Smart Mechatronics

Hochschule Bochum
Bochum University
of Applied Sciences

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

Entwicklung einer Bildverarbeitung mit dem Schwerpunkt Personenerkennung für ein autonomes Logistik-Fahrzeug

Autor: Giuliano Montorio
giuliano.montorio@hs-bochum.de
Matrikelnummer: 015202887

Erstgutachter: Prof. Dr.-Ing. Arno Bergmann

Zweitgutachter: M.Sc. Bernd Möllenbeck

Abgabedatum: tt.mm.jjjj

Eidesstattliche Erklärung

Eidesstattliche Erklärung zur Abschlussarbeit:

«Entwicklung einer Bildverarbeitung mit dem Schwerpunkt Personenerkennung für ein autonomes Logistik-Fahrzeug»

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Bochum, 14. September 2020	
Ort, Datum	Giuliano Montorio

Danksagung

Ein großes Dankeschön gilt all jenen Personen, die mich im Rahmen dieser Masterarbeit begleitet und geholfen haben. Insbesondere möchte ich Herrn Prof. Dr.-Ing. Arno Bergmann, Herrn Bernd Möllenbeck, M.Sc. und Herrn Dr.-Ing. Christoph Krimpmann danken, die unsere Arbeit durch ihre fachliche und persönliche Unterstützung begleitet haben. Auch beim Fachbereich Elektrotechnik und Informatik der Hochschule Bochum, insbesondere Herrn Dipl.-Ing. Thorsten Bartsch möchten wir uns bedanken. Für die Bereitstellung von Informationen und Dokumente sind wir Herrn Dennis Hotze, M.Sc. und der Smart Mechatronics GmbH sehr dankbar, ohne deren Hilfe und finanzielle Unterstützung dieses Projekt nicht möglich gewesen wäre.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ALF	Autonomes Logistik Fahrzeug
BLDC	Brushless Direct Current
CAN	Controlled Area Network
CONSENS	Conceptual Design Specification Technique for the Engineering of Complex Systems
EPOS	Entwicklungsplattform Ortsfrequenzfilter-Sensor
FMEA	Failure Mode and Effects Analysis
LIDAR	Light Detection and Ranging
LTI	Linear, zeitinvariantes System
MCM	MotorController Module
RALF	Regelung eines Autonomen Logistik Fahrzeugs
ROS	Robot Operating System
RVIZ	ROS Visualization
SLAM	Simultaneous Localization and Mapping
TEB	Timed Elastic Band
TFEST	Transfer Function Estimation
URDF	Unified Robot Description Format
USBFS	Universal Serial Bus Filesystem

Symbolverzeichnis

Symbol	Bedeutung
D	Dämpfung der Übertragungsfunktion
G	Übertragungsfunktion
$\underline{H}$	Hilfsmatrix
K_P	Verstärkungsfaktor
K_s	Streckenverstärkung
$\mathcal{L}$	Laplace-Transformation
M	Momentanpol
$\mathcal{O}_\mathcal{T}$	Menge aller Odometriedaten
P_a	Skalierungsfaktor für manuellen Betrieb
P_m	Skalierungsfaktor für automatischen Betrieb
R_c	Circumscribed Radius
R_i	Inscribed Radius
T	Abklingzeitkonstante
T_g	Anstiegszeit
T_n	Nachstellzeit
T_u	Verzugszeit
T_v	Vorhaltezeit
U	Laplacetransformierte Eingangsgröße

Symbol	Bedeutung
$\mathcal{X}_\mathcal{T}$	Menge aller Positionsvektoren
$\mathcal{Z}_\mathcal{T}$	Menge aller Umgebungsmessungen
Y	Laplacetransformierte Ausgangsgröße
$\vec{a}$	Umrechnungsvektor
a_i	Koeffizienten der Differentialgleichung der Ausgangsgröße
b_j	Koeffizienten der Differentialgleichung der Eingangsgröße
$\vec{b}$	Allgemeines Bewegungsziel
c	Rotatorischer Bewegungsbefehl
d_i	Reelle Zahl
f	Cost Scaling Factor
g	Impulsantwort
h	Übergangsfunktion
$\vec{h}$	Hilfsvektor
i	imaginäre Einheit $i = \sqrt{-1}$
j	Komplexe Zahl $j = \sqrt{-1}$
k	Komplexe Zahl $k = \sqrt{-1}$
m	Karte der Umgebung
m_i	Landmarken
o	Odometrie
p	Wahrscheinlichkeitsfunktion
$\vec{p}$	Orientierungsvektor
q	Quaternion

Symbol	Bedeutung
r	Distanz
$\vec{r}_a$	Rotatorisches Bewegungsziel aus automatischen Betrieb
$\vec{r}_m$	Rotatorisches Bewegungsziel aus manuellen Betrieb
s	Komplexe Frequenz
t	Zeit
$\vec{t}_{ma}$	Translatorisches Bewegungsziel aus manuellen oder automatischen Betrieb
u	Eingangsgröße
$\hat{u}$	Sprunghöhe
$\vec{v}$	Geschwindigkeitsvektor
w	Führungsgröße
$\vec{x}_t$	Positionsvektor
$\vec{x}_0$	Startpositionsvektor
y	Ausgangsgröße eines Systems
z	Messwert der Umgebung
α	Fahrtwinkel
β	Posenwinkel
δ	Impulsfunktion
σ	Sprungfunktion

1 Einleitung

1.1 Motivation

Das Thema der künstlichen Intelligenz (KI) ist heutzutage allgegenwärtig. Smart Home Geräte wie Amazons Alexa, Siri der Firma Apple oder der Google Assistant gehören mittlerweile in jeden ... deutschen Haushalt und enthalten KI zur Spracherkennung. Derartige Technologien begleiten den Menschen jedoch nicht nur Zuhause sondern auch in der Transport- und Logistikbranche. Eine Potenzialanalyse zur künstlichen Intelligenz der Firma Sopra Steria zeigt, dass bereits im Jahr 2017 20% aller befragten Unternehmen solche Systeme einsetzten. 37% planten den zukünftigen Einsatz. Die Implementierung solcher Systeme hat Einfluss auf verschiedenste Eigenschaften der Wertschöpfungskette. Die Qualität des Fachprozesses wird mit ebenfalls steigender Geschwindigkeit erhöht. Die zur Logistikbranche gehörenden Transportfahrzeuge sind ebenfalls mit KI ausgestattet und sorgen so für weniger Arbeitsunfälle und eine schnellere, präzisiere Abarbeitung der Logistikaufgaben.

Das Projekt dieser Masterarbeit wird praktisch am autonomen Logistikfahrzeug angewendet, das aus dem Labor für Antriebstechnik der Hochschule Bochum stammt. Die Idee des ALFs ist es ein Fahrzeug zu entwickeln, das nach seiner Fertigstellung Logistikaufgaben am Standort der Hochschule Bochum lösen soll. Der Entwicklungsprozess stellt sich aus diversen Bachelor- und Masterarbeiten zusammen, die sowohl Hardware, als auch Softwareimplementierungen vorsehen. Bisher wurden zwei Abschlussarbeiten inklusive der praktischen Anwendung am ALF geschrieben. M.Sc. Dennis Hotze und M.Sc. Dominik Eickmann entwickelten in ihrem Masterprojekt das Fahrzeug und konnten Fahraufgaben ferngesteuert und manuell erledigen. Während der darauffolgenden Bachelorarbeit wurde eine Schlupfkompensation entwickelt, die den Drift am Fahrzeug durch Eingabe von Umgebungsinformationen verhindert. Weiterhin wurden Funktionen entwickelt, um grundlegende und autonome Fahraufgaben zu lösen. Das autonome Logistikfahrzeug aus der vorangegangenen Bachelorarbeit dient auch in dieser Masterarbeit als

Versuchsplattform.

1.2 Zielsetzung des Projekts

Die Grundidee und Herausforderung dieser Masterarbeit ist die Interaktion zwischen Menschen und Roboter und der dadurch resultierenden Bedienung des Systems ohne Eingabegerät. Bisher wurden anzufahrende Posen per Mausklick eingegeben oder Fahrmodi manuell gewechselt. Der Informationsfluss wird hierbei rein visuell und akustisch passieren. Letzteres wird in der Masterarbeit von Herrn Dittmann behandelt und durch ausgewählte Schnittstellen mit diesem Projekt verknüpft, um ein Gesamtsystem zu bilden. Die visuelle Komponente kann im weiteren Entwicklungsprozess des Roboters für verschiedene Anwendungsbereiche genutzt werden. Ziel dieser Masterarbeit ist die Erkennung und Unterscheidung von Personen. Das System wird zwischen bekannten und unbekannten Personen unterscheiden können und Informationen aus den gegebenen Daten generieren, die in der weiteren Entwicklung nützlich sind.

2 Grundlagen

Für ein besseres Verständnis, der in Kapitel Konzept... angewandten Methoden, werden anbei die Grundlagen behandelt. Informationen zu der verwendeten Hard- und Software wurden bereits in der vorangegangenen Bachelorarbeit vermittelt. Aufgrund Anforderung A... ist während der praktischen Anwendung keine Änderung der Hardware vorgesehen.

2.1 Neuronale Netze

Das Neuronennetz des menschlichen Gehirns dient als Vorbild für künstliche, neuronale Netze (KNN). Diese werden heutzutage als Lösung diverser Anwendungsprobleme angewendet, in denen komplexe Strukturen und Muster aus großen Datenmengen erkannt werden sollen. Das in diesem Projekt zugrundeliegende Bildverarbeitungsproblem besitzt die beschriebenen Eigenschaften und eignet sich somit für den Einsatz zur Erkennung von Personen. Anders als bei den meisten programmierten Applikationen ist die Ausgabe von KNN's lediglich probabilistisch. Beim vorliegenden, autonomen Logistikfahrzeug werden zur Personenerkennung derartige neuronale Netze verwendet.

2.1.1 Eigenschaften von neuronalen Netzen

Das Grundlage für die Eingabe in ein neuronales Netz ist die Skalierung der vorliegenden Daten auf eine definierte Größe. Diese wäre beispielsweise bei einem Anwendungsfall mit einer Audiospur die Frequenzspektren oder bei einem Bildverarbeitungsproblem die Pixel eines Bildes. Die skalierten Daten werden in einem Tensor gegeben der die Dimensionen der Eingabe hat. Somit unterteilt sich ein Bild in die drei Dimensionen, die Höhe, die Weite und die Farbwerte der Primärfarben pro Pixel, auch Kanäle genannt.

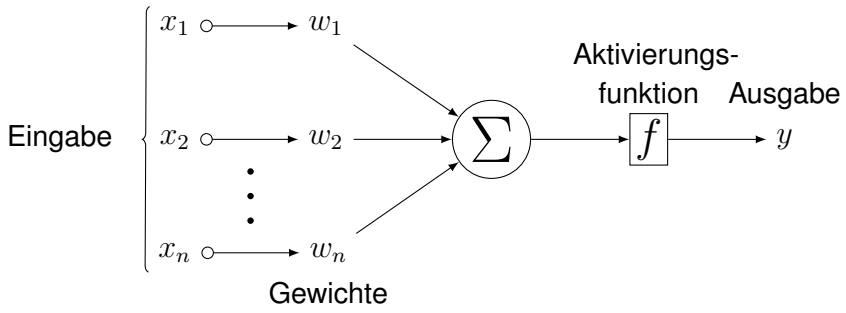


Abbildung 2.1: https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Der grundlegende Aufbau eines neuronalen Netzes besteht aus miteinander verbundenen Schichten, die häufig aus Neuronen bestehen. Die typische Struktur eine Neuron ist Abbildung 2.1 zu sehen. Es verarbeitet im wesentlichen eingehende Zahlenwerte x_n und gibt diese durch die Ausgabe y aus. Genauer wird mit den eingehenden Zahlenwerten eine gewichtete Summe gebildet. Diese wird dann auf eine Aktivierungsfunktion angewendet.

$$s = \sum_{j=1}^{n} w_{ij} x_j \tag{2.1}$$

Gleichung 2.1 zeigt das mathematische Modell der gewichteten Summe s. Das jeweilige Gewicht w wird mit dem Index j inkrementiert und mit dem dazugehörigen Eingang des Neurons i multipliziert. Alle Produkte werden aufsummiert und ergeben die gewichtete Summe. Es gibt verschiedene Varianten der Aktivierungsfunktion, die je nach Netzart zur Anwendung kommen können. Häufig werden für Aktivierungsfunktion Schwellwertfunktionen angewendet.

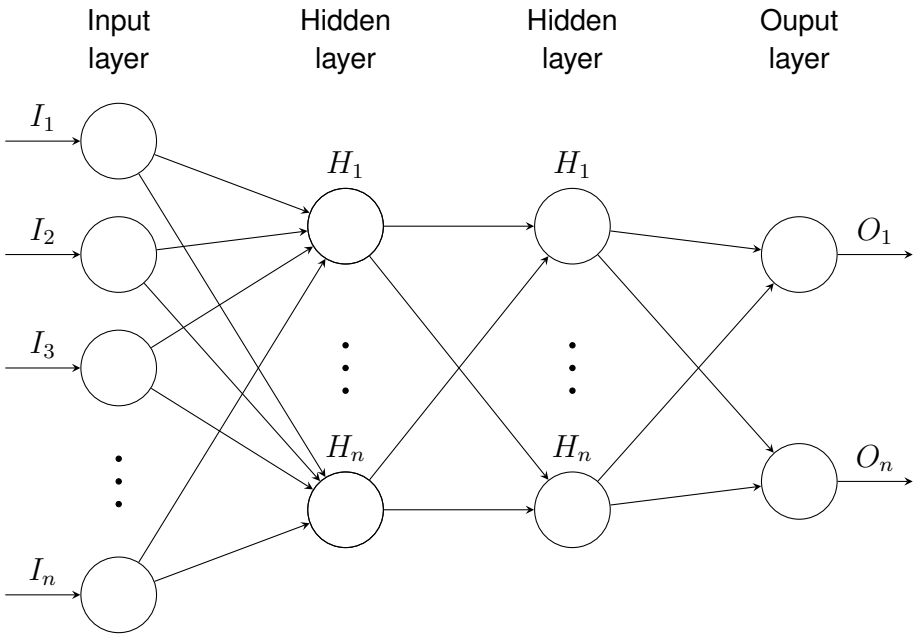


Abbildung 2.2: https://newbiettn.github.io/2016/12/16/tikz/

In Abbildung 2.2 ist Grundstruktur eines neuronalen Netzes veranschaulicht. Neuronen sind hier als Kreise dargestellt und bilden in den beispielhaft vertikal veranschaulichten Formationen einzelne Schichten. Hierbei wird zwischen Eingabe-, Zwischen- und Ausgabeschichten unterschieden. Die Eingabeschicht nimmt Informationen in Form von Daten auf und gibt diese an die erste Zwischenschicht weiter. Die Anzahl der Zwischenschichten, oder auch verdeckte Schichten, ist in der Anwendung der neuronalen Netzen variabel. Am rechten Bildrand ist die Ausgabeschicht gezeigt, die die entsprechende Ausgabe des Netzes generiert.

2.1.2 Lernprozess

Der Lernprozess von neuronalen Netzen zielt darauf hinaus, einer Netzstruktur ein gewünschtes Verhalten beizubringen. Genauer sollen die in Kapitel... beschriebenen Gewichte modifiziert werden.

Zunächst wird zwischen drei Lernverfahren unterschieden, dem unüberwachten, dem bestärkenden und dem überwachten Lernen. Beim unüberwachten Lernen erkennt das Netz selbst Muster und Klassen aus der eingegebenen Menge. Anders als beim unüberwachten Lernen, lernt das Netz beim bestärkten Lernen mit einer Rückmeldung. Diese enthält Informationen dar-

über, ob ein errechnetes Ergebnis einer Trainingseinheit richtig oder falsch ist. Das überwachte Lernen setzt eine Trainingsmenge voraus, die neben der Eingabedaten auch das dazugehörige korrekte Ergebnis enthält. So wird in der Vorwärtspropagation durch eine Eingabe eine entsprechende Ausgabe erzeugt und diese mit dem korrekten Ergebnis verglichen. Das KNN wird dann mithilfe des aus dem vorangegangenen Vergleich entstandenen Fehler korrigiert.[1]

Die meist genutzte Form des überwachten Lernens ist die Rückwärtspropagierung (engl. Backpropagation) oder Fehlerrückführung genannt [2]. Auch in dieser Masterarbeit werden Netze mithilfe der Rückwärtspropagierung trainiert und in diesem Kapitel genauer erläutert. Die mathematische Grundlage für dieses Lernverfahren sind Gradientenabstiegsverfahren [1]. Durch die Fehlerrückführung werden Gewichte durch die Ausgabeschicht, die dann als Eingabeschicht genutzt wird, mithilfe des Fehlervektors optimiert [1]. Dafür ist eine Trainingsmenge in Form eines Datensatzes nötig. Im Falle einer Personenerkennung wäre beispielsweise ein Datensatz aus Bildern von Personen eine geeignete Trainingsmenge. Jedes enthaltene Bild verfügt die Trainingsmenge

2.1.3 Evaluation neuronaler Netze

Die Ausgabe von neuronalen Netzen ist probabilistisch und nicht vorhersehbar. Folglich bestehen diverse Metriken für Evaluationen, die derartige Systeme messbar machen. Im Rahmen dieser Masterarbeit wird die Methode *Precision and Recall* verwendet. Da die Erkennung von Personen auf ein Klassifikationsproblem binärer Natur reduziert werden kann, eignet sich die genannte Herangehensweise. Außerdem werden die neuronalen Netze anhand des Top-5 und Top-1 Fehlers verglichen. Die Grundlagen der entsprechenden Metriken werden im Folgenden vermittelt.

Precision and Recall ist ein traditionelles Werkzeug zur Evaluation und Leistungsmessung [3]. Der *Recall*-Wert $r(t)$, oder $TPR(t)$ für *Truepositiverate*, beschreibt die Fähigkeit eines System, tatsächlich positive Stichproben zu erkennen. Angewandt auf die Personenerkennung sind Bilder, auf denen Personen zu sehen sind, als tatsächlich positive Stichproben einzustufen. In Gleichung 2.2 wird der *Recall*-Wert durch eine Division von allen wahren positiven Werten $TP(t)$ und die Anzahl aller tatsächlich positiven Werte n_{pos} berechnet. Die Variable t definiert den eingestellten Schwellwert. Im Sachkontext ist der Wert als Konfidenz zu betrachten, bei der

eine Person als solche klassifiziert wird. Für die Anwendung dieser Methode ist die Kenntnis über negativen Beispielen der Stichprobe nicht notwendig [4].

$$r(t) = TPR(t) = \frac{TP(t)}{TP(t) + FN(t)} \tag{2.2}$$

Der *Precision*-Wert $p(t)$ berechnet sich durch das Verhältnis von allen wahren positiven Werten $TP(t)$ durch alle als positiv bewerteten Beispielen $TP(t)$ und $FP(t)$. Durch diesen Wert wird verdeutlicht, wie gut ein System in der Lage ist tatsächlich wahre Werte von tatsächlich falschen Werten zu unterscheiden.

$$p(t) = \frac{TP(t)}{TP(t) + FP(t)} \tag{2.3}$$

Häufig muss in der Praxis ein Kompromiss zwischen *Precision* und *Recall* gefunden werden. Dies lässt sich anhand eines Beispiel in der Personenerkennung veranschaulichen. Das System zur Erkennung wird beispielhaft mit einem hohen *Precision*-Wert betrieben. Ein damit verbundener, niedriger *Recall*-Wert ist in der Praxis üblich. Dies führt dazu, dass irrelevanter Bildinhalt selten als Person klassifiert wird. Jedoch kommt es eher häufig vor, dass keine Person detektiert wird obwohl eine zu sehen ist. Legt man nun den Fokus auf einen hohen *Recall*, sinkt der *Precision*-Wert. Personen würden dann zwar häufiger als Person klassifiziert werden, jedoch wird irrelevanter Bildinhalt ebenfalls häufig als Person klassifiziert. Je nach Anwendungsfall eines Netzes wird dann der entsprechende Betriebspunkt zwischen *Precision* und *Recall* gewählt.

Eine einfache Betrachtung zur Evaluierung der Qualität hinsichtlich der Genauigkeit eines Netzes liefert der Top-x Fehler. Der Platzhalter x kann zunächst durch eine beliebige Zahl ersetzt werden. In der Praxis hat sich jedoch der Top-5 und der Top-1 Fehler als Vergleich durchgesetzt. Hierbei wird zunächst ein Bild durch ein beliebiges KNN analysiert und über die Ausgabeschicht extrahiert. Als Beispiel sollte sich im optimalen Fall die tatsächliche Klasse des Bildes unter den x wahrscheinlichsten Klassen befinden, die über das Netz ausgegeben wurden. Folglich sagt

der Top-1 Fehler aus, wie oft ein Netz ein eingegebenes Bild falsch klassifiziert hat. In vielen Paper zu neuen, künstlichen, neuronalen Netzen wird so die Genauigkeit mit bereits bestehenden Netzen verglichen.

2.2 Objekterkennung

Bei der visuellen Objekterkennung wird ein Objekt, das auf einem Bild gezeigt ist, mit einer gewissen Wahrscheinlichkeit inklusive der Position in der Abbildung erkannt. Die drei Abstraktionsebenen einer solchen Erkennung unterteilen sich in Bildklassifikation, Objektlokalisierung und semantische Segmentierung ...2014Bild. Letzteres kommt in dieser Arbeit nicht zur Anwendung und wird aufgrund dessen im Folgenden nicht behandelt. Die Bildklassifikation beschreibt eine Zuweisung von Objektkategorien zu einem gegebenen Bild. Mithilfe einer Merkmalsextraktion werden Merkmalsvektoren extrahiert und können so in einem Klassifikator berechnet werden. In den folgenden Kapiteln wird auf die in dieser Arbeit eingesetzten Methoden zur Objekterkennung eingegangen. Hierbei werden insbesondere alternative Verfahren mit modernen state-of-the-art Lösungen zur Objekterkennung gegenübergestellt.

2.2.1 Objekterkennung durch alternative Verfahren

Neben der neuronalen Netzen gibt weitere Methoden zur Objekterkennung. Ein gängiges Verfahren zur Merkmalsextraktion ist das sogenannte Histogram of oriented gradients (HoG) von *Dalal* in Verbindung mit der *Linear Support Vector Machine* (SVM).

Bei diesem Verfahren wird ein Bild in kleine Bereiche, sogenannte Zellen, aufgeteilt [5]. Für jede Zelle wird ein eindimensionales Histogram extrahiert. Dieses enthält Gradienten die aus den Informationen der Pixel enstehen, wie zum Beispiel durch der Lichtintensität oder der Farbe. Hieraus lassen sich Kanten und Ecken und somit auch Konturen und Muster aus einem Bild erkennen.

Abbildung 2.3: Flower one.

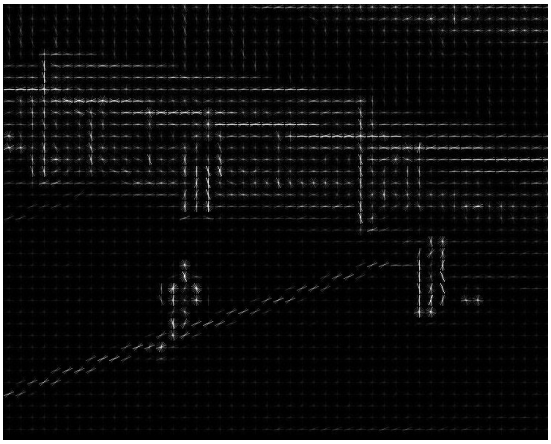


Abbildung 2.4: Flower two.

Die *SVM* ist ein typischer Funktionsapproximator für eine Objektklassifikation. Es handelt sich hierbei um ein mathematisches Verfahren, das Klassen durch Trennungsebenen, die sogenannten Hyperebenen, voneinander trennt. Auch hierbei gibt es ein Lernprozess in Form des überwachten Lernens. Ziel der Algorhitmen des Trainings einer *SVM* ist es, die Hyperebenen so zu konstruieren, dass Objekte sicher klassifiziert werden können. In der Realität lassen sich Objekte rein mathematisch häufig nicht linear trennen. Nicht-lineare Trennbarkeit bedeutet oft einen höheren Rechenaufwand. Somit verwendet die Methode der *SVM* den sogenannten *Kernel-Trick*. Dieser transformiert Daten in eine höhere Dimension, um eine lineare Trennbarkeit zu erreichen. Diese Methode hat sich vor allem aufgrund ihrer kurzen Rechenzeit durchgesetzt.

Die *HoG* Methode in Verbindung mit der *SVM* bringt jedoch auch Nachteile mit sich. Einerseits könnten Objekte problemlos als Person klassifiziert werden, die dieselben Richtungen der Gradienten aufweisen. Anderseits unterscheidet sich das Histogramm einer sitzenden Person zu einer stehenden und würde dann falsch oder gar nicht klassifiert werden.

2.2.2 Objekterkennung durch neuronale Netze

Die bisher besten Ergebnisse in der Bildverarbeitung im Zusammenspiel mit neuronalen Netzen wurden durch *Convolutional Neural Network* (CNN) ermöglicht [6]. Anders als bei den bereits erwähnten Methoden geschieht die Merkmalsextraktion hierbei innerhalb des Netzes. Derartige Netzwerke nutzen Faltung zur Verarbeitung der Eingangsdaten statt der üblichen Matritzenmultiplikation [6]. Der Aufbau eines CNNs setzt sich aus einer Merkmalsextraktion und die darauffolgende Klassifikation zusammen.

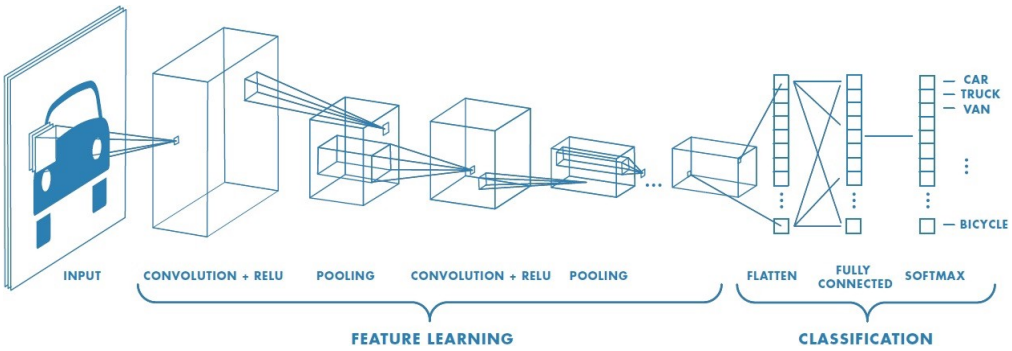


Abbildung 2.5: bla

Eine Einheit der Merkmalsextraktion besteht im grundlegenden Fall aus drei Unterschichten. Dabei können sich diese innerhalb der Merkmalsextraktion hintereinander wiederholen. Dies hat jedoch Einfluss auf die Eigenschaften eines Netzes. Die erste Unterschicht führt Faltungsprozesse mit den Eingangsdaten durch [6]. Im zweiten Schritt wird eine nichtlineare Aktivierungsfunktion wie der *Rectified Linear Unit* (ReLU) Funktion auf die Ausgangsdaten der Kovolutionsschicht angewendet. In der dritten Unterschicht wird das sogenannte *Pooling* durchgeführt. In einigen Fällen wird die Zusammensetzung der drei Stufen als Kovolutionsschicht bezeichnet [6]. Im Laufe dieses Kapitels wird auf die Motivationen und der Funktionsweise eines CNNs eingegangen.

Es gibt drei Motivationen für die Nutzung von Faltung in einem neuronalen Netz. Hierzu gehören die eingeschränkte Konnektivität, die Parameterverteilung und die äquivariante Darstellung [6]. Im folgenden Abschnitt werden diese Punkte näher erläutert.

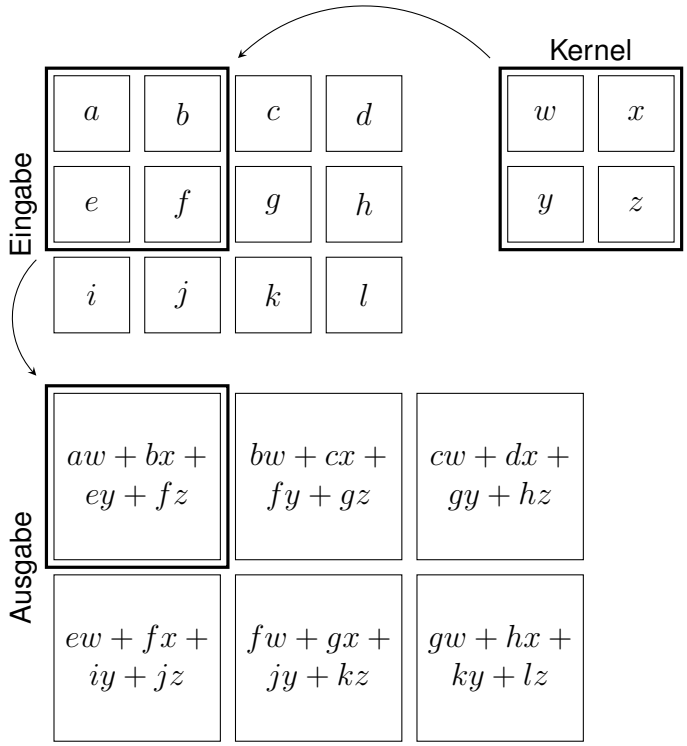


Abbildung 2.6: Prinzipielle Darstellung des Faltungsprozess. Adaptiert aus [6]

Während der Konvolution, oder auch Faltung genannt, werden eingehende Daten in Filter, sogenannte Kernels, eingegeben. Abbildung 2.6 zeigt den prinzipiellen Vorgang der Faltung. Die Konfiguration des Kernels ist beispielhaft mit w, x, y und z dargestellt. Eingehend Datenpunkte sind hier mit den Buchstaben a bis l gekennzeichnet. Die Filter extrahieren bestimmte Merkmale je nach Konfiguration, wie im unteren Teil der Abbildung als Ausgabe dargestellt. So können verschiedene Schichten diverse Merkmale extrahieren. Die Ausgänge der Schichten üblicher, neuronaler Netzen sind mit jedem Eingang der folgenden Schicht verknüpft. Ein typischer Aufbau wurde bereits in Kapitel 2.1.1 in Abbildung 2.2 gezeigt. Wird bei derartigen Netzen eine Schicht mit n Ausgaben und eine mit m Eingaben verknüpft, werden $m \cdot n$ Parameter benötigt [6]. Durch den Faltungsprozess wird diese Konnektivität eingeschränkt. Beispielsweise wird der Datenpunkt b der Eingabe aus der Darstellung 2.6 lediglich in zwei von sechs Datenpunkten der Ausgabe berücksichtigt. Die Ausmaße der eingeschränkten Konnektivität lassen sich in der folgenden Abbildung 2.7 verdeutlichen. Die Eingabepunkte x_n geben je nach Netzart ihre Informationen an alle oder benachbarten Ausgabepunkte s_n weiter. Die Einschränkung der Konnektivität hängt von der Dimension des Kernels ab. Folglich nehmen CNNs deutlich weniger Speicher ein im Vergleich zu herkömmlichen KNNs [6]. Gleichzeitig wird durch die Faltung eine

höhere Statistische Effizient erreicht [6].

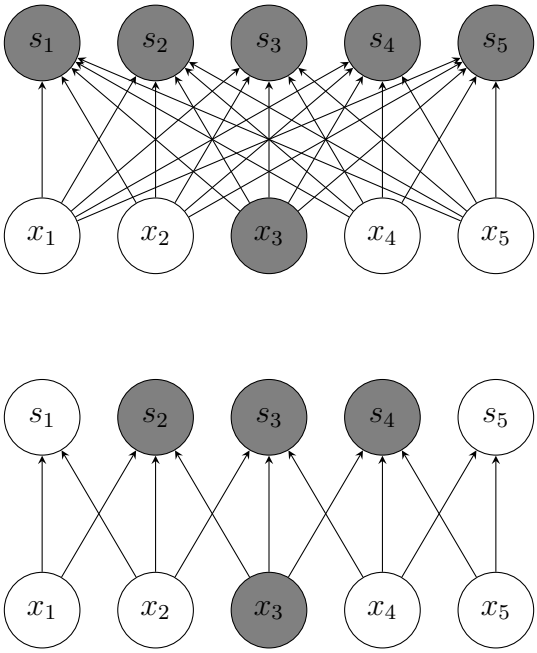


Abbildung 2.7: deep learning

Als Parameterverteilung bezeichnet man die Nutzung eines Parameters pro Funktion eines Netzes [6]. Bei grundlegenden künstlichen neuronalen Netzen wird jede Eingabe eines Neurons, wie in Kapitel 2.1.1, mit einem Gewicht verrechnet [6]. Wie bereits beschrieben, wird bei CNNs ein Kernel pro Schicht für die Merkmalsextraktion verwendet. Dies führt zu deutlich weniger Speicheraufwand, da das Netz lediglich die Kernels abspeichert statt aller Gewichte pro Neuron.

Die äquivariante Darstellung bezieht sich auf die Auswirkung der Ausgabe bei einer Änderung der Eingabedaten. Die Konvolution erzeugt eine zweidimensionale Karte, die sogenannte *Featuremap* in der Merkmale eines Bildes eingetragen sind. Wird ein Objekt in dem eingegebenen Bild des CNNs bewegt, verändert sich die Merkmalskarte im selben Maße. ...

Beim *Pooling* werden die stärksten Merkmale der eingehende Daten weitergegeben [6]. Kleine Änderungen der Eingabewerte haben durch *Pooling* keinen oder einen kleinen Einfluss auf die Ausgabe. Ein Beispiel hierfür liefert die folgende Darstellung 2.8.

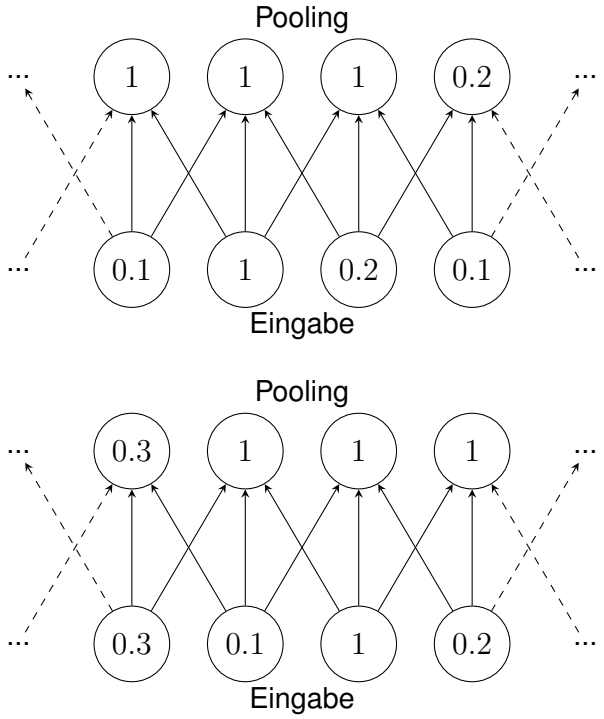


Abbildung 2.8: deep learning

In Abbildung 2.8a werden beispielhaft die stärksten Merkmale der Eingangsdaten durch das *Pooling* weitergeleitet. Obwohl in der unteren Darstellung alle Eingangsdaten um eine Stelle nach rechts verschoben wurden, hält das Verfahren zwei Merkmale konstant. Dies untermauert die Motivation der äquivrianten Darstellung.
/bild Klassifikation

Für die Klassifikation der Daten setzen sich die letzten Schichten meist aus einer oder mehrerer vollständig verbundenen Schichten und einer *Softmax*-Schicht zusammen [6]. Die vollständig verbundene Schicht gibt einen Vektor mit K Elementen aus, wobei K für die Anzahl der ausgegebenen Klassen steht. Eine *Softmax*-Funktion repräsentiert grundlegend eine Wahrscheinlichkeitsverteilung des Eingabevektors K. Durch die *Softmax*-Schicht wird der Vektor in einem Zahlenbereich von Null bis Eins transformiert. Die Summe aller Elemente des Vektors ergeben 1. Jedes Element wird als Konfidenz der jeweiligen Klasse interpretiert. An dieser Stelle sind alle Daten vollständig bearbeitet und werden als Vektor aus dem CNN ausgegeben. In Abbildung 2.9 wird die Funktionsweise eines Faltungsnetzwerks anhand eines Minimalbeispiels dargestellt.

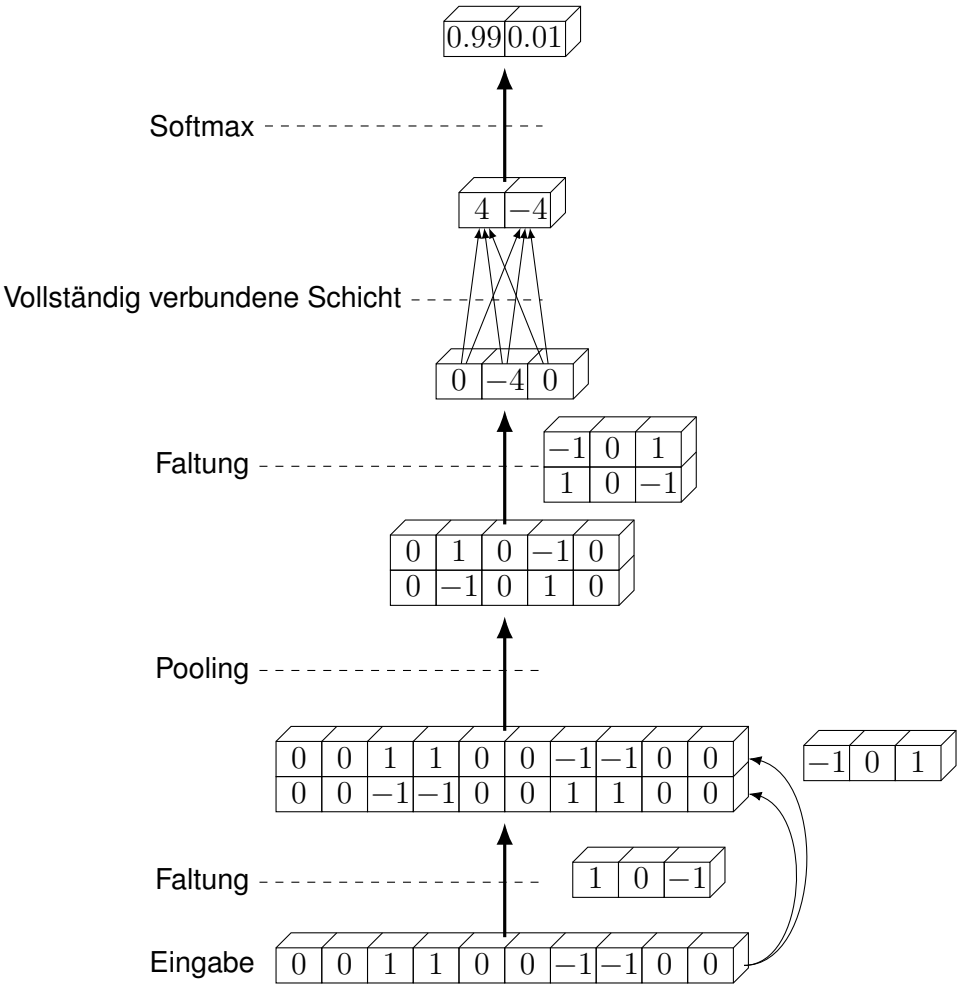


Abbildung 2.9: https://tex.stackexchange.com/questions/519268/write-convolutional-neural-networks-using-tikzg

2.3 Vergleich möglicher Konvolutionsnetze

Die Artenvielfalt der CNNs ist sehr breit gefächert. Jedes Netzwerk unterscheidet sich in der jeweiligen Architektur der Schichten. Durch die Änderung verschiedener Parameter, beispielsweise bei der Faltung oder beim Pooling, können CNNs im Einsatz jeweils anders reagieren. Hierbei entscheidet man bei der Auswahl des Modells häufig unter den Gesichtspunkten Bearbeitungszeit, Genauigkeit und je nach Anwendungsfall spielt der Speicherplatz ebenfalls eine große Rolle.

Auf dem ALF werden je Kamera 10 Bilder in der Sekunde eingelesen. Folglich wird eine Netzarchitektur verwendet, die in der Lage ist mindestens 20 Bilder in der Sekunde zu verarbeiten. Für eine zukünftige Auslagerung auf ein eingebettetes System soll das neuronale Netz nach den dafür notwendigen Aspekten entwickelt werden. Für die Auswahl der für diese Masterarbeit entsprechenden Architektur werden state-of-the-art Lösungen hinzugezogen. In *Canzianis* Paper [7] werden bekannte CNNs nach der Genauigkeit über die Anzahl der Rechenoperationen pro eingegebenes Bild auf einer Grafik aufgetragen. Als dritte Eigenschaft ist dort ebenfalls die Größe der Architekturen als Parameteranzahl gezeigt. Die höchsten Genauigkeiten erzielten hierbei die *ResNet*, *Inception* und *VGG* Architekturen. *Canziani* veröffentlichte sein Paper im Jahr 2016. Ein Jahr später entwickelte *Howard* [8] die *Mobilenet* Architektur. Sie zeichnet sich durch die ihre Schnelligkeit bei teilweise höherer Genauigkeit im Vergleich zu bekannten Architekturen aus. Insbesondere soll das ALF Personen auf Bildern erkennen und lokalisieren können. Die genannten Konvolutionsnetze sind in ihrer Grundform lediglich in der Lage Bilder zu klassifieren. Für die Umsetzung einer vollständige Objekterkennung werden die gängigen Lösungen *R-CNN* und *SSD* präsentiert.

VGG

Die VGG Architektur wurde im Jahr 2015 von *Simonyan* und *Zisserman* [9] vorgestellt. Oft wird die die Bezeichnung *VGG-xx* verwendet, wobei *xx* für die Anzahl der Konvolutionsschichten in Addition mit den vollständig verbundenen Schichten steht. Im Vergleich zur *Alexnet* Architektur werden hierbei ausschließlich sehr kleine 3×3 Filter genutzt [10] [9]. Im Gegenzug wurde die Tiefe der Netze erhöht. In dem Paper sind Netze bis zu einer Tiefe von 19 Schichten untersucht worden[9]. Es hat sich herausgestellt, dass die Veränderung der Tiefe eines Netzes über die Genauigkeit der Klassifikation bestimmt. So ist das Netz in der Lage sehr hohe Genauigkeiten zu erzielen. Wiederum hängt die Anzahl der Schichten direkt mit dem Speicher- und dem Rechenaufwand eines Netzes zusammen. Ein typisches *VGG-16* Netz benötigt circa 530 MB wegen seiner 138 Millionen Parameter und ist aufgrund dessen für die bereits erwähnte Anwendung an dem ALF nicht geeignet.

ResNet

ResNet steht für *residual network* und wurde erstmals im Jahr 2015 durch das Paper von *He* veröffentlicht. Diese Architektur fällt besonders durch Verbindungen auf, die es Bildinformationen ermöglicht Schichten zu überspringen. So sind tiefe Zwischenschichten nicht nur von der Ausgabe der vorherigen Schicht abhängig. In Abbildung **??** ist eine Prinzipdarstellung der Architektur gezeigt. Hierbei wird eine allgemeine Eingabe x in beliebig viele Zwischenschichten eingegeben, die eine Ausgabe $F(x)$ erzeugt. Außerdem wird x durch eine Verbindung mit $F(x)$ addiert.

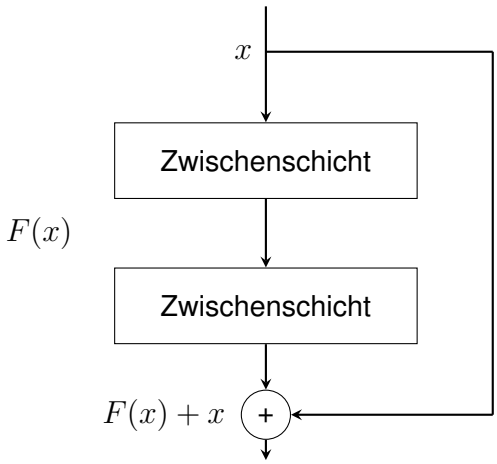


Abbildung 2.10: Prinzipielle Darstellung des Faltungsprozess. Adaptiert aus [6]

ResNet Netze erreichen je nach Größe eine höhere Genauigkeit als die bereits erwähnten *VGG* Architekturen. Die Größe der Netze erreichen hierbei 152 Schichten. Trotz der entsprechenden Tiefe enthält eine derartiges CNN circa die Hälfte der Parameter im Vergleich zu *VGG* Netzen. Dementsprechend liegt der Speicherbedarf bei circa 230 MB. Für den Einsatz auf eingebetteten Systeme würde ein Netz dieser Größe zu viel Rechenkapazität in Anspruch nehmen, sodass die verarbeitung eines Bildes unter Umständen mehrere Sekunden dauern könnte.

Inception/GoogleNet

Gemessen an der Genauigkeit liegen *Inception/GoogLeNet* Netze höher als *VGG* und *ResNet* Architekturen. *Szegedy* veröffentlichte die erste Version der *Inception* CNNs in seinem Paper

[11] im Jahr 2015. Die Besonderheit hierbei ist der Einsatz von multiplen Kernels auf die jeweiligen Schichten. Durch diesen Aufbau benötigt ein solches Netz keine vollständig verbundene Schicht zur Klassikation, da die Genauigkeit nur geringfügig beeinträchtigt wird. Gleichzeitig wird ein Großteil der Parameter eingespart. Bei *VGG* Architekturen befinden sich beispielsweise circa 90$\%$ aller Parameter in den vollständig verbundenen Schichten am Ende des Netzes. Das klassische *Inception* Architektur enthält insgesamt 27 Schichten, 22 davon enthalten Parameter [11]. Somit sind übliche *Inception* Netze circa 90 MB groß und verfügen über 23 Millionen Parameter.

Mobilenets

Zu den bekanntesten state-of-the-art Lösungen gehört die *MobileNet* Architektur aus dem Jahr 2017. *Howard* beschreibt in seiner Paper, dass das Netz effizient hinsichtlich des Zusammenspiels zwischen Geschwindigkeit und Genauigkeit agiert. Anders als bei den bisher genannten Methoden besitzen einige Kernels der Konvolutionsschichten eine dritte Dimension. Diese sind somit in der Lage tiefenorientierte Faltungsprozesse durchzuführen. Weiterhin wird der Rechenaufwand durch sogenannte Weiten- und Auflösungsmultiplikatoren reduziert. In diesem Kapitel wird näher auf die Funktionsweise eines *MobileNet* CNNs eingegangen.

Ein *MobileNet* Modell basiert grundlegend auf tiefenorientierte, trennbare Konvolution [8]. Diese setzt sich aus einer Tiefenkonvolution und einer 1×1 Faltung, auch punktuelle Konvolution genannt, zusammen [8]. Der Rechenaufwand R einer konventionellen Konvolutionsschicht kann mit Gleichung 2.4 beschrieben werden. Hierbei wird die Dimension der Eingangsdaten durch D_F und die Tiefe durch M ausgedrückt. Ein übliches RGB-Bild besitzt beispielsweise die Tiefe .... Die Größe des Kernels wird in der Gleichung durch die Variable D_K, sowie die Tiefe der Ausgangsdaten mit N dargestellt. Im Gegensatz zu üblichen Faltungsprozessen werden die Eingangskanäle während einer tiefenorientierten Konvolution lediglich gefiltert und nicht zusammengeführt [8]. Somit ist die Tiefe der Ausgangsdaten für die Berechnung des Rechenaufwands bei einer tiefenorientierten Faltung zu vernachlässigen und ist in Gleichung 2.5 gezeigt.

$$R_p = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \tag{2.4}$$

$$R_d = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \tag{2.5}$$

Durch die Zusammensetzung einer tiefenorientierten und einer punktuellen Faltung wird die Rechenbelastung der tiefenorientierte, trennbare Konvolution durch die in Gleichung 2.8 gezeigte Addition errechnet. Die Kombination der beiden Methoden reduziert den Rechenaufwand um den in Gleichung 2.7 dargestellten Faktor $\frac{1}{N} + \frac{1}{D_K^2}$ [8]. In Zahlen ausgedrückt erreicht die *MobileNet* Architektur eine Reduktion der Rechenoperationen um den Faktor 8 bis 9 gegenüber konventionellen Faltungsmethoden [8].

$$R_{dp} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \tag{2.6}$$

$$R_{rdp} = \frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \tag{2.7}$$

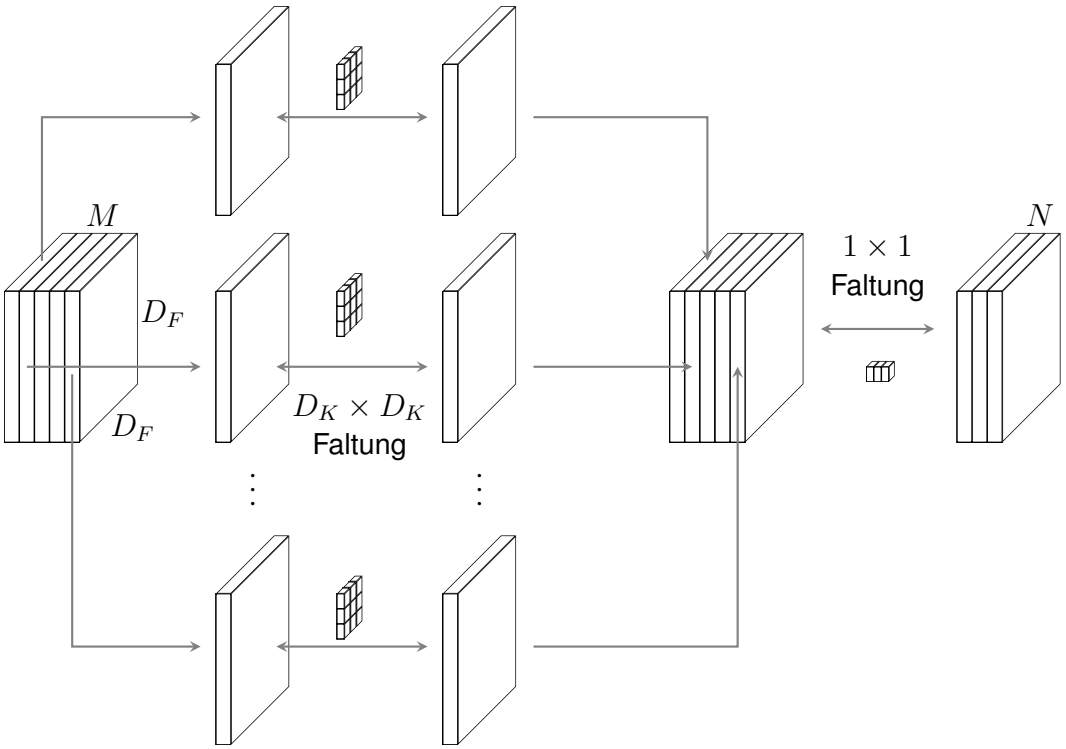


Abbildung 2.11: dddddd

Weiterhin nutzt *MobileNet* Weiten- und Auflösungsmultiplikatoren für die Optimierung des Netzes. Der Weitenmultiplikator α reduziert sowohl die Tiefe M der Eingangsdaten durch αM, als auch die Tiefe N der Ausgangsdaten durch αN. Eine Minimierung der Dimension D_F der Eingangsdaten erfolgt durch den Auflösungsmultiplikator ρ durch ρD_F. Beide Multiplikatoren reduzieren den Rechenaufwand nochmals um α^2 und ρ^2. Dieser kann somit mit folgender Gleichung dargestellt werden.

$$R_{dpm} = D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F \text{ mit } \alpha \in (0, 1]; \rho \in (0, 1] \quad (2.8)$$

Howard vergleicht in seinem Paper die *MobileNet* Architektur mit anderen Netztypen, wie zum Beispiel *Inception* oder *VGG 16*. Bei dem Vergleich ist zu erkennen, dass die verwendeten *MobileNet* CNNS deutlich weniger Parameter nutzen als übliche Netze. Trotzdem wurden lediglich

minimale negative Auswirkung bezüglich der Genaiugkeit festgestellt. Gleichzeit ist das Netz um ein Vielfaches schneller als herkömmliche Architekturen mit hoher Genauigkeit. Zudem bewirkt die Einsparung der enthaltenen Parameter einen geringeren Speicheraufwand.

R-CNN

In Kapitel 2.2 wurde die Aufgabe der Objekterkennung mit damit verbundenen Ausgabe diskutiert. Die bisher präsentierten Lösungen sind ledglich in der Lage eine entsprechende Einghabe zu klassifizieren. Sie können jedoch nicht Objekte in einem Bild lokalisieren. R-CNN steht für *Region-based Convolutional Neural Network* und wurde 2014 im Paper von *Girshick* erstmals vorgestellt [12]. *R-CNN* bietet die Komplettlösung zur vollständigen Objekterkennung eines Bildes. Mittlerweile gibt es durch diverse Modifikationen dieser Methode die Erweiterungen *fast R-CNN* und *faster R-CNN* [13] [14]. Alle zielen darauf hinaus eine Echtzeitanalyse durchzuführen.

In der grundlegendsten Form besteht ein derartiges Netz aus drei Modulen. Das erste Modul stellt kategorieunabhängige Lokalisierungsvorschläge bereit in denen sich Objekte befinden könnten. Danach folgt ein Faltungsnetzwerk zur Merkmalsextraktion. In den drei Entwicklungsstufen des *R-CNN* wird die *VGG* Architektur verwendet. Die extrahierten Merkmale werden dann im dritten Modul mithilfe des in Kapitel 2.2.1 erwähnten, linearen SVM klassifiziert.
/eventuell Bild des Prozesses wie im Paper rcnn

SSD

Der *Single Shot Multibox Detector* (SSD) bietet ebenfalls eine Komplettlösung zur Objekterkennung. Anders als bei *R-CNN* wird beim *SSD* kein herkömmlicher Detektor als letztes Modul verwendet. Vielmehr handelt es sich hierbei um ein vollständiges CNN.

Im November 2016 präsentierte *Liu* in seinem Paper den *SSD*, modifiziertes *VGG-16* Netzwerk. Hierbei wurden die vollständig verbundenen Schichten des *VGG-16* Netzes entfernt und durch Konvolutionsschichten ersetzt. Im erwähnten Paper wird ausdrücklich darauf hingewiesen, dass es nicht ausgeschlossen sei andere Architekturen als Basis zu nutzen. Der Name unterteilt sich in *Single Shot*, *Multibox* und *Detector*. Anhand dessen lässt sich die Funktionsweise

dieser Methode erklären. *Single Shot* drückt aus, dass die Architektur mit nur einer Bildanalyse auskommt und die Objekterkennung durchführen kann. Die *Multibox* Methode wurde von *Szegedy* entwickelt [15]. Diese gibt klassenbasierte Vorschläge zur Objekterkennung in Form von Begrenzungsrahmen aus. Der Aufbau ähnelt in der Grundidee der bereits erwähnten *Inception* Architektur. Die *Multibox* Methode hat einen großen Einfluss auf die Schnelligkeit der gesamten *SSD* Architektur.

2.4 Zustandsautomat

Die Idee der Nutzung eines Zustandsautomaten oder auch endlicher Automat (EA) ergab sich im Laufe der Entwicklungsphase. In der in Kapitel ... erwähnten Bachelorarbeit werden diverse Modi beschrieben, die den Aufruf von unterschiedlichen ROS-Knoten vorraussetzen. Aufgrund der Analogie zwischen den beschriebenen Modi und der Zustände eines Zustandsautomaten wird die Nutzung eines solchen Automaten begründet. Im Folgenden wird auf die Eigenschaften eines endlichen Automats eingegangen.

Im Allgemeinen geht es bei einem Zustandsautomaten um die Beschreibung der Zustände (engl. States) eines Objekts. Dabei stellt das Objekt meist das Gesamtsystem dar, etwa ein Getränkeautomat oder wie dieser Arbeit autonomes Fahrzeug. States sind durch Bedingungen verknüpft und lösen während sogenannter Ereignisse eine Transition aus, die den Wechsel des Zustands ausübt. Weiterhin Bilden die Zustände in ihrer Gesamtheit den Lebenszyklus des Objekts. Ein Getränkeautomat befindet sich bekanntermaßen beim Eintreffen eines Kunden in einer Art Bereitschaft. Übertragen auf die Theorie eines Zustandsautomaten wäre dies ein Bereitschaftszustand. Die Auswahl des Getränks und die Eingabe des entsprechenden Geldbetrags können beispielhaft als Ereignisse interpretiert werden. Somit wird ein Transition durchgeführt und der Zustand der Getränkeausgabe wird losgetreten. Wurde das Getränk ausgegeben und entnommen, geschieht der Wechsel in den Bereitschaftszustand und der beschriebene Zyklus ist komplettiert.

Seit dem Bestehen der endlichen Automaten haben sich in der Praxis zwei Typen durchgesetzt. Mealy und Moore Automaten unterscheiden sich grundlegend in ihrem Verhalten und können durch folgende Gleichungen beschrieben werden.

$$\alpha_{t+1} = \phi(\zeta_t, \alpha_t) \text{ mit t} \in \mathbb{N} \tag{2.9}$$

$$\gamma_t = \psi(\zeta_t, \alpha_t) \text{ mit t} \in \mathbb{N} \tag{2.10}$$

In den Gleichung ... beschreibt φ die Transitionsfunktion und ψ die Ausgabefunktion des Mooreautomats. Die Transition steht in Abhängigkeit von ζ_t, die aktuelle Eingabe, und α_t, der aktuelle Zustand selbst. Mithilfe der Transitionsfunktion lässt sich der Zustand bestimmen, der im folgenden Zeitschritt t angestrebt werden soll. Der Ausgang des Moore Automaten wird durch die Ausgangsfunktion $psiup$ berechnet. Diese hängt genau wie die Transitionsfunktion von der Eingabe und dem Zustand zum Zeitpunkt t ab.

$$\alpha_{t+1} = \phi(\zeta_t, \alpha_t) \text{ mit t} \in \mathbb{N} \tag{2.11}$$

$$\gamma_t = \psi(\alpha_t) \text{ mit t} \in \mathbb{N} \tag{2.12}$$

Beim Vergleich der beiden Gleichungen ... und ... stellt sich heraus, dass die Ausgangsfunktion psi in der Beschreibung des Verhaltens eines endlichen Automaten durch Moore lediglich vom Ausgang zum Zeitpunkt t abhängig ist.
Eine Unterkategorie der Finiten Automaten ist der Hierarchische Zustandsautomat. Die Besonderheit hierbei ist die Zusammensetzung aller vorangegangenen Zustände eines aktiven Zustands. Diese sind bei der hier beschriebenen hierarchisch aufgebauten Maschine nämlich ebenfalls aktiv. So besteht die Möglichkeit eines aufeinander aufbauenden Endzustands.

2.5 Bestimmung von Positionskoordinaten

Während der Durchführung autonomer Fahr- bzw. Logistikaufgaben können diverse Probleme auftreten, die eine erfolgreiche Bearbeitung verhindern können. Beispielsweise können Türen geschlossen sein oder Gegenstände die geplante Route blockieren. Da das ALF nicht über die

technischen Möglichkeiten besitzt derartige Problemstellungen zu lösen, müssen Menschen Abhilfe schaffen. Für diese Zwecke ist die Kenntnis über die letzte Position der erfassten Personen realtiv zur statischen Karte notwendig. Anstehende Fahraufgaben werden, bedingt durch das Vorgängerprojekt, mithilfe des Robot Operating Systems gelöst. Personen können folglich als Position in das ROS Netzwerk veröffentlicht. Dies ermöglicht dem Roboter die veröffentlichten Positionen anzufahren. Die Eintragung der Position in die statische Karte setzt die Beschreibung der Position als Koordinaten vorraus. Für die Bestimmung der Positionskoordinaten wird ein zweidimensionales Bild und die dazugehörigen Tiefeninformationen genutzt. Die Koordinate xloc beschreibt hier die longitudinale Entfernung von der Kamera zur Person. Eingehende laterale Distanzen werden durch die Koordinate yloc dargestellt.

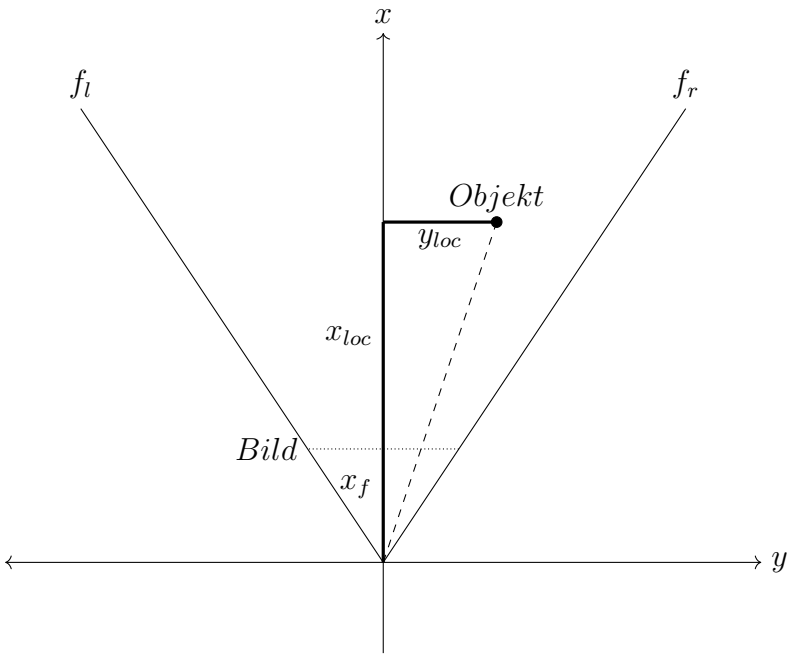


Abbildung 2.12: (a) Die Abbildung zeigt die Übergangsfunktion $h(t)$ eines Verzögerungsglieds erster Ordnung als Antwort auf die sprungförmige Eingangsgröße $u(t)$ mit $\hat{u} = 1$. (b) Die Impulsantwort auf das Eingangssignal $\delta(t)$. Zwischen den Funktionen $h(t)$ und $g(t)$ besteht der Zusammenhang $g(t) = \frac{\mathrm{d}}{\mathrm{d}t}\, h(t)$.

2.6 Schnittstelle zwischen ROS und Python

3 Konzeptionierung

3.1 Anforderungserhebung mit CONSENS

3.2 Konzept und Aufbau der Personenerkennung

3.2.1 Wirkstruktur der Personenerkennung

3.2.2 Auswahl und Training der verwendeten neuronalen Netze

Im diesem Kapitel werden Vergleiche zwischen den genannten Netzen gezogen und eine Auswahl für die praktische Anwendung am ALF getroffen.

Tabelle 3.1: Default Table

Eigen-schaften	Netze			
	VGG-16	ResNet	Inecption	MobileNet
Parameter	27.54	13.23	6.80	6.80
Eingabe	29.55	9.61	6	6.80
Top-5 error	25.99	11.49	6.15	6.80

Hinsichtlich der genannten Aspekte wird im Zuge dieser Arbeit ein KNN mit der MobileNet Architektur und dem *Single Shot Detector* (SSD) Framwork verwendet. Die Grundlage der Auswahl wird mithilfe der Paper *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* [8] und *The Implementation of CNN-based Object Detector on ARM Embedded Platforms* [16] begründet.

Die *MobileNet* Architektur ermöglicht im Vergleich zu anderen Lösung zur Merkmalsextraktion den gesuchten Spagat zwischen Schnelligkeit, Genauigkeit und Größe des Netzes [8]. Das Hauptmerkmal ist der Aufbau der Konvolutionsschichten, der zu den genannten Eigenschaften in der Praxis führt [8] [16]. Vergleichsmessungen zu anderen Architekturen werden in den erwähnten Paper durchgeführt. Der *Single Shot Detector* dient als Klassifikator und zeichnet sich besonders durch seine Schnelligkeit aus [17]. Er erreicht eine höhere Genauigkeit als andere state-of-the-art Lösungen wie zum Beispiel *Faster R-CNN* [14] und ist zudem dreimal schneller[17].

3.2.3 Entwickeltes neuronales Netz

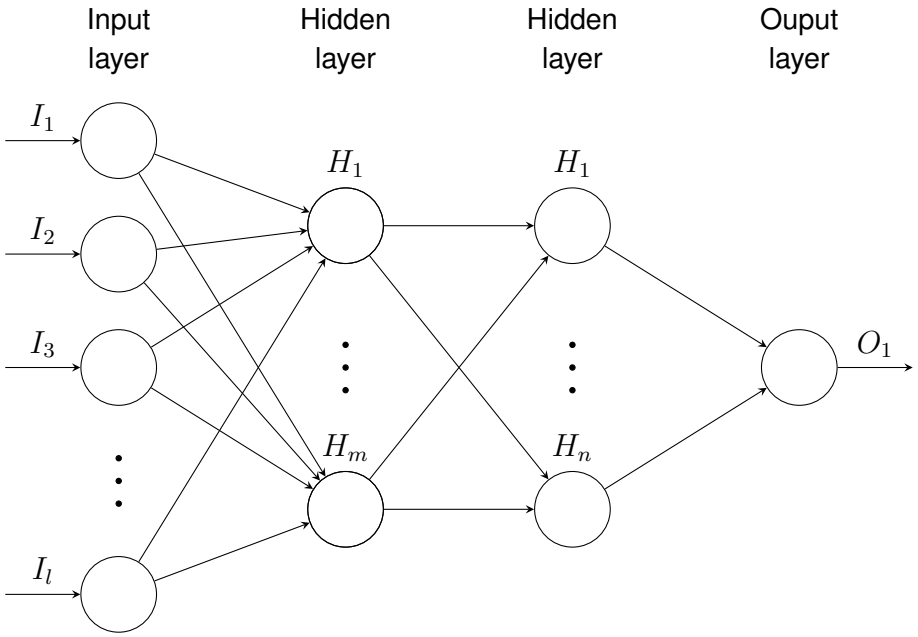


Abbildung 3.1: https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

3.2.4 Schnittstelle zwischen Python und ROS

3.2.5 Erstellung von Objektinformationen

3.3 Funktionsweise des Gesamtsystems

Der Kern dieser Arbeit ist die Personenerkennung im praktischen Kontext des im Kapitel ... beschriebenen autonomen Logistikfahrzeugs. Aufgrund dessen sind einzelne Programme nicht als abgeschlossenes System zu betrachten. In Kapitel .../consens wurden bereits alle Schnittstellen zu verbauten Hardware- und Softwarekomponenten präsentiert. Das vollständige System der Personenerkennung und der Aufbau des entwickelten, endlichen Automats wird im folgenden Kapitel erklärt.

Die Personenerkennung am ALF wird mithilfe der Bildinformationen von zwei *Kinect*-Kameras betrieben. In Kapitel.../rospython wurden bereits zwei Lösungsansätze in der Softwarentwicklung in Zusammenspiel mit ROS und Python präsentiert. Aufgrund einer starken Belastung der im Roboter verbauten Recheneinheit während der parallelen Bildverarbeitung beider eingehender Bilder, wurde das Programm auf eine serielle Verarbeitung umgestellt. Der Befehl der parallelen Verarbeitung erzwingt Berechnungsprozesse mit derselben Frequenz, die durch den Eingang der Bilder beider Kameras vorgegeben wird. Mithilfe der seriellen Abarbeitung war es ebenfalls möglich die Häufigkeit der Berechnungen zu steuern und damit verbundene Programmoptimierungen vorzunehmen. Auslastungen des verbauten Computers können so eingespart werden und für weitere, parallel laufende Prozesse genutzt werden. Dies wird zum Beispiel durch eine gezielte Verzögerung der Personenerkennung erreicht. Hierbei werden Pausen mit der gewünschten Dauer zwischen Bildverarbeitungsprozessen eingelegt bis ein relevantes Bild erkannt wird. Erst dann arbeitet die Personenerkennung mit der maximalen Geschwindigkeit. Als relevant werden Bilder eingestuft, die eine Person enthalten.

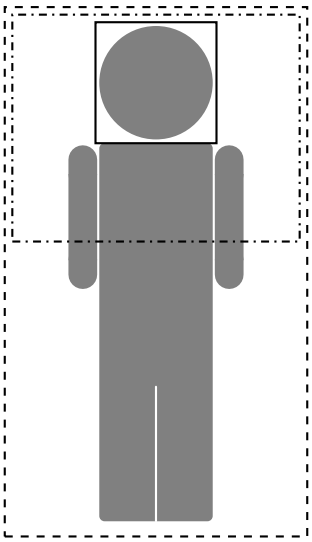
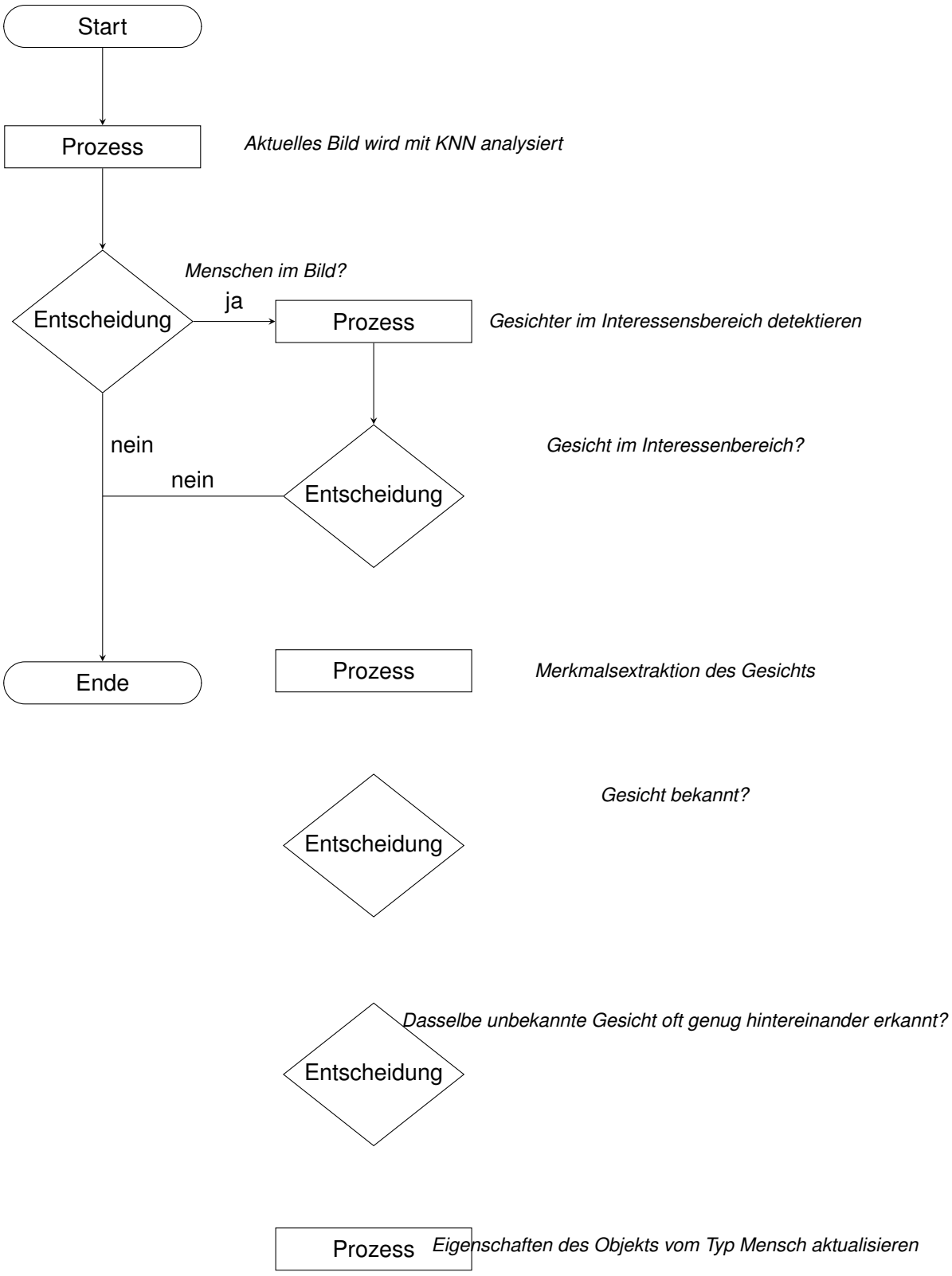


Abbildung 3.2: test

In Abbildung 3.2 wird der Ablauf der Personenerkennung in Form eines Programmablaufplans dargestellt. Die Darstellung zeigt die Funktionsweise des Programms ab dem Zeitpunkt, an dem eine Person vor der jeweiligen Kamera detektiert wird. Weiterhin zeigt die Abbildung den Informationsfluss eines Bildes von einer Kamera durch die Personenerkennung. Zu Beginn der Analyse gelangt jedes Bild zunächst in das eingestellte künstliche neuronale Netz. Je nach Anzahl der erkannten Person werden korrespondierende Koordinaten ausgegeben, die die Position des Interessensbereich beschreiben. Anhand dieser Informationen kann an eine Aussage darüber getroffen werden, ob eine Person im Bild zu sehen ist und wo sich diese befindet. Wird keine Person erkannt arbeitet das Programm wieder reduziert, wie bereits beschrieben. Sollten jedoch Personen erkannt worden sein, wird versucht ein Gesicht zu erkennen. In den meisten Fällen sitzen und stehen Menschen aufrecht. So kann davon ausgegangen werden, dass sich das Gesicht einer Person im oberen Teil des Bereichs befindet. Dafür wird der Interessensbereich verkleinert, um Rechenkapazitäten einzusparen. Sollte sich kein Gesicht im relevanten Bereich befinden, wird davon ausgegangen, dass die Person stark von der Kamera abgewandt ist. Somit ist keine eindeutige Identifikation möglich und das Programm schaltet in den reduzierten Modus. Detektiert das Netz ein Gesicht wird eine Merkmalsextraktion durchgeführt.

Im Anschluss werden die extrahierten Merkmale mit den der bereits abgespeicherten Gesichtern verglichen. Wird kein übereinstimmendes Gesicht gefunden versucht die Software das

Gesicht zu registrieren. Je nach Einstellung wird ein Gesicht registriert, wenn es entsprechend oft hintereinander erkannt worden ist. Der Registrierungsprozess und die Sammlung von Bildinformationen wird in Kapitel 3.2.5 behandelt. Im Falle der Erkennung eines bekannten Gesichts wird die Informationen der erkannten Person wie im bereits erwähnten Kapitel aktualisiert.


Start
Prozess
Aktuelles Bild wird mit KNN analysiert
Menschen im Bild?
Entscheidung
ja
Prozess
Gesichter im Interessensbereich detektieren
nein
Gesicht im Interessenbereich?
nein
Entscheidung
Ende
Prozess
Merkmalsextraktion des Gesichts
Gesicht bekannt?
Entscheidung
Dasselbe unbekannte Gesicht oft genug hintereinander erkannt?
Entscheidung
Prozess
Eigenschaften des Objekts vom Typ Mensch aktualisieren

3.4 Umsetzung der Statemachine

Für die Steuerung des autonomen Logistikfahrzeugs durch eine Spracherkennung wurde ein endlicher Automat entwickelt. Die Entwicklung der Spracherkennung wird in der Masterarbeit von Hannes Dittmann erläutert. In der vorangegangenen Bachelorarbeit werden die Fahrfunktionen des Roboters erklärt. Der Grundgedanke und die daraus resultierende Auswahl des Zustandsautomats wird im folgendem Kapitel näher ausgeführt.

3.4.1 Auslegung des Zustandsautomats

Die Problematik der Steuerung über Sprache ist die Extraktion der eigentlichen Aussage eines Satzes. In der Masterarbeit von Hannes Dittmann werden aufgrund dessen Sprachbefehle kategorisiert. Die KI ist in der Lage verschiedene Sätze einer für den Roboter relevanten Kategorie zuzuordnen. Diese werden als Eingabeparameter für den EA genutzt. Weiterhin wird zwischen einer manuellen und einer autonomen Fahraufgabe unterschieden und als zweiten Parameter für den Zustandsautomaten genutzt. Anhand der technischen Fähigkeiten des Roboters wurden die Kategorienamen so gewählt, dass alle möglichen Handlungen abgedeckt sind. Der Zustandsautomat wurde so entworfen, dass er trotz der vielen Handlungsmöglichkeiten des Roboters mit möglichst wenig Zuständen arbeitet. Für die größtmögliche Effizienz hinsichtlich der Dimension des EAs wurde ein mathematisches Modell entworfen.

$$\vec{z} = \sum_{z=0}^{z_f} \begin{bmatrix} k_n \\ k_{n+1} \\ . \\ . \end{bmatrix} \circ \begin{bmatrix} b_n(z) \\ b_{n+1}(z) \\ . \\ . \end{bmatrix} \circ \begin{bmatrix} 1 \\ 1 \\ l(z_f) \\ l(z_f) \\ 1 \\ f_m \\ f_a \\ l(z_f) \cdot f_a \\ l(z_f) \cdot f_a \end{bmatrix} \tag{3.1}$$

$$b_n(z) = \begin{cases} 1 \text{ für } z = n \\ 0 \text{ für } z \neq n \end{cases} . \tag{3.2}$$

Dies ermöglicht einen sich aufbauenden Endzustand in Form eines Zustandsvektors $\vec{z}$ und wird im folgenden Beispiel anhand der Gleichung 3.1 und 3.2 erläutert. Es wird angenommen, dass der Endzustand $z_f = 6$ erreicht werden soll. Der sechste, finale Zustand ist die manuelle Steuerung des Roboters über den Controller. Durch die Variable z werden Zustände beschrieben, die vor Erreichen des Endzustands durchlaufen werden müssen. Jeder Zustand z verfügt über eine eigene Knotenmenge k_n, die in Form von ROS-Knoten aufgerufen werden. Die Hilfssunktion $b_n(z)$ ist eine binäre Funktion und aktiviert im jeweiligen Index der Summe die benötige Knotenmenge k_n

In diesem Zusammenhang kam während der Entwicklung die Idee einer Fusion der Personen- und Spracherkennung in der Praxis auf.

4 Verifikation

5 Zusammenfassung und Ausblick

Quellenverzeichnis

[1] Kriesel, D. *Ein kleiner Überblick über Neuronale Netze*. 2005.

[2] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz*. 3. Auflage. Springer Vieweg, 2013.

[3] Ben Carterette. *Encyclopedia of Database Systems - Precision and Recall*. Springer US, 2009, S. 2126–2127.

[4] Süße, H. und Rodner, A. *Bildverarbeitung und Objekterkennung - Computer Vision in Industrie und Medizin*. Springer Vieweg, 2014.

[5] Dalal, N. und Triggs, B. *Histograms of Oriented Gradients for Human Detection - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005.

[6] Goodfellow, I. , Bengio, Y. und Courville, A. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[7] Alfredo Canziani, Adam Paszke und Eugenio Culurciello. *An Analysis of Deep Neural Network Models for Practical Applications*. 2016.

[8] Howard, A. G. , Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. und Adam, H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Apr. 2017.

[9] g. *g*. 2015.

[10] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks - Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, S. 1097–1105.

[11] C. Szegedy u. a. *Going deeper with convolutions - 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, S. 1–9.

[12] 2015, S. 1–9.

[13] Ross Girshick. *Fast R-CNN - Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[14] Shaoqing Ren u. a. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, S. 91–99.

[15] Christian Szegedy u. a. *Scalable, high-quality object detection*. 2015.

[16] Zhang, Y., Bi, S., Dong, M. und Liu, Y. *The Implementation of CNN-based Object Detector of ARM Embedded Platforms. 2018 IEEE 16th Int. Conf. on Dependable, Autonomic & Secure Comp., 16th Int. Conf. on Pervasive Intelligence & Comp., 4th Int. Conf. on Big Data Intelligence & Comp., and 3rd Cyber Sci. and Tech. Cong.* 2018.

[17] Liu, W. , Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. und Berg, A. C. *SSD: Single Shot MultiBox Detector. European conference on computer vision*. pp. 21-37. 2016.

A Anhang

A.1 Abbildungen

A.2 Inhalt Datenträger

1. Datenblätter
2. Programm
3. Lastenheft