Smart Mechatronics

HOCHSCHULE HAMM-LIPPSTADT

Hochschule Bochum
Bochum University
of Applied Sciences

Masterarbeit

Zur Erlangung des akademischen Grades
Master of Engineering (M. Eng)

Entwicklung und Verifikation einer Sprachverarbeitung für das autonome Logistik-Fahrzeug ALF

Autor: B.Eng Hannes Dittmann
hannes.dittmann@stud.hshl.de
Matrikelnummer: 2199809

Erstgutachter: Prof. Dr.-Ing. Mirek Göbel

Zweitgutachter: Dr.-Ing. Christoph Termühlen

Abgabedatum: 20.10.2020

Danksagung

Ein großes Dankeschön gilt all jenen Personen, die mich im Rahmen dieser Masterarbeit begleitet haben. Ganz besonderen Dank möchte ich Herrn Prof. Dr.-Ing. Mirek Göbel für die Betreuung in dieser Konstellation aussprechen. Weiterhin gilt mein Dankeschön für die *Smart Mechatronics* GmbH und Herrn Dr.-Ing. Guido Stollt für die finanzielle Unterstützung während des Studiums und dem entgegengebrachten Vertrauen als Bachelor- und Masterand. In diesem Zuge möchte ich nachdrücklich Herrn Dr.-Ing Christoph Termühlen für all die konstruktiven Vorschläge, Hinweise und kritischen, aber stets fairen Anmerkungen danken. Außerdem gilt mein Dank Herrn M. Sc. Bernd Möllenbeck, welcher diese Arbeit ebenfalls fachlich begleitet hat. Besonderer Dank gilt Herrn Dr.-Ing Arno Bergmann für die ungezwungenen und lehrreichen Gespräche während des Studiums sowie der Betreuung verschiedenster Projekte. Dies hat mir stets große Freude bereitet. Weiterer Dank gilt meinem Kommilitonen Giuliano, mit dem ich den Großteil des Studiums bewältigt habe und eine tiefe Freundschaft pflege. Zu guter Letzt danke ich meiner Familie und dabei besonders Jochen und Anne für die Unterstützung jeglicher Art auf meinem bisherigen Lebensweg.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ALF	Autonomes Logistik-Fahrzeug
BOW	Bag Of Words
CONSENS	Conceptual Design Specification Technique for the Engineering of Complex Systems
DFT	Diskrete Fourier-Transformation
DS	DeepSpeech
FFW	Vorwärtspropagierendes neuronales Netzwerk
FFWE	Vorwärtspropagierendes neuronales Netzwerk mit Embedding-Schicht
MFCC	Mel-Frequency-Cepstral Coefficients
OHE	One-Hot Encoding
PS	Pocketsphinx
ReLu	Rectified Linear Unit
RNN	Rekurrentes neuronales Netzwerk
RNNE	Rekurrentes neuronales Netzwerk mit Embedding Schicht
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
STFFT	Short-Time Fast Fourier-Transformation
WE	Word Embedding

Symbolverzeichnis

Symbol	Bedeutung
Mathematische Notationen	
x	Skalar
$\underline{x}$	Vektor
$\boldsymbol{X}$	Matrix
$\boldsymbol{X}_{m \times n}$	Matrix der Größe $m \times n$
$\mathcal{X}$	Menge
$\overline{x}$	Mittelwert
Lateinische Symbole	
A	Anzahl von ausgetauschten Wörtern bzgl. einer Grundwahrheit
a	Werte einer Umsetzungstabelle
b	Schwellwert für Klassifikationen
$\mathcal{C}$	Menge von Klassen
d	Dimensionen der Merkmalsvektoren für Textklassifizierung
D	Anzahl von entfernten Wörtern bzgl. einer Grundwahrheit

Symbol	Bedeutung
$\mathcal{D}$	Menge aus Sätzen mit zugehörigen Kategorien
E	Anzahl von eingesetzten Wörtern bzgl. einer Grundwahrheit
$\boldsymbol{E}$	Matrix einer Umsetzungstabelle
f	Aktivierungsfunktion eines künstlichen Neurons
F	Anzahl einer fehlerhaften Klasse
g	Genauigkeit
h	Verstecktes Neuron
$\mathcal{H}$	Menge von Hypothesen
I	Eingangsgröße eines Neurons
$\underline{I}$	Eingangsvektor eines künstlichen neuronalen Netzes
$\boldsymbol{I}$	Einheitsmatrix
i, j, k, n	Indizes mit $i, j, k, n \in \mathbb{N}$
K	Anzahl aller richtigen oder falschen Klassifikationen
l	Länge des längsten Satz eines Textkorpus
m	Fensterbreite
N	Anzahl aller Wörter einer Grundwahrheit
P	Wahrscheinlichkeit
$\mathcal{P}$	Menge der Ausgangsvektoren eines neuronalen Netzwerkes
q	Wortfehlerrate

Symbol	Bedeutung
s	Verschiebung der Fensterung
S	Sätze eines Datensatzes
t	Zeit
T	Transkript
$\boldsymbol{U}, \boldsymbol{V}$	Gewichtsmatrizen eines neuronalen Netzwerkes
$\mathcal{V}$	Menge des Vokabulars
$\tilde{v}$	$\tilde{v} = \lvert\mathcal{V}\rvert + 1$
$\boldsymbol{W}$	Gewichtsmatrix eines neuronalen Netzwerkes
$\mathcal{W}$	Menge der Wortfehlerraten
w	Gewicht eines neuronalen Netzes
x	Argument einer Aktivierungsfunktionen
y	Ausgangsgröße Neuron
$\underline{y}$	Ausgangsvektor eines neuronalen Netzes
$\hat{y}$	Ausgangsgröße Neuron als Wahrscheinlichkeit
$\underline{\hat{y}}$	Ausgangsvektor eines neuronalen Netzes als diskrete Wahrscheinlichkeitsverteilung
z	Anteil einer fehlerhaften Klasse

Symbol	Bedeutung
Griechische Symbole	
α	Konfidenz einer Klassifizierung
$\underline{\theta}$	Merkmalsvektor gebildet mit *Word Embedding*
η	Index mit $\eta \in \mathbb{N}$
σ	Sigmoid Aktivierungsfunktion
ψ	Softmax Aktivierungsfunktion
$\underline{\phi}$	Merkmalsvektor gebildet mit *One-Hot Encoding*

1 Einleitung

1.1 Motivation und bisherige Arbeiten am autonomen Logistik-Fahrzeug

In vielen Haushalten befinden sich Apparaturen, die auf einen Sprachbefehl hin Musik abspielen, tagesaktuelle Informationen weitergeben oder Gerätschaften im Haushalt steuern [1]. Eine Umfrage der Unternehmensberatung *Capgemini* zeigt, dass weltweit fast jeder Vierte bei der Suche nach Informationen eher auf einen Sprachassistenten zurückgreifen würde, als eine Website zu nutzen [2].

Diese Sprachassistenten nutzen Algorithmen des Themengebiets künstlicher Intelligenz, um die alltäglichen Anfragen der Nutzer umzusetzen [3, 4]. Die Technologien auf Basis der künstlichen Intelligenz werden jedoch nicht nur zur Spracherkennung eingesetzt, sondern auch in der Transport- und Logistikbranche genutzt [5]. Laut einer Potenzialanalyse der Firma *Sopra Steria* setzen 57 % der befragten Unternehmen bereits künstliche Intelligenz ein oder planen dies zu tun [6].

Das autonome Logistik-Fahrzeug (ALF) ist ein solches Transportfahrzeug und dient als Versuchsplattform für die Entwicklung autonomer Fahrfunktionen [7]. Das Ziel ist die automatisierte Abwicklung von Logistikprozessen an dem Standort der Hochschule Bochum [7]. Das ALF aus den Abbildungen 1.1 und A.1 stellt die Grundlage für verschiedene Projekte in den Bereichen des autonomen Fahrens, künstlicher Intelligenz, *Model-Based Design* sowie dem *Model-Based Systems Engineering* [8]. Das Fahrzeug wurde im Rahmen der Masterarbeit „Entwicklung und Verifikation eines autonomen Logistik-Fahrzeugs“ von M.Sc. Dennis Hotze, M.Sc. Dominik Eickmann und Prof. Dr. Ing. Arno Bergmann an der Hochschule Bochum entwickelt.

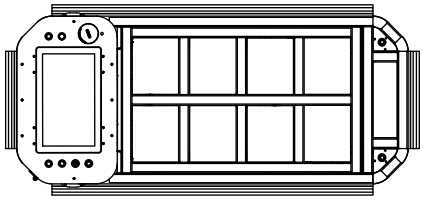
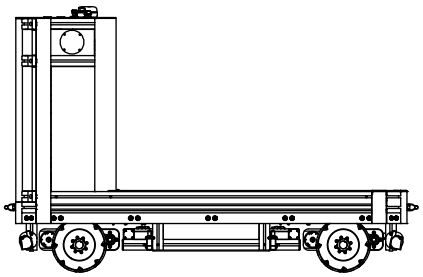
(a)  (b) 

Abbildung 1.1: (a) Darstellung des autonomen Logistik-Fahrzeugs aus der Draufsicht mit Blick auf die Ladefläche. (b) Darstellung aus der Seitenansicht.[8]

Nach der Entwicklung wurde in der Bachelorarbeit „Implementierung einer Schlupfregelung per *Model-Based Design* sowie einer *SLAM*-Kartografierung für ein autonomes Logistik-Fahrzeug“ eine Driftkompensation und das Abfahren einer definierten Trajektorie bei gleichzeitiger Kartografierung der Umgebung integriert [8]. Damit wurde die Einbindung autonomer Fahrfunktionen ermöglicht [8]. Weiterhin kann die Roboterplattform über eine Fernbedienung gesteuert und dadurch bewegt werden [7].

Die vorangegangenen Arbeiten dienen als Grundlage dieser Masterarbeit. In der Bachelorarbeit wurde das *Robot Operating System* (ROS) als Framework zum Datenaustausch zwischen verschiedenen Softwareartefakten eingeführt. Parallel zu der Entwicklung der Sprachverarbeitung wurde neben einer Bildverarbeitung mit dem Schwerpunkt Personenerkennung ein Zustandsautomat entwickelt. Dieser dient zur Verwaltung von verschiedenen Betriebsmodi des Fahrzeugs, die aus der Entwicklungshistorie entstanden sind. Die Betriebsmodi behandeln verschiedene autonome Fahrfunktionen des Roboters.

Der Zustandsautomat ist nicht Teil dieser Arbeit, jener wird in Montorio [9] behandelt. Diese Arbeit dient im Wesentlichen der Bereitstellung der Transitionsbedingungen für den genannten Automaten durch Sprachbefehle. Der Aufruf dieser Betriebsmodi erfolgt bisher durch Eingaben in der Eingabeaufforderung des verwendeten *Linux*-Systems. Dabei müssen für jeden Modus verschiedene Eingaben in der Konsole getätigt werden [8]. Der Zustandsautomat und die Steuerung dessen mit der Sprachverarbeitung vereinfachen das Anwählen dieser Betriebsmodi, da Sprachbefehle diese manuellen Eingaben ersetzen.

Für diese Funktion ist es nicht notwendig, Sprache beliebigen Inhalts zu erkennen. Hinsichtlich der Aufgaben am ALF reicht ein begrenztes Vokabular, da die Anzahl der Tätigkeiten und Sprachbefehle begrenzt ist. In dieser Arbeit wird der eingeschränkte Wortschatz als bedienungsorientiert bezeichnet. Das Ziel ist es, dem ALF eine Interpretation von Sätzen mit definiertem abgegrenzten Inhalt (zum Beispiel *„Drive to location beta“* oder *„Start to localize yourself in known environment“* etc.) zu ermöglichen.

1.2 Aktueller Stand der Sprachverarbeitung

Die Umsetzung einer automatischen Spracherkennung ist ein lang ersehntes Anliegen der Wissenschaft, so wird seit über 50 Jahren an einer ingenieurwissenschaftlichen Lösung geforscht [10]. Der erste vollständige Spracherkenner geht zurück in das Jahr 1952. Die *Bell Labs* in den Vereinigten Staaten fertigten den Ersten dieser Art an [11]. Einen besonderen Fortschritt erzielte man durch die Beschreibung einer Theorie der menschlichen Sprachbildung als auch durch die Verwendung von statistischen Modellen [11]. Des Weiteren trugen große, kostengünstige und leistungsfähige Computer zur Weiterentwicklung bei [10, 12]. Eine Spracherkennung basiert heute auf Methoden des maschinellen Lernens, also einem Teilgebiet der künstlichen Intelligenz [11]. Die Fähigkeit solcher Modelle, bestimmte Muster und Gegebenheiten zu erkennen, muss antrainiert werden um Erfahrungen zu generieren [13]. Dabei lernen diese Modelle aus Datenbeständen immer wiederkehrende Merkmale, sodass Wahrscheinlichkeiten berechnet und Vorhersagen getroffen werden können [14].

Dennoch waren lange Zeit Spracherkennungssysteme durch ihre Rechenleistung und Speicher limitiert [10]. Erst durch die Möglichkeit des *Cloud Computing*, als auch Projekte zur Datensammlung. haben Spracherkennungssystemen ihre Leistungsfähigkeit steigern können [15]. Dadurch ist die Erkennung eines größeren Vokabulars sowie verschiedener Sprachen und Akzente möglich geworden [15].

Sprachassistenten wie *Siri, Amazon Alexa, Google Assistant* und *Microsoft Cortana* stellen nicht nur Dienste zur Spracherkennung bereit, sondern bieten eine komplette Erkennung und Verarbeitung von gesprochener Sprache [16]. Diese Sprachassistenten

zielen hauptsächlich darauf ab, alltägliche Dinge oder Prozesse zu vereinfachen. Die genannten Systeme verarbeiten die Sprachsignale jedoch online mithilfe von *Cloud Computing* [16]. Das *Cloud Computing* stellt dabei eine über das Internet bereitgestellte IT-Infrastruktur dar, bei der Speicherplatz, Rechnerkapazitäten oder Software-Dienste genutzt werden können [17].

Bei einem Spracherkennungssystem ist zwischen einem Dienst und einer Softwarelösung zu unterscheiden. Die Dienste erzeugen ein Transkript mithilfe von *Cloud Computing* [18]. Die Softwarelösungen stellen Modelle und Bibliotheken für entsprechende Entwicklungsumgebungen bereit. Bekannte Online-Dienste zur automatischen Spracherkennung bieten die Firmen *Google*, *IBM Watson*, *Amazon* und *Microsoft* [19]. Die Unterschiede zeichnen sich dabei im Wesentlichen durch die Unterstützung von verschiedenen Sprachen und Akzenten aus [19]. Die genannten Spracherkennungsdienste und Sprachassistenten benötigen eine Internetverbindung. Diese ist aufgrund der technischen Gegebenheiten des Fahrzeugs nicht dauerhaft gegeben, weshalb eine offline Sprachverarbeitung umgesetzt wird.

Als Offlinelösung sind *DeepSpeech*, *DeepSpeech II*, *Pocketsphinx*, *Espnet*, *Wav2Letter* und *Kaldi* hervorzuheben. Diese Anwendungen stellen fertige Spracherkennungssysteme und Modelle offline zur Verfügung [20, 21, 22, 23, 24]. Außerdem bietet Picovoice ebenfalls eine kommerzielle *closed-source* Offline-Softwarelösung an [25, 26].

1.3 Ziel und Struktur dieser Arbeit

Das Ziel dieser Arbeit ist die Entwicklung eines Offline-Sprachverarbeitungssystems für das autonome Logistik-Fahrzeug. Diese Verarbeitung schafft die Grundlage einer Mensch-Maschine-Interaktion, um die Bedienung des Roboters zu vereinfachen. Dabei werden die aus der Entwicklungshistorie entstandenen Betriebsmodi des ALF anhand von Spracheingaben eines Benutzers ausgeführt als auch gewechselt. Per Sprachbefehle werden Trajektorien zu entsprechenden Zielen geplant und abgefahren, sodass das ALF daraufhin Logistikprozesse hochautomatisiert abwickelt.

Diese Arbeit besteht aus insgesamt sechs Kapiteln. In den Kapiteln 1 und 2 wird dabei auf die Motivation und die notwendigen Grundlagen hinsichtlich dieser Arbeit eingegangen. Eine besondere Bedeutung wird in dem Kapitel 2 der Funktionsweise von verschiedenen neuronalen Netzwerken und deren Anwendung beigemessen.

Das Kapitel 3 analysiert das Umfeld des bestehenden Systems und wie die Sprachverarbeitung in eben jenes eingebettet wird. Aus dieser Analyse werden Anforderungen an die Sprachverarbeitung erhoben und in einem Lastenheft festgehalten. Die hinsichtlich der Sprachverarbeitung entwickelten Unterprozesse *Spracherkennung* und *Sprachklassifizierung* werden ebenfalls im dritten Kapitel näher betrachtet. Des Weiteren erfolgt eine Vorstellung der Entwicklungsumgebungen und Netzwerkarchitekturen, welche für die Entwicklung der Sprachverarbeitung in dieser Arbeit von Bedeutung sind.

Das Kapitel 4 stellt die Evaluation der entwickelten Methoden dar. Dabei werden die Prozesse mithilfe verschiedener Datensätze und Metriken quantitativ bewertet und anschließend diskutiert. In Kapitel 5 werden die entwickelte Sprachverarbeitung und der erstellte Datensatz bezüglich der erhobenen Anforderungen sowie anhand einer Plausibilitätsprüfung verifiziert. Zuletzt wird in Kapitel 6 ein Fazit dieser Arbeit gezogen und ein Ausblick auf Folgethemen zur Weiterentwicklung gegeben.

2 Theoretische Grundlagen

Das folgende Kapitel behandelt die Grundlagen von verschiedenen neuronalen Netzwerken und sprachverarbeitenden Systemen. Diese dienen dem besseren Verständnis der in den Kapiteln 3 und 4 vorgestellten Methoden, Konzepte und Lösungen. Besonderer Fokus liegt auf der Extraktion von Merkmalsvektoren für Klassifikationsaufgaben von vorwärtsgerichteten und rekurrenten neuronalen Netzwerken. Derartige Netzwerke finden in dieser Arbeit Anwendung und bieten darüber hinaus ein breites Anwendungsfeld in Wissenschaft und Technik [27]. Des Weiteren werden Definitionen von Grundbegriffen aus der Sprachforschung vorgestellt.

2.1 Sprachwissenschaftliche Grundbegriffe

Phon:

Ein Phon ist das kleinste Lautsegment, welches als selbstständig wahrgenommen wird. Ein Phon unterscheidet sich in der Klangfarbe, zeitlichen Länge, Stärke der Betonung und der Tonhöhe. Ein Phon beschreibt den tatsächlich wahrzunehmenden Schall. [12]

Phonem:

Das Phonem ist ein Sprachzeichen, das nicht selbst wieder aus Zeichen besteht, das heißt das kleinste bedeutungsunterscheidende Sprachzeichen [28]. Ein Phonem ist demnach eine aus Lauten abgeleitete Einheit [12]. Ein Wort ist im Gegensatz dazu die kleinste, aus Phonemen bestehende, selbstständige sprachliche Einheit mit einer Bedeutung [29]. Mehrere Wörter bilden eine Wortgruppe [30].

Satz:

Ein Satz ist im Allgemeinen eine aus mehreren Wörtern bestehende, in sich geschlossene sprachliche Einheit. Dieser enthält eine Frage, Aussage oder Aufforderung. [31]

2.2 Vorwärtsgerichtete neuronale Netzwerke

Künstliche neuronale Netze stellen eine Abstraktion der Funktionsweise von Neuronen des menschlichen Gehirns dar [12]. Das Ziel ist, einzelne Neuronen und deren Bestandteile mathematisch zu modellieren und miteinander zu vernetzen. Dies simuliert kognitive Prozesse des menschlichen Gehirns, wodurch eine Mustererkennung mithilfe eines solchen Modells möglich wird [14]. Die grafische Modellierung eines einzelnen Neurons ist in Abbildung 2.1 dargestellt.

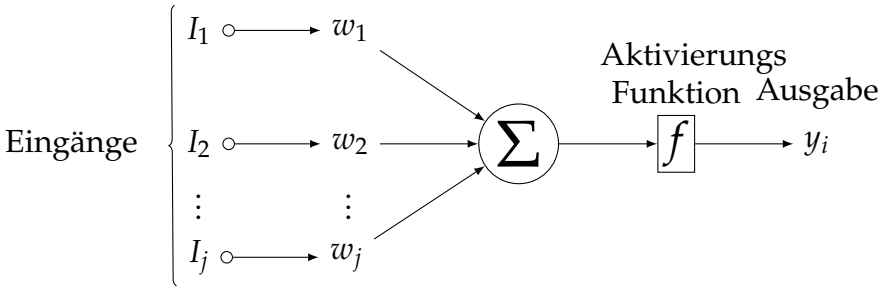


Abbildung 2.1: Abbildung eines Neurons h_i mit den Eingängen I_j, Gewichten w_j, Aktivierungsfunktion f und dem Ausgang y_i. Die Produkte aus Eingangswerten und Gewichten werden aufsummiert und anschließend mit einer Aktivierungsfunktion transformiert. Adaptiert aus [13].

Mathematisch wird ein Neuron modelliert, indem auf die gewichtete Summe aller Eingänge eine Aktivierungsfunktion f angewendet wird [14]. Dadurch ergibt sich die Ausgabe:

$$y_i = f(\sum_{\eta=1}^{j} w_\eta I_\eta) \text{ mit } j, \eta \in \mathbb{N} \, . \tag{2.1}$$

Angesichts der Aktivierungsfunktionen existieren eine Reihe von Möglichkeiten. Die Gleichungen (2.2) sowie (2.3) beschreiben die *Rectified Linear Unit* (ReLU)- und Sigmoid-Funktion [4]. Für die Funktionen aus den Gleichungen (2.2), (2.3) gilt $x \in \mathbb{R}$.

$$ReLU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Bei den in Abbildung 2.2 dargestellten Aktivierungsfunktionen $ReLU(x)$ und $\sigma(x)$ ergibt sich zwischen den Ein- und Ausgangsgrößen ein nichtlinearer Zusammenhang [4]. Der Ausgang eines solchen Neurons wird als Aktivität oder Erregungszustand der Zelle bezeichnet [13].

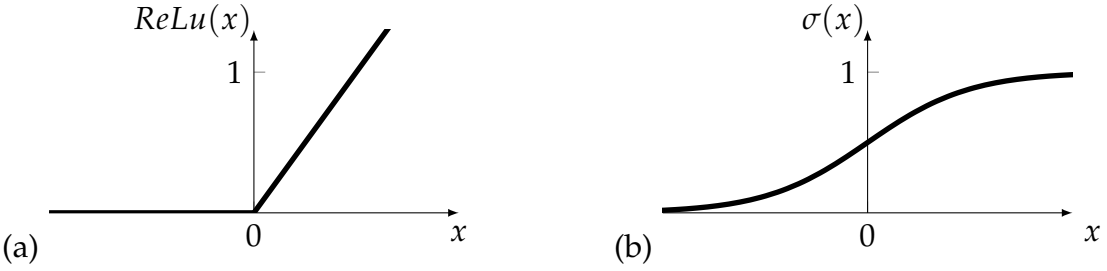


Abbildung 2.2: (a) Darstellung der *Rectified Linear Unit*-Funktion. Die Ausgangswerte für $x < 0$ werden auf 0 gesetzt. (b) Darstellung der Sigmoid-Funktion für den Eingangswert x. Für beide Funktionen gilt $x \in \mathbb{R}$. Adaptiert aus [32].

Durch Vernetzen dieser inneren Struktur mit weiteren Neuronen über verschiedene Ebenen hinweg entsteht ein sogenanntes vorwärtsgerichtetes neuronales Netz. Das Kennzeichen der Verbindung ist die Gewichtung w zwischen Neuronen unterschiedlicher Schichten. Dabei kann jedes Neuron beliebig viele Eingangsverbindungen mit gleichen oder variierenden Gewichtungen besitzen. Als Einschränkung gilt bei dem vorwärtspropagierenden Netz (engl. *Feedforward Net*), dass die Verbindungen vorwärtsgerichtet sind. Das heißt, dass keine Rückkopplung eines Neuronenausgangs auf bereits zurückliegende oder benachbarte Neuronen stattfindet. Die Abbildung 2.3 zeigt eine solche Vernetzung mehrerer Neuronen. [12]

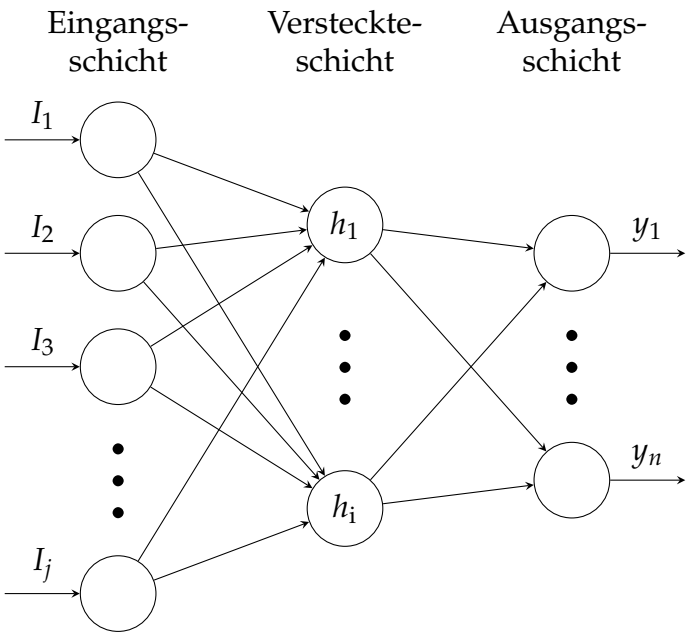


Abbildung 2.3: Abbildung eines vorwärtspropagierenden neuronalen Netzwerks mit jeweils einer Eingangs-, Versteckten- und Ausgangsschicht. Die verschiedenen Schichten entstehen aus der Vernetzung beliebig vieler Neuronen $h_1, ..., h_i$. Adaptiert aus [12].

Befinden sich vor einer Schicht keine gewichteten Eingangsverbindungen, wird diese als Eingangsschicht bezeichnet. Besitzen Neuronen keine gewichteten Ausgänge, so ist dies eine Ausgangsschicht. Die Schichten dazwischen werden als versteckte Schichten bezeichnet. [4]

Aus der Abbildung 2.3 kann entnommen werden, dass die Ein- und Ausgänge Vektoren sind. Demzufolge ergibt sich hinsichtlich der Abbildung 2.3 ein j-dimensionaler Eingangsvektor $\underline{I}$ und ein n-dimensionaler Ausgangsvektor $\underline{y}$. Die Gewichte zwischen den verschiedenen Schichten vereinen sich zu der Matrix $\boldsymbol{U}_{i\times j}$. Die gewichtete Summe aus der Gleichung (2.1) ist somit eine Matrixmultiplikation zwischen der Gewichtsmatrix und Ausgangswerten von Neuronen. [4]

Der durch die Aktivierungsfunktionen erzeugte nichtlineare Zusammenhang zwischen dem Ein- und Ausgang bei einem einzelnen Neuron ruft bezüglich der Ein- und Ausgangsvektoren des gesamten Netzes ebenfalls einen nichtlinearen

Zusammenhang hervor [4]. Der Ausgangsvektor errechnet sich durch Vorwärtspropagieren, das heißt durch eine schrittweise Berechnung der Neuronenausgänge [14]. Der Eingangsvektor stellt dabei den Anfang dar, dieser nimmt im Fall einer Spracherkennung ein Spektrogramm eines Audiosignals an [20]. Die Ermittlung eines solchen Spektrogramms erfolgt exemplarisch in Abschnitt 2.4. Mit dieser Struktur lassen sich unter anderem Klassifizierungsaufgaben lösen [33]. Eine Klassifizierungsaufgabe ist in diesem Kontext eine rechentechnische Aufgabe. Dabei ordnet ein Algorithmus eine Eingangsgröße, zum Beispiel den Eingangsvektor $\underline{I}$, in verschiedene Kategorien

$$\mathcal{C} = \{0, 1, 2, ..., n\} \text{ mit } n \in \mathbb{N} \tag{2.4}$$

ein. Um diese Aufgabe zu lösen, unterliegt ein Klassifikator der Funktion:

$$g : \mathbb{R}^j \rightarrow \{1, 2, ..., n\}. \tag{2.5}$$

Das Abbild der Funktion stellt dabei die numerische Codierung einer Klasse dar [33]. Zugleich können die Ausgangswerte eines Netzes auch einer diskreten Wahrscheinlichkeitsverteilung der verschiedenen Kategorien $\mathcal{C}$ entsprechen [33]. Dies tritt auf, wenn eine Transformation des Ausgangsvektors $\underline{y}$ mit der *Softmax*-Funktion ψ aus der Gleichung (2.6) zu $\underline{\hat{y}}$ stattfindet [4]. Die Funktion bewirkt, dass die Ausgangswerte eines Netzes als Wahrscheinlichkeitsverteilung

$$\hat{y}_j = P(c_j = j|\underline{y}) = \psi(\underline{y}) = \frac{e^{y_j}}{\sum_{i=1}^{n} e^{y_i}} \text{ für } j = 0, 1, 2, ..., n \tag{2.6}$$

interpretierbar sind [4, 20]. Die Grundlage der Funktion ist die Reihe von Einträgen $y_1, ..., y_n$ des Ausgangsvektors $\underline{y}$ eines neuronalen Netzes. Die Ausgabe $\hat{y}$ beschreibt eine Wahrscheinlichkeit P für die Zugehörigkeit zur Kategorie c_j angesichts der Gegebenheit $\underline{y}$. Das Vorgehen erfolgt für alle Elemente des Vektors $\underline{y}$, sodass die diskrete Wahrscheinlichkeitsverteilung $\underline{\hat{y}}$ entsteht [33]. Der Einsatz dieser Funktion erfolgt am Ende eines Netzwerkes [33]. Dabei findet eine Transformation von einer Reihe reellwertiger Zahlen in den Bereich von 0 bis 1 statt [32].

Die Gewichte w beziehungsweise die Einträge der Matrix $\boldsymbol{U}$, der diversen Schichten eines solchen Netzes, werden mit zufälligen Anfangswerten initialisiert [14]. Dadurch sind Klassifikationsaufgaben noch nicht zu bewältigen.

Um diese Aufgabe zu erfüllen, ist ein Training des Netzes mithilfe von geeigneten Daten notwendig [12]. Zu diesem Zweck wird ein Vektor $\underline{I}$ aus einem Datensatz $\mathcal{D}$ erzeugt, um anschließend durch Vorwärtspropagieren den Ausgangsvektor $\underline{y}$ des Netzes zu berechnen [12]. Der Datensatz beinhaltet dabei ein zu klassifizierendes Objekt und eine zugehörige Grundwahrheit (engl. *Label*). Im Fall einer Spracherkennung wäre diese eine Tonspur mit dazugehörigem, korrektem Transkript. Bei der Sprachklassifikation demzufolge ein Text mit korrespondierender Kategorie.

Nach dem Vorwärtspropagieren erfolgt die Ermittlung eines Fehlers zwischen dem erwarteten Ausgangsvektor (Grundwahrheit) und dem vom Netz bestimmten (Hypothese). Der resultierende Fehler wird genutzt, um die Gewichte rückwärts von Schicht zu Schicht anzupassen. Diese Vorgehensweise wird als Rückwärtspropagieren (engl. *Backpropagation*) bezeichnet. Die Anpassung erfolgt anhand eines Optimierungsverfahrens. Das Bekannteste und am häufigsten eingesetzte Optimierungsverfahren ist das Gradientenabstiegsverfahren. Das Rückwärtspropagieren wird solange wiederholt, bis der ermittelte Fehler unter einer festgelegten Schwelle liegt. Dieses Vorgehen gilt heute als Standardverfahren hinsichtlich des Trainings derartiger Strukturen. [12]

Ein neuronales Netzwerk kann überangepasst (engl. *overfit*) werden. Dabei weist das Netz eine sehr hohe Genauigkeit an einem Trainingsdatensatz auf. Ein Testdatensatz stellt eine Datenansammlung ähnlich des Trainingsdatensatzes dar. Unterschied ist, dass das Netzwerk diese Daten noch nicht zur Berechnung verwendet hat. Bei Überanpassung weist die Klassifikation eine geringe Genauigkeit am Testdatensatz auf. [34]

In die vorwärtsgerichteten Netzwerke werden die Eingangsdaten diskret eingegeben [32]. Die berechnete Ausgabe hat demzufolge keine Abhängigkeit zu bereits durchgeführten Prädiktionen [32]. Im Folgenden wird eine Netzwerkstruktur vorgestellt, welche Berechnungen aufgrund von vorausgegangen und aktuellen Eingaben tätigt.

2.3 Rekurrente neuronale Netzwerke

Rekurrente neuronale Netzwerke (RNN) sind spezialisierte neuronale Netze, mit denen aufeinanderfolgende Datensequenzen verarbeitet werden. Des Weiteren sind im Gegensatz zu vorwärtsgerichteten neuronalen Netzen Rückkopplungen von Neuronenausgängen auf zurückliegende Schichten gegeben. [32]

Solche Datensequenzen beinhalten mehrere Vektoren $\underline{I}_k$, kennzeichnend ist der Zeitschrittindex $k \in \mathbb{N}$. Die Abbildung 2.4 zeigt ein Signal $s(t)$, dieses wird beispielhaft in die drei Datensequenzen $\underline{I}_{k-1}$, $\underline{I}_k$ und $\underline{I}_{k+1}$ zerlegt. Die Datenpunkte aus den entsprechenden Bereichen bilden dann den Eingangsvektor eines RNN zum jeweiligen Zeitschrittindex. Der Zeitschrittindex referenziert dabei einen bestimmten Bereich der gesamten Datensequenz. Ein RNN benötigt dabei nicht zwingend zeitabhängige Daten, auch mehrdimensionale Daten, wie zum Beispiel Bilder, können in verschiedene Sequenzen aufgeteilt werden. [33]

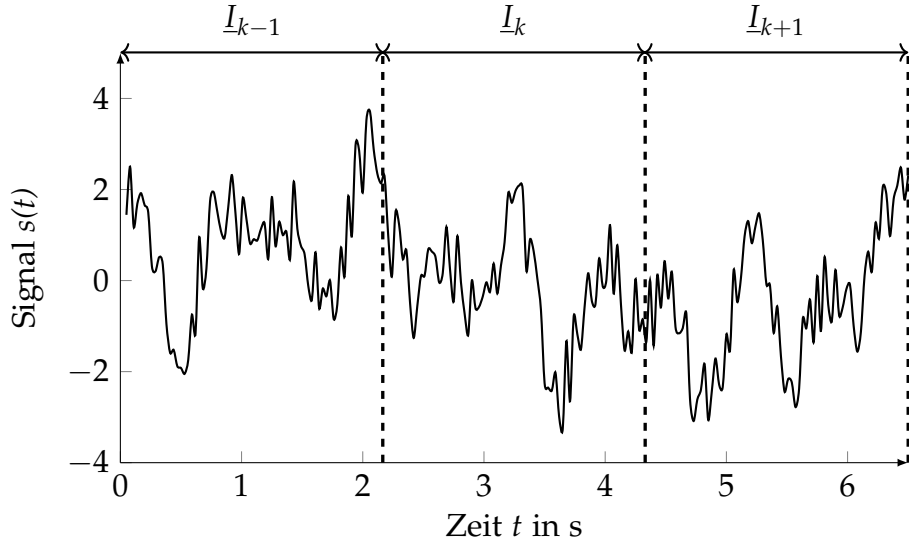


Abbildung 2.4: Die Abbildung zeigt ein beliebiges Signal, welches in drei Abschnitte eingeteilt wird. Die Datenpunkte der Bereiche entsprechen einzelnen aufeinanderfolgenden Datensequenzen und können als Eingangsvektor für ein RNN betrachtet werden.

Für jedes Element der Sequenz werden äquivalente Rechnungen durchgeführt, lediglich der Zeitschrittindex wird inkrementiert. Die Besonderheit dabei ist, dass die Ausgabe abhängig von der davor stattgefundenen Berechnung ist [32]. RNN besitzen demzufolge durch die Rückkopplung einen Speicher, in der Literatur auch als Assoziativspeicher bezeichnet [14].

In modernen Spracherkennungssystemen finden solche Strukturen Anwendung, so auch bei den in dieser Arbeit angewendeten Softwarelösungen [20, 24]. Weiterhin werden mit RNN Texte klassifiziert, da diese ebenfalls aufeinanderfolgende Datensequenzen darstellen können [32]. Für die Kategorisierung von Texten und Tonspuren ist jedoch eine vorausgehende Merkmalsextrahierung von Nöten [10]. Die notwendigen Schritte hierfür werden in den Abschnitten 2.4, 2.5 und 2.6 erläutert. Die Abbildung 2.5 stellt ein solche Netzstruktur dar.

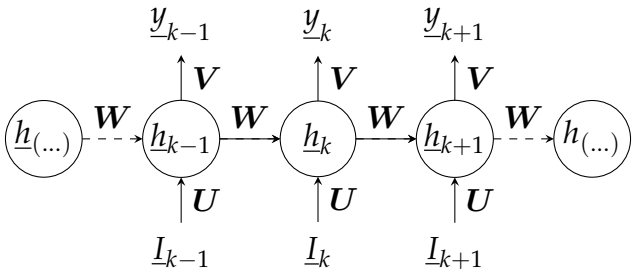


Abbildung 2.5: Darstellung eines rekurrenten neuronalen Netzwerks mit den Gewichtsmatrizen $\boldsymbol{U}, \boldsymbol{V}$ und $\boldsymbol{W}$. In aufgefalteter Illustration. Adaptiert aus [33, 32].

Durch diese Art der Vernetzung entstehen verschiedene Verbindungen, welche durch folgende Gewichtsmatrizen parametrisiert werden [32]:

- Eingangssequenz $\underline{I}_k$ zu versteckter Schicht $\underline{h}_k$ mit der Matrix $\boldsymbol{U}$
- versteckte Schicht $\underline{h}_k$ zur versteckten Schicht zum nächsten Zeitschrittindex $\underline{h}_{k+1}$ durch $\boldsymbol{W}$
- versteckte Schicht $\underline{h}_k$ zum Ausgang mit der Matrix $\boldsymbol{V}$

Der Ausgang $\underline{y}_k$ einer Schicht ergibt sich durch:

$$\underline{y}_k = \psi(\boldsymbol{V}\ \underline{h}_k) \text{ mit } \underline{h}_k = f(\boldsymbol{W}\ \underline{h}_{k-1} + \boldsymbol{U}\ \underline{I}_k). \tag{2.7}$$

Der Ausgang ist abhängig von den Gewichtsmatrizen $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$ und den Aktivierungsfunktionen f aus den Gleichungen (2.2) und (2.3) [33]. Die Vektoren $\underline{h}_k$ bilden einen Zustand des Netzwerkes zu dem entsprechenden Zeitschrittindex k ab [32]. Der sogenannte versteckte Zustand berechnet sich aus dem Zustand $\underline{h}_{k-1}$ und dem aktuellen Eingang $\underline{I}_k$ [32]. Durch Inkrementieren des Zeitschrittindexes wird für jede Sequenz fortlaufend der neue Zustand berechnet, bis die gesamte Datensequenz durchlaufen ist. Die Ausgaben dieser Netzwerkstruktur können, wie bei den vorwärtspropagierenden Netzwerken mit der *Softmax*-Funktion $\psi(\underline{y})$ aus Gleichung (2.6) transformiert werden [32]. Dadurch entsteht die diskrete Wahrscheinlichkeitsverteilung $\hat{\underline{y}}_k$ für jede Zeitsequenz. Nachfolgend werden die Grundlagen hinsichtlich einer Spracherkennung beschrieben.

2.4 Spracherkennung - Sprache zu Text

Dieser Abschnitt dient zum Verständnis einer automatischen Spracherkennung. Es werden Schritte beschrieben, um von einer Audiodatei mit Sprachaufnahme zu einem Transkript zu gelangen. Sprache zu Text heißt, aus einer gesprochenen Äußerung die Wörter richtig zu verstehen und als Text darzustellen [12].

Ausgangspunkt dieses Prozesses ist ein analoges Sprachsignal. Im Hinblick auf eine automatische Spracherkennung liegen zwei große Prinzipien zugrunde, die Merkmalsextraktion und Klassifikation eben jener [12]. Die Merkmalsextraktion dient der Reduktion großer Datenmengen und dem Herausarbeiten von charakteristischen Merkmalen eines Sprachsignals [12]. Die signifikanten Spracheigenschaften spiegeln sich am besten im Frequenzbereich wider und werden aus einem analogen Signal entsprechend transformiert [12].

Die Abbildung 2.6 stellt die Amplituden $A(t)$ des Wortes *Hallo* im Zeitbereich dar. Ein solches Signal unterzieht sich einer Kurzzeitspektralanalyse. Dadurch entsteht ein kontinuierlicher Fluss von Merkmalsvektoren eines definierten Zeitausschnitts.

Der Übergang von Zeit- in den Frequenzbereich erfolgt mittels der Diskreten Fourier-Transformation (DFT). Um ein korrektes Ergebnis der DFT zu erhalten, muss es sich um ein periodisches Signal handeln [10].

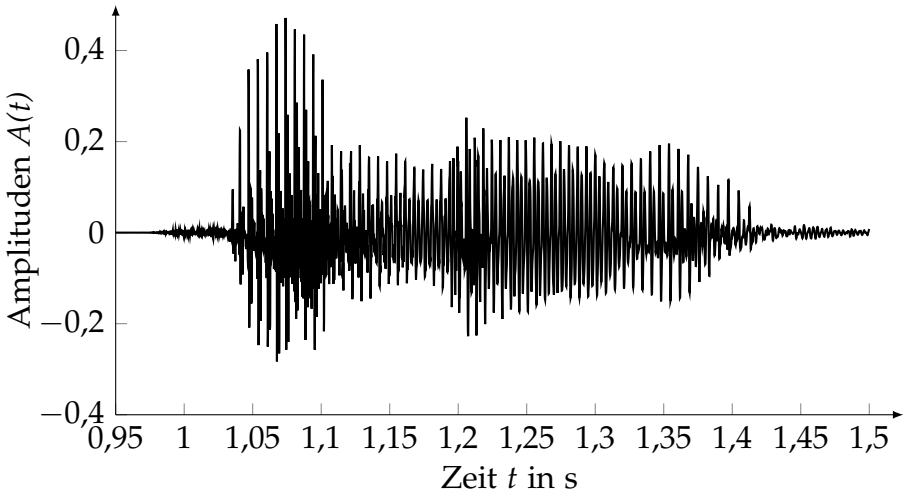


Abbildung 2.6: Darstellung der Amplituden des Wortes „Hallo“ im Zeitbereich. Aufgenommen mit einer Abtastrate von $f_s = 4000\,\text{Hz}$.

Dies ist nicht der Fall, wenn die Aufteilung eines solchen Zeitsignals aus Abbildung 2.6 in mehreren kurzen Zeitabschnitten w erfolgt [10, 35]. Die DFT setzt automatisch den Ausschnitt außerhalb des Zeitabschnittes periodisch fort [36, 10]. Mittels der dadurch entstehenden Periodizität kann das Sprachsignal innerhalb dieses Zeitbereichs als stationär angenommen werden [10].

Durch die periodische Fortsetzung entstehen an den Abschneidekanten Sprungstellen. Diese rufen bei der Transformation in den Spektralbereich sehr hohe Frequenzanteile hervor [10]. Die Abbildung 2.7 zeigt diesen Effekt in einem Zeitausschnitt. Der Ausschnitt wurde aus dem Signal der Abbildung 2.6 entnommen und zeigt zwei Perioden. Dem auftretenden *Leck*-Effekt wird entgegengewirkt, indem die einzelnen Zeitausschnitte (*engl. Frames*) mit einer sogenannten Fensterfunktion gewichtet werden [36]. Um das komplette Signal zu transformieren, erfolgt eine Verschiebung der Ausschnitte um eine bestimmte Anzahl von Zeitschritten (*engl. Frame-Shift*) s und die anschließende Gewichtung mit einer Fensterfunktion.

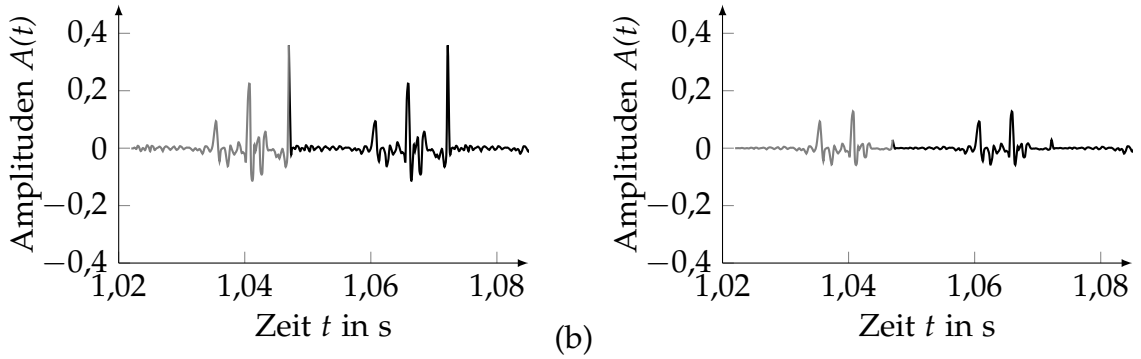


Abbildung 2.7: (a) Darstellung eines Ausschnitt des Signals aus Abbildung 2.6. Der Zeitausschnitt hat eine Dauer von $t = 25\,\text{ms}$. Dargestellt sind zwei Perioden. (b) Der gleiche Ausschnitt wie in (a), jedoch mit einem *Hamming*-Fenster der Breite $m = 25\,\text{ms}$ gewichtet. Zum Zeitpunkt $t = 1.047\,\text{s}$ entsteht durch das Abschneiden und anschließende Fortsetzung die beschriebene Sprungstelle. Diese wird durch eine passende Fensterfunktion vermieden.

Die typische Fensterbreite beträgt $m = 25\,\text{ms}$ bei einer Verschiebung von $s = 10\,\text{ms}$ [10]. Dadurch ergibt sich eine Überlappung und je nach Wahl der Fensterfunktion werden Sprungstellen und die daraus resultierenden hohen Frequenzanteile vermieden [10, 35]. Die Signalausschnitte werden nun fortschreitend transformiert. Dieses Vorgehen wird auch als *Short-Time Fast Fourier-Transformation* (STFFT) bezeichnet [10].

Das menschliche Hörvermögen ist bezüglich verschiedener Frequenzbereiche nicht konstant ausgeprägt [12]. Dadurch kann das Gehör nur ganze Frequenzbereiche zu einer Gruppe zusammenfassen, nicht aber einzelne Frequenzen aus dem Gesamtspektrum wahrnehmen [10]. Dieser Effekt nimmt mit höheren Frequenzen zu und wird in der *Mel*-Skala festgehalten [10]. Dabei werden die Frequenzen in *Mel*-Frequenzen überführt [10]. Diese Skalierung erfolgt mit einer Filterbank. Eine Filterbank ist eine Anordnung von digitalen Filtern, welche ein Eingangssignal in verschiedene Frequenzbänder zerlegt [35]. Die Abbildung 2.8 zeigt das aus den einzelnen Kurzzeitspektren resultierende Leistungsspektrogramm des Wortes *Hallo*, aufgetragen über die *Mel*-Frequenz Skala und der Zeit t.

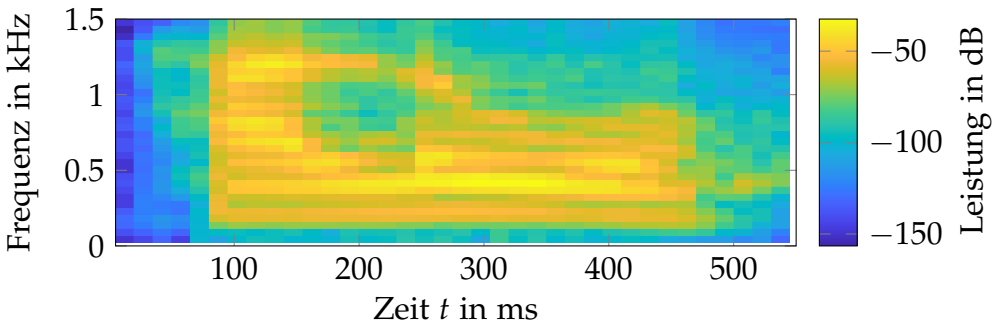


Abbildung 2.8: Abgebildet ist das gesamte Leistungsspektrogramm des Wortes *Hallo*, aufgetragen über die Mel-Frequenz Skala. Die Fensterbreite beträgt $m = 25\,\text{ms}$ und der *Frame-Shift* $s = 10\,\text{ms}$.

Die aus der STFFT ermittelten Kurzzeitspektrogramme ermöglichen ein direktes Auswerten und Klassifizieren. Die zeitversetzten Spektren entsprechen der Datensequenz aus Abschnitt 2.2 und beinhalten die Merkmale. Diese dienen zum Beispiel als Eingangsvektoren $\underline{I}$ für ein klassifizierendes RNN. [12]

Eine weitere Möglichkeit ist das Ableiten spezifischer Koeffizienten aus den Kurzzeitspektrogrammen. Dies hat sich als besonders vorteilhaft erwiesen, da die Verwendung von solchen Parametern den Aufwand reduziert und ebenso sprachspezifische Merkmale repräsentiert [12]. In der Spracherkennung haben vor allem die *Mel-Frequency-Cepstral Coefficients* (MFCC) eine große Bedeutung, da diese auf die hörpsychologischen Gegebenheiten angepasst sind [12]. Wendemuth und Fellbaum [10, 12] beschrieben zur Extraktion der MFCC eine präzise Vorgehensweise. Für jedes betrachtete Zeitfenster werden in der Regel 12 dieser Koeffizienten extrahiert und als Merkmale für einen Klassifikator genutzt [12].

Die Kurzzeitspektrogramme oder MFCC dienen als Eingangsvektoren $\underline{I}_k$ für ein in Kapitel 2.2 vorgestelltes RNN. Dabei wird mit den extrahierten Merkmalen aus einem *Frame* ein Eingangsvektor zu einem bestimmten Zeitschrittindex k gebildet. Demnach wird das Sprachsignal in eine Datensequenz zerlegt, bei der eine Sequenz die extrahierten Merkmale enthält. [20]

Dadurch ist es möglich, den *Frames* einzelne Phoneme oder ganze Wörter aus dem antrainierten Vokabular zuzuordnen. Die einzelnen Zeitausschnitte ermöglichen anhand der Merkmale demnach eine Klassifizierung. Finden als Klassen zum Beispiel die verschiedenen Buchstaben des Alphabets Verwendung, so entsteht aus dem Sprachsignal ein Transkript [20]. Bei einem RNN hat die Kategorisierung zum Zeitschrittindex k einen Einfluss auf die Klassifizierung zum Index $k+1$ [32]. Dadurch entstehen Klassifikationen, welche der tatsächlich gesprochenen Sprache sehr nah kommen. Denn das gesprochene Wort funktioniert in Form von Lautgruppen und nachfolgende beziehungsweise vorhergehende Laute stehen ebenfalls in Abhängigkeit zueinander [10].

Das errechnete Transkript T kann Fehler aufweisen. Daher wird das Resultat einer zusätzlichen Überprüfung unterzogen. Dies erfolgt mit einem Sprachmodell, welches die Wahrscheinlichkeit der resultierenden Wortgruppe berechnet [20]. Das Modell bestimmt die Wahrscheinlichkeit

$$P(T_{1:n}) = P(T_1)P(T_2|T_1)P(T_3|T_{1:2})P(T_4|T_{1:3}), ..., P(T_n|T_{1:n-1}) \tag{2.8}$$

für das Auftreten aller in dem Transkript enthaltenen Wörter $T_{1:n}$ [4]. Die Wahrscheinlichkeit $P(T_{1:n})$ des Sprachmodells und $\underline{\hat{y}}$ des vorher verwendeten Spracherkenners werden anschließend zusammengefasst und maximiert [20]. Das Sprachmodell stellt ebenfalls einen Algorithmus des maschinellen Lernens dar. An dieser Stelle sei auf Goldberg [4] und Goyal [32] verwiesen, welche die Sprachmodelle tiefer gehend analysieren.

2.5 Merkmalsextrahierung aus Transkripten durch One-Hot Encoding

Die Spracherkennung aus Abschnitt 2.4 erzeugt aus einer akustischen Eingabe das sogenannte Transkript T [12, 32]. Um eine Sprachverarbeitung in einem technischen System umzusetzen, muss aus dem Transkript eine Handlung erkannt werden [32]. In dieser Arbeit werden Transkripte kategorisiert. Diese Klassifikation dient als Transitionsbedingung für den in Abschnitt 1.1 erwähnten Zustandsautomaten.

Ein bekanntes Beispiel einer Textklassifizierung ist das Zuordnen eines Artikels zu einem Thema aus einer Liste von verschiedenen Themen wie Sport, Politik oder Technik [37, 38]. Zu Beginn der Textklassifizierung mithilfe von Algorithmen des maschinellen Lernens steht das Extrahieren von Merkmalen [37]. Entsprechende Algorithmen können nicht mit den rohen Textdaten eine Klassifizierungsaufgabe lösen. Vielmehr ist eine numerische Darstellung von Wörtern, Sätzen oder ganzen Texten nötig [38]. Eine Möglichkeit der Umsetzung ist das *Bag of Words* (BOW) Modell. Der erste Schritt bei der Erstellung eines BOW besteht darin, einen Datensatz zur Klassifikation zu generieren. Die aufgelisteten Sätze aus Tabelle 2.1 stellen einen Datensatz beispielhaft dar.

Tabelle 2.1: Ein Datensatz, auch Textkorpus genannt, bestehend aus vier Beispielsätzen. Die satzweise Zuweisung von Klassen c_j ergibt einen Ausschnitt des Datensatzes $\mathcal{D}$ aus Abschnitt 4.2. Weiterhin sind dies Beispielsätze aus dem in Abschnitt 1.1 erwähnten bedienungsorientierten Vokabular.

Nr.	Satz
1	*Drive to location beta*
2	*Localize yourself in known environment*
3	*Localize yourself in unknown environment*
4	*Wait for new commands*

Aus den in der Tabelle dargelegten Sätzen wird das Vokabular $\mathcal{V}$ gewonnen, indem jedes auftretende Wort einmal aus dem Korpus entnommen wird. Aus dem Vokabular entsteht der BOW-Vektor:

$$\underline{\phi}^T = [\textit{drive, to, location, beta, localize, yourself, in, known,} \\ \textit{environment, unknown, wait, for, new, commands}]. \tag{2.9}$$

Dabei repräsentiert jede Dimension $d_{1:14}$ des Vektors ein Merkmal [4]. Der Vektor beinhaltet 14 einzigartige Wörter, wobei der gesamte Textkorpus aus 18 Wörtern besteht. Der Eingangsvektor für ein Modell zur Klassifikation von Texten entsteht unter anderem durch *One-Hot Encoding* (OHE). Die Größe des Eingangsvektors ist abhängig

von der Anzahl an Dimensionen des Vektors $\underline{\phi}$, in diesem Beispiel also $d = 14$. Um Vektoren für eine Berechnung zu erzeugen, werden die Beispielsätze aus der Tabelle 2.1 genutzt und die Präsenz eines Merkmals mit einer 1 und die Abwesenheit mit einer 0 in dem BOW-Vektor markiert. Der Logik folgend entstehen hinsichtlich der Sätze Nr. 1, 2, 3 und 4, die Vektoren:

$$\underline{\phi}_1^T = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \tag{2.10}$$

$$\underline{\phi}_2^T = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0] \tag{2.11}$$

$$\underline{\phi}_3^T = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0] \tag{2.12}$$

$$\underline{\phi}_4^T = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]\ . \tag{2.13}$$

Diese Vektoren können als Eingangsvektoren $\underline{I}$ eines neuronalen Netzes fungieren. Die Gleichungen (2.10), (2.11), (2.12) und (2.13) stellen die OHE-Vektoren eines Satzes dar. Wird jetzt ausschließlich ein einzelnes Wort mithilfe der vorgestellten Methode codiert, so entsteht ein dünnbesetzter Vektor, der sogenannte *Sparse*-Vektor [38, 4]. In dem gezeigten Beispiel entsteht für das Wort *drive* der *Sparse*-Vektor:

$$\underline{\phi}_{\mathrm{d}}^T = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\ . \tag{2.14}$$

Daraus lässt sich schlussfolgern, dass lediglich ein Eintrag des Vektors ungleich 0 ist und für den Betrag $|\underline{\phi}_{\mathrm{d}}^T| = 1$ gilt. Bezüglich der restlichen Wörter des Textkorpus stellt sich ein äquivalentes Verhalten dar. [38]

Die Wahl, die Textklassifikationsaufgaben durch OHE und BOW-Vektoren zu lösen, ergibt sich aufgrund der Simplizität und Robustheit der Methodik. Die Technik ist aber limitiert, da diese abhängig von dem Transkript eines automatischen Spracherkennungssystems (ASR) ist. [39]

Eine weitere Möglichkeit, aus Wörtern und Sätzen entsprechende Vektoren zu ermitteln, ist das *Word Embedding*. Dahinter steht das Konzept, Wörtern mit ähnlicher Bedeutung eine gleichartige reellwertige Darstellung in einem vorher definierten Vektorraum zu geben [4].

Liegt ein größerer Textkorpus vor, steigt bei der OHE Methode die Dimensionalität der Merkmalsvektoren mit steigender Größe des Vokabulars [4]. Dabei ist weiterhin nur ein Eintrag ungleich null und der Betrag des Vektors verharrt bei eins. Die Darstellung liegt neuronalen Netzwerken aus rechentechnischen Gründen nicht [4]. Darin liegt der Vorteil des *Word Embedding*, da die Darstellung auf einen dicht besetzten Vektor, dem *Dense*-Vektor, basiert [4]. Dabei wird jedem Wort aus dem Vokabular ein *Dense*-Vektor definierter Dimension mit Betrag ungleich 1 zugewiesen [4, 32]. Die Dimension ist vor Beginn der Trainingsphasen zu definieren [38]. Im nächsten Abschnitt wird eine Technik vorgestellt, um zu dieser Darstellung zu gelangen.

2.6 Merkmalsextrahierung aus Transkripten durch Embedding-Schichten

Wie der Name bereits andeutet, handelt es sich bei der *Embedding*-Schicht um eine Ebene eines neuronalen Netzwerkes [38]. Die Schicht ersetzt die Eingangsschicht eines Text-, Wort- oder Dokumentenklassifizierenden Netzwerkes [4]. Das Trainieren der Gewichte $\boldsymbol{U}$ erfolgt gemeinsam mit den nachfolgenden Schichten [4]. In der Literatur sind als Name dieser Konstellation auch die Bezeichnungen *projection layer* oder *lookup layer* angegeben [32, 39, 4]. Die Anzahl der Dimensionen des Vektorraums für die reellwertige Darstellung der Wörter ist ein Teil des Modells und unterliegt einer vorherigen Definition [38]. Nachfolgend werden die Berechnungsvorschriften dieser Schicht näher erläutert.

Um eine reellwertige *Dense*-Vektor Darstellung für den Textkorpus aus 2.1 umzusetzen, wird die Dimension $d = 8$ für nachfolgende Beispiele definiert. Weiterhin ist das Vokabular $\mathcal{V}$ mit dem Betrag $|\mathcal{V}| = 14$ aus den vorangegangen Beispiel bekannt. Aus diesen Informationen entsteht die Umsetzungstabelle 2.2.

Tabelle 2.2: Umsetzungstabelle, auch *lookup table* oder *projection matrix* genannt, mit $d \times (|\mathcal{V}| + 1)$ Einträgen, wobei für $a \in \mathbb{R}$ gilt. Die Addition um 1 wird im Folgenden begründet.

Nr.	Werte der Umsetzungstabelle			
0	$a_{0,1}$	$a_{0,2}$	$\cdots$	$a_{0,8}$
1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,8}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
8	$a_{8,1}$	$a_{8,2}$	$\cdots$	$a_{8,8}$

Die Einträge $a_{0,1}$,...,$a_{|\mathcal{V}|,8}$ der Tabelle werden mit Zufallszahlen initialisiert und im Verlauf des Trainings angepasst. Das Ableiten von *Sparse*-Vektoren wie zum Beispiel $\underline{\phi}_{\mathrm{d}}^T$ aus der Gleichung (2.14) ist in diesem Fall nicht zwingend erforderlich. Stattdessen können einzelne Wörter als Skalar mit der Spalten- oder Zeilennummer des BOW-Vektors aus der Gleichung (2.9) aufgeschlüsselt werden [38]. Den Wörtern *drive* und *commands* werden exemplarisch die Werte 1 und 14 zugeordnet. Eine vektorielle Darstellung entsteht durch eine Zuordnung entsprechender Skalare aus der zugehörigen Zeile der Tabelle 2.2. Für das Wort *drive* entsteht demzufolge der Vektor:

$$\underline{\theta}_{\mathrm{d}}^T = [a_{1,1}\ a_{1,2}\ \cdots\ a_{1,8}]. \tag{2.15}$$

Werden die Werte der Umsetzungstabelle als Matrix $\boldsymbol{E}$ definiert, so entsteht

$$\boldsymbol{E} = \begin{pmatrix} a_{0,1} & a_{0,2} & \cdots & a_{0,8} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,8} \\ \vdots & \vdots & \ddots & \vdots \\ a_{|\mathcal{V}|,1} & a_{|\mathcal{V}|,2} & \cdots & a_{|\mathcal{V}|,8} \end{pmatrix}, \tag{2.16}$$

der Größe $8 \times (|\mathcal{V}| + 1)$. Mithilfe von $\boldsymbol{E}$ wird durch die Matrixmultiplikation

$$\underline{\theta}_{\mathrm{d}}^T = \underline{\phi}_{\mathrm{d}}^T \boldsymbol{E} \tag{2.17}$$

eine weitere Möglichkeit zur Darstellung von $\underline{\theta}_{\mathrm{d}}^T$ erlangt [4].

Dies gilt auch für die anderen Wörter des Vokabulars. Für einen ganzen Satz resultiert eine Verschlüsselung entsprechend der darin enthaltenen Wörter. Zusätzlich wird ein Satz mit Nullen aufgefüllt (engl. *Padding*), wobei der längste Satz des Textkorpus als Referenz l dient [38]. Für dieses Beispiel gilt demzufolge $l = 5$. Durch das Auffüllen mit Nullen steigert sich der Betrag des Vokabulars um den Wert 1, da die nullwertigen Einträge mit Hilfe der Umsetzungstabelle ebenfalls einen Vektor zugewiesen bekommen. Daraus folgen angesichts der Sätze eins bis vier die Verschlüsselungen

$$\underline{\theta}_{\mathrm{a}}^{T} = [1\ 2\ 3\ 4\ 0] \tag{2.18}$$

$$\underline{\theta}_{\mathrm{b}}^{T} = [5\ 6\ 7\ 8\ 9] \tag{2.19}$$

$$\underline{\theta}_{\mathrm{e}}^{T} = [5\ 6\ 7\ 10\ 9] \tag{2.20}$$

$$\underline{\theta}_{\mathrm{f}}^{T} = [11\ 12\ 13\ 14\ 0] \tag{2.21}$$

in vektorieller Darstellung. Die dicht besetzten Vektoren entstehen, indem die Skalare der Spalten den entsprechenden Zeilen aus der Tabelle 2.2 zugeordnet und verkettet werden [38]. Für die vier Sätze ergeben sich somit die Vektoren der Dimension $d_{\mathrm{neu}} = l\,d = 5 \cdot 8 = 40$:

$$\underline{\theta}_{1}^{T} = [a_{1,1}\ ...\ a_{1,8}\ a_{2,1}\ ...\ a_{2,8}\ a_{3,1}\ ...\ a_{3,8}\ a_{4,1}\ ...\ a_{4,8}\ a_{0,1}\ ...\ a_{0,8}] \tag{2.22}$$

$$\underline{\theta}_{2}^{T} = [a_{5,1}\ ...\ a_{5,8}\ a_{6,1}\ ...\ a_{6,8}\ a_{7,1}\ ...\ a_{7,8}\ a_{8,1}\ ...\ a_{8,8}\ a_{9,1}\ ...\ a_{9,8}] \tag{2.23}$$

$$\underline{\theta}_{3}^{T} = [a_{5,1}\ ...\ a_{5,8}\ a_{6,1}\ ...\ a_{6,8}\ a_{7,1}\ ...\ a_{7,8}\ a_{10,1}\ ...\ a_{10,8}\ a_{9,1}\ ...\ a_{9,8}] \tag{2.24}$$

$$\underline{\theta}_{4}^{T} = [a_{11,1}\ ...\ a_{11,8}\ a_{12,1}\ ...\ a_{12,8}\ a_{13,1}\ ...\ a_{13,8}\ a_{14,1}\ ...\ a_{14,8}\ a_{0,1}\ ...\ a_{0,8}] \tag{2.25}$$

Die Abbildung 2.9 zeigt beispielhaft die Umsetzungstabelle einer *Embedding*-Schicht. Die ausgehende Verkettung mit der Dimension d_{neu} dient als Eingang für nachfolgende Netzwerkschichten [32, 38]. Das Durchführen einer Klassifikationsaufgabe wird möglich, indem die Optimierung der Werte $a_{0,1},...,a_{|\mathcal{V}|+1,8}$ durch Rückwärtspropagieren erfolgt.

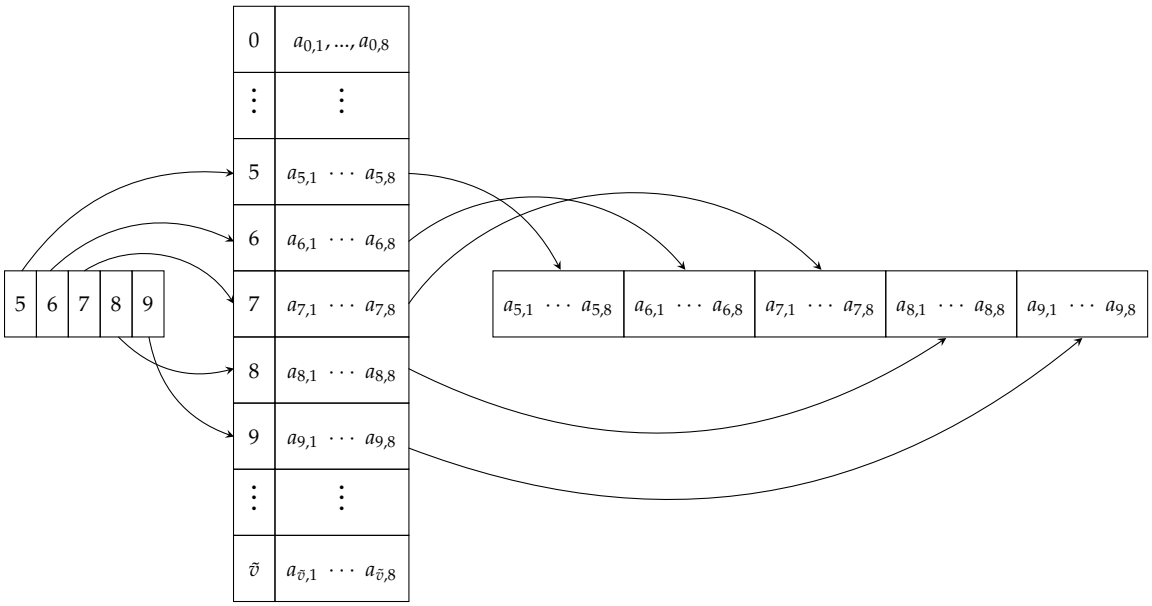


Abbildung 2.9: Abgebildet ist die Funktionsweise einer *Embedding*-Schicht mit dem Beispielsatz *„Localize yourself in unknown environment"*. In codierter Form ergibt sich der Vektor $\underline{\phi}_b^T$ aus der Gleichung (2.19). Den einzelnen Vektorkomponenten wird der entsprechende Zeilenvektoren aus der Tabelle zugeordnet und anschließend verkettet. Dadurch entsteht die reellwertige Darstellung $\underline{\theta}_2^T$ aus der Gleichung (2.23). Es gilt $\tilde{v} = |\mathcal{V}|$.

Aus dieser Technik lässt sich die *Word2Vec* Methode ableiten. Ähnlich wie die *Embedding* Schicht ist *Word2Vec* eine Technik, welche in Form von neuronalen Netzwerken ihre Anwendung findet. Der Unterschied zwischen den Verfahren ist, dass *Word2Vec* nur aus einer versteckten Schicht besteht und das Training dadurch effizienter ist [32, 39]. Die resultierende Gewichtsmatrix dieser einzelnen Schicht kann dann als *projection* beziehungsweise *lookup* Schicht interpretiert werden [32, 39]. Durch die vektorielle Darstellung der Sätze erfolgt in Abschnitt 4.2 eine Plausibilitätsprüfung des bedienungsorientierten Vokabulars.

Wie in diesem Abschnitt deutlich wurde, ordnet das *Word Embedding* einzelnen Wörtern beziehungsweise Sätzen mit ähnlicher Bedeutung eine vergleichbare vektorielle Darstellung zu. Durch eine Hauptkomponentenanalyse des Textkorpus kann die Datenmenge auf eine vorstellbare Dimension reduziert und dargestellt werden. Dadurch entsteht die Möglichkeit, Sätze und deren zugehörige Kategorie c_j darzustellen und fehlerhafte Datenpunkte zu identifizieren.

2.7 Textabgleich und phonetische Suche

Eine weitere Möglichkeit, um Sprachverarbeitungssystemen aus einem Transkript eine Handlung abzuleiten, ist das Abgleichen von Textfragmenten. Anstelle des neuronalen Netzes aus Abschnitt 2.2 und der Merkmalsextrahierung durch OHE oder *Word Embedding*, wird direkt nach Merkmalen in dem Transkript gesucht. Merkmale sind im Falle eines Textabgleichverfahrens Wörter, welche als Transitionsbedingungen genutzt werden können. In dieser Arbeit werden die erkannten Merkmale zur Veröffentlichung von Zielposen in *ROS* verwendet. Die Algorithmen funktionieren dabei in der Regel ähnlich und basieren auf den Abgleich von Schriftzeichen [40].

Dabei wird die Zeichenlänge des zu findenden Merkmals als Fenster genutzt und durch das Transkript geschoben. Abhängig vom verwendeten Algorithmus erfolgt ein Abgleich der Zeichen in dem Fenster mit denen des Merkmals. Dies wird solange wiederholt, bis der gesamte Text verglichen ist. In der Abbildung 2.10 ist die Fensterung eines Satzes zu sehen. [40]

D	R	I	V	E	T	O	L	O	C	A	T	I	O	N	B	E	T	A

Abbildung 2.10: Die Abbildung der Wortgruppe *Drive to location beta* wurde von Leerzeichen bereinigt. Das Wort *beta* dient als Merkmal, daraus folgt die Fensterbreite von 4 Zeichen.

Da in dieser Arbeit ein Transkript aus natürlich gesprochener Sprache analysiert wird, besteht die Möglichkeit, dass dieses fehlerhaft ist. Das hat zur Folge, dass der Benutzer beispielsweise eine Zielpose gesprochen hat, aber diese nicht im Textfragment gefunden wurde. Des Weiteren ist es möglich, dass ähnlich klingende Laute

erkannt werden, aber nicht zu den gesuchten Merkmalen passen. Wird statt *beta*, *better* transkribiert, so könnten die Algorithmen das gesuchte Merkmal nicht finden. Die phonetische Ähnlichkeit und dazugehörige Algorithmen schaffen hierbei Abhilfe. [41]

Das Ziel der Algorithmen ist es, Wörter aufgrund ihrer phonetischen Ähnlichkeit zu verschlüsseln. Dadurch erhalten ähnlich klingende Wörter den gleichen Schlüssel, obwohl diese eine andere Schreibweise besitzen. Die bekanntesten Verfahren zur Verschlüsselung von Begriffen sind *Soundex*, *Metaphone* und *Double-Metaphone* [42, 41]. Letztgenannte basieren auf den *Soundex* Algorithmus, bei dem der erste Buchstabe übernommen wird und die übrigen als Zahl verschlüsselt werden [41]. Aufeinanderfolgende gleiche Zeichen reduzieren sich auf ein einziges [41]. Für deutschsprachige Wörter wurde als Vorgehensweise die *Kölner-Phonetik* eingeführt. Der Algorithmus ist an das *Soundex* Verfahren angelehnt und verschlüsselt ebenfalls Buchstaben zu Ziffern [43]. Die Tabelle 2.3 stellt die Verschlüsselung von Buchstaben nach dem *Soundex*-Verfahren dar.

Tabelle 2.3: Verschlüsselung von Buchstaben zu Ziffern mit dem *Soundex*-Verfahren [41].

Buchstabe	Zugehörige Ziffer
a, e, h, i, o, u, w, y	0
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Wird das Codierverfahren aus Tabelle 2.3 auf das Beispiel *beta* und *better* angewandt, ergeben sich die Verschlüsselungen *b*030 und *b*0306. Das Merkmal *beta* kann jetzt von einem Textablgeichverfahren gefunden werden, obwohl ein fehlerhaftes Transkript vorlag.

3 Konzeptionierung

Das folgende Kapitel dient zur Konzeptionierung und Einordnung der Sprachverarbeitung in die bestehende Systemarchitektur. Weiterhin werden die Schritte dargelegt, mit denen aus einem Sprachsignal eine Handlung des Fahrzeugs eingeleitet wird. Die Einordnung der Sprachverarbeitung erfolgt mithilfe der Umfeldmodellierung aus der *„Conceptual design specification technique for the engineering of complex Systems“* (CONSENS). Das Verfahren dient der Modellierung von Umfeld- und Wirkstrukturen von mechatronischen Systemen und stellt eine Methode des *Model Based Systems Engineering* dar [44, 45]. Die Einordnung in das bestehende Modell basiert auf der in Kapitel 1.1 erwähnten Bachelorarbeit. Des Weiteren werden Anforderungen an den Sprachverarbeitungsprozess erhoben und die verwendeten Entwicklungsumgebungen sowie verschiedene Netzwerkarchitekturen erläutert.

3.1 Einordnung der Sprachverarbeitung in die bestehende Systemarchitektur

Die CONSENS-Methode dient unter anderem dazu, Anforderungen an die Sprachverarbeitung zu erheben und in einem Lastenheft festzuhalten [44, 45]. Neben der Erweiterung des Umfeldmodells wurde die vorhandene Wirkstruktur um die Sprachverarbeitung erweitert. Das Wirkelement *ALF* interagiert sowohl mit Umfeld- als auch Wirkelementen. Durch eine ausführlichere Modellierung der Wirkelemente entsteht ein Modell der entwickelten System- und Softwarearchitektur.

Die Wirkstrukturen aus Abbildung 3.2, 3.3, 3.4 und 3.5 repräsentieren demnach den Inhalt von einzelnen Wirkelementen. Das bestehende Umfeldmodell aus Abbildung 3.1 ist um das Umfeldelement *Zustandsautomat* und die dazugehörigen Transitionsbedingungen erweitert.

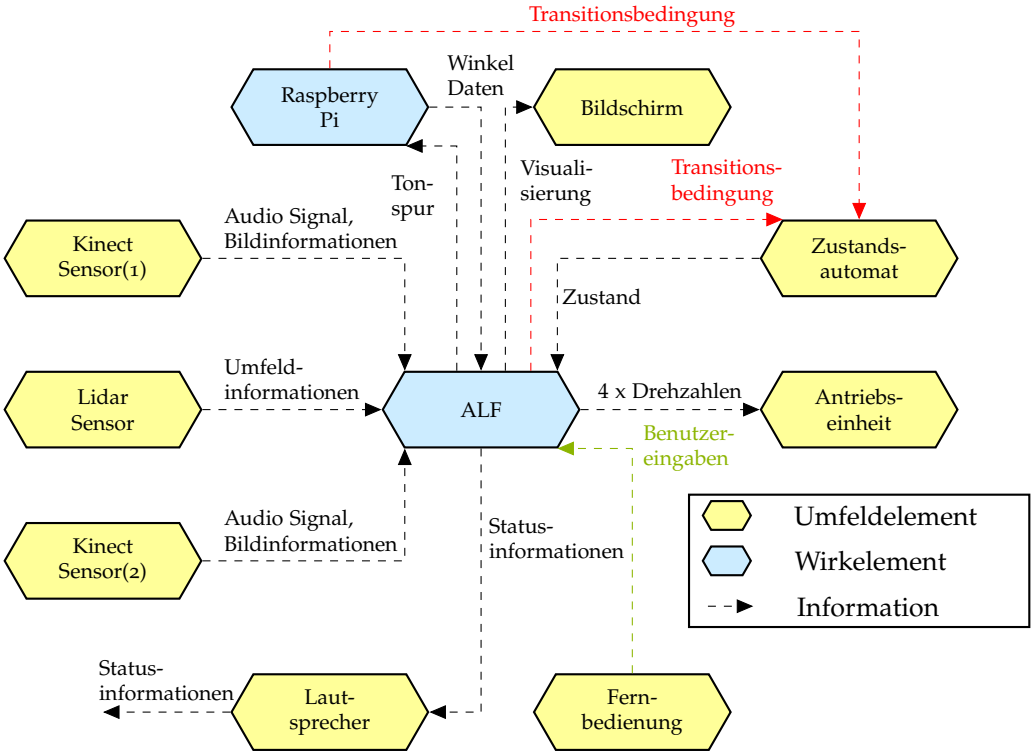


Abbildung 3.1: Die Abbildung zeigt das aus der vorangegangenen Arbeit mit der CONSENS-Methode weiterentwickelten Umfeldmodell. Eine besondere Bedeutung bekommt der Informationsfluss der Wirkelemente *Raspberry Pi* und *ALF* zu dem Umfeldelement *Zustandsautomat*, dieser beinhaltet die Transitionsbedingungen, um die Betriebsmodi des ALF zu steuern. Dies ermöglicht eine Sprachverarbeitung auf einem externen Gerät. Der Informationsfluss zwischen verschiedenen Elementen wird mit Strichpunktlinien gekennzeichnet. Adaptiert aus [8].

Die Wirkstruktur des Elements *ALF* ist mit dem Wirkelement *Sprachverarbeitung* erweitert. Der Inhalt des Elements ist in Abbildung 3.2 dargestellt. Das Wirkelement *ALF* wird detaillierter aufgeschlüsselt und stellt hierarchisch eine tiefer liegende Ebene des Umfeldmodells dar. Dort befinden sich die Verknüpfungen von einzelnen Softwareartefakten, welche unter anderem die unterschiedlichen Betriebsmodi sowie die Sprachverarbeitung abbilden.

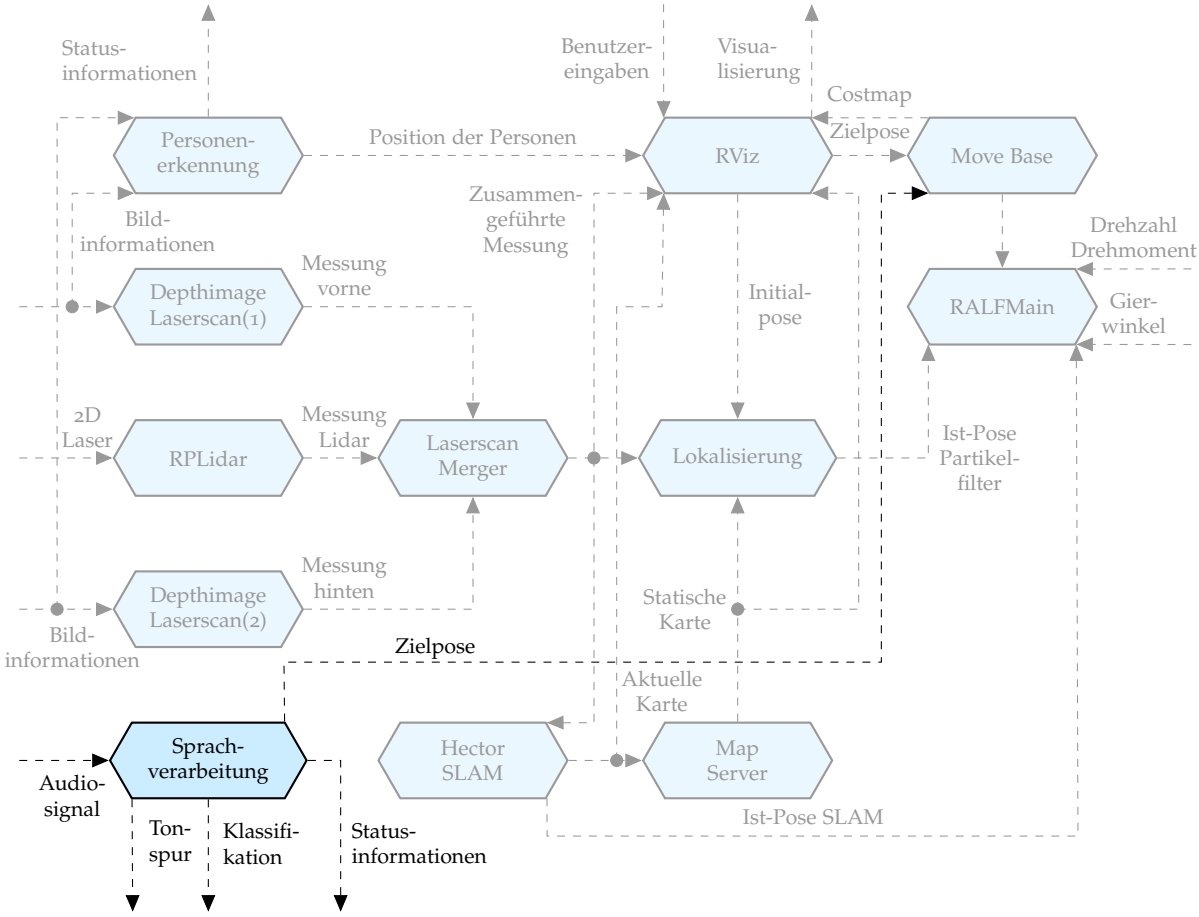


Abbildung 3.2: Darstellung der Wirkstruktur des Wirkelementes *ALF*. Die für diese Arbeit relevanten Wirkelemente und Informationsflüsse sind hervorgehoben. Die Sprachverarbeitung gibt eine Zielpose, die Tonspur und die Klassifikation des Transkripts aus. Die eingehende Verbindung ist das Audiosignal aus der *Kinect*-Sensorik. Die Tonspur ist ausgehend, um diese auch auf einem externen System zu nutzen. Für diese Arbeit irrelevanten Wirkelemente sind transparent dargestellt. Adaptiert aus [8].

In dieser Ebene ist die entwickelte Sprachverarbeitung mit deren aus- und eingehenden Verbindungen lokalisiert. Besonderer Fokus liegt dabei auf der ausgehenden Klassifikation. Diese ist im Umfeldmodell als Transitionsbedingungen des Zustandsautomaten interpretierbar. *ROS* wird angewendet, um einen Datenaustausch zwischen Softwareartefakten sowie die Integration hochautomatisierter Fahrfunktionen zu ermöglichen [8].

Ein *ROS*-Netzwerk besteht aus verschiedenen Knoten (*Nodes*). Die Knoten kommunizieren über einen *Parameter-Server*, *Topics* oder *Services* miteinander. Der *Parameter-Server* wird auf einem *Master* ausgeführt, über diesen werden Konfigurationsdateien ausgetauscht. *Topics* ermöglichen es, Datenpakete periodisch bereitzustellen (*publish*) oder zu abonnieren (*subscribe*). Nachkommend werden die Anforderungen an die Sprachverarbeitung beschrieben. [46]

3.2 Anforderungserhebung und Verifikationsplan

Auf Grundlage der Umfeldmodellierung aus Abbildung 3.1 werden an den gesamten Prozess der Sprachverarbeitung Anforderungen erhoben und in einem Lastenheft festgehalten. Die Anforderungen spezifizieren einzelne signalverarbeitende Schritte in der Sprachverarbeitung. Ferner wird ein Verifikationsplan erstellt, welcher Tests für jede erstellte Anforderung enthält. Die Tabelle 3.1 listet die erhobenen Anforderungen auf.

Tabelle 3.1: Anforderungen gegen die Sprachverarbeitung des ALF.

Nr.	Titel der Anforderung
1	Auslösen einer Sprachaufnahme durch manuelle Betätigung
2	Erzeugen und Bereitstellen einer Tonspur der Dauer $t = 5\,\mathrm{s}$
3	Erkennung und Klassifizierung von bedienungsorientierter Sprache
4	Erkennen von benutzerdefinierten Schlagwörtern

Die Wortgruppen, welche als bedienungsorientiert eingestuft werden, sind im Anhang A.1.3 aufgelistet. Eine genauere Beschreibung der Anforderungen ist im Lastenheft festgehalten. Dies und der zugehörige Verifikationsplan befinden sich im Anhang A.1.2 sowie in Abschnitt 5.2. Im Weiteren Fortgang wird die Modellierung der Wirkstrukturen erläutert, welche in der Verarbeitungskette der Sprachverarbeitung notwendig sind.

3.3 Wirkstruktur der Sprachverarbeitung

Die Abbildung 3.3 zeigt das Wirkelement *Sprachverarbeitung*. Dieses stellt eine weitere, tiefer liegende Schicht der Modellierung aus Abbildung 3.2 dar. Die Wirkstruktur zeigt den Prozess des Sprachverarbeitungssystems. Dabei wird ein Audiosignal mit der vorhandenen *Kinect*-Sensorik aufgenommen und durch die Spracherkennung transkribiert. Die Erklärung und Modellierung dessen wird in Abschnitt 3.4 weiter verdeutlicht. Das resultierende Transkript wird einer Schlagworterkennung sowie Sprachklassifizierung unterzogen.

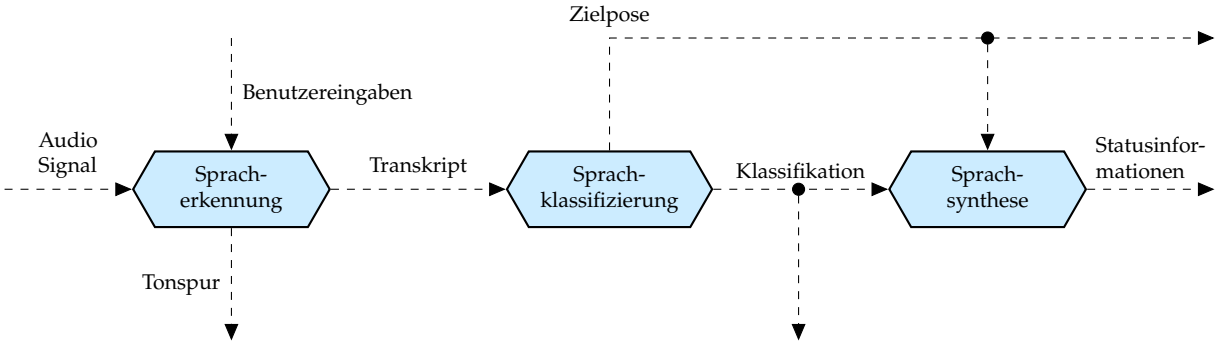


Abbildung 3.3: Darstellung der Wirkstruktur des Wirkelements *Sprachverarbeitung*. Die Sprachverarbeitung transkribiert Sprache aus einem Audiosignal und klassifiziert das resultierende Transkript. Außerdem wird in dem Transkript nach Zielen gesucht und bei Existenz im *ROS* Netzwerk veröffentlicht. Die ausgehenden Statusinformationen werden über einen Lautsprecher an das Umfeld ausgegeben. Die Tonspur wird im gesamten System durch *ROS* bereitgestellt. Dies ermöglicht eine Sprachverarbeitung auf externen Geräten.

Die Sprachsynthese gibt über den Lautsprecher des Systems entsprechende Statusinformationen an das Umfeld aus. Die Beschreibung und Modellierung der Sprachklassifizierung erfolgt in Abschnitt 3.5. Nachfolgend wird der Unterprozess *Spracherkennung* erläutert und modelliert.

3.4 Prozess der Spracherkennung

Die Spracherkennung verarbeitet eine Tonspur zu einem Transkript, zu diesem Zweck ist eine Sprachaufnahme von Nöten. Der Anforderungsliste ist zu entnehmen, dass die Aufnahme durch eine manuelle Betätigung ausgelöst und eine Dauer von $t = 5\,\mathrm{s}$ haben muss. Eine Fernbedienung ist bereits durch die Vorgängerprojekte in das *ROS*-Netzwerk eingebunden [8, 7]. Der dazugehörige Knoten veröffentlicht die Schalterbetätigungen, jene werden durch das repräsentierende Wirkelement *Schalter-Detektion* periodisch aus dem *ROS*-Netzwerk abonniert und folglich ausgewertet [47, 48]. Dies ermöglicht die Nutzung als manuelle Betätigung und das Auslösen der Sprachaufnahme. Die Tonspur wird mit einem Mikrofon der vorhandenen *Kinect*-Sensorik aufgenommen und in das *ROS*-Netzwerk veröffentlicht. Die Modellierung der Wirkstruktur des Elementes *Spracherkennung* ist in Abbildung 3.4 dargestellt.

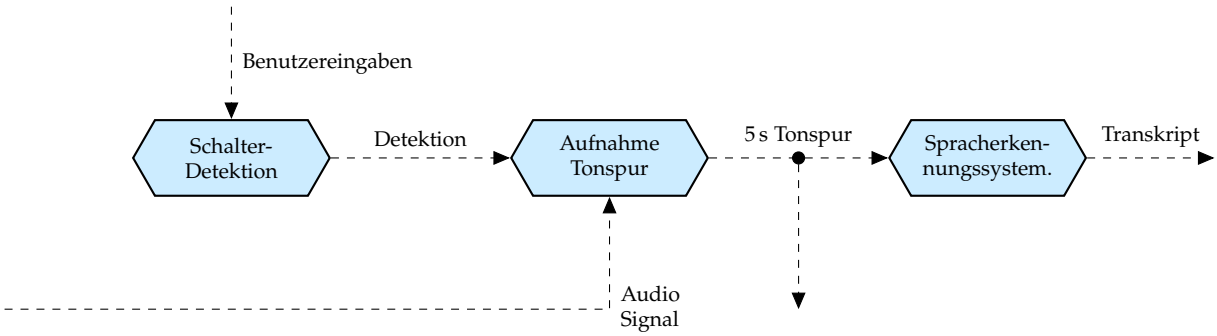


Abbildung 3.4: Abbildung des Wirkelements *Spracherkennung*. Auf eine manuelle Betätigung hin wird eine Tonspur der Dauer $t = 5\,\mathrm{s}$ aufgenommen und anschließend transkribiert. Die Tonspur und das Transkript werden im *ROS*-Netzwerk veröffentlicht.

Ein Spracherkennungssystem abonniert und transkribiert nachfolgend die Sprachaufnahme und stellt das Transkript ebenfalls in dem Netzwerk bereit. Jenes wird nachfolgend einer Klassifizierung und Schlagworterkennung unterzogen. Hinsichtlich des Spracherkennungssystems können verschiedene Modelle und Entwicklungsumgebungen eingesetzt werden. Der Abschnitt 3.6 beschreibt und erörtert, die in dieser Arbeit verwendeten. Anschließend erfolgt eine Konkretisierung der Sprachklassifizierung.

3.5 Prozess der Sprachklassifizierung

Aus dem bereitgestellten Transkript werden durch OHE oder den Spaltennummern des zugehörigen Vokabulars die Merkmale extrahiert. Grundlage dafür bildet der BOW, welcher in der Trainingsphase des Klassifizierenden neuronalen Netz gebildet wird. Wie bereits erwähnt, wird das Transkript auch einem Textabgleichverfahren unterzogen. Dadurch lassen sich potenzielle Zielposen des Fahrzeugs identifizieren. Der Klassifikator erkennt Handlungen aus dem Transkript und ordnet diesen Wahrscheinlichkeiten zu. Die Abbildung 3.5 zeigt den Inhalt des Wirkelements *Sprachklassifizierung*.

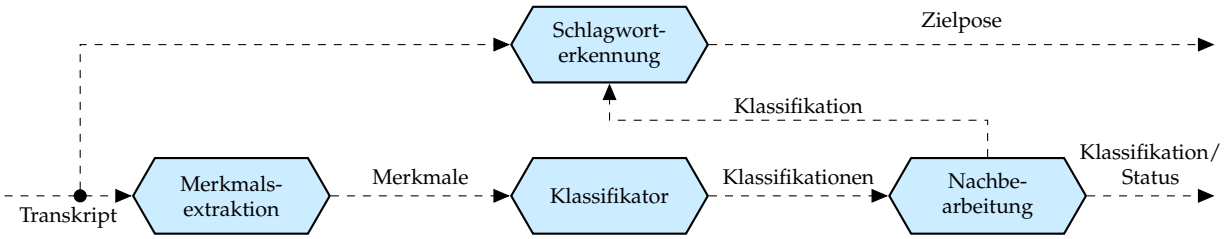


Abbildung 3.5: Dargestellt ist die Wirkstruktur des Elementes *Sprachklassifizierung*. Die aus dem Transkript extrahierten Merkmale werden mithilfe eines Klassifikators in Kategorien eingeordnet. Vor Veröffentlichung des finalen Ergebnisses werden diese nachbearbeitet. Eine Schlagworterkennung sucht mit einem Textabgleichverfahren nach vorher definierten Zielen und veröffentlicht diese.

Die grundlegenden Tätigkeiten des Fahrzeugs wie zum Beispiel *Fahren*, *Lokalisieren* oder *Warten* bleiben dabei immer gleich. Ziele und Gegenstände können sich aber laufend ändern, sodass Textabgleichverfahren zur Erkennung von Objekten und Positionen genutzt werden. Demnach definiert sich die Menge der zu kategorisierenden Klassen zu:

$$\mathcal{C} = \{0, 1, 2, 3, 4, 5\}. \tag{3.1}$$

Die Elemente der Menge $\mathcal{C}$ stellen eine numerische Codierung der Kategorien dar [33]. Dabei stehen die Elemente $0, ..., 4$ für Handlungen respektive Tätigkeiten des Fahrzeugs, während die Klasse 5 lediglich eine Ausschussklasse ist.

Die ausgehenden Klassifikationen entsprechen der beschriebenen diskreten Wahrscheinlichkeitsverteilung:

$$\hat{y}_j = P(c_j = j|\underline{y}) = \psi(\underline{y}) = \frac{e^{y_j}}{\sum_{i=0}^{|C|} e^{y_i}} \text{ mit } j = 0, 1, ..., 5. \tag{3.2}$$

Die einzelnen Wahrscheinlichkeiten werden auch als Konfidenzen bezeichnet [33]. Darüber hinaus findet eine Zuweisung zwischen den numerischen Codierungen und den tatsächlichen Handlungen des Fahrzeugs statt. Dieser Zusammenhang wird in Tabelle 3.2 dargestellt. Diese Klassen gelten als potenzielle Transitionsbedingungen für den erwähnten Zustandsautomaten.

Tabelle 3.2: Die Zuordnung der numerischen Klassen zu den Ausführenden Handlungen des Fahrzeugs.

Klasse c_j	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
Handlung	*Drive*	*SLAM*	*Wait*	*Localization*	*Stop*	*Unknown*

Der resultierende Vektor $\hat{\underline{y}}$ unterliegt im Folgenden einer Nachbearbeitung. Dabei wird das Maximum der sechs vorliegenden Konfidenzen ermittelt und die Klasse mit der höchsten Wahrscheinlichkeit vor Veröffentlichung einer Schwellwertprüfung unterzogen. Hier werden Grenzwerte für jede Kategorie festgelegt. Beim Unterschreiten der Schwelle wird lediglich die Statusinformation weitergeleitet und nicht die Klassifikation. In der Tabelle 3.3 werden die zulässigen Schwellwerte den entsprechenden Klassen zugeordnet. Die Statusinformation beinhaltet den Vektor $\hat{\underline{y}}$ und das Transkript.

Tabelle 3.3: Die Schwellwerte zur finalen Veröffentlichung von Klassifikationen.

Nr.	Zulässige Wahrscheinlichkeit
0	$P(c_j = 0\|\underline{y}) \geq 0,8$
1	$P(c_j = 1\|\underline{y}) \geq 0,8$
2	$P(c_j = 2\|\underline{y}) \geq 0,4$
3	$P(c_j = 3\|\underline{y}) \geq 0,4$
4	$P(c_j = 4\|\underline{y}) \geq 0,4$
5	$P(c_j = 5\|\underline{y}) \geq 0,4$

Die Kategorien mit einem Schwellwert von $P(c_j|\underline{y}) \geq 0,4$ stellen risikominimale Klassifikationen dar. Die Zuordnungen mit dem Schwellwert $P(c_j|\underline{y}) \geq 0,8$ lösen in der Regel Fahrfunktionen des ALF aus, daher wird für diese Werte eine höhere Konfidenz verlangt. Die Begründung der Werte erfolgt mit der Evaluation aus Kapitel 4. Nachfolgend werden die Spracherkennungssysteme und Sprachklassifikatoren erläutert. Zudem erfolgt ein Blick auf die verwendeten Entwicklungsumgebungen.

3.6 Verwendete Entwicklungsumgebungen und Netzwerkarchitekturen

Zur Umsetzung der entwickelten Wirkstrukturen werden verschiedene Entwicklungsumgebungen, Spracherkennungssysteme und Sprachklassifikatoren eingesetzt. Neben *ROS* wurden *PyCharm* mit der Programmiersprache *Python* sowie *Tensorflow* und *Keras* eingesetzt. *Tensorflow* als auch *Keras* bieten eine *Python* Entwicklungsumgebung, welche eine breite Anwendung in den Bereichen des maschinellen Lernens findet [49]. Die Softwarebibliothek stellt Schichten und Trainingsalgorithmen für jegliche Art von neuronalen Netzwerken bereit [50]. Viele Dienste und Softwarelösungen in den Bereich der automatischen Spracherkennung stellen eine *Python* Bibliothek bereit, weshalb diese Programmiersprache hier Anwendung findet [18]. Des Weiteren bietet *Python* eine Schnittstelle zu dem *ROS*, wodurch der Datenaustausch zwischen bereits bestehender Softwarearchitektur und des Sprachverarbeitungssystems ermöglicht wird [51].

In der Schlagworterkennung wird die *Brute Force* Methode in Kombination mit den *Soundex* und *Metaphone* Verschlüsselungsalgorithmen verwendet. Ein weniger rechenintensiver und komplizierterer Algorithmus ist dabei nicht notwendig, da es sich um einen kurzen zu überprüfenden Text handelt. Durch die Überprüfung des nach *Soundex* und *Metaphone* verschlüsselten Textes wird die phonetische Ähnlichkeit verschiedenster Wörter mit in Betracht gezogen. Dies ermöglicht die Erkennung eines Schlagwortes trotz vermeintlich fehlerhafter Transkription.

Als Spracherkennungssysteme werden in dieser Arbeit die offline Softwarelösungen *DeepSpeech* (DS), *Espnet* sowie *Pocketsphinx* (PS) eingesetzt und bezüglich des bedienungsorientierten Vokabulars evaluiert. Die ausgewählten Softwarelösungen bieten *Python* Bibliotheken und ermöglichen dadurch die Integration in das Fahrzeug [18]. Weiterhin erreichen die Lösungen ähnliche Ergebnisse hinsichtlich der in Abschnitt 4.1 eingeführten Wortfehlerraten wie vergleichbare Spracherkennungsdienste und Softwarelösungen [20, 52, 21, 25]. Des Weiteren wird der Online-Dienst von *IBM-Watson* genutzt, um die Ergebnisse mit einem *Cloud Computing*-Dienst zu vergleichen.

Für die Sprachklassifikatoren werden neuronale Netzwerke eingesetzt. Um das beschriebene Ziel, Sätzen mit bedienungsorientierten Inhalt die Klassen aus Tabelle 3.2 zuzuweisen, werden dafür unterschiedliche Netzwerkschichten kombiniert. Ein rekurrentes neuronales Netzwerk findet Anwendung, da Wortgruppen als in Abhängigkeit stehende Datensequenzen betrachtet werden können [32]. Weiterhin wird durch eine *Embedding*-Schicht die kontextbezogene vektorielle Darstellung gewonnen [38].

Die Auswahl der Netzwerkarchitekturen ist bedingt durch die zugrundeliegende Merkmalsextrahierung der Abschnitte 2.5 und 2.6. Dabei wird das *Word Embedding* und *One-Hot Encoding* angewendet, um Wörter, Sätze und Wortgruppen als Vektoren zur numerischen Berechnung darzustellen. Bei dem OHE stellt jede Dimension des resultierenden Vektors ein Merkmal dar, dabei spielt die Reihenfolge respektive Anordnung in einem Satz keine Rolle. Deshalb stellt die dadurch verschlüsselte Wortgruppe nicht mehr die ursprüngliche Datensequenz dar. Diese Codierung repräsentiert lediglich die vorhandenen Wörter, jedoch nicht deren ursprüngliche Reihenfolge. Die Verwendung eines RNN ist deshalb mit einer *Embedding*-Schicht verknüpft [32]. Durch den strukturbedingten Speicher einer rekurrenten Schicht liegt die Verwendung für eine Klassifikation von Wortgruppen beziehungsweise Sätzen nahe. Die Anordnung von Wörtern in einem Satz stehen in Abhängigkeit zueinander und stellen eine Datensequenz dar [12]. Für die Prädiktion einer bestimmten Klasse eines Satzes ist dabei die Klassifikation der zurückliegenden Sequenz von Bedeutung [32]. Daraus ergeben sich drei verschiedene Netzwerkarchitekturen, welche im Folgenden charakterisiert werden.

Vorwärtspropagierendes neuronales Netzwerk:

Die Erste in dieser Arbeit genutzte Netzwerkarchitektur ist ein vorwärtspropagierendes Netzwerk (FFW) aus Abschnitt 2.2 und ist in Abbildung 3.6 schematisch dargestellt. Der Eingangsvektor $\underline{I}$ wird durch die OHE Methode erzeugt und besitzt demzufolge die Dimension $|\mathcal{V}|$. Dieser wird zu einem als diskrete Wahrscheinlichkeitsverteilung der Klassen $\mathcal{C}$ interpretierbaren Vektor $\underline{\hat{y}}_{1\times 6}$ transformiert.

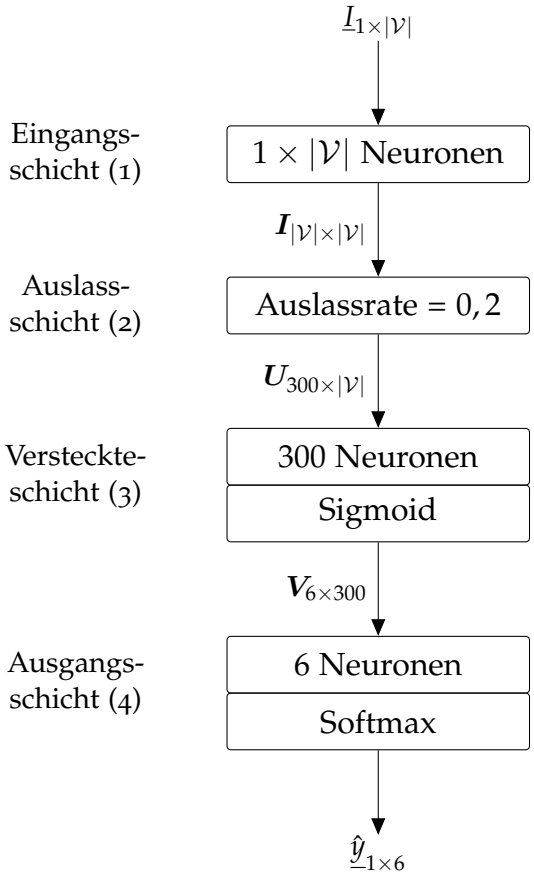


Abbildung 3.6: Abbildung einer Netzwerkarchitektur des Vorwärtspropagierenden neuronalen Netzwerk zur Textklassifikation. Zwischen den abgebildeten Schichten finden Matrizenmultiplikationen statt. Die Matrix $\boldsymbol{I}$ ist die Einheitsmatrix der angegebenen Dimension. Die Größe der Gewichtsmatrizen $\boldsymbol{U}_{300\times|\mathcal{V}|}$ und $\boldsymbol{V}_{6\times 300}$ steht in Abhängigkeit zur Neuronenanzahl der Schichten.

Zwischen den Schichten befinden sich Gewichtsmatrizen $\boldsymbol{U}$, um die mathematische Modellierung von Neuronen aus der Gleichung (2.1) zu erfüllen. Die Dimension der Matrizen variiert in Abhängigkeit zur Mächtigkeit des Vokabulars und der Neuronenanzahl in den verschiedenen Schichten. Eine Auslassschicht setzt bei dem Trainingsvorgang entsprechend der angegeben Auslassrate zufällig die Einträge des anliegenden Vektors auf 0 [53]. Wird ausschließlich vorwärtspropagiert, so werden keine Einträge ausgelassen. Dies soll eine potenzielle Überanpassung an die verwendeten Trainingsdaten verhindern [34].

Vorwärtspropagierendes neuronales Netzwerk mit Embedding-Schicht:

Die zweite verwendete Netzwerkarchitektur ist das vorwärtspropagierende Netzwerk mit einer *Embedding*-Schicht (FFWE). Diese Schicht befindet sich statt der herkömmlichen Eingangsschicht am Anfang eines Netzwerkes. Dem Eingangsvektor wird mit der in Abschnitt 2.6 beschriebenen Methodik eine neue vektorielle Darstellung zugewiesen. Im Anschluss daran erfolgt eine Klassifikation, die mit der aus dem vorherigen Abschnitt vergleichbar ist. Die Abbildung 3.7 stellt die Anordnung der Schichten und die zugehörigen Gewichtsmatrizen dar.

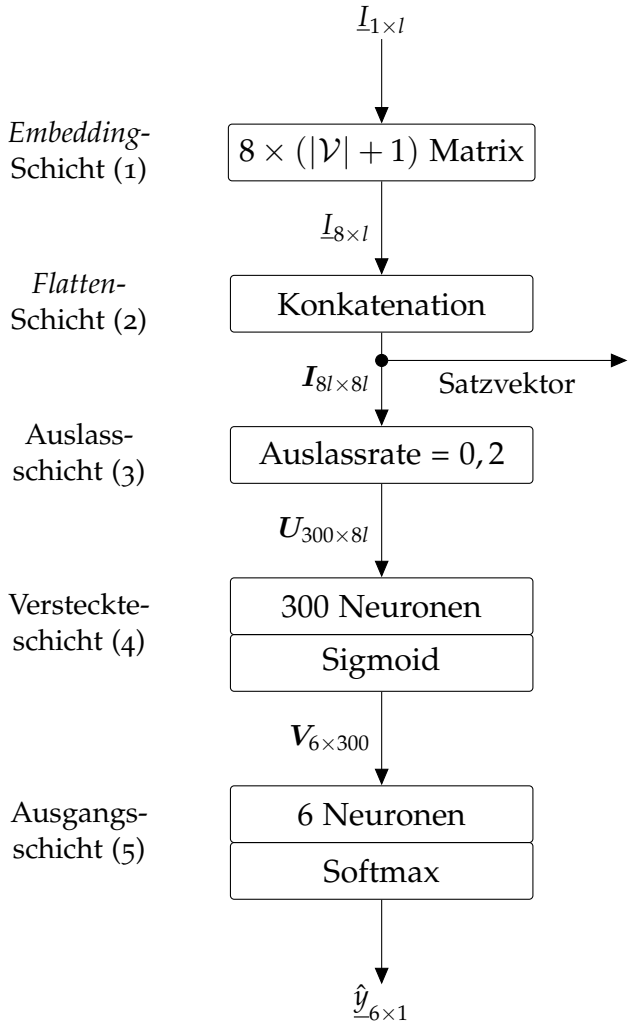


Abbildung 3.7: Prinzipdarstellung der Netzwerkarchitektur mit *Embedding*-Schicht und darauffolgendem vorwärtspropagierenden Netzwerk zur Klassifikation von Sätzen. Durch Unterbrechen des vorwärtspropagieren kann die dicht besetzte vektorielle Darstellung des zu klassifizierenden Satzes abgeleitet werden.

Die Länge l beschreibt hierbei den längsten Satz und die Umsetzungstabelle der *Embedding*-Schicht erzeugt für jedes enthaltene Wort einen Vektor der Dimension $d = 8$. Die *Flatten*-Schicht konkateniert die einzelnen Komponenten, sodass eine reellwertige vektorielle Darstellung eines gesamten Satzes entsteht. Speichert man die Resultate nach der *Flatten*-Schicht und wendet eine Hauptkomponentenanalyse an, so entstehen die Darstellungen 5.2 und 5.3 aus Kapitel 5.

Rekurrentes neuronales Netzwerk mit Embedding-Schicht:

Bei der dritten verwendeten Netzwerkarchitektur handelt es sich um ein rekurrentes neuronales Netzwerk mit vorausgehender *Embedding*-Schicht (RNNE). Die Abbildung 3.8 zeigt die Architektur mit den Gewichtsmatrizen und deren Dimensionen.

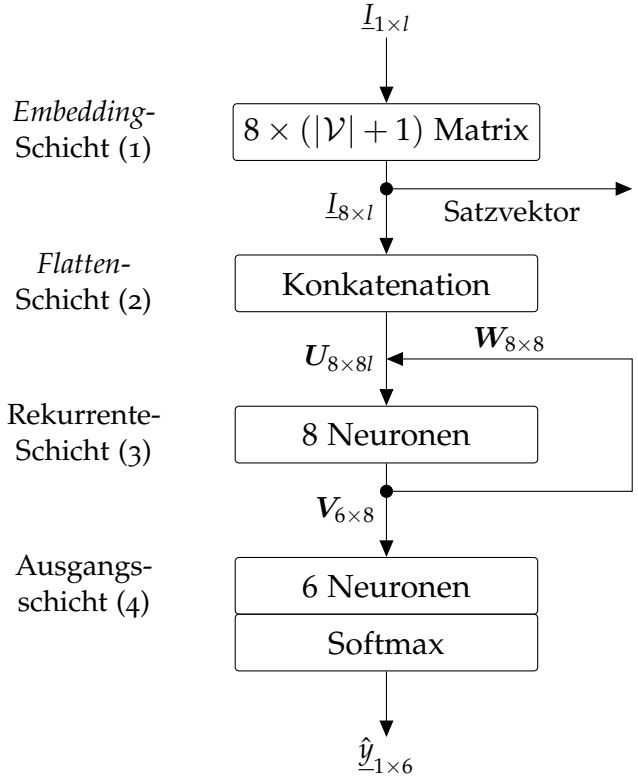


Abbildung 3.8: Schematische Abbildung des rekurrenten neuronalen Netzwerks mit *lookup* Ebene. Die Verarbeitung einer Datensequenz wird durch die *Embedding*-Schicht ermöglicht. Die Verknüpfung der verschiedenen Zustände erfolgt nach der Gleichung (2.7).

Die Transformation der *Embedding*-Schicht ist als Datensequenz aus Abschnitt 2.3 interpretierbar. Jedes Wort stellt dabei eine Teilsequenz dar, welche in folgender Rekurrenten-Schicht weiterverarbeitet wird. Der Ausgang dieser Zelle wird zurückgekoppelt und mit einer anliegenden Eingangssequenz nach der Gleichung (2.7) verrechnet. So entsteht der in Abschnitt 2.3 beschriebene versteckte Zustand. Nachfolgend findet die Evaluation der Spracherkennungssysteme und der vorgestellten Netzwerkstrukturen hinsichtlich verschiedener Datensätze statt.

4 Evaluation der Methodiken

Das folgende Kapitel dient der Evaluation der verwendeten Methoden. Die verschiedenen Systeme und Architekturen werden dafür anhand selbst erstellter Datensätze verglichen. Die Gegenstände der Evaluation sind die in Kapitel 3 vorgestellten Unterprozesse *Spracherkennung* und *Sprachklassifikation*. Die qualitative Bewertung der einzelnen Prozesse findet mithilfe von verschiedenen Metriken statt, welche im Folgenden vorgestellt werden.

4.1 Metriken

Zur Evaluation der Sprachverarbeitung werden die Bewertungskriterien Wortfehlerrate, Genauigkeit der Sprachklassifikatoren und Geschwindigkeit vorgestellt. Da die Fahrmanöver des Fahrzeugs von der korrekten Klassifikation abhängig sind, wird ein prozentualer Anteil der Klassen bei fehlerhafter Klassifikation berechnet. Nachfolgend wird diese Metrik durch $z_{1:j}$ bezeichnet. Dies bildet den sicherheitskritischen Aspekt der falschen Klassenzuordnung ab.

Des Weiteren wird eine mittlere Wortfehlerrate und Konfidenz sowohl für die fehlerhaften als auch die korrekten Klassifikationen bestimmt. Wie Kapitel 3 verdeutlicht, hängt die Klassifikation von der Qualität des Transkripts ab. Darüber hinaus unterliegt die Genauigkeit der Sprachklassifikation einer Schwellwertprüfung. Die beiden Zusammenhänge werden daher durch die berechneten Mittelwerte in die Bewertung eingehen.

Die Grundlage der Evaluation bildet der Datensatz $\mathcal{D}$, aus dem die Teilmengen $\mathcal{D}_{\text{Train}} \subseteq \mathcal{D}$ und $\mathcal{D}_{\text{Val}} \subseteq \mathcal{D}$ hervorgehen. Jener Datensatz wird in Abschnitt 4.2 eingeführt und beinhaltet die Grundwahrheiten sowohl für die Spracherkennung als auch

für die Sprachklassifikation. Durch Berechnungen des Spracherkennungssystems und der Sprachklassifikation entstehen anhand eines zu analysierenden Datensatzes $|\mathcal{D}_x|$ die Ergebnismengen:

$$\mathcal{H} = \{T_1, T_2, ..., T_i\} \text{ und} \tag{4.1}$$

$$\mathcal{P} = \{\underline{\hat{y}}_1, \underline{\hat{y}}_2, ..., \underline{\hat{y}}_i\} \text{ mit } i = |\mathcal{D}_x|. \tag{4.2}$$

Dabei sind die Elemente $T_1, T_2, .., T_i$ der Menge $\mathcal{H}$ die resultierenden Transkripte aus dem verwendeten Spracherkennungssystem. Die Einträge $\underline{\hat{y}}_1, \underline{\hat{y}}_2, ..., \underline{\hat{y}}_i$ der Ergebnismenge $\mathcal{P}$ beschreiben nach der Gleichung (3.2) die als diskrete Wahrscheinlichkeitsverteilung interpretierbaren Ausgangsvektoren der Netzwerkarchitekturen. Im Folgenden werden die Metriken zur Bewertung vorgestellt.

Wortfehlerrate:

Die Wortfehlerrate ist eine bekannte Metrik im Bereich der Spracherkennung und wird durch

$$q = \frac{S + D + I}{N} \text{ mit } S, D, E, N \in \mathbb{N} \tag{4.3}$$

berechnet [12]. In der Gleichung (4.3) beschreiben A, D und E die Anzahl der ausgetauschten, entfernten sowie in eine Hypothese T_i eingesetzten Wörter. Die Summe dieser Werte wird durch die Wortanzahl N der zugehörigen Grundwahrheit des Datensatzes dividiert. Die Grundwahrheit entstammt der vorliegenden Wortgruppe und ist ein Teil des Datensatzes. Die resultierenden Fehlerraten aller Vergleiche zwischen den Hypothesen und Grundwahrheiten eines Datensatzes bilden zusammen die Menge:

$$\mathcal{W} = \{q_1, q_2, ..., q_i\} \text{ mit } q_i \in \mathbb{R}. \tag{4.4}$$

Hinsichtlich der einzelnen Wortfehlerraten berechnet sich der Mittelwert $\overline{q}$ durch:

$$\overline{q} = \frac{1}{i} \sum_{\eta=1}^{i} q_\eta. \tag{4.5}$$

Da die nachfolgende Klassifikation vom Transkript abhängig ist, werden die mittleren Wortfehlerraten

$$\overline{q}_{\mathrm{T}} = \frac{1}{K_{\mathrm{T}}} \sum_{\eta=1}^{K_{\mathrm{T}}} q_{\mathrm{T},\eta} \text{ mit } q_{\mathrm{T},\eta} \in \mathcal{W}_{\mathrm{T}} \ \wedge \ \mathcal{W}_{\mathrm{T}} \subseteq \mathcal{W}, \text{ sowie} \tag{4.6}$$

$$\overline{q}_{\mathrm{F}} = \frac{1}{K_{\mathrm{F}}} \sum_{\eta=1}^{K_{\mathrm{F}}} q_{\mathrm{F},\eta} \text{ mit } q_{\mathrm{F},\eta} \in \mathcal{W}_{\mathrm{F}} \ \wedge \ \mathcal{W}_{\mathrm{F}} \subseteq \mathcal{W} \ , K_{\mathrm{T}}, K_{\mathrm{F}} \in \mathbb{N} \tag{4.7}$$

für die Gesamtheit aller richtig und falsch klassifizierten Hypothesen K_{T} respektive K_{F} berrechnet. $\mathcal{W}_{\mathrm{T}}$ und $\mathcal{W}_{\mathrm{F}}$ stellen dabei Teilmengen von $\mathcal{W}$ dar und beinhalten die Wortfehlerraten der korrekt (T) beziehungsweise falsch (F) klassifizierten Hypothesen. Die Zugehörigkeit wird in den entsprechenden Indizes gekennzeichnet. Dies setzt die Qualität des Transkripts in Beziehung zur Klassifikation.

Genauigkeit:

Die Genauigkeit g ist in dieser Arbeit das Verhältnis zwischen der Anzahl aller korrekt klassifizierten Hypothesen und dem Betrag eines zu analysierenden Datensatzes $\mathcal{D}_x$. In der Gleichung (4.8) ist die dazugehörige Berechnungsvorschrift dargestellt:

$$g = \frac{K_{\mathrm{T}}}{|\mathcal{D}_x|}. \tag{4.8}$$

Darüber hinaus erfolgt eine Bestimmung der Mittelwerte für die Konfidenzen α der korrekten und fehlerhaften Klassifikationen mit:

$$\overline{\alpha}_{\mathrm{T}} = \frac{1}{K_{\mathrm{T}}} \sum_{\eta=1}^{K_{\mathrm{T}}} \max(\underline{\hat{y}}_{\mathrm{T},\eta}) \text{ mit } \underline{\hat{y}}_{\mathrm{T},\eta} \in \mathcal{P}_{\mathrm{T}} \ \wedge \ \mathcal{P}_{\mathrm{T}} \subseteq \mathcal{P}, \text{ sowie} \tag{4.9}$$

$$\overline{\alpha}_{\mathrm{F}} = \frac{1}{K_{\mathrm{F}}} \sum_{\eta=1}^{K_{\mathrm{F}}} \max(\underline{\hat{y}}_{\mathrm{F},\eta}) \text{ mit } \underline{\hat{y}}_{\mathrm{F},\eta} \in \mathcal{P}_{\mathrm{F}} \ \wedge \ \mathcal{P}_{\mathrm{F}} \subseteq \mathcal{P}. \tag{4.10}$$

$\mathcal{P}_T$ und $\mathcal{P}_F$ stellen dabei Teilmengen von $\mathcal{P}$ dar und beinhalten die Ausgangsvektoren der korrekt (T) beziehungsweise falsch (F) klassifizierten Hypothesen. Die Zugehörigkeit wird ebenfalls in den entsprechenden Indizes gekennzeichnet. Die Berechnungen dienen zur Bewertung der resultierenden Konfidenzen. So wird bei einer korrekten Klassifikation ein Wert von $\alpha_T \rightarrow 1$ angestrebt. Bei einer falschen Kategorisierung hingegen ein Wert $\alpha_F << \overline{\alpha_T}$. Mithilfe des α_F Maßes sind die Schwellwerte aus Abschnitt 3.3 zu dimensionieren.

Geschwindigkeit:

Die Geschwindigkeit der Sprachverarbeitung ist zum größten Teil abhängig von der Spracherkennung. Die Dauer der Sprachklassifikation ist dabei zu vernachlässigen. Während der Entwicklung dieser traten keine nennenswerten Latenzen auf. Infolgedessen werden ausschließlich die eingesetzten Spracherkennungssysteme überprüft. Gemessen wird die Transkriptionszeit t_t bis eine Audiodatei der Länge $t = 7\,s$ transkribiert wurde. Die Messung erfolgt anhand der Datensätze aus Abschnitt 4.2. Als Grundlage dient die Linux basierte Hauptsteuerung des ALF. Die Tabelle 4.1 listet die verbauten Komponenten, welche Einfluss auf die Messungen nehmen, auf.

Tabelle 4.1: Auflistung von Komponenten der Linux Hauptsteuerung des ALF [7].

Komponente	Typ	Eigenschaften
CPU	Intel i7-7700	4 x 3,6 - 4,0 GHz+ HT, iGPU
RAM	Corsair 2 x 4 GB	DDR4 PC-2400

Prozentualer Anteil fehlerhafter Klassen:

Mithilfe der Werte: $z_{1:j}$ wird der prozentuale Anteil einer Klasse bezüglich aller fehlerhaften Klassifikationen angegebenen. Dies ermöglicht die Bewertung des sicherheitstechnischen Aspekts von fehlerhaften Klassifikationen. Der Anteil einer Klasse errechnet sich durch:

$$z_{c_j} = \frac{F_{c_j}}{K_F} \text{ mit } F_{c_j} \in \mathbb{N}. \tag{4.11}$$

Dabei steht F_{c_j} für die Anzahl aller fehlerhaften Klassifikationen der Kategorie c_j, jene wurden in der Gleichung (3.2) eingeführt. Die Klassen c_0 und c_1 können dabei eine Bewegung des Fahrzeugs auslösen. Deshalb werden diese bei fehlerhafter Klassifikation als sicherheitskritisch bewertet. Nachfolgend werden die Datensätze vorgestellt, anhand derer die Sprachverarbeitung evaluiert wird.

4.2 Datensätze

Da für die Sprachklassifikation aus Abschnitt 3.5 neuronale Netze eingesetzt werden, sind diese vor einer erfolgreichen Klassifikation zu trainieren. Wie bereits in Kapitel 1 beschrieben, führt das ALF eine Interpretation von Sätzen mit bedienungsorientiertem Inhalt durch. Dazu wurden in Abschnitt 3.5 Klassen eingeführt, mit denen dies umgesetzt wird. Hinsichtlich der Lösung dieses spezifischen Ziels, wird ein eigener, auf das Problem angepasster Datensatz beziehungsweise Textkorpus der Form

$$\mathcal{D} = \{\{S_1, c_j\}, \{S_2, c_j\}, ..., \{S_i, c_j\}\} \text{ mit } i = 462 \tag{4.12}$$

erstellt. Dieser repräsentiert eine Menge aus 462 Grundwahrheiten $S_1, ..., S_i$ mit einer zugehörigen Kategorie c_j. Aus den erstellten Grundwahrheiten entsteht das bedienungsorientierte Vokabular. Die Abbildung 4.1 zeigt die Verteilung der einzelnen Grundwahrheiten und deren zugehörige Kategorie.

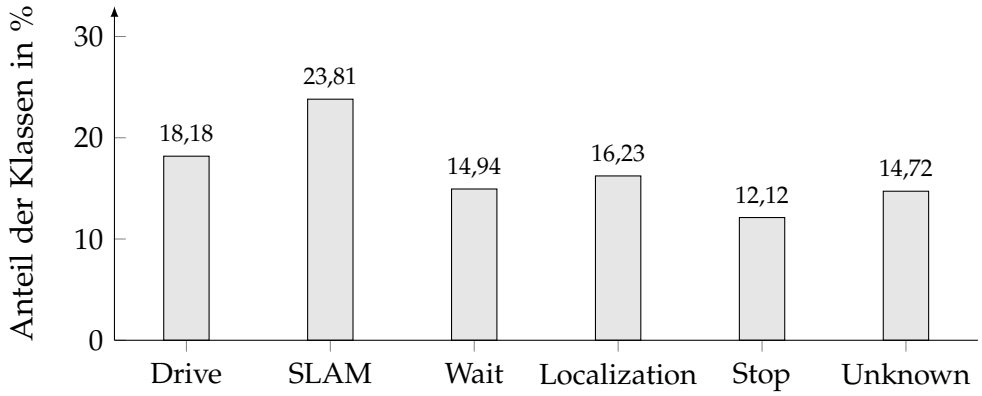


Abbildung 4.1: Die Abbildung der prozentualen Klassenverteilung des erstellten Datensatzes. Den numerischen Klassen c_j wird mithilfe der Tabelle 3.2 die auszuführende Handlung des Fahrzeugs zugewiesen.

Der Datensatz wird anschließend in einen Trainings- und Validierungsdatensatz aufgeteilt. Dies erfolgt durch eine zufällige Auswahl der darin enthaltenden Grundwahrheiten samt zugehöriger Klassen. Dadurch werden entwicklungsbedingte Häufungen von bestimmten Wortgruppen beziehungsweise Sätzen in einen der beiden resultierenden Datensätze vermieden.

Das Verhältnis der Teilung beträgt drei zu eins. Daraus resultiert der Trainingsdatensatz $\mathcal{D}_{\text{Train}}$ mit $|\mathcal{D}_{\text{Train}}| = 346$, sowie der Validierungsdatensatz $\mathcal{D}_{\text{Val}}$ mit $|\mathcal{D}_{\text{Val}}| = 116$. Die Darstellung 4.2 zeigt die Klassenverteilung nach der Aufteilung in Trainings- und Validierungsdatensatz. Letzterer wird nicht für den Lernprozess der erstellten Modelle genutzt, dieser dient ausschließlich zur Überprüfung der Leistungsfähigkeit eben jener [33].

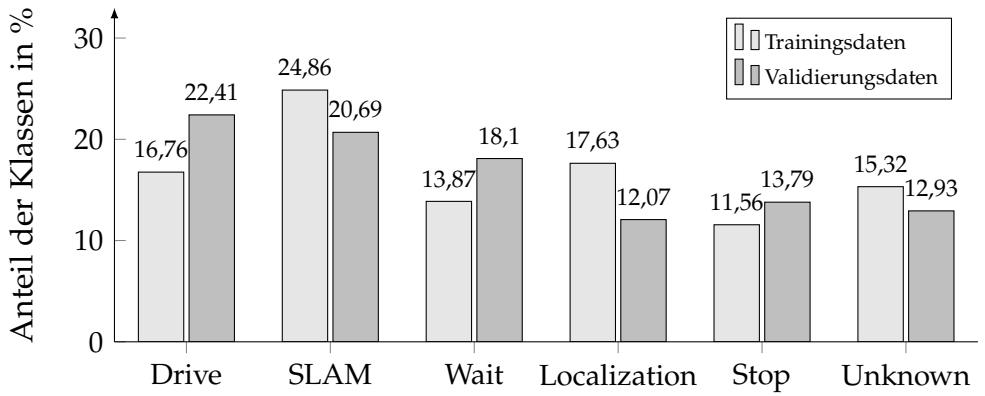


Abbildung 4.2: Darstellung der prozentualen Klassenverteilung $\mathcal{D}_{\text{Train}}$ und $\mathcal{D}_{\text{Val}}$ nach der Aufteilung von $\mathcal{D}$. Der Trainingsdatensatz wird verwendet, um die in Abschnitt 3.6 eingeführten Netzwerkarchitekturen zu trainieren. Der Validierungsdatensatz dient zur Evaluation.

Die Evaluation ausschließlich mit dem Validierungsdatensatz $\mathcal{D}_{\text{Val}}$ würde nur den Prozess der Sprachklassifikation bewerten. Die gesamte Sprachverarbeitung besteht, wie Kapitel 3 zeigt, aus mehreren Schritten. Darüber hinaus wird in Abschnitt 4.1 bereits beschrieben, dass die Klassifikation vom Transkript abhängt. Das Evaluieren nur mit dem Datensatz $\mathcal{D}_{\text{Val}}$, lässt die genannte Abhängigkeit außer Acht, da eine Hypothese in nahezu jedem Vorgang mit auftretender Wortfehlerrate ≥ 0 zu klassifizieren ist.

Deshalb werden mithilfe der angelegten Datensätze Audiodateien aufgenommen. Darauf befinden sich Tonaufnahmen mit Sätzen aus der zugehörigen Datenmenge, welche von unterschiedlichen Sprechern aufgenommen wurden. Die Sprecher waren in diesem Fall 24 synthetische Stimmen, welche sich in Akzent und Qualität unterschieden.

Daraus ergeben sich für die Validierungs- und Trainingsdatensätze 2784 respektive 8304 Tonaufnahmen mit einer Länge von 7 s. Diese bilden die Datensätze $\mathcal{D}_1$ und $\mathcal{D}_2$. Ein dritter Datensatz $\mathcal{D}_3$ wird mithilfe von Probanden erstellt. Dabei sprechen 12 Personen 10 zufällig ausgewählte Sätze aus $\mathcal{D}$ ein. Hieraus ergibt sich durch Aussortieren von Fehlerhaften eine Anzahl von 119 Aufnahmen. Jede Audioaufnahme besitzt dabei die Grundwahrheiten S_i und c_j aus der zugehörigen Datenmenge. In der Tabelle 4.2 werden die erstellten Datensätze bezüglich ihrer Parameter zusammengefasst.

Tabelle 4.2: Zusammenfassende Übersicht der sechs erstellten Datensätze. WAV (syn-S) steht dabei für die Aufnahmen mit synthetischen Stimmen. WAV (Prob) für Tonspuren von Probanden.

Datensatz	Datenpunkte	Art	Herkunft	Format
$\mathcal{D}$	462	Wortgruppen + Klassen	-	Text
$\mathcal{D}_{\text{Train}}$	346	$\mathcal{D}_{\text{Train}} \subseteq \mathcal{D}$	-	Text
$\mathcal{D}_{\text{Val}}$	116	$\mathcal{D}_{\text{Val}} \subseteq \mathcal{D}$	-	Text
$\mathcal{D}_1$	2784	Tonspuren + Klassen	$\mathcal{D}_{\text{Val}}$	WAV (syn-S)
$\mathcal{D}_2$	8304	Tonspuren + Klassen	$\mathcal{D}_{\text{Train}}$	WAV (syn-S)
$\mathcal{D}_3$	119	Tonspuren + Klassen	$\mathcal{D}$	WAV (Prob)

Nachfolgend werden die Ergebnisse hinsichtlich der drei Datensätze $\mathcal{D}_1$, $\mathcal{D}_2$ und $\mathcal{D}_3$ mithilfe der eingeführten Kriterien präsentiert. Zudem wird der Trainingsprozess der eingeführten Netzwerkarchitekturen beschrieben.

4.3 Training der Netzwerkarchitekturen

Die drei offline Spracherkennungssysteme erzeugen anhand des Datensatzes $\mathcal{D}_1$ die Hypothesen $\mathcal{H}$, welche zusammen mit den Grundwahrheiten des Datensatzes $\mathcal{D}_{\text{Train}}$ zum Trainieren genutzt werden. Die Gesamtanzahl der Trainingssätze betrug daher:

$$24 \cdot 346 \cdot 3 + 346 = 25258. \tag{4.13}$$

Mithilfe des Datensatzes $\mathcal{D}_2$ und der vier Spracherkennungssysteme werden Hypothesen erzeugt, die als Testsätze genutzt werden. Daraus entsteht die Gesamtanzahl der zu überprüfenden Sätze mit:

$$24 \cdot 116 \cdot 4 = 11136. \tag{4.14}$$

Die zu den Hypothesen zugehörigen Klassen c_j wurden dabei im Transkriptionsprozess der Spracherkennungssysteme gespeichert. Die Sätze zur Validierung werden aus dem Datensatz $\mathcal{D}_{\text{Val}}$ entnommen. Das Training der Netzwerke erfolgte mit 200 Lernepochen und einer *Batch Size* von 50. Nach jeder Lernepoche fand eine Kalkulation der Genauigkeiten g_{Train}, g_{Val} und g_{Test} statt. In Abbildung 4.3 sind die Genauigkeiten der unterschiedlichen Netzwerke über die Lernepochen aufgetragen.

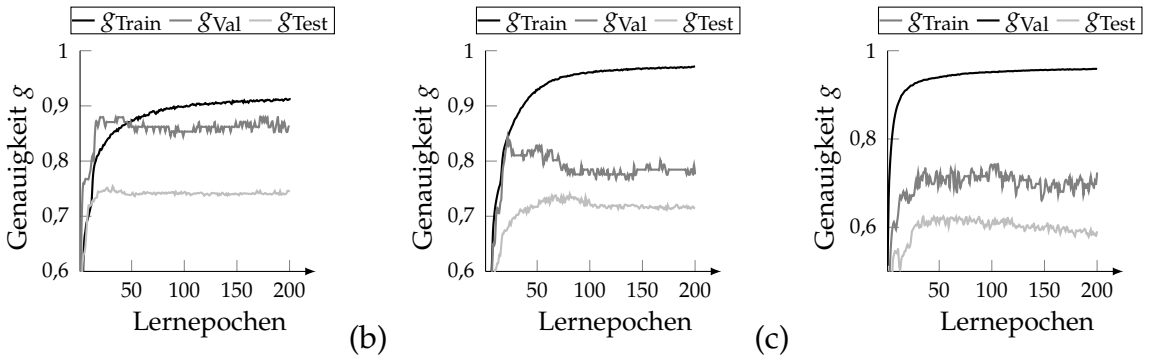


Abbildung 4.3: (a) Darstellung von den Lernkurven der FFW Struktur über die Lernepochen. Die resultierenden Genauigkeiten betragen $g_{\text{Train}} = 0.905$, $g_{\text{Val}} = 0.862$ und $g_{\text{Test}} = 0.746$. (b) Abbildung der Lernkurven für die FFWE Architektur. Die finalen Genauigkeiten betragen $g_{\text{Train}} = 0.970$, $g_{\text{Val}} = 0.801$ und $g_{\text{Test}} = 0.714$. (c) Darstellung der Lernkurven hinsichtlich der RNNE Struktur mit den resultierenden Genauigkeiten $g_{\text{Train}} = 0.959$, $g_{\text{Val}} = 0.724$ und $g_{\text{Test}} = 0.584$.

Die Berechnung erfolgte anhand der Trainings- und Testsätze der Gleichungen (4.13) und (4.14) sowie mit den 116 Grundwahrheiten des Datensatzes $\mathcal{D}_{\text{Val}}$. Wahl der Trainingsparameter und Netzwerkarchitekturen kann angepasst werden, um die Genauigkeiten g_{Val} und g_{Test} der einzelnen Netzwerke zu steigern.

Die Wahl von 50 und 200 erfolgte, um eine Grundlage für den Vergleich der Architekturen hinsichtlich g_{Train} zu schaffen. Mithilfe der Lernkurven kann ein Optimum zwischen Trainings- und Testgenauigkeit bestimmt werden [33]. Zudem entsteht durch die Lernkurven eine Erwartungshaltung an die Resultate bezüglich der Kombination von Spracherkennung und Sprachklassifikation. Nachfolgend werden die Ergebnisse für die Hypothesen aus den erstellten Audiodateien präsentiert.

4.4 Ergebnisse des Datensatzes I

Der Datensatz $\mathcal{D}_1$ beinhaltet 2784 aus dem Validierungsdatensatz erzeugte Audiodateien. Diese wurden mithilfe von vier ASR Lösungen transkribiert und anschließend klassifiziert. Die Ausgaben der entsprechenden Teilprozesse werden bezüglich der eingeführten Metriken aus Abschnitt 4.1 ausgewertet. Die Ergebnisse dieser Auswertung sind in Tabelle 4.3 aufgelistet.

Tabelle 4.3: Zusammenstellung der Ergebnisse für den Datensatz $\mathcal{D}_1$. Eingetragen sind die Ergebnisse der verschiedenen Netzwerkarchitekturen aus Abschnitt 3.6 für die verwendeten ASR. Festgehalten sind die Werte der eingeführten Metriken des Abschnittes 4.1.

Netz	FFW				FFW + E				RNN + E			
ASR	Espnet	PS	DS	IBM	Espnet	PS	DS	IBM	Espnet	PS	DS	IBM
q	0,446	0,660	0,562	0,403	0,446	0,660	0,562	0,403	0,446	0,660	0,562	0,403
$\overline{q}_{\text{T}}$	0,398	0,618	0,493	0,382	0,400	0,616	0,499	0,374	0,399	0,595	0,490	0,359
$\overline{q}_{\text{F}}$	0,578	0,769	0,698	0,473	0,562	0,760	0,677	0,488	0,531	0,743	0,649	0,475
g	0,733	0,724	0,659	0,776	0,717	0,695	0,644	0,749	0,641	0,561	0,544	0,623
$\overline{\alpha}_{\text{T}}$	0,957	0,936	0,937	0,948	0,981	0,975	0,978	0,981	0,983	0,981	0,984	0,983
$\overline{\alpha}_{\text{F}}$	0,773	0,804	0,794	0,829	0,913	0,900	0,916	0,902	0,927	0,934	0,939	0,930
z_{c_0}	0,139	0,164	0,119	0,125	0,048	0,072	0,068	0,039	0,089	0,097	0,130	0,058
z_{c_1}	0,197	0,216	0,168	0,214	0,303	0,324	0,227	0,324	0,215	0,186	0,178	0,177
t_{t} in s	3,776	0,465	2,372	2,317	3,776	0,465	2,372	2,317	3,776	0,465	2,372	2,317

Die verwendeten Spracherkennungslösungen sind *Espnet*, *DeepSpeech* (DS) Version 0.7.4, *Pocketsphinx* (PS) und *IBM-Watson Speech to text*. Die Wortfehlerraten und Transkriptionszeiten bleiben trotz unterschiedlicher Netzwerkarchitekturen zur Textklassifizierung gleich, wurden zum Zwecke der Vollständigkeit dennoch mit aufgetragen.

Die höchste Genauigkeit wird von einem herkömmlichen vorwärtspropagierenden neuronalen Netzwerk mit *IBM-Watson* als Spracherkennungslösung erzielt. Da es sich dabei um eine Online-Lösung handelt, werden die Ergebnisse gesondert betrachtet.

In der Abbildung 4.4 sind die Genauigkeiten der Sprachklassifikation in Kombination mit der Spracherkennung über das Verhältnis $\frac{\overline{\alpha}_\mathrm{T}}{\overline{\alpha}_\mathrm{F}}$ aufgetragen. Des Weiteren werden die vier unterschiedlichen Transkriptionszeiten t_t mithilfe verschiedener Radien dargestellt. Dem Abschnitt 4.1 folgend, soll bei korrekter Klassifikation eine hohe und bei falscher eine geringe Konfidenz resultieren. Daher befindet sich der anzustrebende Zustand der Verknüpfungen im oberen rechten Bereich der Darstellung.

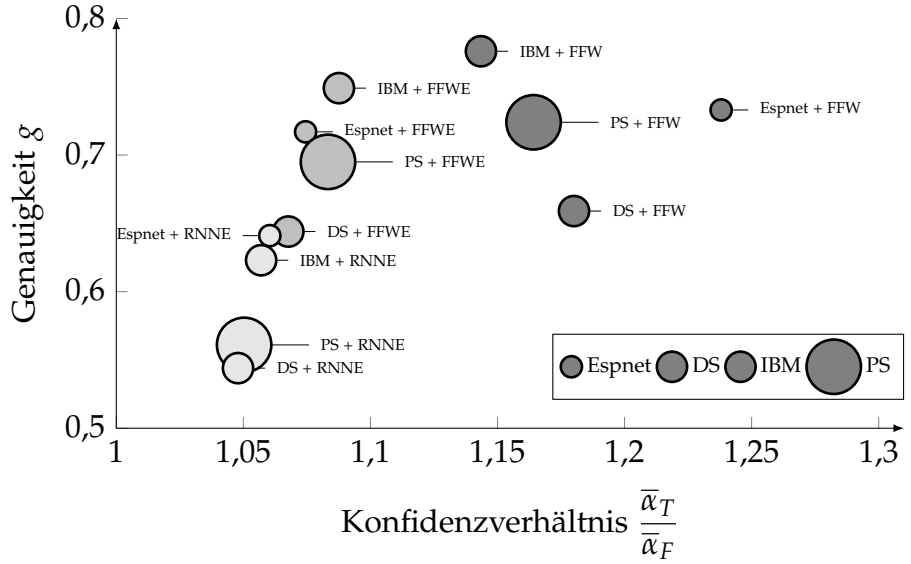


Abbildung 4.4: Darstellung der Kombinationen von Spracherkennung und Sprachklassifikator. Die verschiedenen Radien der Kreise geben die Geschwindigkeit zur Erzeugung eines Transkripts aus einer 7 s Tonspur. Dies entspricht den vier Zeiten der Tabelle 4.3, Zeile 9.

Die Klassifikationen der Netzwerkarchitekturen FFW und FFWE erreichen am Datensatz $\mathcal{D}_2$ eine höhere Genauigkeit als die RNNE Struktur. Mittels der Werte z_{c_0} und z_{c_1} sowie g wird darüber hinaus die Schnittwahrscheinlichkeit P_S der jeweiligen Spalten bestimmt und anschließend für die zugehörige Netzwerkarchitektur gemittelt. Hinsichtlich der Tabelle 4.3 ergeben sich die Wahrscheinlichkeiten:

$$\overline{P}_{\mathrm{S}_1} = \frac{1}{4}\sum_{i=1}^{4}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.0921 \tag{4.15}$$

$$\overline{P}_{\mathrm{S}_2} = \frac{1}{4}\sum_{i=5}^{8}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.1041 \tag{4.16}$$

$$\overline{P}_{\mathrm{S}_3} = \frac{1}{4}\sum_{i=9}^{12}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.1156. \tag{4.17}$$

Dabei kennzeichnet der Index i in diesem Fall die Spaltennummer der Ergebnistabelle. Die Werte beschreiben die Wahrscheinlichkeit, dass eine fehlerhafte Klassifikation und die Kategorie *Drive* oder *SLAM* resultiert. Die weiteren $z_{2:j}$ Werte sind nicht für zusätzliche Berechnungen vorgesehen, weshalb diese hier nicht präsentiert werden.

Aus den Ergebnissen für die Schnittwahrscheinlichkeit folgt, dass bei circa 10 % der Klassifikationen eine fehlerhafte, sicherheitskritische Kategorie erkannt wird. Daraus resultiert die Schwellwertprüfung. Anhand der $\overline{\alpha}_\mathrm{T}$ Ergebnisse aus dieser Tabelle kann der Schwellwert aus Abschnitt 3.5 eingestellt werden. Abhängig von der Wahl des Klassifikators wird der zugehörige Zellwert genutzt.

4.5 Ergebnisse des Datensatzes II

Der Datensatz $\mathcal{D}_2$ beinhaltet 8304 Audiodateien. Diese wurden durch drei Spracherkennungssysteme transkribiert. Dies und die anschließende Klassifikation ergeben die Werte aus der Tabelle 4.4. Die *IBM*-Lösung wurde indes nur an dem Datensatz $\mathcal{D}_1$ verwendet.

Tabelle 4.4: Zusammenstellung der Ergebnisse für die vorgestellten Metriken des Datensatzes $\mathcal{D}_2$. Die weiteren z Werte wurden zwar ermittelt, jedoch nicht für weitere Berechnungen genutzt. Daher wurde auf das Auflisten dieser verzichtet.

Netz	FFW			FFW + E			RNN + E		
ASR	Espnet	PS	DS	Espnet	PS	DS	Espnet	PS	DS
q	0,433	0,673	0,572	0,433	0,673	0,572	0,433	0,673	0,572
$\overline{q}_{\mathrm{T}}$	0,407	0,649	0,546	0,420	0,667	0,568	0,409	0,647	0,559
$\overline{q}_{\mathrm{F}}$	0,723	0,920	0,820	0,819	0,980	0,835	0,654	0,884	0,698
g	0,918	0,912	0,905	0,969	0,982	0,985	0,901	0,892	0,908
$\overline{\alpha}_{\mathrm{T}}$	0,970	0,964	0,955	0,998	0,999	0,999	0,998	0,998	0,999
$\overline{\alpha}_{\mathrm{F}}$	0,717	0,778	0,709	0,927	0,903	0,921	0,737	0,737	0,663
z_{c_0}	0,062	0,116	0,108	0,081	0,093	0,033	0,217	0,182	0,168
z_{c_1}	0,191	0,204	0,146	0,327	0,320	0,223	0,177	0,129	0,123
t_{t} in s	3,965	0,502	2,346	3,965	0,502	2,346	3,965	0,502	2,346

Die resultierenden Hypothesen der ASR-Systeme werden für das Training der neuronalen Netzwerke genutzt. Dadurch sind die deutlich höheren Genauigkeiten zu erklären, da die Netzwerke mit den Hypothesen trainiert wurden. Die Genauigkeitswerte entsprechen im Wesentlichen den Trainingsgenauigkeiten aus Abschnitt 4.3. In der Abbildung 4.5 sind die Genauigkeiten über das Verhältnis $\frac{\overline{\alpha}_{\mathrm{T}}}{\overline{\alpha}_{\mathrm{F}}}$ der Ergebnisse des Datensatzes $\mathcal{D}_2$ aufgetragen.

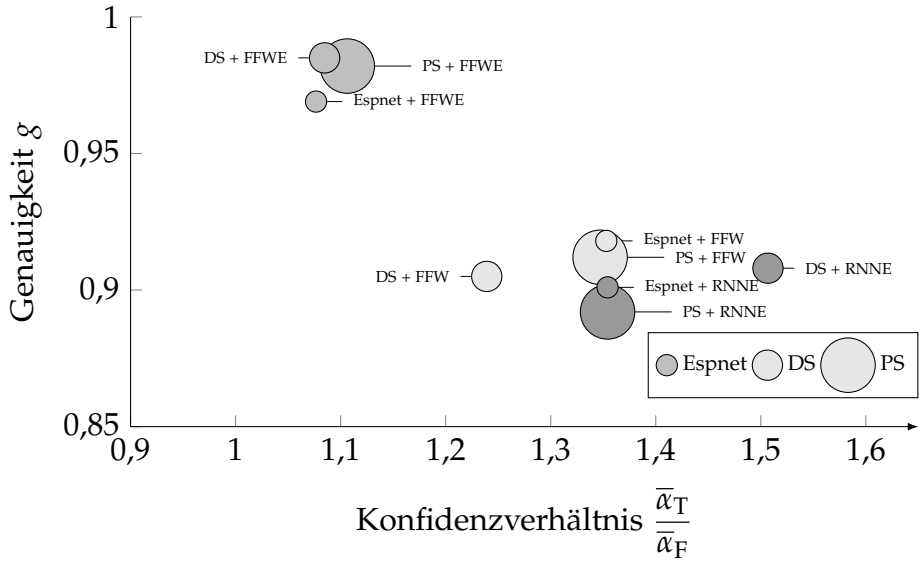


Abbildung 4.5: Darstellung der Verknüpfung von Spracherkennung und Sprachklassifikator für den Datensatz $\mathcal{D}_2$. Die ermittelten Transkriptionszeiten bezüglich einer Tonspur der Länge $t = 7\,\mathrm{s}$ werden erneut durch unterschiedliche Radien dargestellt.

Die höchsten Genauigkeiten erreicht die ausschließlich vorwärtspropagierende Netzwerkstruktur mit einer *Embedding*-Schicht. Die FFW und RNNE Architekturen erreichen nahezu äquivalente Resultate. Hinsichtlich der Ergebnisse aus den Abschnitten 4.3 und 4.4 ist bei der RNNE Architektur eine Neigung zur Überanpassung erkennbar. Da bezüglich der Trainingsdaten eine hohe Genauigkeit erzielt wird, während die Resultate bei dem Test- und Validierungsdatensatz deutlich geringer ausfallen [33]. Zudem fallen die sicherheitskritischen Schnittwahrscheinlichkeiten geringer aus, da die Genauigkeit bei dem Datensatz $\mathcal{D}_2$ deutlich höher liegt. Bezüglich der Tabelle 4.4 betragen die Schnittwahrscheinlichkeiten:

$$\overline{P}_{\mathrm{S}_4} = \frac{1}{3}\sum_{i=1}^{3}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.0243 \tag{4.18}$$

$$\overline{P}_{\mathrm{S}_5} = \frac{1}{3}\sum_{i=4}^{6}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.0080 \tag{4.19}$$

$$\overline{P}_{\mathrm{S}_6} = \frac{1}{3}\sum_{i=7}^{9}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.0331. \tag{4.20}$$

4.6 Ergebnisse des Datensatzes III

Der Datensatz $\mathcal{D}_3$ beinhaltet 119 Audiodateien. Der wesentliche Unterschied zu den vorher genannten ist die Art des Sprechers. Wie bereits erwähnt, handelt es sich um Aufnahmen von Probanden, nicht um synthetische Stimmen. Die Ergebnisse dieser Sätze sind in der Tabelle 4.5 aufgelistet.

Tabelle 4.5: Zusammenstellung der Ergebnisse für die vorgestellten Metriken des Datensatzes $\mathcal{D}_3$.

Netz	FFW			FFW + E			RNN + E		
ASR	Espnet	PS	DS	Espnet	PS	DS	Espnet	PS	DS
q	0,424	0,930	0,514	0,424	0,930	0,514	0,424	0,930	0,514
$\overline{q}_{\mathrm{T}}$	0,380	0,777	0,438	0,368	0,743	0,470	0,345	0,747	0,431
$\overline{q}_{\mathrm{F}}$	0,701	1,228	0,813	0,725	1,362	0,806	0,630	1,123	0,730
g	0,862	0,661	0,798	0,844	0,697	0,872	0,725	0,514	0,725
$\overline{\alpha}_{\mathrm{T}}$	0,975	0,960	0,984	0,999	0,969	0,985	0,983	0,954	0,988
$\overline{\alpha}_{\mathrm{F}}$	0,815	0,831	0,777	0,944	0,920	0,897	0,895	0,916	0,948
z_{c_0}	0,000	0,270	0,182	0,059	0,152	0,143	0,100	0,075	0,133
z_{c_1}	0,400	0,297	0,091	0,471	0,333	0,286	0,300	0,245	0,067
t_{t} in s	4,048	0,865	2,427	4,048	0,865	2,427	4,048	0,865	2,427

Die Spracherkennungslösung von *Pocketsphinx* erzeugt dabei Transkripte mit vergleichsweise sehr hoher Wortfehlerrate. Daraus folgen die niedrigen Genauigkeiten der Kombinationen mit der *Pocketsphinx* Lösung.

Wiederholt weisen die FFW und FFWE Architekturen bessere Ergebnisse im Vergleich zur RNNE Struktur auf. Dabei ist zu beachten, dass die Beispielrate aufgrund des kleineren Datensatzes deutlich niedriger ist. Dies führt zu weniger belastbaren Resultaten, da bereits minimale Änderungen in den Berechnungen großen Einfluss haben. Wie in Abschnitt 4.5 ist bezüglich der RNNE Architektur eine Neigung zu Überanpassung erkennbar. In der Abbildung 4.6 sind die Genauigkeiten über das Verhältnis $\frac{\overline{\alpha}_{\mathrm{T}}}{\overline{\alpha}_{\mathrm{F}}}$ der Ergebnisse des Datensatzes $\mathcal{D}_3$ aufgetragen.

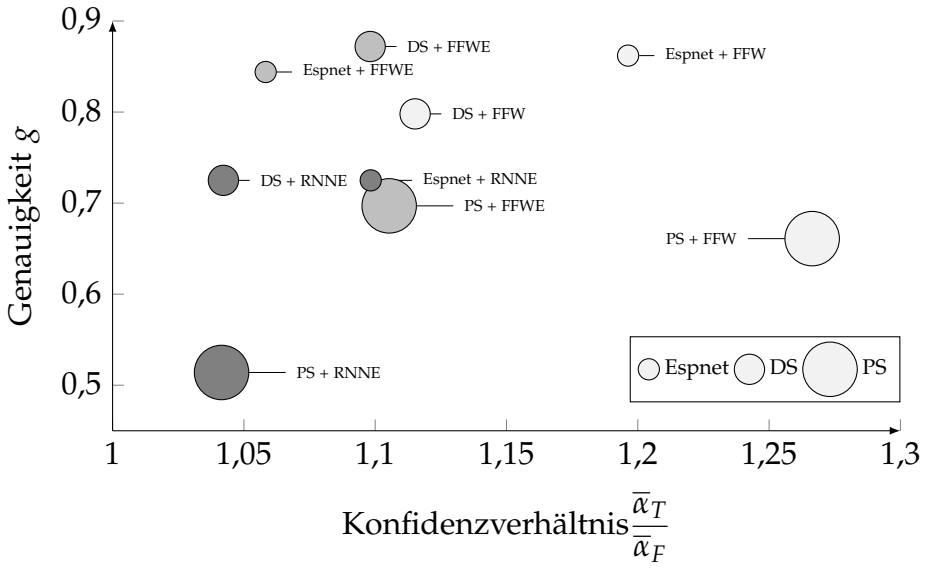


Abbildung 4.6: Abbildung der Verknüpfung von Spracherkennung und Sprachklassifikator für den Datensatz $\mathcal{D}_3$.

Hinsichtlich einer fehlerhaften Klassifikation in der Kategorie *Drive* oder *SLAM* ergeben sich für $\mathcal{D}_3$ die Schnittwahrscheinlichkeiten:

$$\overline{P}_{S_7} = \frac{1}{3}\sum_{i=1}^{3}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.1009 \tag{4.21}$$

$$\overline{P}_{S_8} = \frac{1}{3}\sum_{i=4}^{6}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.0948 \tag{4.22}$$

$$\overline{P}_{S_9} = \frac{1}{3}\sum_{i=7}^{9}(1-g_i)(z_{c_0,i}+z_{c_1,i}) = 0.1068. \tag{4.23}$$

4.7 Diskussion der Ergebnisse

Das beste Spracherkennungssystem bietet der Online-Dienst von *IBM-Watson*. Diese Lösung wird nicht verwendet, da eine Offline-Erkennung anzustreben ist. Anhand der $\overline{q}_{\mathrm{T}}$ und $\overline{q}_{\mathrm{F}}$ Werte ist festzustellen, dass bei geringerer Wortfehlerrate eine korrekte Klassifikation wahrscheinlicher ist. Deshalb empfiehlt es sich ein Spracherkennungssystem mit niedriger Wortfehlerrate zu nutzen. Die Tabelle 4.6 fasst die Resultate der Spracherkennungssysteme zusammen. Bezug bei der Zusammenfassung ist jeweils der Beste in gelisteter Kategorie beziehungsweise die Ergebnisse aus der Literatur. Für die Tabellen 4.6 und 4.7 gelten die Notationen: o (neutral), + (befriedigend) und ++ (empfohlen). Die Wortfehlerraten werden nur mit befriedigend oder neutral bewertet, da Veröffentlichungen wie *Wav2Letter* [21], *DeepSpeech* [20] und *Espnet* [24, 52] deutlich niedrigere Wortfehlerraten angeben.

Tabelle 4.6: Zusammenfassender Vergleich der vier Spracherkennungssysteme.

	Espnet	PS	DS	IBM
Geschwindigkeit	o	++	+	+
Wortfehlerrate	+	o	+	+
Offline	Ja	Ja	Ja	Nein

Die Netzwerke klassifizieren an dem Trainingsdatensatz besser. Dies entspricht den Erwartungen, da die Testdaten nicht Teil des Trainings waren. Die RNNE Netzwerkarchitektur weist an den Testdatensätzen die niedrigsten Genauigkeiten auf und zeigt eine Tendenz zur Überanpassung. Anhand der Lernkurven aus Abschnitt 4.3 kann zur Steigerung der Genauigkeiten eine Verringerung der Lernepochen vorgenommen werden.

Die Anpassung der Trainingsparameter kann erfolgen, da die Netzwerke bezüglich der Test- und Validierungsgenauigkeiten ein Maximum in vorausgehenden Lernepochen aufweisen. Wird der Trainingsprozess nicht durch eine Stoppbedingung abgebrochen, so neigt die FFWE Architektur anhand der Randbedinungen ebenfalls zu einer Überanpassung. In der Tabelle 4.7 werden die verwendeten Netzwerkarchitekturen anhand der eingeführten Notation verglichen.

Tabelle 4.7: Zusammenfassender Vergleich der verschiedenen Netzwerkarchitekturen. Die Kategorie Darstellung basiert auf *One-Hot Encoding* (OHE) oder *Word Embedding* (WE). Weiterhin wird eine Einschätzung bezüglich einer Tendenz zur Überanpassung (Ü) aufgelistet.

Architektur	FFW	FFWE	RNNE
g	++	++	o
$\frac{\overline{\alpha}_T}{\overline{\alpha}_F}$	++	+	+
P_S	++	++	o
Tendenz Ü	Nein	Ja	Ja
Darstellung	OHE	WE	WE

Generell haben die vorwärtspropagierenden Netzwerke, dass auf Rückkopplung basierende, anhand der vorgegebenen Randbedingungen übertroffen. Die Wahl der Parameter bezüglich der RNNE Struktur erfolgte, um diese mit der FFWE zu vergleichen. Daher wird in beiden *Embedding*-Schichten ein Wordvektor der Dimension $d = 8$ erzeugt. Da jedes Wort eine Sequenz darstellt, wurden nur 8 Neuronen gewählt. Die Anzahl der Neuronen einer Rekurrenten-Schicht kann demzufolge hinsichtlich einer Verbesserung der Ergebnisse angepasst werden.

Durch die *Embedding*-Schicht der FFWE Architektur wird indes eine Darstellung der Sätze erzeugt, welche zum Verifizieren des Datensatzes genutzt wird. Weiterhin können die Klassifikatoren gleichzeitig eingesetzt werden, um bezüglich der Klassifizierung eine Redundanz und daraus zusätzliche Sicherheit zu gewinnen. Die berechneten Schnittwahrscheinlichkeiten geben an, wie hoch die Wahrscheinlichkeit bei bekannten und unbekannten Daten ist, eine falsche und dazu sicherheitskritische Klassifikation zu erzeugen. Durch eine passende Wahl der Schwellwerte kann dies beschränkt werden.

Die Wahl der Kombination aus Spracherkenner und Sprachklassifizierer ist abhängig von der Nutzung. Eine schnelle und ungenauere Lösung liefert *Pocketsphinx* und das FFW. Im Gegensatz dazu ist die Verwendung von *Espnet* und FFW genauer, dafür aber langsamer. Anhand der Wahl sind die Schwellwerte der Klassifikationen anzupassen. Die Festlegung der Schwellwerte aus Abschnitt 3.5 erfolgt mit der mittleren Konfi-

denz $\overline{\alpha}_{\mathrm{F}}$ bei falschen Klassifikationen und hängt von der verwendeten Architektur ab. In Abhängigkeit von der Höhe dieser Schwelle entsteht ein längerer andauernder Vorgang, bis eine finale Klassifizierung als Transitionsbedingung genutzt werden kann. Anschließend erfolgt die Verifikation der entwickelten Sprachverarbeitung.

5 Verifikation von Datensatz und Anforderungen

Dieses Kapitel dient der Verifikation der verwendeten Methoden. Hierbei werden die Anforderungen aus Abschnitt 3.2 mithilfe eines Verifikationsplans verifiziert. Die genaue Beschreibung der Anforderungen und des Verifikationsplans erfolgt im Lastenheft. Die Ergebnisse der Verifikation werden tabellarisch festgehalten und in diesem Kapitel präsentiert. Weiterhin wird eine Plausibilitätsprüfung des erstellten Datensatzes vorgenommen. Dies erfolgt mit den Resultaten der *Embedding*-Schicht aus Abschnitt 3.6.

5.1 Verifikation des Datensatzes

Wie in Abschnitt 2.6 beschrieben, kann die vektorielle Darstellung der Sätze durch eine *Embedding*-Schicht eines klassifizierenden neuronalen Netzwerks gewonnen werden. Die Sätze mit ähnlichem Inhalt erhalten demnach eine gleichartige vektorielle Darstellung. Erfolgt kein Training des zweiten vorgestellten Netzwerks, so sind keine klaren Abgrenzungen zwischen verschiedenen Gebieten auszumachen. Die Satzvektoren des Datensatzes $\mathcal{D}$ sind demnach nicht in gleichartige Gebiete gruppiert.

Um die hochdimensionale Darstellung der Satzvektoren vorstellbar darzustellen, wird eine Hauptkomponentenanalyse der resultierenden Vektoren durchgeführt [38]. Dabei werden für jeden Satz drei Hauptkomponenten (X-, Y- und Z) bestimmt. Hinsichtlich eines untrainierten Netzwerkes zeigt Abbildung 5.1 die X- und Y-Projektionen aller 462 Vektoren. Dabei sind keine klaren Abgrenzungen zwischen den Kategorien zu erkennen. Dies entspricht im Wesentlichen der Erwartungshaltung, da die Initialisierung des Netzwerkes mithilfe von Zufallszahlen erfolgt.

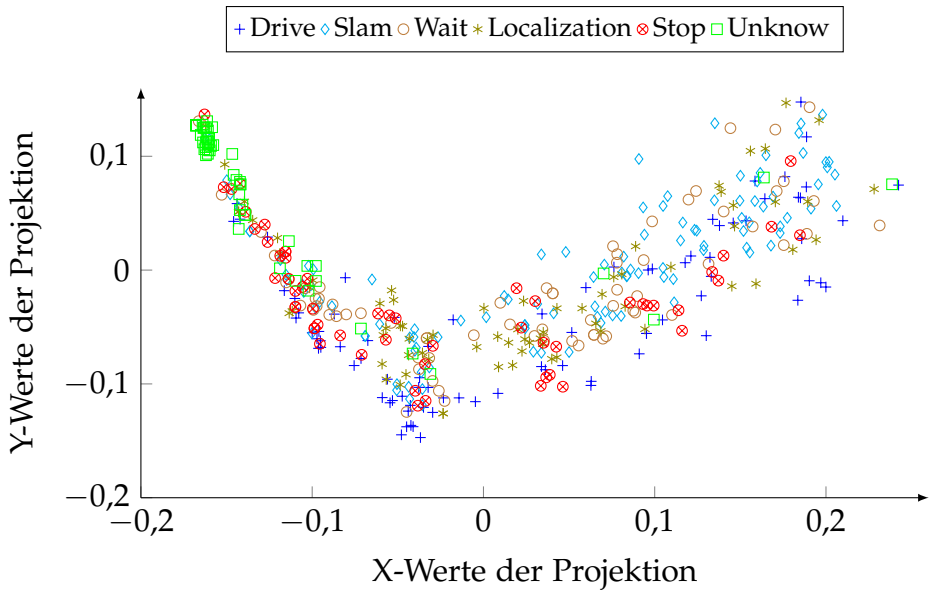


Abbildung 5.1: Darstellung von Hauptkomponenten der 426 Satzvektoren des erstellten Datensatzes $\mathcal{D}$. Die X- und Y-Projektionen der Vektoren sind hier dargestellt. Weiterhin ist keine deutliche Abgrenzung zwischen Satzvektoren unterschiedlicher Kategorien zu erkennen. Die Klassenzugehörigkeit der Sätze wird durch verschiedene Marker und Farben gekennzeichnet.

Ist das Training des zweiten vorgestellten Netzwerks aus Abschnitt 3.6 mit einer entsprechenden Genauigkeit abgeschlossen, so kann der Klassifikator die Wortgruppen bezüglich ihrer Kategorie sortieren [12]. Folglich entsteht durch die *Embedding*-Schicht eine inhaltsabhängige räumliche Verteilung der verschiedenen Sätze des Datensatzes $\mathcal{D}$ und zugehörigen Klassen [4]. Die Kategorien schließen folglich verschiedene Bereiche in einem Vektorraum ein, wobei die Satzvektoren je nach Bedeutung in den zugehörigen Gebieten liegen.

Durch nachfolgende Hauptkomponentenanalyse entstehen die Projektionen in X-, Y- und Z-Richtung. Die Abbildungen 5.2 und 5.3 zeigen diese in X- und Y-Richtung, sowie X- und Z-Richtung. Aus Gründen der Übersichtlichkeit wurden die Nummern der zugehörigen Satzvektoren nicht aufgetragen. In den Darstellungen ist eine Gruppierung der definierten Klassen zu erkennen. Anhand dieser Darstellung erfolgt die Plausibilitätsprüfung. Eine Darstellung der zugehörigen Satznummern kann Aufschluss über die Datensatzzugehörigkeit geben. Dabei können verschiedene Projektionen von Sätzen betrachtet und auf die zugehörige Kategorie überprüft werden.

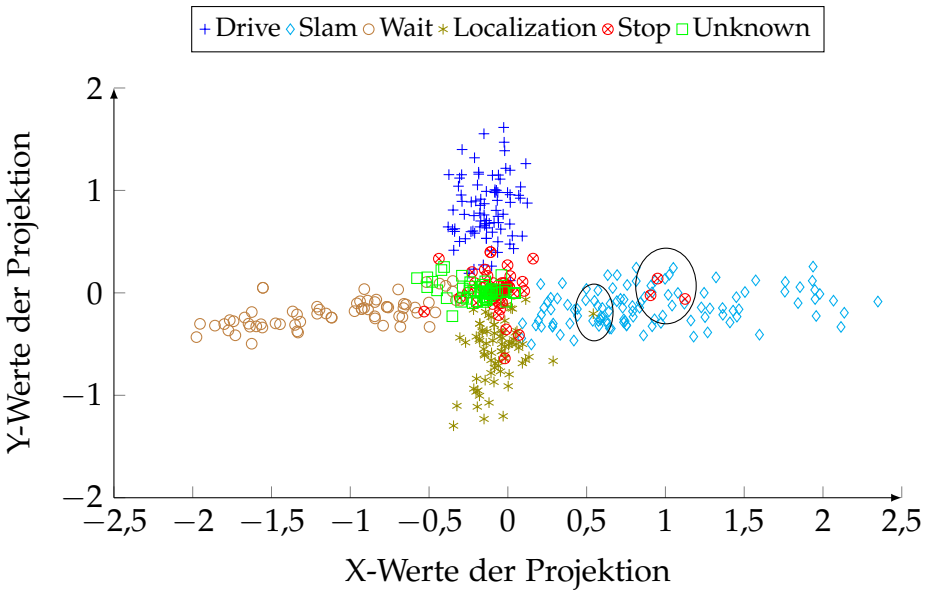


Abbildung 5.2: Darstellung von Hauptkomponenten der 462 Satzvektoren des erstellen Datensatzes $\mathcal{D}$ nach Training des neuronalen Netzwerks. Die Projektionen der X- und Y-Komponenten der Vektoren werden hier nach einem Lernvorgang dargestellt. Eine Abgrenzung zwischen den Kategorien wird erkennbar. Zusätzlich sind beispielhaft zwei Gebiete durch Ellipsen gekennzeichnet. Dort befinden sich projizierte Satzvektoren, die einer anderen Kategorie angehören.

Die in der rechten Ellipse eingerahmten Datenpunkte repräsentieren die Sätze *„finish the solution of the slam problem“*, *„stop the slam mode“* und *„stop to solve the slam problem“*. Die zwei Erstgenannten gehören dem Datensatz $\mathcal{D}_{\text{Test}}$ an, wurden also nicht mittrainiert. Aufgrund von phonetischer Ähnlichkeit liegen fehlerhafte Klassifikationen vor. Der dritte Satz in diesem Bereich befindet sich im Datensatz $\mathcal{D}_{\text{Train}}$ und wurde ebenfalls falsch klassifiziert. Der zweite markierte Bereich beinhaltet die Projektion des Satzes *„can you start with pose estimation and mapping“*. Dieser Satz enthält eine falsche Grundwahrheit, statt der Klasse c_1, besitzt der Satz die Kategorie c_3. *SLAM* und *Localization* wurden demnach beim Anlegen des Datensatzes vertauscht.

Die Abbildung 5.3 zeigt X- und Z-Komponenten der Projektionen. Daran ist zu erkennen, ob die Satzvektoren in einer gleichen Ebene liegen. Die Darstellung dient der Vollständigkeit, da drei Hauptkomponenten berechnet wurden. Weiterhin ist zu erkennen, dass die in Abbildung 5.2 gekennzeichnete Bereiche erneut die falsch klassifizierten und markierten Sätze beinhalten.

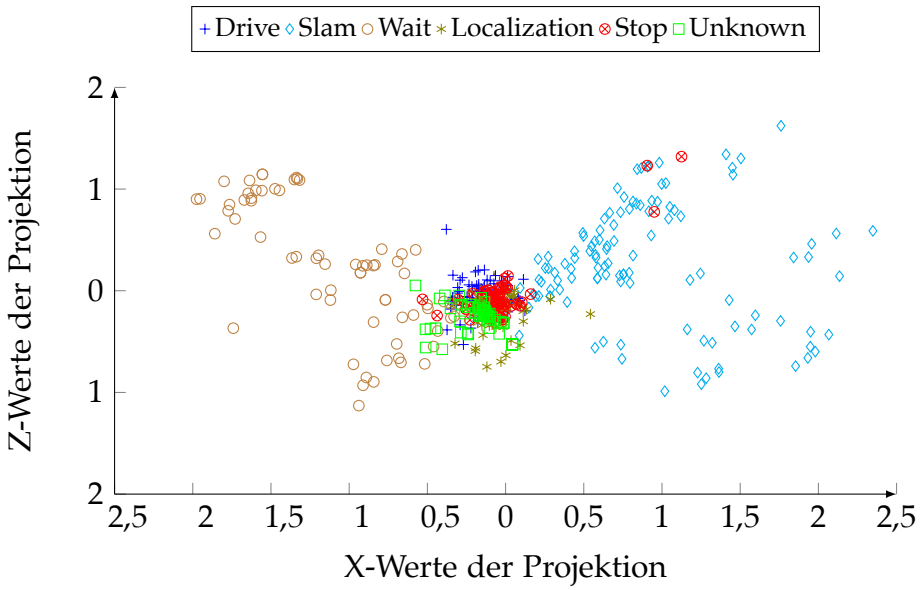


Abbildung 5.3: Darstellung von Hauptkomponenten der 426 Satzvektoren des erstellen Datensatzes $\mathcal{D}$ nach Training des neuronalen Netzwerks. Die Projektionen der X- und Z-Komponenten der Vektoren werden hier dargestellt. Die Abbildung zeigt eine weitere Ebene der Projektion.

In Abbildung 5.2 ist zwischen den Kategorien *Stop* und *Unknown* keine räumliche Trennung zu erkennen. Durch die Projektion der X- und Z-Komponenten kann in Darstellung 5.3 jedoch ein räumlicher Unterschied festgestellt werden.

5.2 Verifikation der Anforderungen

Die zu erreichenden Ziele sind in dem Lastenheft im Anhang festgehalten. Die erarbeiteten Anforderungen an die Sprachverarbeitung werden anhand des folgenden Verifikationsplans geprüft. Im Folgenden ist die Vorgehensweise der Verifikation mit den entsprechenden Hilfsmitteln geschildert.

Nr./ID	Titel	Verifikation der Anforderung	Hilfsmittel
ANF_01	Auslösen einer Sprachaufnahme durch manuelle Betätigung	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. Launch-File starten (./launch.sh) und bis zur Initialisierung warten 3. Abonnieren der veröffentlichten *ROS-Topic /audio_stream* 4. *Start* Knopf der Fernbedienung betätigen 5. In dem ROS log wird *Start streaming audio...* geschrieben 6. Nach der Aufnahme, wird in den ROS log *Finish streaming audio...* geschrieben 7. Abonnement der *ROS-Topic* erhält eine neue Nachricht Ergebnis: **Test bestanden**	*ROS* *Kinect-Mikrofon* *rostopic echo topicname*

Nr./ID	Titel	Verifikation der Anforderung	Hilfsmittel
ANF_02	Erzeugen und bereitstellen einer Tonspur	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. Launch-File starten (./launch.sh) und bis zur Initialisierung warten 3. Abonnieren der veröffentlichten *ROS-Topic /audioStream/pub/stream/topic* 4. *Start* Knopf der Fernbedienung betätigen 5. Über die vorgegebene Zeitspanne von $t = 5$ s eine Wortgruppe in das Mikrofon sprechen 6. Abonnement der *ROS-Topic* erhält eine neue Nachricht, diese wird zu einer *WAV*-Datei *test.wav* konvertiert und liegt im Ordner des Knotens 7. Abspielen der Audio Datei mit entsprechender Software Ergebnis: **Test bestanden**	*ROS Kinect-Mikrofon rostopic echo topicname*

Nr./ID	Titel	Verifikation der Anforderung	Hilfsmittel
ANF_03	Erkennung und Klassifizierung von bedienungsorientierter Sprache des Benutzers	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. Launch-File starten (./launch.sh) und bis zur Initialisierung warten 3. Abonnieren der veröffentlichten *ROS-Topic transcript/task* 4. *Start* Knopf der Fernbedienung betätigen und über die vorgegebene Zeitspanne eine bedienungsorientierte Wortgruppe in das Mikrofon sprechen. (Hinweis: Bitte sprechen sie ca. 1 m von dem Aufnahmgerät entfernt in Richtung des Mikrofons) 5. Vergleich der veröffentlichten Klassifikation mit der zugehörigen Kategorie der Wortgruppe aus dem Datensatz *random-distributed-dataset.json* Ergebnis: **Test bestanden**	*ROS Kinect-Mikrofon Datensatz*

Nr./ID	Titel	Verifikation der Anforderung	Hilfsmittel
ANF_04	Erkennen von benutzerdefinierten Schlagwörtern	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. Launch-File starten (./launch.sh) und bis zur Initialisierung warten 3. Abonnieren der veröffentlichten *ROS-Topic transcript/buzz* 4. *Start* Knopf der Fernbedienung betätigen und über die vorgegebene Zeitspanne den Satz „drive to location“in das Mikrofon sprechen 5. *Start* Knopf der Fernbedienung betätigen und über die vorgegebene Zeitspanne ein Schlagwort der Liste *buzzword.json* in das Mikrofon sprechen 6. Vergleich des veröffentlichten Schlagworts mit dem eingesprochenen Ergebnis: **Test nicht bestanden**	*ROS Kinect-Mikrofon*

6 Zusammenfassung und Ausblick

Im Rahmen dieser Masterarbeit wurde eine Sprachverarbeitung für das autonome Logistik-Fahrzeug konzipiert, entwickelt und evaluiert. An dem Fahrzeug wird durch manuelle Betätigung des *Start* Schalters der Fernbedienung eine Tonspur der Dauer $t = 5\,\mathrm{s}$ erstellt. Die Aufnahme erfolgt dabei mit einem Mikrofon der verbauten *Kinect*-Sensorik und enthält eine bedienungsorientierte Wortgruppe.

Anschließend erfolgt eine Transkription sowie eine Klassifikation und Schlagworterkennung des resultierenden Textes. Das Ergebnis dessen wird in *ROS* veröffentlicht. Die Veröffentlichungen der Sprachverarbeitung dienen zur Steuerung eines Zustandsautomaten, welcher die aus der Entwicklung entstandenen Betriebsmodi des Fahrzeugs steuert.

In der Konzeptionierung erfolgte eine Einordnung in die bestehende Systemarchitektur. Für die Evaluierung wurden verschiedene Systeme zur Sprachklassifikation und Spracherkennung eingesetzt. Zur Umsetzung gehört die Erzeugung eines bedienungsorientierten Datensatzes und eine Liste von Schlagwörtern. Diese dienen der Klassifikation von Handlungen und dem Veröffentlichen von Zielposen. Dadurch wird die Möglichkeit geboten, hierarchisch übergeordnete Aufgaben, Tätigkeiten oder Prozesse in einem Folgeprojekt einzubinden. Dabei dient die Sprachverarbeitung und der Zustandsautomat als Hilfsmittel zur Abwicklung von hochautomatisierten Logistikprozessen. Die Einbindung solcher Abläufe in einer geeigneten Entwicklungsumgebung kann als Folgethema behandelt werden.

Weiterhin wurde mithilfe des Datensatzes und synthetischer Sprachausgabe ein Testfeld zur Evaluation und Verifikation angelegt. Das Testfeld beinhaltet 11207 von 24 synthetischen Akzenten und 12 Probanden eingesprochene *WAV*-Dateien mit bedienungsorientierten Wortgruppen. In der Evaluation erfolgte eine Analyse und

Gegenüberstellung von den Prozessen Spracherkennung und Sprachklassifikation. Dies findet durch vorher eingeführter Metriken anhand der erstellten Datensätze statt. Des Weiteren folgte eine Plausibilitätsprüfung des Datensatzes $\mathcal{D}$ anhand des *Word Embedding* sowie eine Verifikation von Anforderungen mit einem Verifikationsplan. Daher ist der Datensatz entsprechend der fehlerhaften Zuweisungen anzupassen. Eine Erweiterung der Menge kann zudem die Genauigkeit der Klassifikationen steigern.

Durch die Plausibilitätsprüfung wird eine falsche Klassifikation von Wortgruppen mit phonetischer Ähnlichkeit erkannt. Eine Unterstützung der Einordnung kann mithilfe von Informationen aus dem Anwendungs-Umfeld erreicht werden. Zum Beispiel kann der aktuelle Zustand des Fahrzeugs Einfluss auf die Klassifikation haben. Dies bedarf weiterer Untersuchungen und stellt ebenfalls ein mögliches Folgeprojekt dar.

Aktuell wird die Sprachverarbeitung auf dem *Linux*-System des ALF eingesetzt. Ein mögliches Folgeprojekt kann die Implementierung der Sprachverarbeitung auf einem eingebetteten System, wie zum Beispiel dem vorhandenen *Raspberry Pi*, sein.

Hinsichtlich der Klassifikationsergebnisse der rekurrenten Netzwerkstruktur aus Kapitel 4 kann die Untersuchung einer anderen Architektur erfolgen, um dadurch die Neigung zur Überanpassung zu verhindern. Das Hinzufügen von Datenpunkten zu dem Datensatz $\mathcal{D}_3$, erweitert die Evaluation um die reale Sprecher Situation. Denn die Datensätze $\mathcal{D}_1$ und $\mathcal{D}_2$ wurden mit synthetischen Stimmen erzeugt.

Eidesstattliche Versicherung

Eidesstattliche Versicherung zur Abschlussarbeit:

„Entwicklung und Verifikation einer Sprachverarbeitung für das autonome Logistik-Fahrzeug ALF “

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht.

Birkenfelde, 20. Oktober 2020	
Ort, Datum	Hannes Dittmann

Quellenverzeichnis

[1] Hansen, S., Jurran, N. und Porteck, S. *Sprachassistenten durchdringen den Alltag*. Zugriff: 13.10.2020. https://www.heise.de/ct/artikel/Sprachassistenten-durchdringen-den-Alltag-4517343.html.

[2] Techbook, o. V. *Sprachassistenten nisten sich im Alltag ein*. Zugriff: 13.10.2020. https://www.techbook.de/entertainment/sprachassistenten-nisten-sich-im-alltag-ein.

[3] Noelle, C. *How Artificial Intelligence Will Disrupt Speech Recognition*. Zugriff: 13.10.2020. https://www.processmaker.com/blog/how-artificial-intelligence-will-disrupt-speech-recognition/.

[4] Goldberg, Y. *Neural Network Methods for Natural Language Processing*. https://www.morganclaypoolpublishers.com/catalog_Orig/samples/9781627052955_sample.pdf: Morgan & Claypool Publishers.

[5] Gesing, B., Peterson, S., und Michelsen, D. *Artificial Intelligence in Logistics - A collaborative report by DHL and IBM on implications and use cases for the logistic industry*. Zugriff: 15.10.2020. https://www.dhl.com/content/dam/dhl/global/core/documents/pdf/glo-core-trend-report-artificial-intelligence.pdf: DHL Customer Solutions & Innovation.

[6] Sopra Steria GmbH, o. V. *Potenzialanalyse Künstliche Intelligenz*. Zugriff: 13.10.2020. https://www.soprasteria.de/docs/librariesprovider2/sopra-steria-de/publikationen/studien/potenzialanalyse-kuenstliche-intelligenz-2017.pdf?sfvrsn=190f45dc_4/.

[7] Eickmann, D. und Hotze, D. *Entwicklung und Verifikation eines autonomen Logistik-Fahrzeugs*. Masterarbeit. Hochschule Bochum - Bochum University of Applied Sciences, Feb. 2018.

[8] Montorio, G. und Dittmann, H. *Implementierung einer Schlupfregelung per Model-Based Design sowie einer SLAM-Kartografierung für ein autonomes Logistik-Fahrzeug*. Bachelorarbeit. Hochschule Bochum - Bochum University of Applied Sciences, Feb. 2019.

[9] Montorio, G. *Entwicklung einer Bildverarbeitung mit dem Schwerpunkt Personenerkennung für ein autonomes Logistik-Fahrzeug*. Masterarbeit. Hochschule Bochum - Bochum University of Applied Sciences, Okt. 2020.

[10] Wendemuth, A. *Grundlagen der stochastischen Sprachverarbeitung*. 1. Auflage. Oldenbourg Verlag, 2004.

[11] Dong Wang, Xiaodong Wang und Shaohe Lv. *An Overview of End-to-End Automatic Speech Recognition*. Symmetry, Aug. 2019, S. 1018.

[12] Fellbaum, K. *Sprachverarbeitung und Sprachübertragung*. 2. Auflage. Springer Vieweg, 2013.

[13] Görz, G., Schneeberger, J. und Schmidt, U. *Handbuch der Künstlichen Intelligenz*. 5. Auflage. Oldenbourg Verlag, 2013.

[14] Ertel, W. *Grundkurs Künstliche Intelligenz - Eine praxisorientierte Einführung*. 3. Auflage. Springer Vieweg, 2013.

[15] van der Velde, N. *Speech Recognition Software: Past, Present & Future*. Zugriff: 26.08.2020. https://www.globalme.net/blog/speech-recognition-software-history-future/.

[16] Jüngst, J. *Alexa, Cortana, Siri und Co.: Wie digitale Assistenten die Internetnutzung revolutionieren*. Zugriff: 12.10.2020. https://blog.iao.fraunhofer.de/alexa-cortana-siri-und-co-wie-digitale-assistenten-die-internetnutzung-revolutionieren/.

[17] Radtke, M. *Was ist Cloud Computing*. Zugriff: 12.10.2020. https://www.cloudcomputing-insider.de/was-ist-cloud-computing-a-563624/.

[18] Gupta, S. C. *Speech Recognition with Python - 9 most prominent alternatives.* Zugriff: 20.09.2020. https://www.slanglabs.in/blog/automatic-speech-recognition-in-python-programs.

[19] Ried, S. *IoT Sprachsteuerung – Das geht auch Offline und sehr Privat*. Zugriff: 19.09.2020. https://de.cloudflight.io/presse/iot-sprachsteuerung-das-geht-auch-offline-und-sehr-privat-35425/.

[20] Hannunn, A. et al. *Deep Speech: Scaling up end-to-end speech recognition*. http://arxiv.org/abs/1412.5567: CoRR, 2014.

[21] Vineel, P. et al. *wav2letter++: The Fastest Open-source Speech Recognition System*. Zugriff: 24.09.2020. http://arxiv.org/abs/1812.07625: CoRR, 2018.

[22] Povey, D. et al. »The Kaldi Speech Recognition Toolkit«. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dez. 2011.

[23] Rizk, B. *Evaluation of State Of Art Open-source ASR Engines with Local Inferencing*. Zugriff: 18.09.2020. https://www.researchgate.net/publication/335524542_Evaluation_of_State_Of_Art_Open-source_ASR_Engines_with_Local_Inferencing, Aug. 2019.

[24] Watanabe, S. et al. *ESPnet: End-to-End Speech Processing Toolkit*. https://arxiv.org/abs/1804.00015.

[25] kenarsa. *Speech-to-Text Benchmark*. Zugriff: 12.10.2020. https://github.com/Picovoice/speech-to-text-benchmark.

[26] Picovoice, o. V. *End-to-End Intent Inference from Speech*. Zugriff: 16.10.2020. https://picovoice.ai/blog/end-to-end-intent-inference-from-speech/.

[27] Dargan, S. et al. *A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning*. Archives of Computational Methods in Engineering volume 27, Juni 2019.

[28] Dudenredaktion, o. V.: „Phonem“. *Duden online*. Zugriff: 17.09.2020. https://www.duden.de/rechtschreibung/Phonem.

[29] Dudenredaktion, o. V.: „Wort“. *Duden online*. Zugriff: 17.09.2020. https://www.duden.de/rechtschreibung/Wort.

[30] Dudenredaktion, o. V.: „Wortgruppe“. *Duden online*. Zugriff: 17.09.2020. https://www.duden.de/rechtschreibung/Wortgruppe.

[31] Dudenredaktion, o. V.: „Satz“. *Duden online*. Zugriff: 17.09.2020. https://www.duden.de/rechtschreibung/Satz.

[32] Goyal, P., Pandey, S., und Jain, K. *Deep Learning for Natural Language Processing - Creating Neural Networks with Python*. Zugriff: 15.10.2020. https://doi.org/10.1007/978-1-4842-3685-7: Apress, 2018.

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

[34] Chollet, F. *Über- und Unteranpassung*. Zugriff: 24.09.2020. https://www.tensorflow.org/tutorials/keras/overfit_and_underfit.

[35] Meyer, M. *Signalverarbeitung - Analoge und digitale Signale, Systeme und Filter*. 6. Auflage. Vieweg + Teubner, 2004.

[36] Butz, T. *Fouriertransformation für Fußgänger*. 7. Auflage. Vieweg + Teubner, 2011.

[37] Bird, S., Klein, E. und Loper, E. *Natural Language Processing with Python*. 1. Auflage. O'Reilly, 2009.

[38] Brownlee, J. *Deep Learning for Natural Language Processing - Develop Deep Learning Models for natural Language in Python*. Edition v1.1. Machine Learning Mastery, 2017.

[39] Mikolov, T., et al. *Efficient Estimation of Word Representations in Vector Space*. Proceedings of the International Conference on Learning Representations (ICLR 2013), Jan. 2013.

[40] Al-Khamaiseh, K. und ALShagarin, S. *A Survey of String Matching Algorithms*. International Journal of Engineering Research und app, Aug. 2014.

[41] Jokisch, O. und Hain, H. *A Trainable Method for the Phonetic Similarity Search in German Proper Names*. International Conference on Speech und Computer, Aug. 2017.

[42] Peng, T., Li, L., und Kennedy, J. *A comparison of techniques for name matching*. Zugriff: 12.10.2020. https://www.semanticscholar.org/paper/A-comparison-of-techniques-for-name-matching.-Peng-Li/67d1f1df6cb05c9541e63a671ebd4a828659146f.

[43] Wilz, M. *Aspekte der Kodierung phonetischer Ähnlichkeiten in deutschen Eigennamen*. Magisterarbeit. Universität zu Köln, 2005.

[44] Smart Mechatronics GmbH, o. V. *Consens*. Zugriff: 24.09.2020. https://www.smartmechatronics.de/consens.

[45] Gausemeier, J. et al. *Specification technique for the description of self-optimizing mechatronic systems*. Zugriff: 18.10.2020. https://doi.org/10.1007/s00163-008-0058-x: Research in Engineering Design Vol. 20, 2009.

[46] TullyFoote. *ROS.org Concepts*. Zugriff: 19.09.2020. http://wiki.ros.org/de/ROS/Concepts.

[47] GvdHoorn. *ROS.org joy*. Zugriff: 19.09.2020. http://wiki.ros.org/joy.

[48] MatthewWilson. *ROS.org CofiguringALinuxJoystick*. Zugriff: 19.09.2020. http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick.

[49] Martin, A. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Zugriff: 19.09.2020. CoRR, 2016.

[50] Tensorflow, o. V.: „Keras“. *The Sequential model*. Zugriff: 12.10.2020. https://www.tensorflow.org/guide/keras/sequential_model#when_to_use_a_sequential_model.

[51] GvdHoorn. *ROS.org Rospy*. Zugriff: 20.09.2020. http://wiki.ros.org/rospy.

[52] sw005320. *ESPnet: end-to-end speech processing toolkit*. Zugriff: 12.10.2020. https://github.com/espnet/espnet.

[53] Tensorflow API, o. V.: „Droput“. *tf.keras.layers.Dropout*. Zugriff: 24.09.2020. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout.

A Anhang

A.1 Abbildungen

Abbildung A.1: Darstellung des autonomen Logistik-Fahrzeugs

A.2 Inhalt Datenträger

1 Masterarbeit
2 Lastenheft
3 Daten
4 Software
   - *Python* Programme
   - *Matlab*-Skript zur Hauptkomponentenanalyse
   - *Matlab*-Skript zur Darstellung von Lernkurven

Eidesstattliche Versicherung

Eidesstattliche Versicherung zur Abschlussarbeit:

„Entwicklung und Verifikation einer Sprachverarbeitung für das autonome Logistik-Fahrzeug ALF “

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht.

Birkenfelde, 20. Oktober 2020	
Ort, Datum	Hannes Dittmann