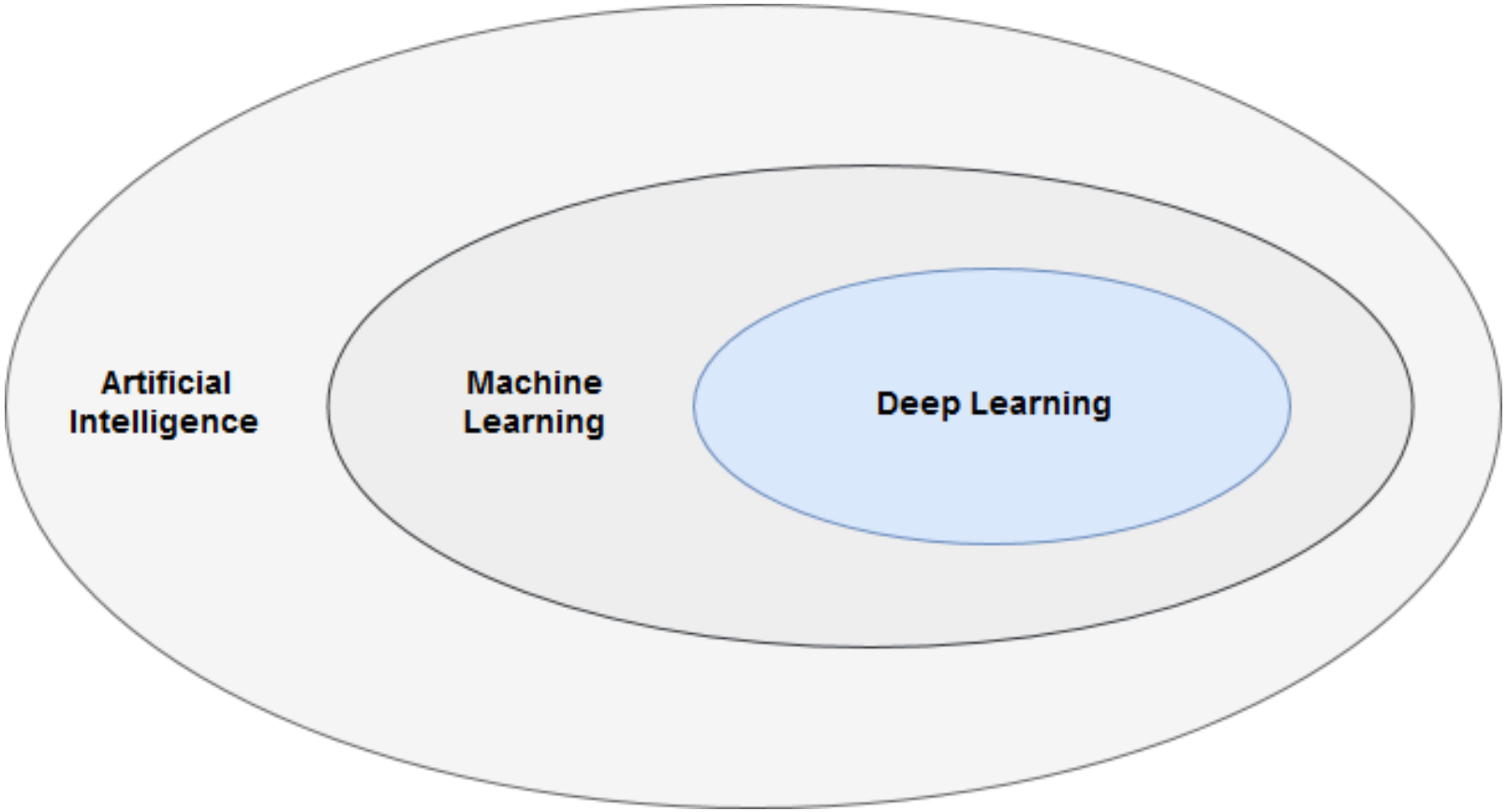
Bildbasierte Navigation mit Neuronalen Netzen: Kolloquium

Jan Robert Rösler

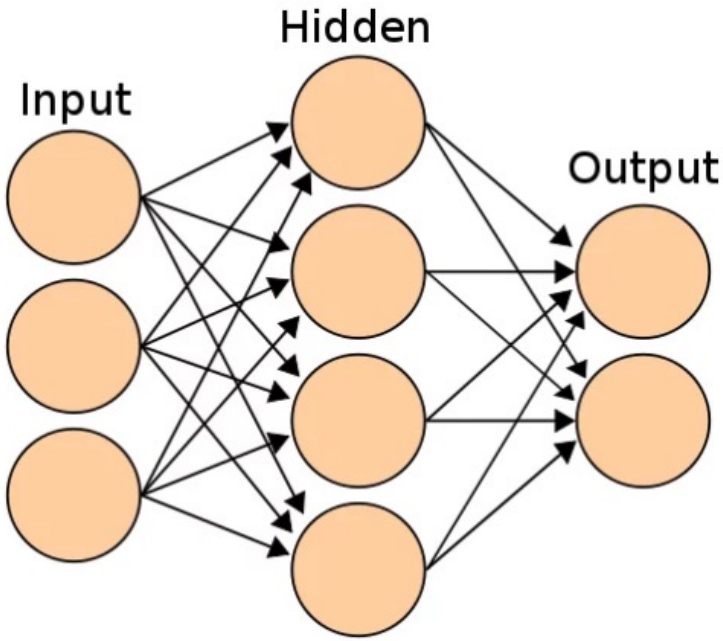
April 28, 2019

Deep Learning



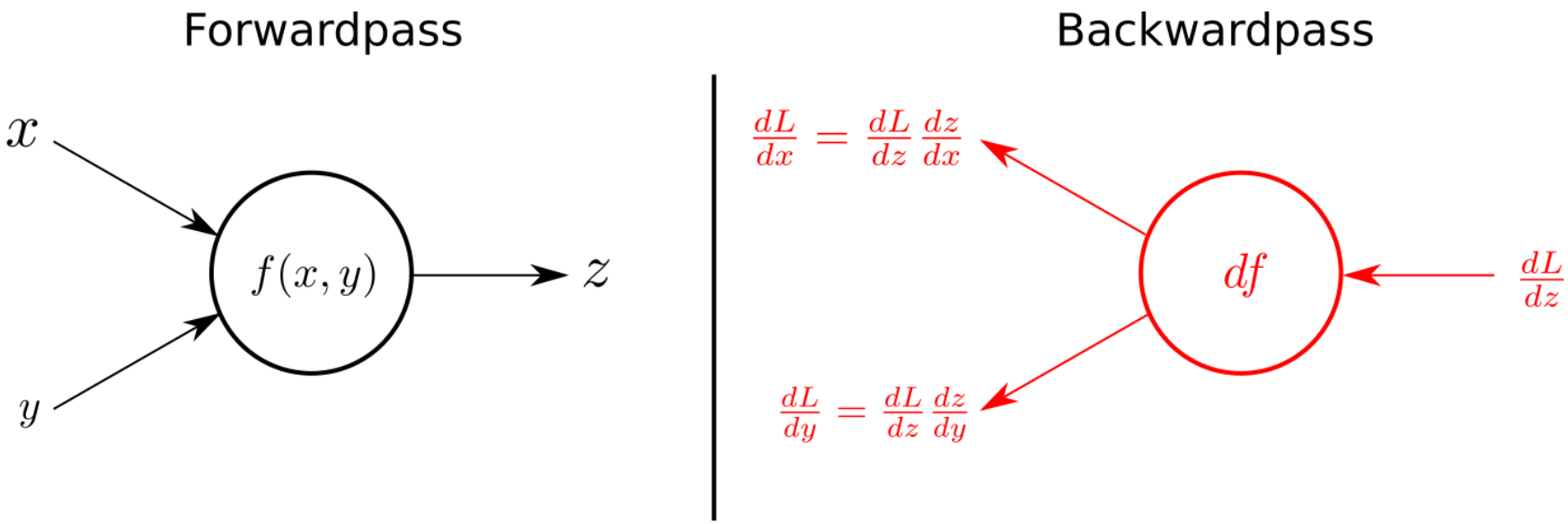
Deep Learning

Künstliche neuronale Netze sind universelle Funktionsapproximatoren.



Deep Learning

Berechnung mit Backpropagation.



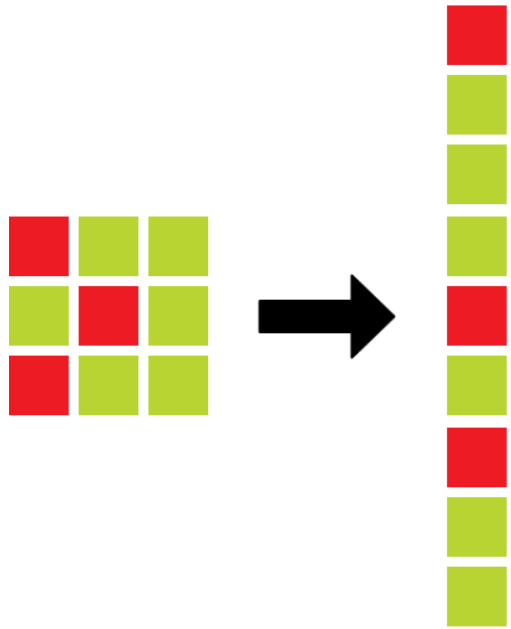
Deep Learning mit Bildern

Wie können Bilder in neuronalen Netzen verarbeitet werden?

Möglich:
Matrix zu einem einspaltigen Inputvektor umwandeln.

Problem:

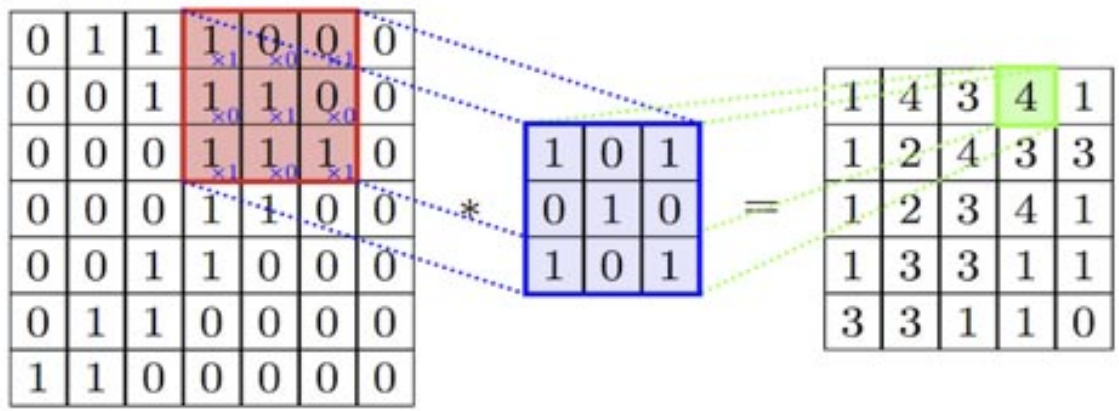
- Räumliche Information geht verloren
- Hoher Rechenaufwand



Convolutional Neural Network

Convolutional-Layer

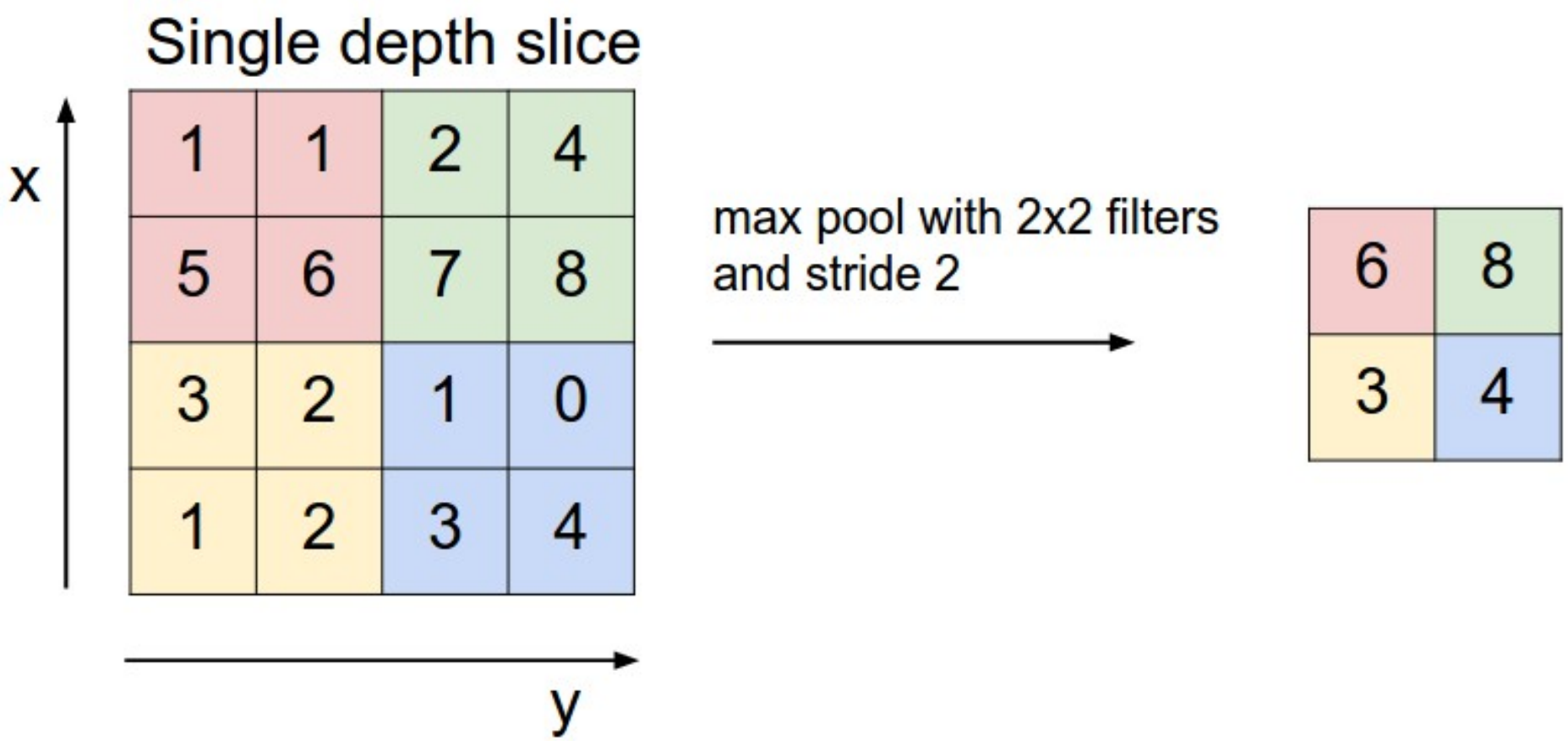
- Input Bilddaten als Matrix
- Neuronenaktivität wird über diskrete Faltung mit Faltungsmatrix (convolution) berechnet
- Extrahierung von Bildeigenschaften in feature-maps



Convolutional Neural Network

Pooling-Layer

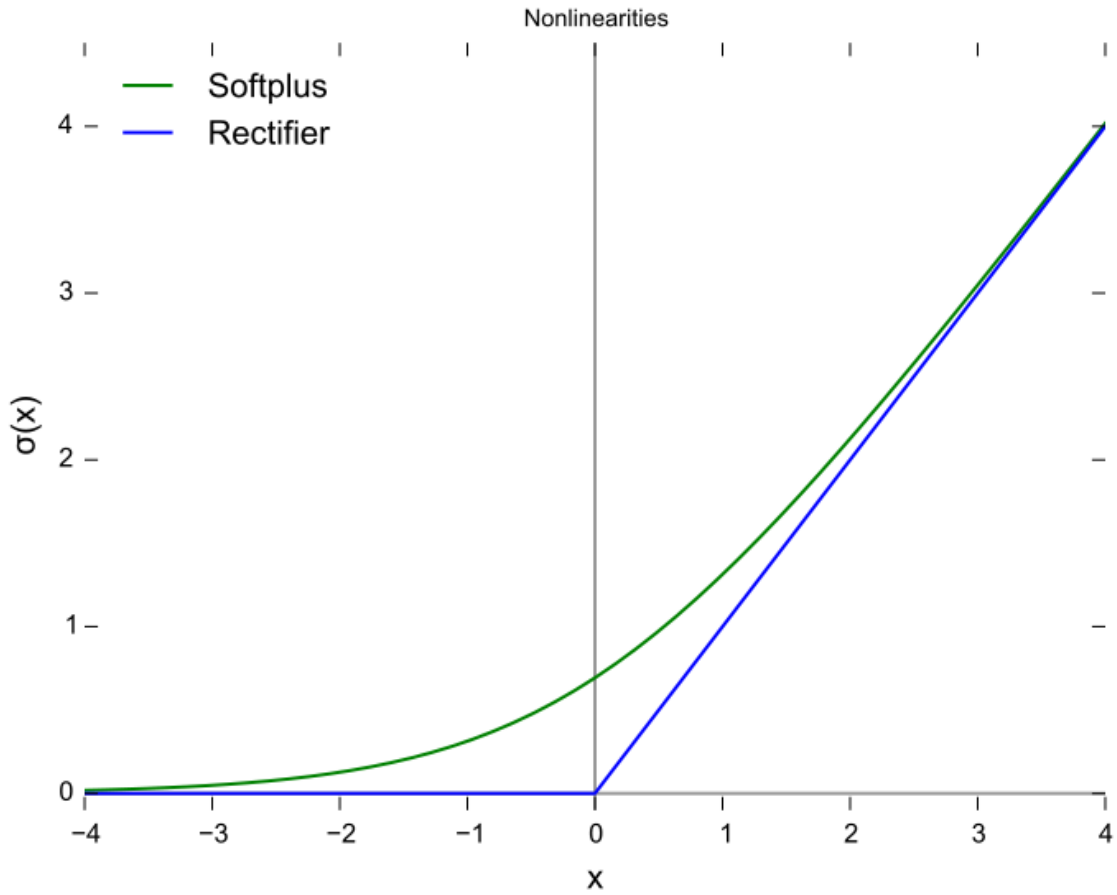
Subsampling der prägnanten Bildteile.



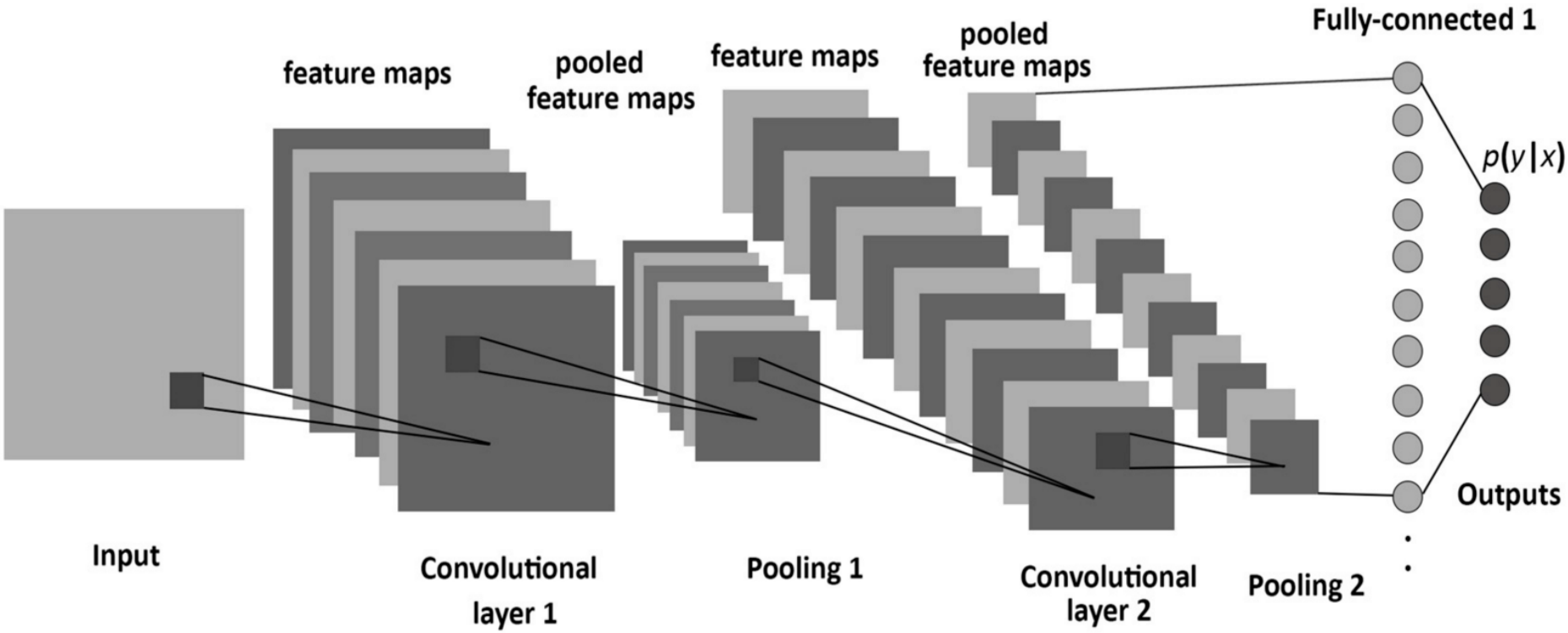
Convolutional Neural Network

Aktivierungsfunktion - ReLu

Die Aktivierungsfunktion wandelt den mittels Faltung ermittelten Neuroneninput in den Output um. **F(x) = max(0,x)**



Convolutional Neural Network

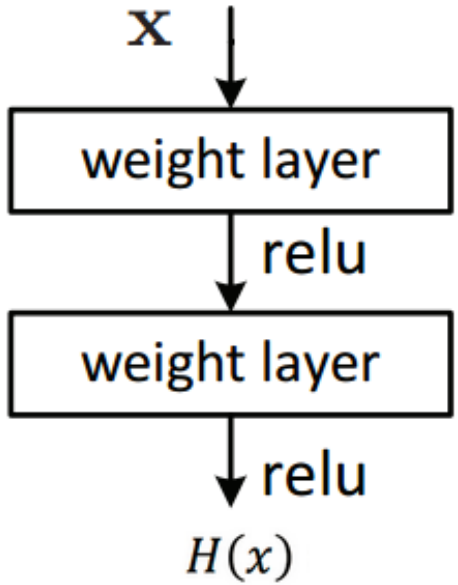


Vanishing Gradient Problem

Wird der durch Backpropagation berechnete Gradient an einer Stelle eines (tiefen) neuronalen Netzes sehr klein oder sogar Null, erfahren die Gewicht der früheren Layer sehr kleine oder gar keine Aktualisierungen mehr. Der "verschwindende" Gradient hält das Netz vom Lernen ab.

Residual Networks

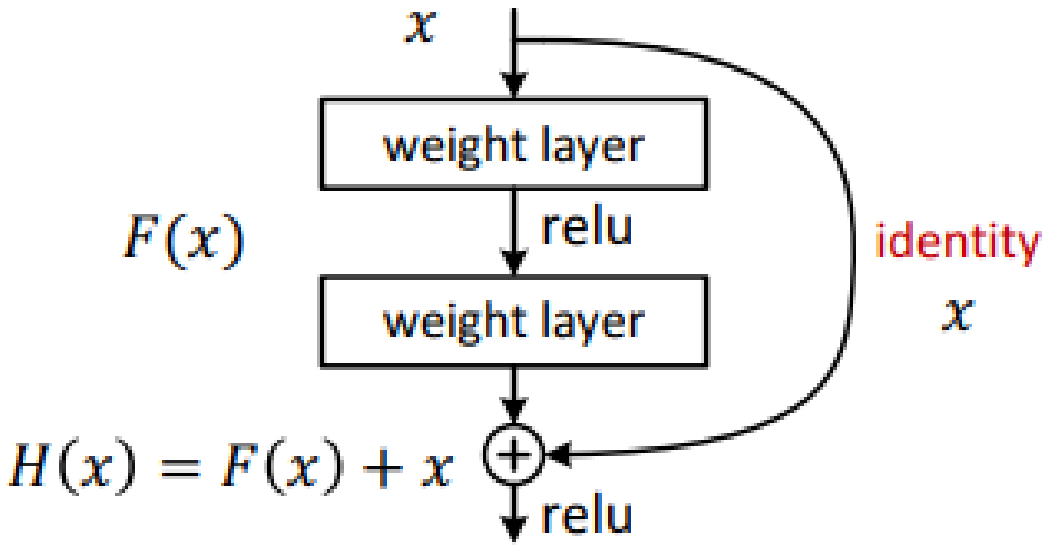
Annahme:
Die Funktion $H(x)$ ist die optimale Lösung für ein Problem. Sie soll approximiert werden.



Residual Networks

Identity Shortcut

Umwandlung:
Statt H(x) wird
$H(x) = F(x) + x$ gelernt.
F(x) ist das Residual, die Differenz $H(x) - x$.

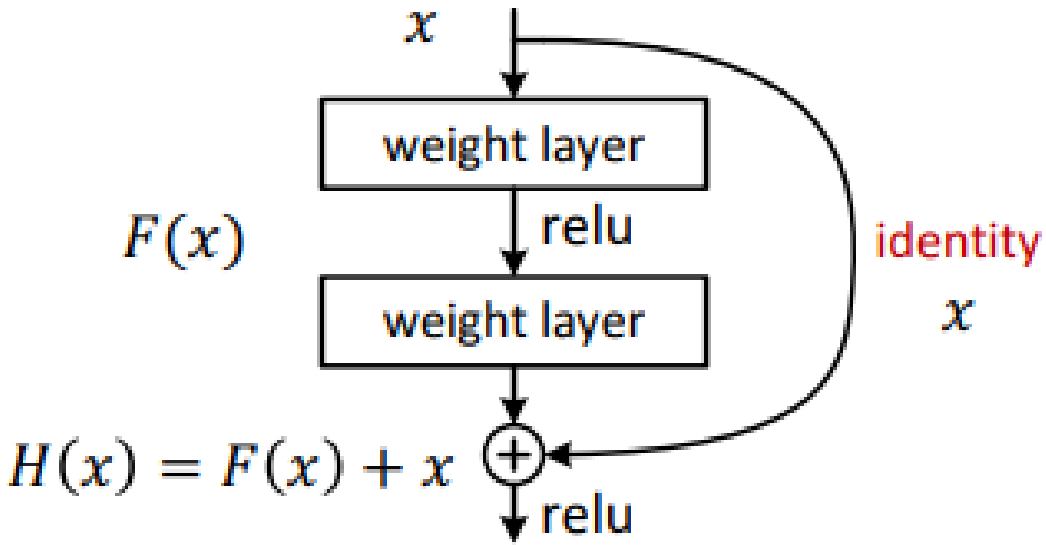


Residual Networks

Identity Shortcut

Vorteile:

- Freier Gradientenfluss
- Durch lernen von F(x)=0 wird H(x) = x
  →Identitätsfunktion leicht zu erlernen



DroNet

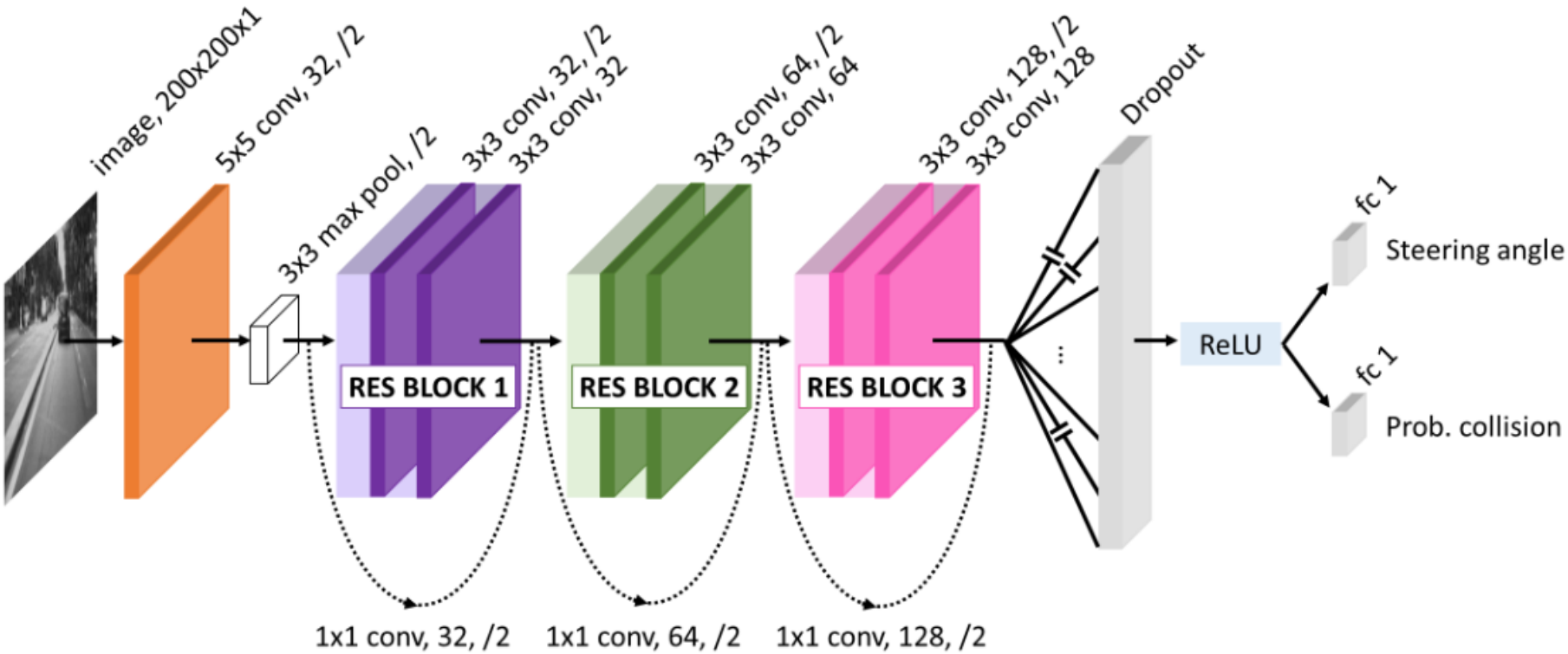
Projekt der ETH Zürich:
Lenkwinkel- und Kollisionsbestimmung mit einem neuronalen Netz.
Trainiert auf einem öffentlichen Datensatz mit Fahrbahnbildern,
angewendet auf einer Drohne.

Input:
200x200 Pixel Graustufenbild

Output:
Lenkwert von -1 (rechts) bis 1 (links) und Kollisionswahrscheinlichkeit in Prozent

DroNet

Architektur



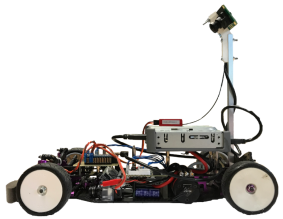
Idee

Anstoß: DroNet Projekt ETH Zürich
Rahmen: Carolo-Cup
Basis: HAW Teststrecke, Carolo-Cup Fahrzeugplattform
Ziel: Entwicklung einer bildbasierten Fahrzeugsteuerung mit einem neuronalen Netz

Daten

Bilder sammeln

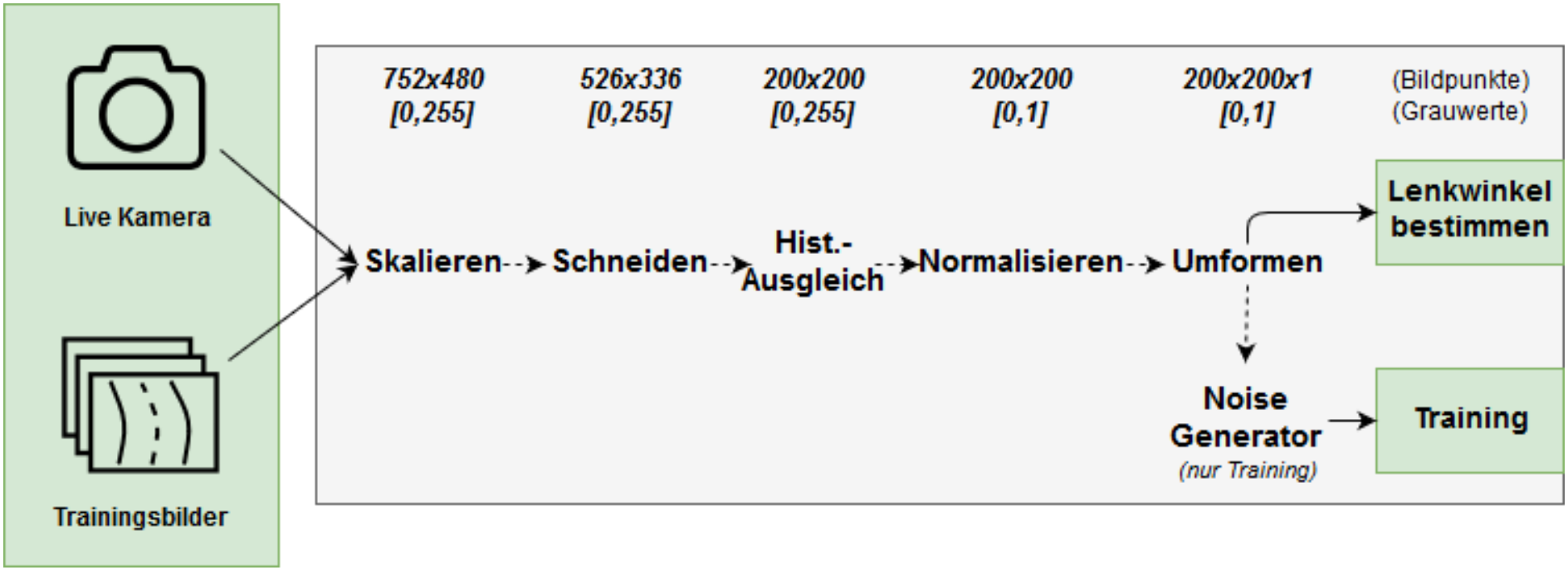
- Algorithmus TeamWorstcase
- ca 20.000 Aufnahmen
- 6000 davon geeignet für Training



Daten

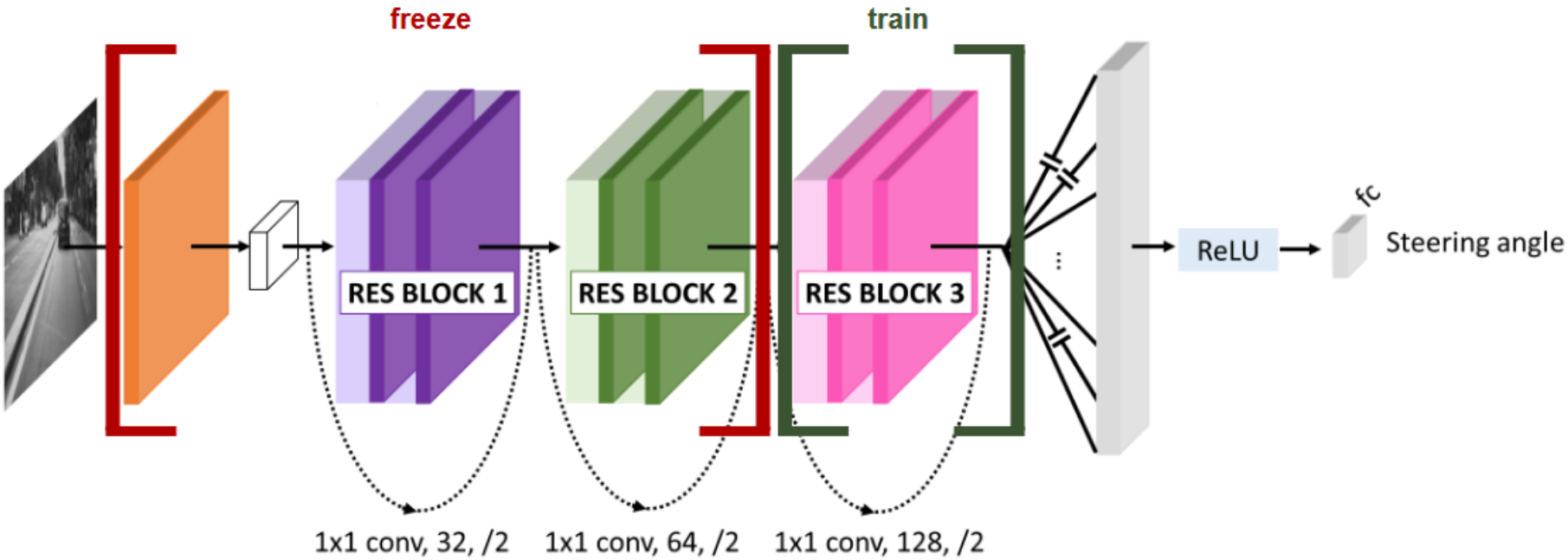
Bildverarbeitung



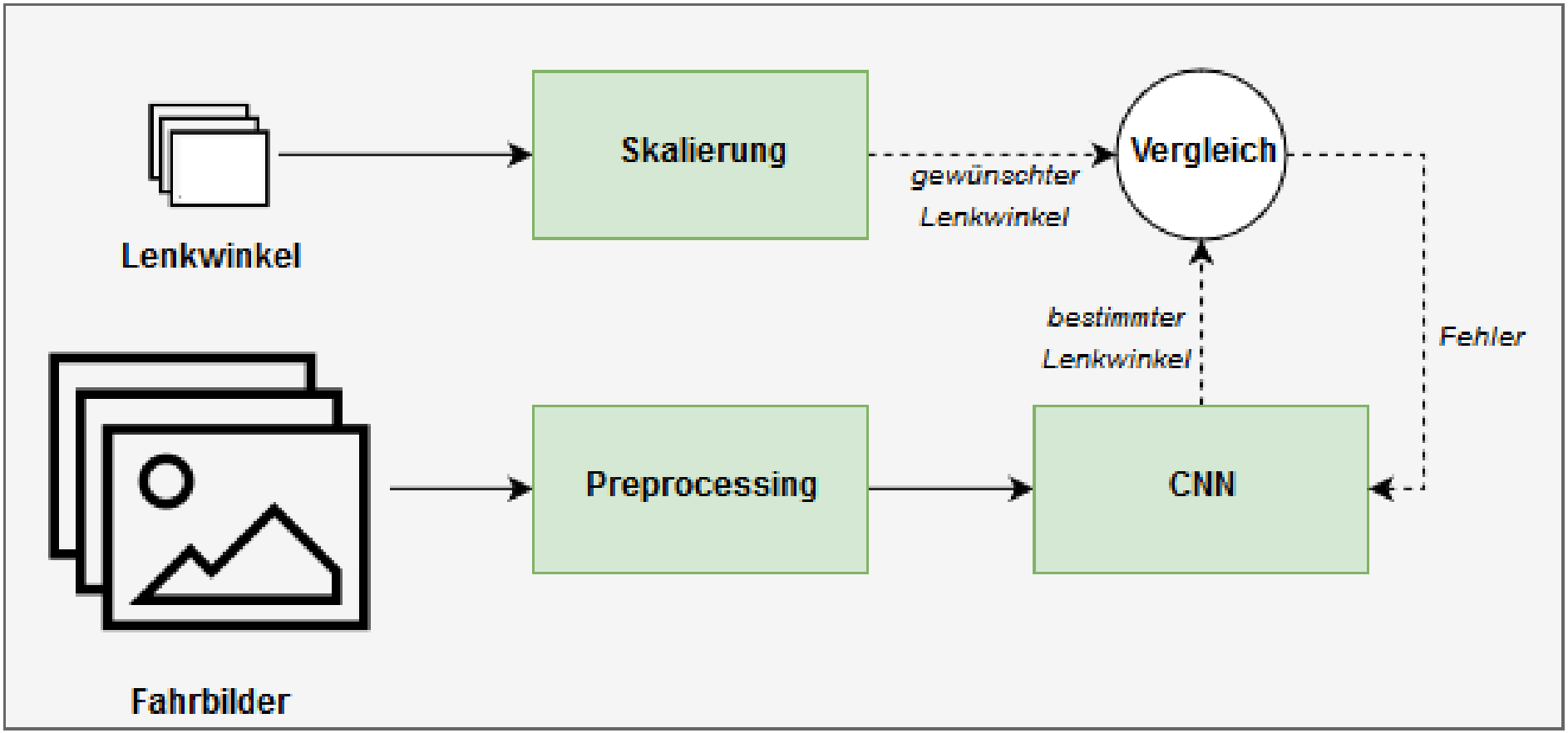


Architektur

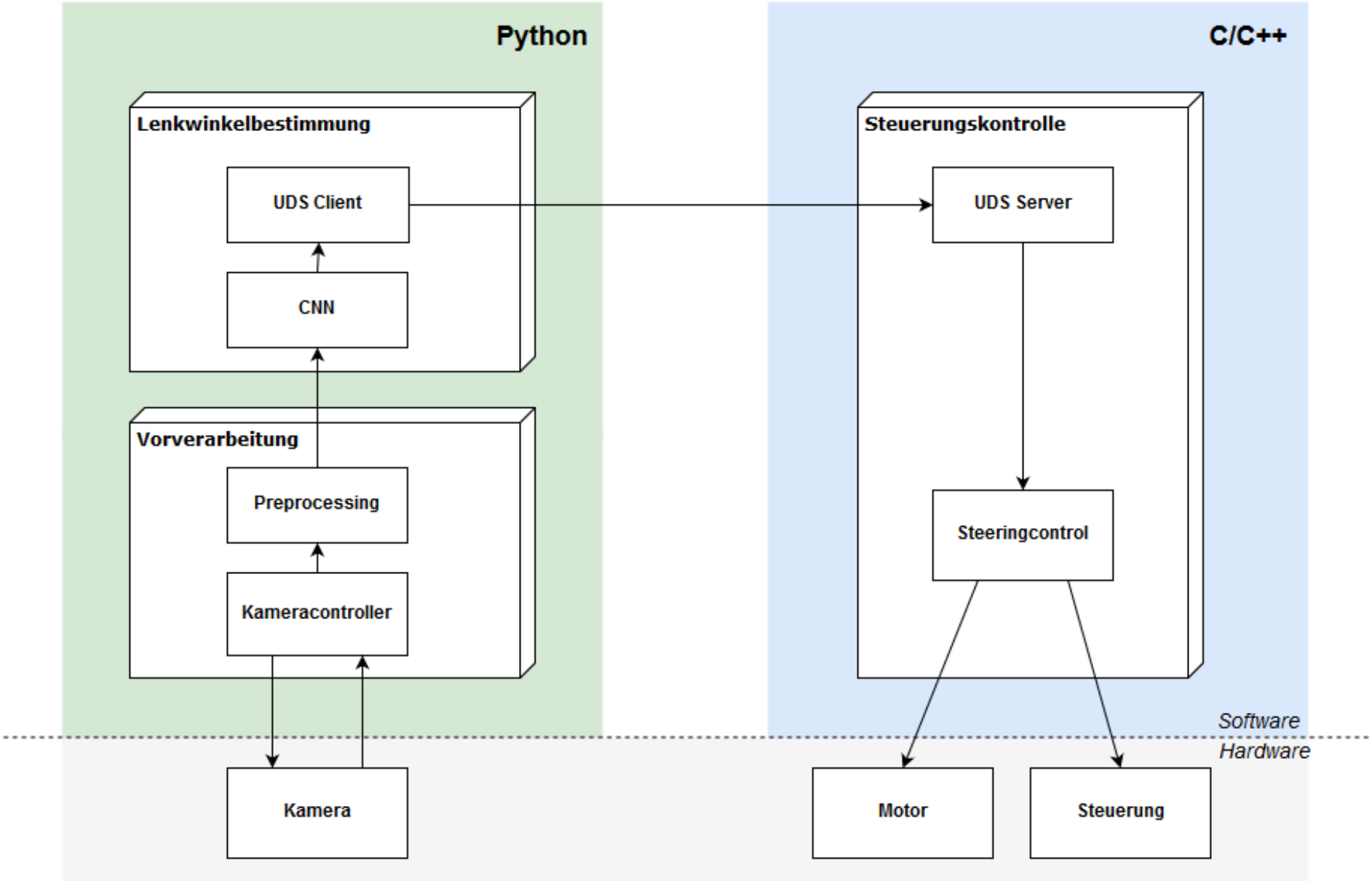
Anpassungen



Training


Lenkwinkel
Skalierung
gewünschter
Lenkwinkel
Vergleich
bestimmter
Lenkwinkel
Fehler
Fahrbilder
Preprocessing
CNN

Steuerung


Python
C/C++
Lenkwinkelbestimmung
UDS Client
CNN
Vorverarbeitung
Preprocessing
Kameracontroller
Kamera
Steuerungskontrolle
UDS Server
Steeringcontrol
Motor
Steuerung
Software
Hardware

Auswertung

Training

Auswertung

Testfahrt

→ Aufnahmen

Testfahrt

Performance messen - Metrik

$$Autonomiewert = (1 - \frac{\text{Anzahl Fehler} \cdot 2s}{\text{Fahrzeit in Sekunden}}) \cdot 100 \quad (1)$$

Testfahrt

Performancemessung

Algorithmus	**Fehler Runde 1**	**Fehler Runde 2**
DroNet	16	12
Carolo-Projekt	7	11
BA-RR	3	5

Gesamtfahrzeit = 120 Sekunden
Runde 1 im Uhrzeigersinn (60 Sekunden)
Runde 2 gegen den Uhrzeigersinn (60 Sekunden)

Testfahrt

Performancevergleich

Algorithmus	**Autonomiewert**
DroNet	53 %
Carolo-Projekt	70 %
BA-RR	87 %

Auswertung

Visualisierung

Sichtbarmachen von wichtigen Bildbereichen

Auswertung

Attention-Heatmap

Bewertung

Ausblick

Quellen