# Abkürzungsverzeichnis

| | |
|---|---|
| **KI** | Künstliche Intelligenz |
| **AI** | Artificial Intelligence |
| **KNN** | Künstlich Neuronale Netze |
| **CNN** | Convolutional Neural Network |

# 1 Einführung Künstliche Intelligenz

Damit ein Fahrzeug autonom, d. h. ohne menschliches Zutun, im Strassenverkehr reagieren kann, muss es Faehig sein, die verschiedenen Verkehrsteilnehmer zu identifizieren. Hierbei wird sich einem Teilgebiet der angewandten Informatik bedient - der Künstliche Intelligenz (KI), auch Artificial Intelligence (AI) genannt.

## Overview

Hier werden folgende Fragen beantwortet:

- Was ist Artificial Intelligence
- Modellbildung Neueronaler Netze
- Technische Umsetzung
- Uebersicht: Convolutional Neueral Networks
- Training eines Neuronalen Netzes
- Der YOLO Algorithmus
- Bisheriger Stand im Projekt - Status Quo

## Was ist Künstliche Intelligenz

*Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.*[1]

KI ist eine Teildisziplin der angewandten Informatik, in der es um automatisierte Problemloesungen geht. Die Urspruenge lagen im Ansatz, allein Logik zu verwenden. Nachteil dabei: hohe Komplexitaet fuer Abdeckung aller Faelle. Es wurde als neuer Ansatz das menschliche Gehirn als Grundlage verwendet. Menschen reagieren auf eine sich veraendernde Umwelt. Wir lernen aus Beispielen. Es erfolgt keine Abarbeitung von Schlussregeln wie "WENN ..., DANN ...". Man stelle sich das Halten des Gleichgewichts beim Fahrradfahren oder das Schreiben auf einem Blatt Papier vor. Die Reaktionszeiten fuer eine sequenzielle Abarbeitung waere viel zu hoch. Beim Fahrradbeispiel waere man schon laengst vor der Abarbeitung aller Regeln umgeflogen. Das selbe gilt analog fuer den Fall des Schreibens. Es wuerde einfach viel zu lange dauern und die Regeln waeren viel zu komplex.

[1] [Bos20]

Unser Gehirn funktioniert anders. Es passt sich den Situationen an und greift dabei auf Erfahrungen von schon erlebten Situationen zurueck. Das wollte man auch fuer Maschinen – dynamische Anpassung auf sich aendernde Randbedingungen. Das ist auch der Grund, warum kuenstliche Netze antrainiert werden muessen. Sie sollen auf Situationen reagieren und dabei auf 'Erfahrung von schon erlebten' zurueckgreifen. Das 'Schon Erlebte' stellt dabei das Trainingsmaterial dar, mit dem das Netzwerk antrainiert wird. Dennoch, im Allgemeinen ist zu entscheiden:

- Existiert ein Algorithmus, dann ist dieser vorzuziehen.
- Ist Erfahrung vorhanden, ein Problem anhand von Regeln zu beschreiben, ist dies vorzuziehen.
- Wenn die ersten beiden Ansaetze nicht zum Erfolg fuehren oder nicht fuehrten, dann kann der Einsatz von KI zielfuehrend sein, wenn genuegend Daten vorhanden sind, aus denen gelernt werden kann.

Im Strassenverkehr aendern sich zu jedem Zeitpunkt die Randbedingungen. Autos biegen ab, bremsen, ueberholen. Es gibt Verkehrsschilder, Ampelanlagen, Fussgaenger- und Fahrradwege und mehr. Menschen greifen im Strassenverkehr auf ihre Erfahrung zurueck. Moechte man als Ziel ansetzen, dass Fahrzeuge in der Lage sein sollen, im Verkehr ohne menschliches Zutun sich zu bewegen, ist eines ersichtlich. Die KI ist prädestiniert fuer den Einsatz im Bereich *autonomes Fahren*.

## Künstlich Neuronale Netze (KNN)

KNNs versuchen das menschliche Gehirn nachzubilden. Wie unsere Nervenzellen trainierbar sind und Steuerungsaufgaben uebernehmen, so moechte man auch, dass Computerprogramme aehnlich befaehigt sein sollen, in analoger Weisse auf neue Situationen aus schon erlebten Beispielen zu reagieren. Man bediehnt sich hierbei dem Gehirn als biologischen Bauplan Abb. 1.1.

Die Dendriten nehmen das Signal auf, leiten es an den Zellkoerper weiter. Dort findet eine Gewichtung der Informationen statt. Die gewichtete Gesamtinformation geht ueber das Axon weiter und verteilt sich auf die Synapsen, welche jeweils mit weiteren Dendriten anderer Neuronen verbunden sind. Ein einzelnes Neuron wird in Form eines Softwarebausteins nachgebildet. Ein Neuron $j$ besteht aus:

- $k$ gewichteten Eingaengen $W_{kj}$
- aus der Summe $net_j$ der jeweils einzelnen Produkten der Gewichte $W_{kj}$ und dem Wert des Eingangs $O_k$. $net_j$ stellt damit die Information zusammen, die aus dem Netz in das Neuron eingehen.
- der Aktivitaet des Neurons, d. h. wie stark der potenzielle Einfluss von $net_j$ auf andere Neuronen sein kann. $\Theta_j$ ist ein *Hyper Parameter*.
- der Ausgabe – Verbindung zu anderen Neuronen

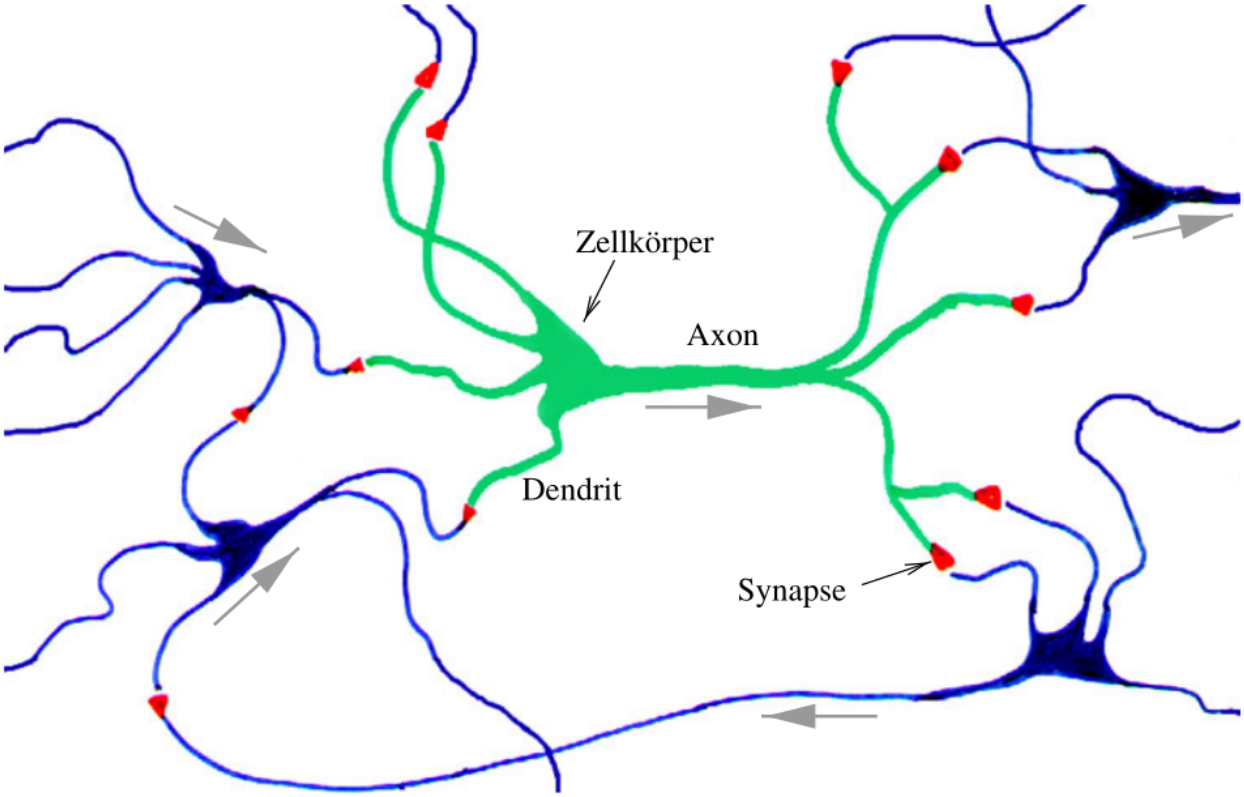


Abbildung 1.1: Menschliches Neuron; [Bos20]

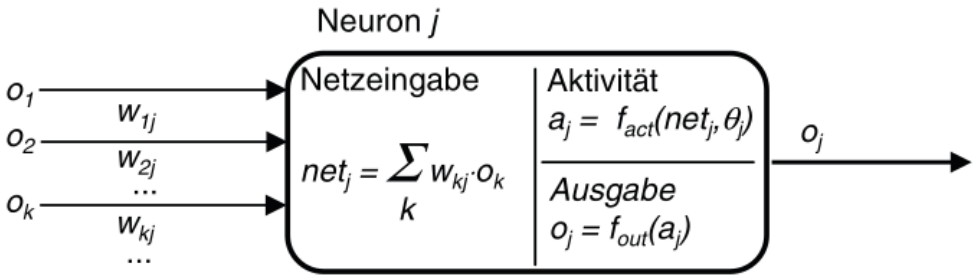


Abbildung 1.2: Modellierung eines einzelnen Neurons – Kuenstliches Neuron; [Bos20]

Abb. 1.2 entspricht der Beschreibung.

Viele Neuronen sind so miteinander verbunden und bilden zusammen ein *Neuronales Netzwerk*. Als technische Modellierung existieren verschiedene Varianten, von denen aber hier nur die *forwaets gerichtete* Variante interessiert.

Abb. 1.3 zeigt verschiedene Ausfuehrungen eines *Feed Forward Neural Networks* oder auch *Fully Connected Neural Network*. Das entspricht der klassischen Vorstellung eines KNNs. Jedes Neuron ist mit jedem vorherigen und jedem nachfolgenden Neuron verbunden.

Die erste Schicht bildet den Eingang. Dort treffen Signale ein, z. B. in Form eines Bildes. Es geht weiter in die zweite Schicht. Tiefer liegende Schichten werden auch als *hidden layer*

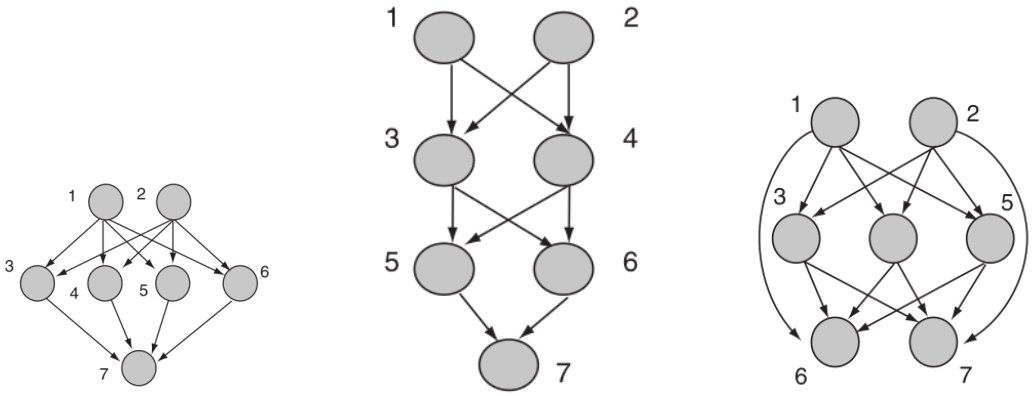


Abbildung 1.3: Forward Looking Networks

bezeichnet, da sie von aussen, d. h von den Schnittstellen, nicht zu sehen sind. Die letzte Schicht ist die Schnittstelle nach Aussen. Jedes Neuron gibt eine Wahrscheinlichkeit ueber den representierenden *Classifier* aus. Zum Beispiel als Anwendungsfall in der Objekterkennung, ob Objekt von der Klasse 'Afrikanischer Elefant' ist. Existieren mehrere Neuronen in der letzten Schicht, kann das Netz Aussagen ueber mehrere Classifier machen.

Es gibt noch wesentlich mehr Architekturen/Topologien, wie ein KNN aufgebaut sein kann. Jede hat ihre Vor- und Nachteile und somit eigene Anwendungsgebiete. In der Objekterkennung, Bildverarbeitung wird eine andere Variante als die fully connected Variante eingesetzt - das Convolutional Neural Network (CNN). Eine nicht vollstaendige Liste ueber verschiedene Topologien / Architekturen von Kuenstliche Neuronalen Netzwerken ist in Abb. 1.4 zu sehen.

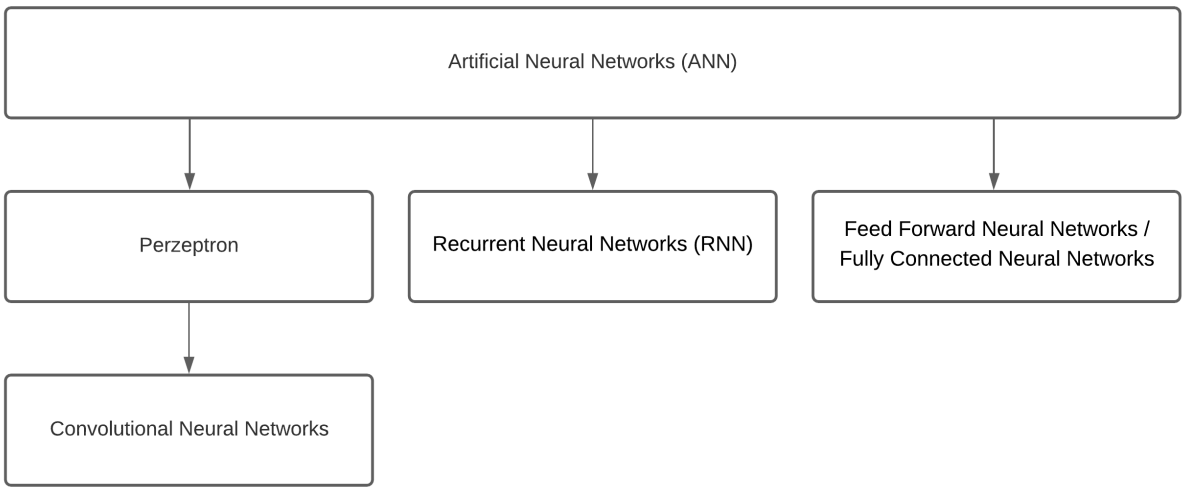


Abbildung 1.4: Eine nicht vollstaendige grobe Uebersicht ueber Topologien / Architekturen von Kuentlich Neuronalen Netzwerken

## Uebersicht CNN

Im Gegensatz zum *Fully Connected Network* sind die Neuronen des CNNs nicht mit jedem Neuron der uebergeordneten Schicht verbunden. Die Verbindung entsteht als Faltungsoperation mit einem Filter bestimmter Groesse (Kernel-Size). Der Filter ‘wandert‘ ueber das Bild und verknuepft so jeden Bildbereich mit dem darueber liegenden Neuronen. Jedes Neuron ist somit mit einem Filter verknuepft. Die Ergebnisse der Faltung entsprechen den Gewichten, mit denen das Neuron mit der darunter liegende Schicht verbunden ist. Allgemein gibt es mehrere Filter pro Schicht; dementprechend besteht jede Schicht aus mehreren Neuronen. In Abb. 1.5 ist das Vorgehen aufskizziert.

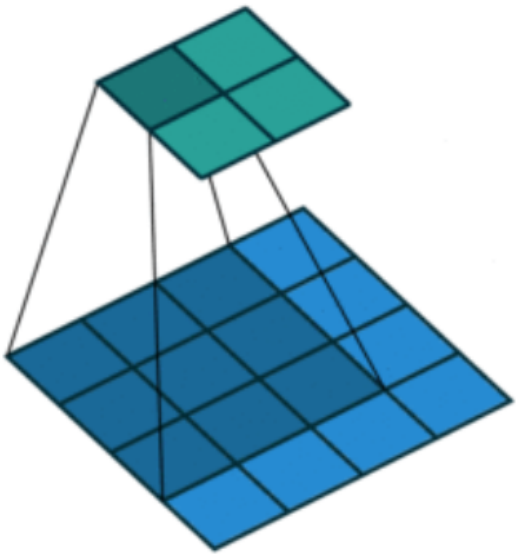

Abbildung 1.5: Jedes Neuron ist mit einer Faltungsoperation (Filter) mit dem darueber liegenden Neuron verbunden. Vorteil eines CNNs ist: gelernte Objekte muessen nur einmal gelernt werden, egal wo im Bild zu sehen (ob oben oder unten, links oder rechts). Das gilt nicht fuer *Fully Connected Networks*. Dort muss ein und das selbe Objekt mehrmals gelernt werden, wenn im Bild wo anders liegt.

Die Parameter der Filter haben anfangs zufaellige Werte. Die Werte werden im Trainingsprozess ‘gelernt‘ oder ‘antrainiert‘. Die Faltung, d. h. der Filter, wird nicht nur auf den Input (das Bild) , sondern auch zur Reduktion *Feature Extraction* der inneren Schichten *Hidden Layers* angewandt. Man kann fuer die inneren Schichten ein schon vortrainiertes, voellig beliebiges CNNs verwenden. Vorteil ist, es muss weniger Zeit fuer das Lernen seines Anwendungsgebietes aufgewendet werden und die *Feature Extraction* ist schon aufgebaut. Es gibt hierbei zwei wesentliche Punkte zu beachten:

- Es muss ‘nur‘ der Eingang auf die korrekte Groesse des Bildes angepasst werden.
- Die letzte Schicht muss ein *YOLO-Layer* sein mit den gewuenschten Ausgangsneuronen fuer jede Klasse an zu detektierenden Objekten.

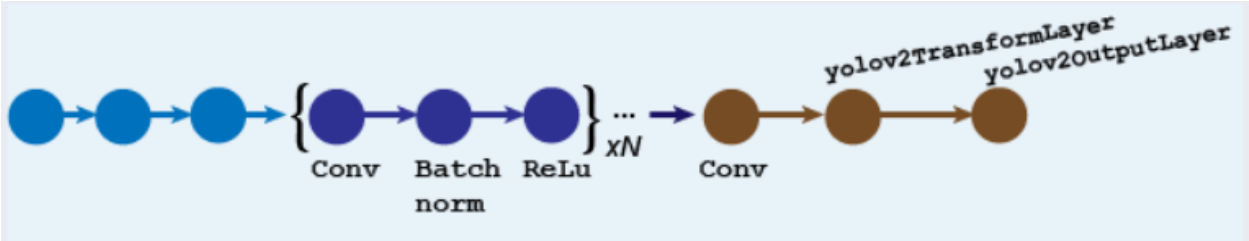


Abbildung 1.6: Links (hellblau) der Eingang (der Situation angepasst), in der Mitte (dunkelblau) das pretrained CNN Network, rechts (braun) das YOLO-Layer (auf Beduerfnisse angepasst)

In Abb. 1.6 ist die Struktur des YOLO-Netzwerks aufskizziert. Es gibt noch sehr viel mehr zu beachten und Parameter einzustellen.

## Training eines Kuenstlich Neuronalen Netzes

## YOLO Algorithmus

# 2 Anwendung der Theorie auf Praxis

In der Objekterkennung hat sich der YOLO Algorithmus bewaehrt. Die YOLO Variante unteschеidet nur in den Letzten Schichten. Ansonsten sind CNN und YOLO von der Architektur identisch.

Es gibt mehrere Technologien, die den Aufbau eines Neuronalen Netzwerks unterstuetzen. Einige Bibliotheken in Python sind beispielsweise: PyTorch, Keras, Tensor Flow. Auch Matworks stellt ein eigenes Framwork bereit und ist in der Matlab eigenen Sprache zu bedienen. Vorteil von Matlab ist: Es gibt sehr viele speziell auf Matlab angepasste Tools fuer den kompletten Erstellungs-, Evaluierung- und Anwendungsprozesses. Hingegen die frei zugaenglichen Alternativen bieten einen kleineren Anwendungsbereich. Beispielweisse, wenn fuer die Evaluierung im Bild die Objekte gekennzeichnet werden sollen, muessen andere Tool als z. B. nur Tensor Flow verwendet werden wie OpenCV zur Bildverarbeitung. OpenCV reichert das Bild mit Rahmen um erkannte Objekte an und setzt Labels. In Matlab ist alles dabei und man externen Tools sind unnoetig.

Hier im Projekt wird Matlab als Entwicklungsumgebung verwendet, weil Vorgaengerarbeiten darauf aufgebaut haben. Wuerde man sich dagegen Entscheiden, muesste der Erstellungsprozess von vorne begonnen werden.

## Overview

Hier werden folgende Fragen beantwortet:

- Erstellen, Trainiern, Testen
- Evaluierung mit schon vorhandenem Netzwerk
- Deployment auf Jetson Nano

## Erstellung YOLO-KNN

## Evaluierung - Vergleich Mit Vorgaengernetz

## Deployment Auf Jetson Nano

Schritte, die Noetig sind:

- YOLO Netzwerk
- Main-Funktion erstellen

**YOLO Netzwerk**: CNN, das ...

**Main-Funktion**: Funktion, die in einer Endlosschleife Bilder von der WebCam aufniehmt und dem Detektor uebergibt. Der Detektor scannt das aufgenommene Bild nach Objekten. Wenn Objekte gefunden wurde, dann wird die entsprechende Bounding Box und Label auf das Bild gebunden und anschliessend auf dem Bildschirm angezeigt. Hierbei muss entweder ein externer Monitor an den Jetson angeschlossen werden oder eine Remoteverbindung hergestellt werden.

# Literaturverzeichnis

[Bos20] Bosl, A.:
*Einführung in MATLAB/Simulink: Berechnung, Programmierung, Simulation.*
Carl Hanser Verlag GmbH & Company KG, 2020 https://books.google.de/books?id=SVoAEAAAQBAJ. –
ISBN 9783446465466