierung eines CNNs - Übung

Ausgehend vom Notebook 02_fashion_mnist_cnn.ipynb probieren Sie

- verschiedene Architekturen,
- Parameter,
- Regularisierungstechniken aus

um bessere Ergebnisse auf den Validierungsdaten zu erzielen.

Falls Sie weights & biases zur Protokollierung Ihrer Ergebnisse nutzen möchten, können Sie auch mit dem Notebook 03_fminst_cnn_wb.ipynb starten.

Nächste Vorlesung und Fragen

- Rekurrente Netze, Transformer Netze und Natural Language Processing
- Fragen?