Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Orhan Aykac

3D-Touchsensor zur Identifikation von Handgesten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Orhan Aykac

3D-Touchsensor zur Identifikation von Handgesten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Andreas Meisel
Zweitgutachter: Prof. Dr. Wolfgang Fohl

Eingereicht am: 22. November 2016

Orhan Aykac

Thema der Arbeit
3D-Touchsensor zur Identifikation von Handgesten

Stichworte
Neuronale Netze, Datenvorverarbeitung, Rekurrentes neuronales Netz, Feedforward neuronales Netz, Mustererkennung, Long short-term memory, Handgesten

Kurzzusammenfassung
In der heutigen Zeit nehmen intelligente Systeme immer mehr Einfluss auf unseren Alltag. In der hiesigen Arbeit wird ein Handgesten-Identifikationssystem entwickelt. Die Handgesten werden über einen 3D-Touchsensor getätigt. Es werden drei Szenarien erstellt, in welchen statische und dynamische Handgesten appliziert werden. Als Klassifikator werden Feedforward- und Rekurrente Neuronale Netze eingesetzt. Zusätzlich werden Verfahren für die Signalvorverarbeitung und die Netzkonfiguration vorgestellt. Weiterhin werden mehrere neuronale Netze mit verschiedenen Netzkonfigurationen entwickelt und bewertet. In der Auswertung wird das neuronale Netz mit der höchsten Klassifikationsleistung ermittelt und gezeigt, dass das System Handgesten zuverlässig erkennt.
Orhan Aykac

Title of the paper
3D touch sensor for identification of hand gestures

Keywords
Neural networks, Data pre-processing, Recurrent neural network, Feedforward neural network, Pattern recognition, Long short-term memory,Hand gestures

Abstract
In modern times, intelligent systems are increasingly influencing our daily lives. In this thesis, a hand gesture identification system is developed. The hand gestures are made over an 3D touch sensor. Three scenarios are created. These scenarios shown static and dynamic hand gestures. Feedforward and recurrent neural networks are used as classifiers. In addition, methods for signal preprocessing and network configuration are presented. Several neural networks with different network configurations are developed and evaluated. In the evaluation, the neural network with the highest classification performance is determined and shown that the system reliably detects hand gestures.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In der heutigen Zeit nimmt die Interaktion zwischen Mensch und Maschine zu. So existieren Systeme, in welche der Mensch anhand von Gesten und Sprachen in Kontakt mit Computern tritt. Dabei erleichtern die Computer die Arbeit der Menschen, indem sie ihnen Aufgaben abnehmen. So werden neuerdings in Fahrzeugen Systeme eingebaut, mit welchen der Mensch mithilfe von Gesten Funktionen im Fahrzeug bedient. Auf diese Weise können die Musiklautstärke, die Temperatur u.v.m. anhand von Handgesten eingestellt werden. Des Weiteren reagieren die Systeme im Fahrzeug auf Sprachkommandos. So können Anrufe anhand von Sprachkommandos getätigt werden, ohne die Tastatur des Handys zu berühren oder sogar ohne das Handy in die Hand nehmen zu müssen. Zudem werden dem Fahrzeugsystem im Wege von Sprachkommandos Navigationsziele und andere Kommandos geliefert. Dank diesen Möglichkeiten werden sowohl der Fahrkomfort als auch die Fahrzeugbedienung erleichtert.

Bei der Interaktion zwischen Mensch und Maschine bedarf es zumeist einer Intelligenz, mit welcher die Maschine fähig ist, mit dem Menschen zu interagieren. Für diesen Zweck können künstliche neuronale Netze angewandt werden. Anhand von künstlichen neuronalen Netzen werden Muster erkannt und klassifiziert. Sie werden unter anderem für die Sprach-, Schrift-, Gesichts- und Gestenerkennung eingesetzt.

Diese Arbeit beschäftigt sich mit der Entwicklung eines Systems, mithilfe dessen Handgesten klassifiziert werden. Als Klassifikator werden künstliche neuronale Netze eingesetzt.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung eines Systems, anhand welches Handgesten klassifiziert werden. Die Handgesten werden über einen 3D-Touchsensor getätigt. Als Klassifikator dienen künstliche neuronale Netze.

Die Handgesten werden in drei Szenarien eingeteilt. Hierbei enthält das erste Szenario statische Handgesten, während das zweite und das dritte Szenario dynamische Handgesten umfassen.

Für jedes Szenario werden mehrere künstliche neuronale Netze entwickelt und aus diesen jenes mit der optimalsten Netzkonfiguration selektiert. Die optimalste Netzkonfiguration ist diejenige, mithilfe welcher die höchste Klassifikationsleistung erzielt werden kann.

1.2 Gliederung

Die vorliegende Arbeit ist in sechs Kapitel gegliedert. Die Inhalte der Kapitel werden im Folgenden erläutert.

In Kapitel 1 erfolgt eine Einleitung in das Thema. Überdies wird das Ziel dieser Arbeit beschrieben.

In Kapitel 2 werden zunächst die Grundlagen behandelt. Im Einzelnen werden Methoden der Datenverarbeitung sowie der Aufbau und die Funktionsweise von neuronalen Netzen beschrieben.

Darauffolgend werden in Kapitel 3 der Aufbau und die Funktionsweise von Rekurrenten Neuronalen Netzen erläutert. Des Weiteren wird die Funktionsweise von Long-short-term-Memory-Netzen erklärt.

In Kapitel 4 wird daraufhin diese Arbeit analysiert. Dabei werden die einzelnen Szenarien und Verfahren zur Verarbeitung der Elektrodensignale vorgestellt. Anschließend werden Verfahren verdeutlicht, mithilfe deren neuronale Netze aufgebaut werden können, welche zu einer hohen Klassifikationsleistung führen.

In Kapitel 5 findet anschließend die Realisierung der Identifikation der Handgesten auf den 3D-Touchsensor statt. In diesem Rahmen wird für jedes einzelne Szenario die Realisierung beschrieben. Dabei kommen die Verfahren aus Kapitel 4 zum Einsatz. Zu jedem Szenario wird eine Auswertung und ein Fazit getätigt.

In Kapitel 6 erfolgt schließlich eine Zusammenfassung der Arbeit.

2 Grundlagen

Das Kapitel beschreibt die Grundlagen, auf denen diese Arbeit basiert. Zunächst werden Datenvorverarbeitungsverfahren vorgestellt. Anhand von Datenvorverarbeitungsverfahren werden u.a. die Eingabemuster für maschinelle Lernverfahren bearbeitet.

Anschließend werden die Grundlagen von neuronalen Netzen erläutert. Für die Identifikation der Handgesten werden neuronale Netze eingesetzt.

2.1 Datenvorverarbeitung

Im Rahmen der vorliegenden Arbeit werden für Trainings- und Testzwecke automatisch Daten generiert. Aus diesen Daten wird für jedes Szenario ein Datensatz erstellt.

Die meisten Datensätze (Runkler, 2015; García u. a., 2014), welche z.B. aus Sensoren gewonnen werden, enthalten Werte, welche außerhalb des Wertebereiches liegen, zu groß, fehlerhaft, redundant, unlesbar und/oder verrauscht sind. Die Fehler in den Datensätzen können aufgrund systematischer oder zufälliger Fehler verursacht werden. Die Qualität der Datensätze ist ein hochrelevanter Faktor, da fehlerhafte Datensätze die Klassifikationsleistung von Klassifikatoren beeinflussen, welche für maschinelle Lernverfahren eingesetzt werden.

Dieser Abschnitt befasst sich mit der Datenvorverarbeitung von Daten, welche für maschinelle Lernverfahren verwendet werden. Die Datenvorverarbeitung lässt sich in die Datenverarbeitung und die Datenreduktion unterteilen.

Bei der Datenverarbeitung werden die Daten in den Datensätzen verarbeitet und in eine einheitliche Form gebracht. Die Verwendung unverarbeiteter Daten als Eingabe führt zu Fehlern oder falschen Ergebnissen. Techniken, welche in der Datenverarbeitung Anwendung finden, sind

- Datensäuberung,
- Datentransformation,
- Datenintegration,

- Datennormalisierung und
- Identifikation von Störungen.

Die Datenreduktion enthält Techniken, um aus einer großen Datenmenge eine äquivalente kleine Datenmenge zu bilden. Diesbezüglich ist es besonders notwendig, die Datenstrukturen und Datenintegrität beizubehalten. Techniken, welche für die Datenreduktion verwendet werden, sind

- Merkmalsauswahl,
- Instanzen-Auswahl,
- Diskretisierung und
- Merkmalsextraktion.

Das Ergebnis der Datenvorverarbeitung ist ein Trainingsdatensatz, welcher frei von Störungen und nicht zu groß ist. Auf diese Weise wird das maschinelle Lernen vereinfacht und eine höhere Klassifikationsleistung erreicht.

Im Folgenden werden einige Datenvorverarbeitungsverfahren genannt. Die verwendeten Datenverarbeitungsverfahren werden in Abschnitt 4.3 näher beschrieben. Weitere Datenvorverarbeitungsverfahren sind in (García u. a., 2014; Runkler, 2015; Zhang u. a., 2015; Verma u. a., 2013; Kotsiantis u. a., 2006) zu finden.

Datenverarbeitung

2.1.1 Identifikation von Ausreißern

In (Runkler, 2015) werden Verfahren vorgestellt, mithilfe derer Ausreißer behandelt werden. Ausreißer entstehen u.a. infolge von Messfehlern und/oder Störungen an Sensoren und befinden sich oft außerhalb des erlaubten Wertebereiches.

Diese können mittels verschiedener Verfahren behandelt werden. Die simpelste Methode zur Festzustellen von Ausreißern erfolgt anhand der Untersuchung der Ober- und Untergrenzen der Wertebereiche. Dabei werden die Ausreißer durch die Maximal- bzw. Minimalwerte ersetzt (Runkler, 2015).

Darüber hinaus werden Ausreißer u.a. durch die Anwendung von Filtern, Clusteralgorithmen und Binningverfahren behoben.

2.1.2 Behandlung von Unvollständige Daten

Ein weiterer Bestandteil der Datenvorverarbeitung ist der Umgang mit fehlenden oder unvollständigen Daten. Bei der Bearbeitung von fehlenden Merkmalswerten müssen einige Aspekte berücksichtigt werden. Zunächst ist ist die Ursache für das Fehlen eines Wertes zu ermitteln. So kann ein Grund für das Fehlen eines Wertes darin bestehen, dass dieser verloren gegangen ist oder dessen Eintragung vergessen wurde.

Die einfachste Behandlung der fehlenden Merkmalswerte ist es, diese zu ignorieren. Sobald ein Merkmalswert einer Instanz fehlt, wird die gesamte Instanz verworfen. Eine andere Methode ist der Ersatz des fehlenden Merkmalswertes durch den am häufigsten auftauchenden Merkmalswert (Kotsiantis u. a., 2006).

2.1.3 Datennormalisierung

Datensätze enthalten oft Werte, deren Wertebereich sehr stark voneinander abweichen. Dies führt bei maschinellen Lernverfahren oft zu Problemen und Schwierigkeiten. Ziel der Normalisierung ist es, die Werte in den Datensätzen auf einen einheitlichen Wertebereich zu skalieren, was zugleich die weitere Verarbeitung vereinfacht.

Zwei bekannte Methoden, mithilfe derer Werte auf einen gemeinsamen Wertebereich skaliert werden, sind die Min-Max-Normalisierung sowie die z-score-Normalisierung.

Daten Reduktion

2.1.4 Auswahl der Instanzen

Ein wichtiger Aspekt der Datenvorverarbeitung und insbesondere der Datenreduktion, ist der Umgang mit großen Datenmengen. Für gewöhnlich enthalten große Datensätze Daten die redundant,unbrauchbar bzw. fehlerhaft sind und/oder mehrfach erscheinen.

Bei der Auswahl der Instanzen (J. Arturo u. a., 2010; Kotsiantis u. a., 2006) wird aus einem großen Datensatz eine äquivalente Teilmenge gebildet. Diese Teilmenge repräsentiert vollständig den originalen Datensatz. Das Ziel ist es, die Laufzeit von Algorithmen zu verkürzen.

Die Auswahl der Instanzen kann u.a. mit Verfahren wie dem k-Nearest-Neighbor-Algorithmus und Filter realisiert werden. Des Weiteren können Verfahren angewendet werden, welche zufällig aus dem großen Datensatz Instanzen auswählen und einen äquivalenten kleinen Datensatz bilden. Zu diesen Verfahren gehören das Simple-random-Sampling sowie das Stratified-Sampling.

Beim Simple-random-Sampling wird eine Teilmenge aus einem großen Datensatz gebildet. Dabei wird jedes Element der Teilmenge zufällig und mit gleicher Wahrscheinlichkeit gewählt. Gleichzeitig darf jedes Element des Datensatzes nur einmal in der Teilmenge enthalten sein. Die Auswahl der Elemente erfolgt durch einen Zufallsgenerator.

2.2 Neuronale Netze

Neuronale Netze ähneln in abstrahierter Form dem Neuronennetz des menschlichen Gehirns. Der Zweck künstlicher neuronaler Netze ist es, künstliche Intelligenz zu entwickeln. Im weiteren Verlauf dieses Kapitels werden die Grundlagen und Bestandteile von künstlichen neuronalen Netzen erläutert. Die Informationen stammen zum großen Teil von (Guenter und Karl F., 2010; Kriesel, 2007; Zell, 1994)

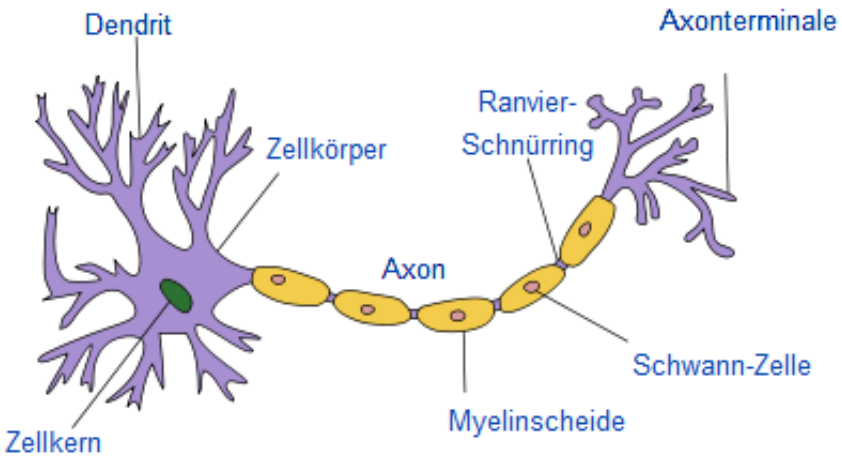

Abbildung 2.1: Nervenzelle des menschlichen Gehirns (Wikipedia, 2016d)

In Abbildung 2.1 ist eine Nervenzelle des menschlichen Gehirns abgebildet. Eine menschliche Nervenzelle besteht u.a. aus einem Zellkörper, ein oder mehreren Eingängen, den Dendriten sowie einem Ausgang, dem Axon. Die Axone sind über Synapsen mit Dendriten anderer Zellkörper verbunden und leiten Informationen an Neuronen weiter.

Die neuronalen Netze des menschlichen Gehirns werden als Analogie und Inspiration für in Computern simulierte künstliche neuronale Netze eingesetzt (Guenter und Karl F., 2010).

2.2.1 Aufbau eines neuronalen Netzes

Neuronale Netze (Kriesel, 2007; Guenter und Karl F., 2010) besitzen eine Eingabeschicht, verdeckte Schichten und eine Ausgabeschicht. Ferner können diese Schichten mehrere Neuronen enthalten, welche über Verbindungsgewichte miteinander verbunden sind. Die Neuronen in der Eingabeschicht erhalten Eingangswerte, die in bearbeiteter Form an die Neuronen der verdeckten Schicht weitergeleitet werden. Die Neuronen in der Ausgabeschicht erhalten die bearbeiteten Eingangswerte und geben diese an das System weiter.

Die einfachste Form eines neuronalen Netzes ist das Perzeptron. In Abbildung 2.2 ist das Perzeptron dargestellt, welches aus zwei Eingabeneuronen(rot) und einem Ausgabeneuron(blau) besteht.

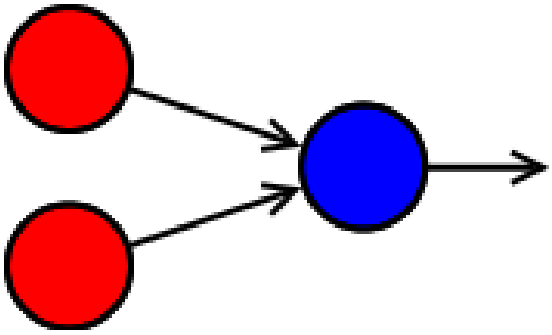

Abbildung 2.2: Perzeptron mit zwei Eingabeneuronen und einen Ausgabeneuron

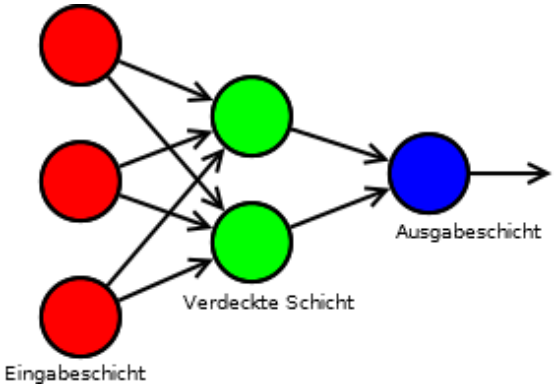

Abbildung 2.3: Dreischichtiges neuronales Netz

In Abbildung 2.3 ist ein neuronales Netz zu sehen, welches aus drei Schichten besteht. Die roten Neuronen entsprechen Eingabeneuronen, in Grün sind die verdeckten Neuronen und

in Blau sind die Ausgabeneuronen zu erkennen. Somit besteht ein neuronales Netz aus einer Eingabeschicht, keinen, einen oder mehreren verdeckten Schichten und einer Ausgabeschicht.

Verbindungen und Gewichte zwischen Neuronen

Eingabeneuronen (Guenter und Karl F., 2010; Kriesel, 2007) erhalten Eingangswerte, welche sie zu den Neuronen in der nächsten Schicht weiterleiten. Jeder Eingangswert eines Neurons wird mit einem, zu ihm zugeordneten, Gewicht multipliziert. Besteht das neuronale Netz aus mehreren verdeckten Schichten, so leiten die Neuronen nach Berechnung der Ausgabefunktion, die Werte weiter an die Neuronen der nächsten Schicht. Ausgabeneuronen reichen die Netzausgabe an das System weiter.

Das Gewicht beschreibt die Stärke zwischen zwei Neuronen zueinander. Im Falle eines positiven Gewichts hat ein Neuron einen anregenden Einfluss auf das andere. Ein negatives Gewicht bedeutet, dass ein Neuron einen hemmenden Einfluss auf das andere Neuron aufweist. Keinen Einfluss hat ein Neuron auf ein anderes, wenn das Gewicht Null ist.

Das Lernen geschieht durch die Änderungen der Gewichte zwischen den Neuronen. Dementsprechend enthalten die Gewichte das Wissen in neuronalen Netzen.

Neuronen einer Schicht haben in der Regel gerichtete Verbindungen zu allen Neuronen der folgenden Schicht. Mithin wird die Anzahl an Verbindungen durch die Anzahl an Neuronen der folgenden Schicht bestimmt. Der Eingangswert eines Neurons wird mit der Übertragungsfunktion berechnet.

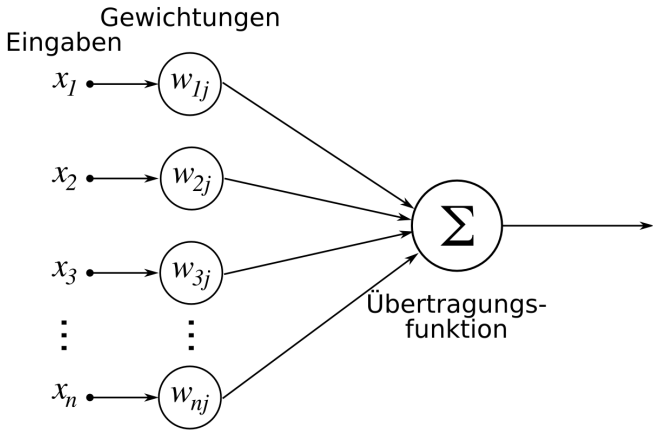

Abbildung 2.4: Darstellung der Übertragungsfunktion (Wikipedia, 2016b)

$$Netzeingabe = \sum_{i=1}^{n} x_i \cdot w_i j \tag{2.1}$$

Dabei entspricht

- n der Anzahl der Eingaben,
- x_i der Eingabe i und
- w_{ij} den Verbindungsgewichten.

In Abbildung 2.4 ist die Berechnung der Übertragungsfunktion eines Neurons dargestellt. Mit der Übertragungsfunktion wird die Netzeingabe des Neurons berechnet, siehe Gleichung 2.1

2.2.2 Aktivierungsfunktion

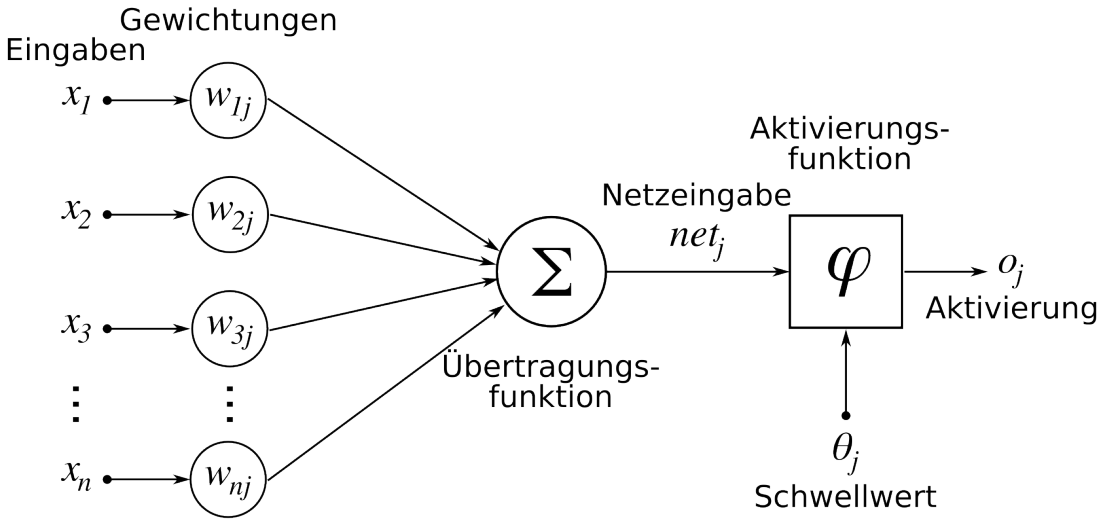

Abbildung 2.5: Darstellung der Aktivierungsfunktion (Wikipedia, 2016b)

In Abbildung 2.5 sind die Bestandteile eines Neurons abgebildet. Ein Neuron hat zu jedem Zeitpunkt einen Aktivierungszustand, welcher sich aus der aktuellen Netzeingabe und dem vorherigen Aktivierungszustand zusammensetzt. Dabei wird ein Neuron aktiv, falls der Schwellenwert überschritten wird.

Im nächsten Absatz werden drei Aktivierungsfunktionen(Transferfunktion) vorgestellt.

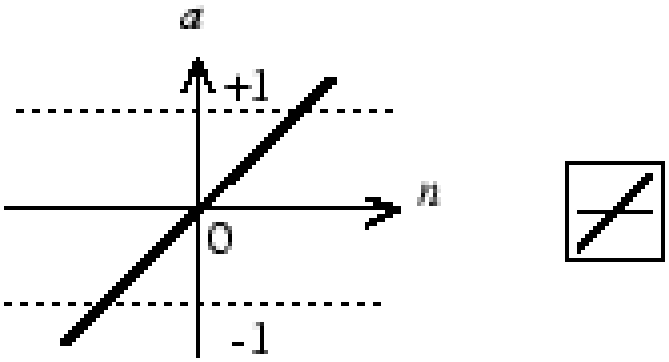

Abbildung 2.6: Lineare Aktivierungsfunktion (Matlab, 2016)

In Abbildung 2.6 ist die lineare Aktivierungsfunktion dargestellt. Die Eingabe der linearen Aktivierungsfunktion entspricht gleichzeitig der Ausgabe.

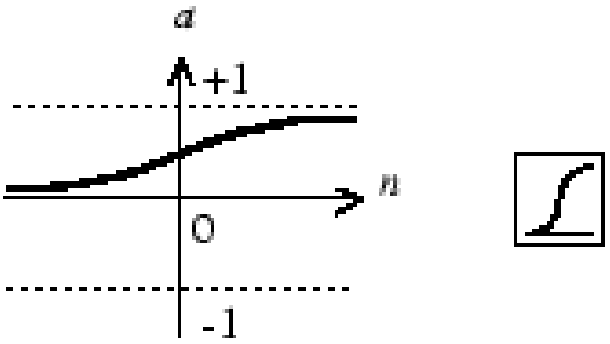

Abbildung 2.7: Logistische(Sigmoid) Aktivierungsfunktion (Matlab, 2016)

In Abbildung 2.7 ist die logistische Aktivierungsfunktion erkennbar. Diese Aktivierungsfunktion ist eine der beliebtesten, da sie differenzierbar ist. Die logistische Aktivierungsfunktion weist einen Wertebereich von 0 bis 1 auf.

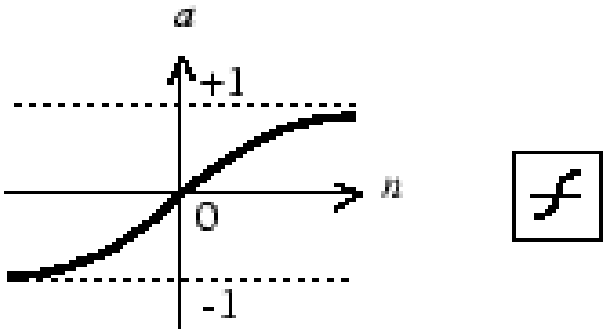

Abbildung 2.8: Tangens-hyperbolicus-Aktivierungsfunktion (Matlab, 2016)

In Abbildung 2.8 ist die Tangens-hyperbolicus-Aktivierungsfunktion abgebildet. Sie ist ebenso wie die logistische Aktivierungsfunktion differenzierbar. Der Wertebereich erstreckt sich von -1 bis 1.

2.2.3 Lernverfahren und Trainingsphase

Das Lernen in neuronalen Netzen erfolgt anhand der Modifikation ihrer Bestandteile. Einige dieser Bestandteile sind die Verbindungen, Verbindungsgewichte und Schwellenwerte. Die gängigste Art des Lernens neuronaler Netze wird durch die Änderung der Verbindungsgewichte realisiert. Hierbei lernt das neuronale Netz, indem es die Verbindungsgewichte nach bestimmten Regeln verändert (Kriesel, 2007).

Drei verschiedene Lernverfahren werden in neuronalen Netzen eingesetzt. Diese sind das überwachte Lernen, das bestärkende Lernen sowie das unüberwachte Lernen. Im folgenden Abschnitt wird das überwachte Lernverfahren näher erläutert, da es in dieser Arbeit zum Zuge kommt.

Trainingsphase

Das Lernen in neuronalen Netzen erfolgt in der Trainingsphase. Dem neuronalen Netz werden ein Trainingsdatensatz von Eingabemustern und abhängig von dem eingesetzten Lernverfahren die dazugehörigen Ausgabemuster vorgegeben. Nach der Trainingsphase sollte das neuronale Netz zu Generalisieren imstande sein.

Im Rahmen des überwachten Lernens besteht der Trainingsdatensatz aus den Eingabemustern und den dazugehörigen Ausgaben. Dem neuronalen Netz werden die Eingabemuster eingegeben, bevor die Netzausgabe mit der erwarteten Ausgabe verglichen wird. Durch die

Differenz zwischen der aktuellen Netzausgabe und der erwarteten Ausgabe wird der Fehler berechnet. Anhand des Fehlers werden schließlich die Verbindungsgewichte des neuronalen Netzes modifiziert. Die Verbindungsgewichte werden so lange modifiziert, bis der Fehler sehr klein ist. Das Ziel ist es, den Fehler in der Ausgabe soweit zu minimieren, dass eine hohe Klassifikationsleistung erzielt wird. Ferner wird angestrebt, das fertig trainierte neuronale Netz in die Lage zu versetzen, unbekannte Eingaben richtig zuzuordnen(generalisieren) (Kriesel, 2007).

Backpropagation

Neuronale Netze lernen durch die Änderung der Verbindungsgewichte. Die Verbindungsgewichte der Neuronen müssen derart gewählt bzw. eingestellt werden, dass der Fehler zwischen der tatsächlichen Ausgabe und der erwarteten Ausgabe des neuronalen Netzes gegen Null geht oder eine gewisse Fehlergrenze nicht überschreitet.

Die Änderungen der Verbindungsgewichte geschehen infolge der Anwendung des Backpropagation-Algorithmus. Bei diesem Algorithmus wird der Fehler in der Ausgabeschicht zurück bis zur Eingabeschicht geführt und dabei anhand des Gradientenabstiegsverfahrens die Verbindungsgewichte zwischen den Neuronen verändert.

Zunächst werden dem neuronalen Netz Eingabewerte aus dem Trainingsdatensatz geliefert und die dazugehörige Ausgabe berechnet. Anschließend wird diese Netzausgabe mit der gewünschten Ausgabe verglichen und der Fehler wird anhand der folgenden Fehlerfunktion berechnet.

$$E = \frac{1}{2} \cdot \sum_{i=1}^{n} (t_i \cdot o_i)^2 \tag{2.2}$$

Hierbei ist

- E der Fehler,
- n die Anzahl der Trainingsdaten,
- t_i die gewünschte Ausgabe und
- o_i die tatsächliche Ausgabe.

In einem mehrschichtigen neuronalen Netz sind die Ausgaben der Neuronen in der Ausgabeschicht bekannt, jedoch in den verdeckten Schichten nicht ermittelbar. Aufgrund dessen wird

der Fehler aus der Fehlerfunktion von der Ausgabeschicht zurück zur Eingabeschicht geführt, währenddessen die Verbindungsgewichte durch das Gradientenabstiegsverfahren verändert werden.

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} = \eta \cdot \delta_j \cdot x_i \tag{2.3}$$

$$\delta_j = \{\varphi'(net_j) \cdot (t_j - o_j) \tag{2.4}$$

$$\delta_j = \{\varphi'(net_j) \sum_k \delta_k \cdot w_j k \tag{2.5}$$

Dabei ist

- Δw_{ij} die Änderung des Gewichts w_{ij} der Verbindung von Neuron i zu Neuron j,
- η eine feste Lernrate,
- x_i die Ausgabe des Neurons i,
- t_j die erwartete Ausgabe des Ausgabeneurons j,
- i_j die tatsächliche Ausgabe des Ausgabeneurons j und
- k der Index der nachfolgenden Neuronen von j(Wikipedia, 2016b).

Gleichung 2.4 wird verwendet, um die Verbindungsgewichte der Ausgabeneuronen zu verändern. Gleichung 2.5 wird angewandt, um die Verbindungsgewichte der Neuronen in den verdeckten Schichten zu verändern.

Die Veränderung der Verbindungsgewichte wird mit der folgenden Gleichung realisiert.

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta_{w_{ij}} \tag{2.6}$$

Dabei ist

- w_{ij}^{neu} der neue Wert des Gewichts,

- w_{ij}^{alt} der alte Wert des Gewichts und
- $\Delta_{w_{ij}}$ die berechnete Änderung des Gewichts(Wikipedia, 2016b).

Diese Schritte werden zyklisch wiederholt, bis der Fehler der Ausgabeschicht soweit wie möglich minimiert oder eine bestimmte Anzahl an Iterationsschritten durchlaufen worden ist.

Testphase

In der Testphase wird das neuronale Netz nicht mehr verändert. In dieser Phase wird geprüft, ob das neuronale Netz, nach der Trainingsphase gelernt hat. Hierfür werden dem neuronalen Netz Eingabemuster aus den Testdaten präsentiert. Ordnet das Netz die Eingabemuster aus den Testdaten richtig zu, so hat das neuronale Netz gelernt und ist demzufolge in der Lage zu generalisieren.

2.2.4 Feedforward-Netz

Neuronale Netze können durch verschieden Strukturen gebildet werden. Dabei bestehen die neuronalen Netze stets aus den Bestandteilen, welche im vorherigen Abschnitt erwähnt wurden.

Ein Feedforward-Netz (Zell, 1994; Kriesel, 2007) entspricht einem neuronalen Netz, das eine bestimmte Struktur und Funktion besitzt. Dabei besteht das Feedforward-Netz aus

- einer Eingabeschicht,
- keiner, einer oder mehreren verdeckten Schicht(en) und
- einer Ausgabeschicht.

Feedforward-Netze, welche keine verdeckten Schichten aufweisen, werden als einschichtige Feedforward-Netze bezeichnet. Hingegen werden Feedforward-Netze mit einer oder mehreren Schicht(en) als mehrschichtige Feedforward-Netze betitelt.

Diese Schichten sind miteinander über Verbindungsgewichte der Neuronen verbunden. Zudem können die Schichten eine beliebige Anzahl von Neuronen enthalten. Das Besondere an Feedforward-Netzen ist, dass sie nur Verbindungen zulassen, welche vorwärtsgerichtet sind. Infolgedessen ist die Richtung des Informationsflusses von der Eingabeschicht über die verdeckten Schichten zur Ausgabeschicht. Insoweit ist es einem Neuron lediglich erlaubt,

Verbindungen zu Neuronen der nächsten Schichten zu haben. Feedforward-Netze besitzen keine rückgekoppelten Verbindungen zu Neuronen vorausgehender Schichten.

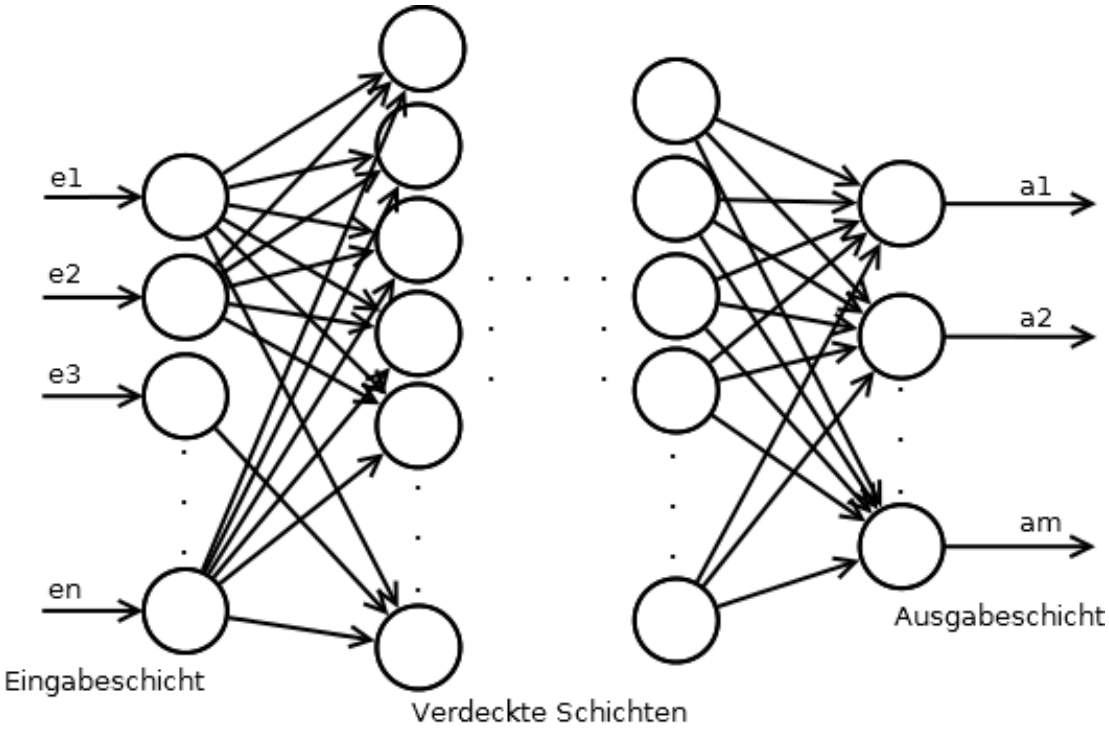

Abbildung 2.9: Struktur eines Feedforward-Netzes

Abbildung 2.9 verbildlicht die typische Struktur eines Feedforward-Netzes, das aus einer Eingabeschicht, mehreren verdeckten Schichten und einer Ausgabeschicht besteht.

Dabei ist

- en die Eingangswerte der Neuronen der Eingabeschicht und
- am die Ausgabewerte der Neuronen der Ausgabeschicht.

Die Neuronen in den Feedforward-Netzen können über zwei verschiedene Wege miteinander verbunden werden.

Bei vollverknüpften Verbindungen ist jedes Neuron einer Schicht mit jedem Neuron der nächsten Schicht verknüpft. Abbildung 2.10 illustriert ein Feedforward-Netz, welches vollverknüpfte Verbindungen zu den Neuronen der nächsten Schicht aufweist.

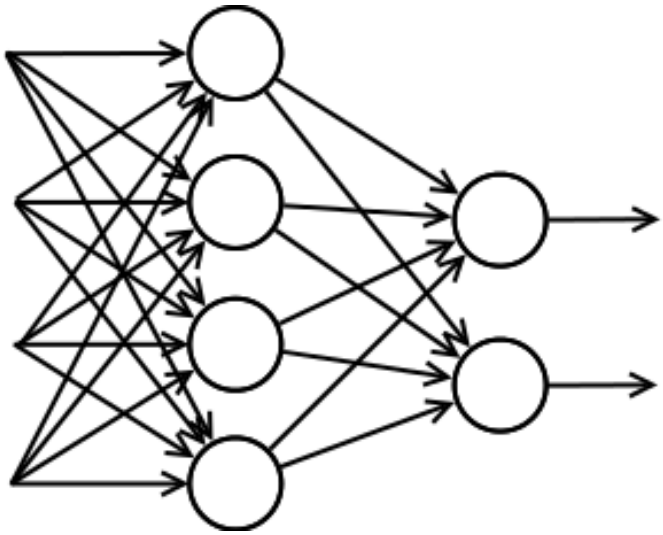

Abbildung 2.10: Vollverknüpftes Feedforward-Netz

In Abbildung 2.11 ist ein Feedforward-Netz mit Shortcut-Connections dargestellt. Diese Verbindungen zwischen Neuronen sind ebenfalls vorwärtsgerichtet. Hier können Neuronen mehrere Schichten überspringen und somit eine Verbindung zu Neuronen folgender Schichten aufbauen.

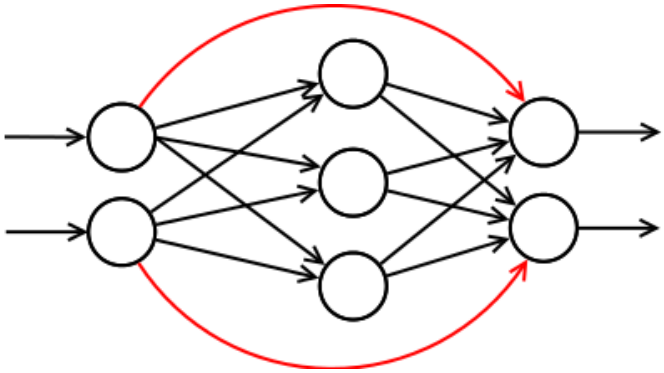

Abbildung 2.11: Feedforward-Netz mit Shortcut-Connections

3 Rekurrente Neuronale Netze

Die in dieser Arbeit entwickelte Anwendung verwendet u.a. Rekurrente Neuronale Netze für die Klassifikation der Handgesten. Rekurrente neuronale Netze sind Erweiterungen der Feedforward-Netze. Diese bestehen aus denselben Bestandteilen wie die Feedforward-Netze, besitzen jedoch zusätzliche rückgekoppelte Verbindungen und Kontextneuronen. Aufgrund dieser Erweiterungen sind Rekurrente Neuronale Netze mächtiger und in der Lage, komplexere Aufgaben zu lösen.

Die Rückkopplungen ermöglichen den Rekurrenten Neuronalen Netzen ein Gedächtnis aufzubauen, infolgedessen diese zustandsbehaftet sind. Damit können Rekurrente Neuronale Netze mehrere Sequenzen in Folge bearbeiten und die Informationen speichern. Auf diese Weise sind sie dazu fähig, Muster in Sequenzen zu erkennen und diese entsprechend zu klassifizieren.

Nachfolgend werden in diesem Kapitel der Aufbau und die Verbindungsmöglichkeiten Rekurrenter Neuronaler Netze vorgestellt. Des Weiteren wird anhand von zwei bekannten Rekurrenten Neuronalen Netzen, Jordannetzen und Elmannetzen, die Funktionsweise erläutert. Zusätzlich werden ein Trainingsverfahren Rekurrenter Neuronaler Netze, das Backpropagation-through-Time, vorgestellt und die Grenzen von Rekurrenten Neuronalen Netzen erwähnt.

3.1 Aufbau und Rückkopplungen

Rekurrente Neuronale Netze (Kriesel, 2007) ähneln ihrer Struktur nach den Feedforward-Netzen. Die Rekurrenten Neuronalen Netze besitzen ebenfalls

- eine Eingabeschicht,
- keine, eine oder mehrere verdeckte Schicht(en) und
- eine Ausgabeschicht.

Mittels der internen Rückkopplungen sind die Rekurrenten Neuronalen Netz in der Lage, sich selbst zu beeinflussen. Die internen Rückkopplungen erfolgen zeitversetzt um einen Taktschritt t^{-1}. Im Folgenden werden drei rückgekoppelte Verbindungsarten vorgestellt.

- Direkte Rückkopplungen,
- indirekte Rückkopplungen und
- laterale Rückkopplungen.

Direkte Rückkopplungen sind Verbindungen eines Neurons zu sich selbst. Auf diese Weise sind Neuronen imstande, ihren eigenen Aktivierungszustand zu stärken oder zu hemmen (Kriesel, 2007). In Abbildung 3.1 ist ein Neuron mit einer direkten Rückkopplung dargestellt.

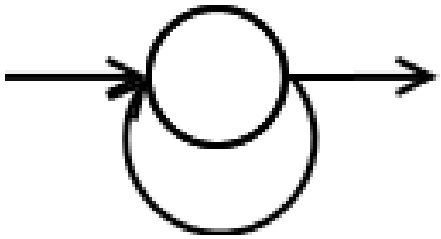

Abbildung 3.1: Ein Neuron mit direkter Rückkopplung

Indirekte Rückkopplungen sind Verbindungen, in welchen Neuronen einer Schicht mit den Neuronen einer vorherigen Schicht verbunden sind. Aufgrund dessen können sich Neuronen selbst mittelbar beeinflussen (Kriesel, 2007). In Abbildung 3.2 ist ein Neuron durch eine indirekte Verbindung abgebildet.

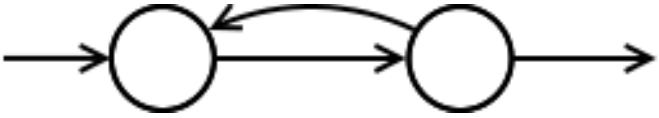

Abbildung 3.2: Darstellung der indirekten Rückkopplung eines Neurons

Laterale Verbindungen zwischen Neuronen sind Verbindungen von Neuronen derselben Ebene. An dieser Stelle versucht ein Neuron, die anderen Neuronen derselben Ebene zu hemmen und auf diese Weise sich selbst zu stärken (Kriesel, 2007). Abbildung 3.3 zeigt eine laterale Verbindung zwischen zwei Neuronen.

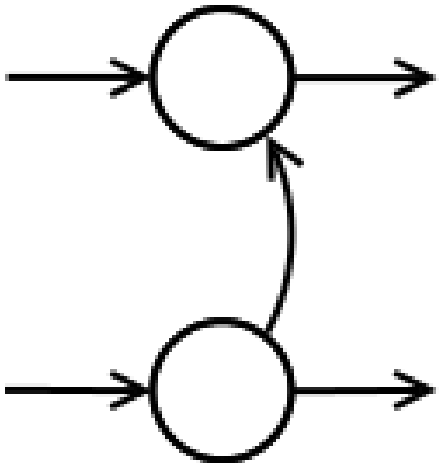

Abbildung 3.3: Darstellung einer lateralen Rückkopplung eines Neurons

Kontextneuronen

Kontextneuronen (Kriesel, 2007) kommt in Rekurrenten Neuronalen Netzen eine sehr gewichtige Funktion zu. Sie arbeiten wie ein Speicher, welcher die letzte Ausgabe eines Neurons der verdeckten Schicht oder der Ausgabeschicht zwischenspeichert. Kontextneuronen befinden sich außerhalb des eigentlichen neuronalen Netzes in der Kontextschicht. Somit weisen Rekurrente Neuronale Netze zusätzlich Kontextschichten auf, in welchen sich die Kontextneuronen befinden.

Die Rückkopplungen in Rekurrenten Neuronalen Netzen werden über Kontextneuronen in das neuronale Netz geführt. Die Kontextneuronen nehmen die aktuelle Ausgabe der Neuronen in den verdeckten Schichten und/oder der Ausgabeschicht auf und leiten sie im nächsten Taktschritt erneut an die Neuronen der jeweiligen Schichten weiter. Kontextneuronen stehen über gewichtete Verbindungen mit den Neuronen der jeweiligen Schichten in Verbindung.

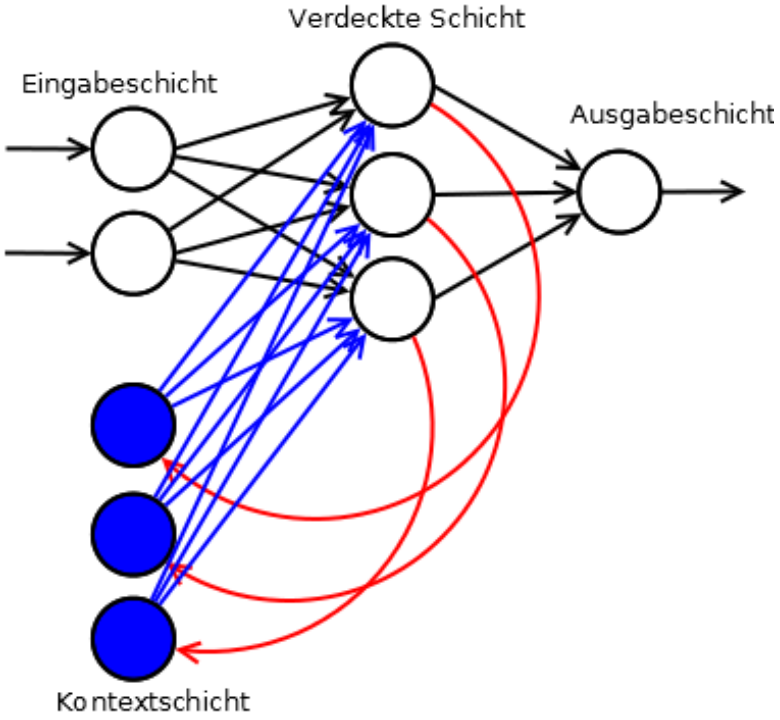

Abbildung 3.4: Rekurrentes Neuronales Netz mit einer Kontextschicht

In Abbildung 3.4 ist ein Rekurrentes Neuronales Netz mit einer Kontextschicht illustriert. Diesbezüglich sind die Neuronen der Kontextschicht in Blau dargestellt, während die roten Pfeile den Rückkopplungen der Neuronen der verdeckten Schicht zu den Neuronen der Kontextschicht entsprechen. Die blauen Pfeile präsentieren vollverknüpften Verbindungen zu den Neuronen der verdeckten Schicht.

3.2 Funktion Rekurrenter Neuronaler Netze

Wie bereits erwähnt, enthalten Rekurrente Neuronale Netze sämtliche Bestandteile eines Feedfoward-Netzes und ergänzend rückgekoppelte Verbindungen. Infolge der Rückkopplungen sind sie mächtiger und komplexer. Rekurrente Neuronale Netze eignen sich insbesondere für die Erkennung von zeitveränderlichen Mustern. Des Weiteren können sie für Prognosen verwendet werden. In diesem Abschnitt werden anhand des Jordan- und Elmannetzes (Kriesel, 2007) die möglichen Strukturen dargestellt und die Funktionsweise Rekurrenter Neuronaler Netze vorgestellt.

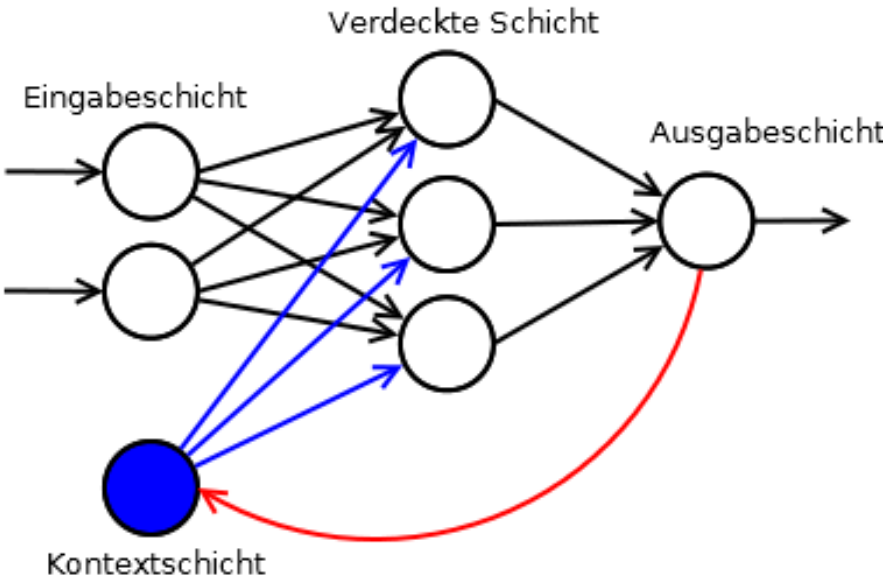

Abbildung 3.5: Struktur eines Jordannetzes

In Abbildung 3.5 ist ein **Jordannetz** dargestellt. In einem Jordannetz befindet sich die Kontextschicht auf der gleichen Ebene wie die Eingabeschicht des Rekurrenten Neuronalen Netzes. Dabei sind die Ausgaben des Rekurrenten Neuronalen Netzes sind mit den Kontextneuronen verbunden. Die Verbindungen zwischen den Neuronen der Ausgabeschicht und den Kontextneuronen in der Kontextschicht sind gewichtet. Die Kontextneuronen sind mit den Neuronen der verdeckten Schicht vollverknüpft und gewichtet verbunden. Die Anzahl der Kontextneuronen entspricht der Anzahl an Neuronen in der Ausgabeschicht.

Die Zustände in den verdeckten Schichten werden in Rekurrenten Neuronalen Netzen durch die aktuelle Eingabe und zusätzlich durch die vorherige Ausgabe des Netzes beeinflusst. Dementsprechend erzeugt das neuronale Netz bei gleichen Eingabewerten verschiedene Ausgaben. Dies wird durch die Rückkopplung verursacht, welche verschiedene netzinterne Zustände erzeugt. Die Kontextneuronen enthalten die vorherige Ausgabe aus dem vorausgehenden Berechnungsschritt.

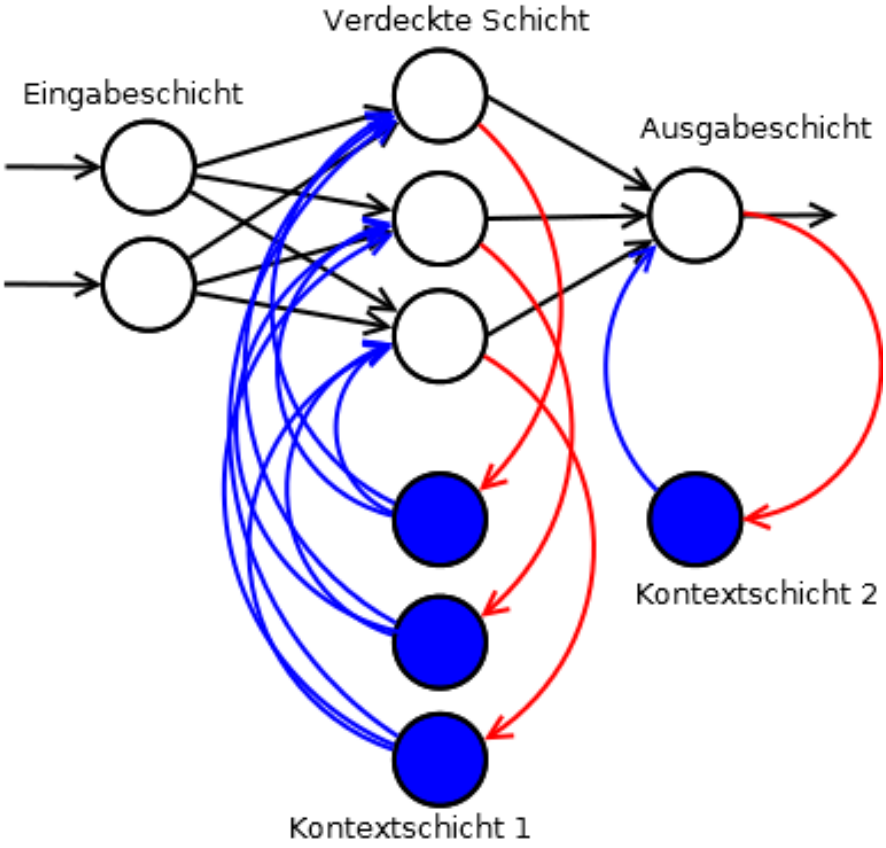

Abbildung 3.6: Struktur eines Elamnnetzes

Ein weiteres Exempel Rekurrenter Neuronaler Netze ist das **Elmannetz**. In Abbildung 3.6 ist die Struktur des Elmannetzes dargestellt. Das Elmannnetz ist eine Modifikation des Jordannetzes. Wie in Abbildung 3.6 erkennbar, weist ein Elmannetz je verdeckte/Ausgabeschicht eine äquivalente Kontextschicht auf. Die Kontextschicht besitzt genauso viele Neuronen wie die zugehörige Netzschicht. Die Ausgaben der Neuronen der verdeckten Schicht und Ausgabeschicht werden an die Neuronen der zugehörigen Kontextschicht geleitet. Im nächsten Taktschritt geben die Kontextneuronen die vorigen Ausgaben an die zugehörigen Neuronen der verdeckten Schichten und Ausgabeschichten vollverknüpft weiter. Sämtliche rückgekoppelten Verbindungen sind ebenfalls gewichtet.

Durch die Rückkopplungen sind die Rekurrenten Neuronalen Netze in der Lage, Folgen von Mustern zu erkennen und diese entsprechend zu klassifizieren. Die Muster können nicht unabhängig voneinander klassifiziert werden, da sie aus nacheinander folgenden Mustern bestehen. Überdies sind die zeitlichen Positionen der Muster von Bedeutung. Diese müssen

in festen Sequenzen geordnet werden. Rekurrente Neuronale Netze können verschiedene Strukturen aufweisen und je nach Anwendungsgebiet variieren.

3.3 Training

Infolge der Rückkopplungen sowie der Zustandsvielfalt in den verdeckten Schichten und in der Ausgabeschicht der Rekurrenten Neuronalen Netze erweist sich das Training mit dem Backpropagation-of-error-Algorithmus als schwierig.

Ein Verfahren, mithilfe dessen Rekurrente Neuronale Netze trainiert werden können, ist der Backpropagation-through-time-Algorithmus(BPTT) (Kriesel, 2007), welcher einer Erweiterung des Backpropagation-of-error-Algorithmus entspricht. Im Rahmen dieses Verfahrens wird das Rekurrente Neuronale Netz mit den Rückkopplungen durch ein weitaus größeres, vorwärtsgerichtetes neuronales Netz ohne Rückkopplungen ersetzt. In Abbildung 3.7 ist ein Rekurrentes Neuronales Netz abgebildet, welches aus

- einer Eingabeschicht mit drei Eingabeneuronen,
- einer Ausgabeschicht mit zwei Ausgabeneuronen und
- einer Kontextschicht mit zwei Kontextneuronen besteht.

Die Kontextschicht befindet sich auf der Ebene der Eingabeschicht und ist mit den Neuronen der Ausgabeschicht rückgekoppelt verbunden.

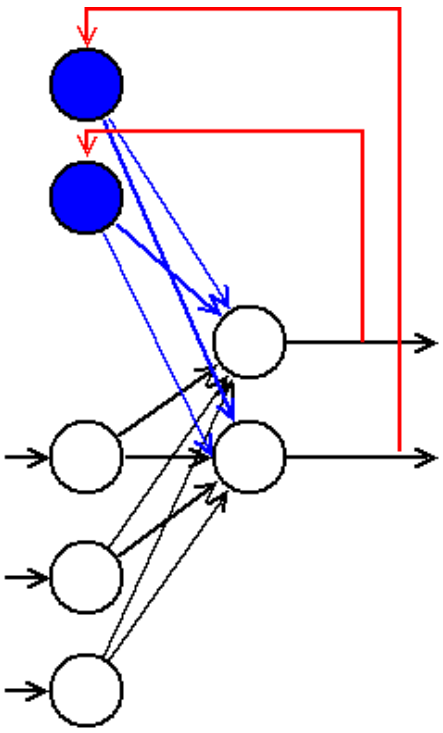

Abbildung 3.7: Rekurrentes Neuronales Netze

Das Rekurrente Neuronale Netz aus Abbildung 3.7 wird mit dem Backpropagation-through-time-Algorithmus trainiert, in dessen Rahmen zunächst die Rückkopplungen aufgelöst werden. Im Anschluss daran wird ein äquivalentes neuronales Netz über die Kontextschichten eingefügt. Dabei stellen die Kontextneuronen die Ausgabe der übergeordneten neuronalen Netze dar. Die Länge der Eingabesequenz bestimmt die Anzahl an äquivalenten neuronalen Netzen, welche über die Kontextschicht übereinander gesetzt werden. Folglich hängt die Größe des neuronalen Netzes von der Eingabesequenz ab. Die Eingabemuster werden in die Neuronen der Eingabeschichten des großen neuronalen Netzes eingegeben. Dabei erhält die oberste Eingabeschicht den ersten Sequenzwert und die unterste Eingabeschicht den letzten Sequenzwert. In Abbildung 3.8 sind die Rückkopplungen des Rekurrenten Neuronalen Netzes aus Abbildung 3.7 aufgelöst. Des Weiteren sind zwei äquivalente neuronale Netze über die Kontextschichten gesetzt.

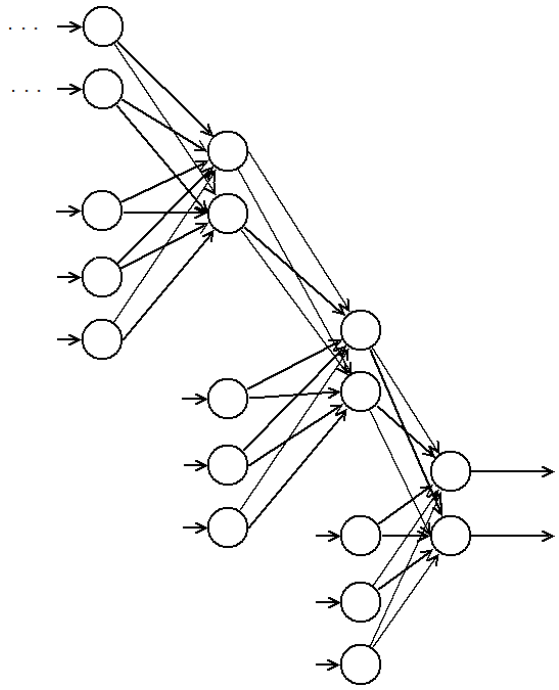

Abbildung 3.8: Auflösung der Rekurrenz durch das BBTT

Nach der Auflösung der Rekursion und dem Bilden des großen äquivalenten neuronalen Netzes kann das neuronale Netz mithilfe des Backpropagation-of-error-Algorithmus trainiert werden.

Ein Nachteil im Zusammenhang mit dem Backpropagation-through-time-Algorithmus ist, dass es zu großen neuronalen Netzen führen kann, infolgedessen die Trainingsdauer erhöht wird. Die Größe der Eingabesequenz entscheidet über die Größe des neuronalen Netzes.

Weitere Verfahren, anhand derer Rekurrente Neuronale Netze trainiert werden können, sind u.a. das Teacher Forcing, Rekurrentes Backpropagation und evolutionäre Algorithmen.

3.4 Vanishing-gradient-Problem

Das Training großer Rekurrenter Neuronaler Netze mit vielen verdeckten Schichten ist sehr zeitaufwendig und führt nicht immer zum Ziel. Eines der Schwierigkeiten ist das sogenannte Vanishing-gradient-Problem (Adam u. a., 2016).

Aufgrund der zahlreichen Schichten eines Rekurrenten Neuronalen Netzes verschwindet der Gradient beim rückwärts propagieren. Die Eingaben werden während des Informationsflusses durch Multiplikationen modifiziert. Somit hat der Gradient, je weiter die jeweiligen Schichten

von der Ausgabeschicht entfernt sind, weniger Einfluss auf die Veränderung der Verbindungsgewichte. Hierdurch wird der Einfluss des Gradientenabstiegsverfahrens, auf frühere verdeckte Schichten verringert. Als Folge des Verschwindens des Gradienten können die Gewichte nicht derart eingestellt bzw. verändert werden, dass sich der Fehler in der Ausgabe minimiert. Somit kann das neuronale Netz nicht lernen.

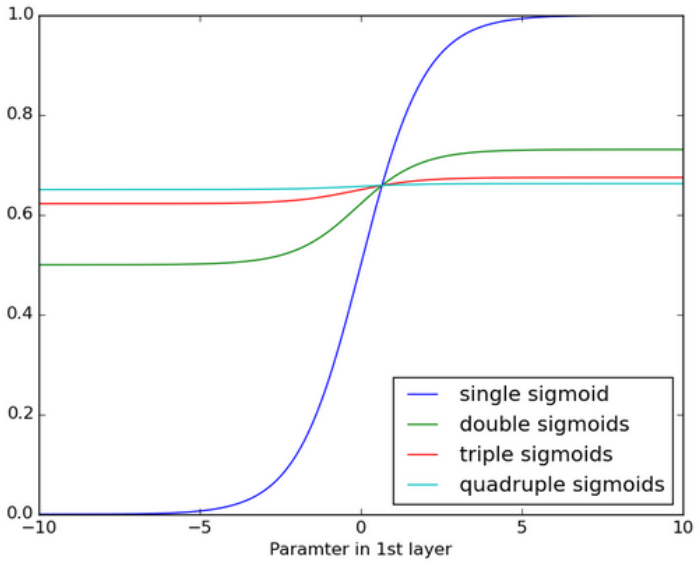

Abbildung 3.9: Vanishing-gradient-Problem (Adam u. a., 2016)

Die mehrfachen Operationen wie Multiplikationen und die Sigmoid-Funktionen, welche auf eine Eingabe Anwendung finden, resultieren in der Verkleinerung des Einflusses des Gradient beim Rückwärtspropagieren, je weiter die verdeckten Schichten von der Ausgabeschicht entfernt sind. In Abbildung 3.9 ist erkennbar, wie sich die Werte nach wiederholter Anwendung der Sigmoid-Funktion ändern. Je öfter die Sigmoid- Funktion angewandt wird, desto flacher werden die Werte und die Kurven verschwinden. Gleiches geschieht in Bezug auf den Gradienten, je weiter man sich von der Ausgabeschicht entfernt. Aufgrund dessen können die Verbindungsgewichte der früheren Schichten kaum verändert werden und das neuronale Netz lernt nicht.

3.5 Long-short-term-memory-Netze (LSTM)

Long-short-Term-memory-Netze (Adam u. a., 2016; Wikipedia, 2016c; Olah, 2015) sind eine besondere Art von Rekurrenten Neuronalen Netzen. Einfache Rekurrente Neuronale Netze sind gut geeignet, falls die Eingabesequenzen klein sind und demnach ein Kurzzeitgedächtnis angelegt wird. Bei langen Eingabesequenzen werden im Rahmen des Trainings große neuronale Netze mit vielen Schichten mittels des Backpropagation-through-time-Algorithmus erzeugt. Dies führt zum Vanishing-gradient-Problem in einfachen Rekurrenten Neuronalen Netzen, wie im vorausgehenden Abschnitt erläutert.

Mitte der 90er Jahre entwickelten Sepp Hochreiter und Jürgen Schmidhuber ein Verfahren,die Long-short-term-memory-Netze, mit welchem das Vanishing-gradient-Problem behoben werden kann (Hochreiter und Schmidhuber, 1997). Dabei wird der Fehler, welcher durch die Schichten rückwärts propagiert wird erhalten, so dass die Verbindungsgewichte durch das Gradientenabstiegsverfahren zugleich in den von der Ausgabeschicht weit entfernten Schichten eingestellt bzw. verändert werden können. Auf diese Weise ist es möglich, große Rekurrente Neuronale Netz mit reichlich Schichten zu trainieren.

In dieser Arbeit werden u.a. einfache Rekurrente Neuronale Netze verwendet, da die Eingabesequenzen klein sind und demzufolge das Vanishing-gradient-Problem umgangen wird. Im Folgenden werden der Aufbau und die Funktionsweise von Long-short-term-memory-Netzen erläutert.

Aufbau von LSTM

Long-short-term-memory-Netze (Adam u. a., 2016; Wikipedia, 2016c; Olah, 2015) weisen sogenannte besondere gated Zellen in den verdeckten Schichten aus. In den gated Zellen können die Informationen gespeichert, gelesen oder gelöscht werden. Die Zellen lernen zu entscheiden, wann die Informationen gelesen, gespeichert oder gelöscht werden. Diese Entscheidungen werden über die gates getroffen, die geöffnet oder geschlossen sein können.

Die Verbindungen in einer Zelle sind alle gewichtet. Die Verbindungsgewichte der Zellen werden u.a., mit dem Backpropagation-Algorithmus, während der Trainingsphase eingestellt. Entscheidungen, ob ein Eingangswert gelesen, geschrieben oder der aktuelle Zellzustand ersetzt wird, hängen von den Operationen ab, die auf den Eingangswerten ausgeführt werden.

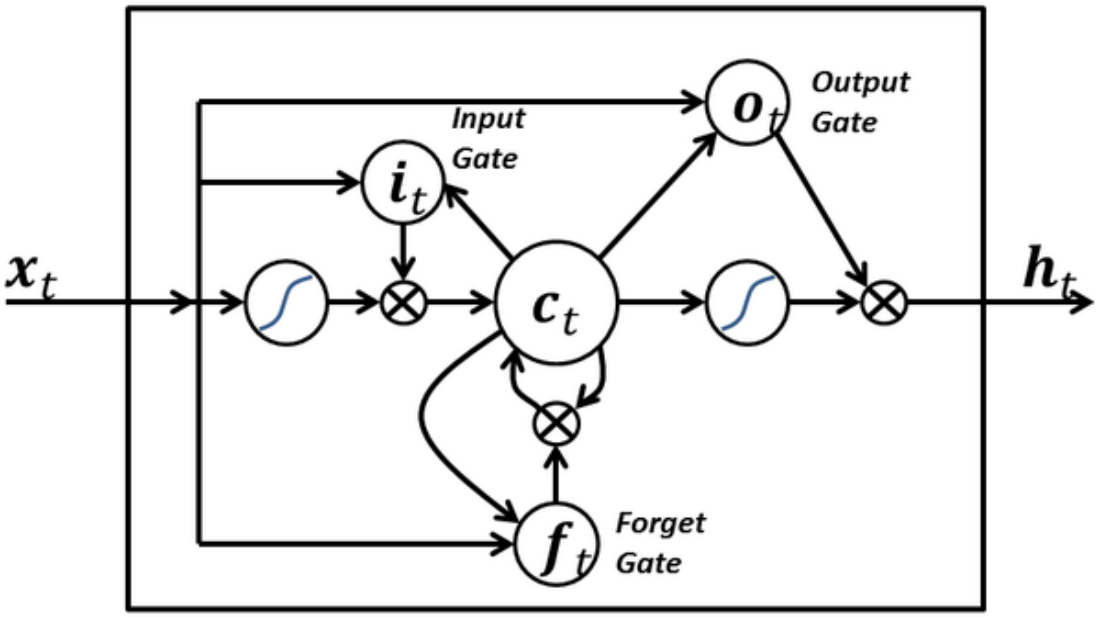

Abbildung 3.10: Long-short-term-Memory Zelle (Wikipedia, 2016c)

In Abbildung 3.10 ist eine Memory Zelle abgebildet. Eine Zelle enthält

- einen oder mehrere Eingangwert(e) x_t,
- ein Eingangsgate i_t,
- ein Vergessgate f_t,
- ein Ausgangsgate o_t
- einen Zellzustand c_t und
- eine Zellenausgabe h_t.

Die Zelle als auch die gates enthalten Funktionen und Operationen, welche auf die eintreffenden Informationen angewandt werden. Die Verbindungsgewichte filtern die Eingänge der gates, wobei sämtliche gates und die Zelle stets die gleichen Eingangswerte erhalten. Die Eingangswerte eines gates und der Zelle entsprechen der aktuellen Eingabe sowie der vorherigen Ausgabe. Dabei entscheiden die gates, ob die aktuelle Eingabe die vorherige ersetzt, Einfluss auf den Zellzustand nimmt und/oder die Ausgabe des neuronalen Netzes beeinflusst.

An allen Eingängen der gates und der Zelle werden die Übertragungsfunktionen der Eingangswerte gebildet. Auf diese Übertragungsfunktionen werden die Sigmoid- oder Tangenshyperbolicus-Aktivierungsfunktionen angewendet. An dem Eingabegate und dem Ausgabegate werden die Einflüsse der aktuellen Eingabe auf den Zellzustand und der nächsten Ausgabe berechnet.

Das Besondere an LSTM-Netzen und die Ursache, weshalb der Fehler konstant durch allen Schichten gehalten wird, ist die Plus Operation, welche in der Zelle ausgeführt wird.

Die Zellen sind in den verdeckten Schichten eines Rekurrenten Neuronalen Netzes. Dabei kann ein Rekurrentes Neuronales Netz zahlreiche Zellen enthalten. Long-short-term-memory-Netze können Mustern mit vielen Sequenzen klassifizieren und sind demgemäß mächtiger als einfache Rekurrente Neuronale Netze.

4 Analyse

Dieses Kapitel beschreibt die Analyse der zu entwickelten Musterkennung von Handgesten. Zunächst wird der 3D-Touchsensor mit seinen Komponenten vorgestellt. Hierzu wird die Funktionsweise der Signalerzeugung erklärt.

Anschließend werden die verschiedenen Handgesten vorgestellt. Die Handgesten sind in drei Szenarien unterteilt. Jedes Szenario besteht aus einem Datensatz mit den zugehörigen Handgesten.

In Abschnitt 4.3 werden Signalverarbeitungs- und Datenreduktionsverfahren präsentiert, mithilfe welcher die Elektrodensignale aus dem 3D-Touchsensor verarbeitet werden können.

Zuletzt werden Verfahren aufgeführt, anhand derer neuronalen Netze u.a. aufgebaut werden.

4.1 3D-Touchsensor MGC3130

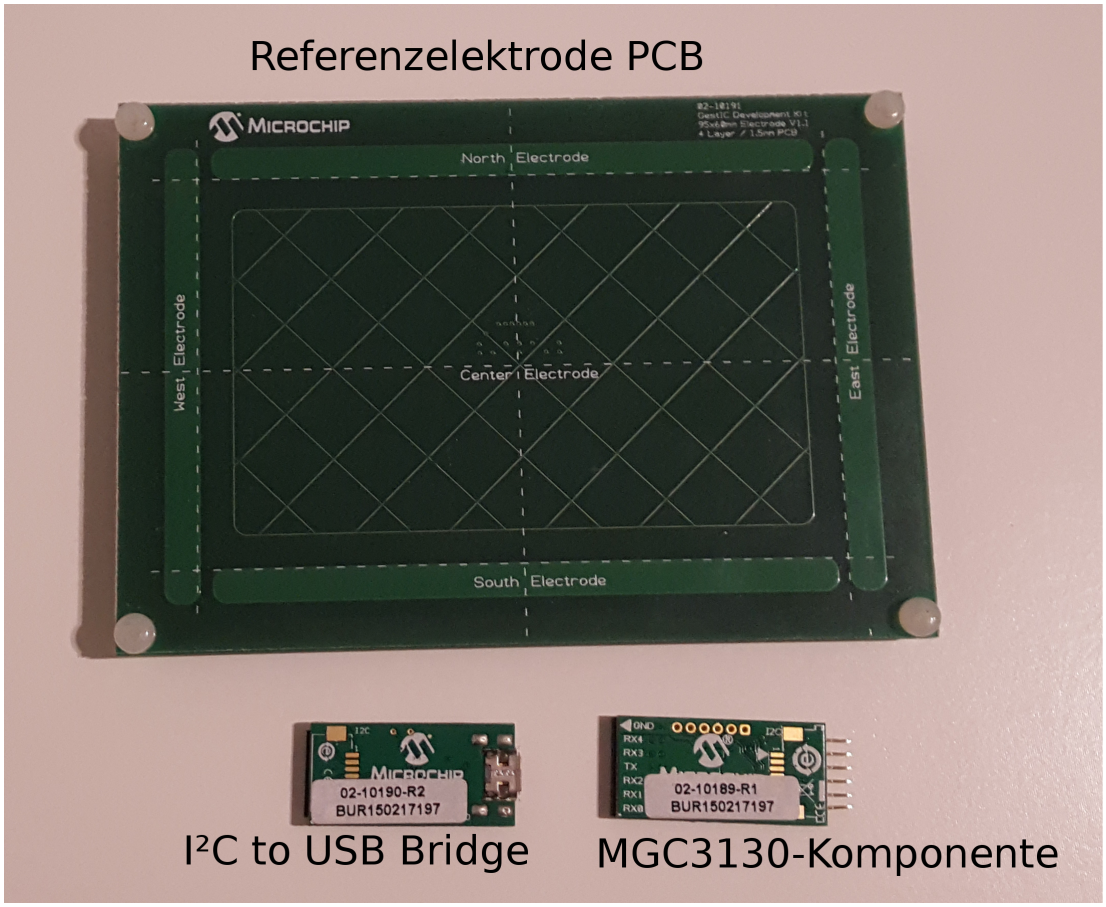

Abbildung 4.1: 3D-Touchsensor MGC3130

Der 3D-Touchsensor in Abbildung 4.1 wurde von Microship entwickelt. Er setzt sich aus drei Komponenten zusammen, einer MGC3130-Einheit, einr I²C to USB Bridge sowie einer Referenzelektrode PCB[1].

[1] Printed circuit board

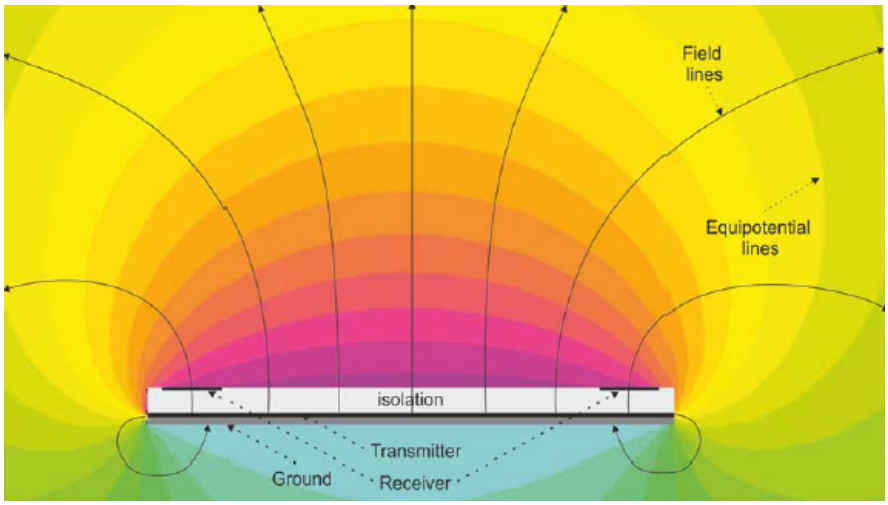

Abbildung 4.2: Ungestörtes elektrisches Feld des 3D-Touchsensors (Inc., 2016b)

In Abbildung 4.2 ist das elektrische Feld des 3D-Touchsensors ersichtlich. Die schwarzen Pfeile stellen die Feldlinien dar, mithilfe welcher die Richtung des elektrischen Feldes dargestellt wird. Elektrische Felder entstehen, falls Körper elektrisch geladen werden, und bereiten sich im Raum aus. Ein elektrisch konstantes Feld wird bei Verwendung von Gleichspannung erzeugt. Im Falle der Anwendung von Wechselspannung variiert die Stärke der Ladung über die Zeit. Elektromagnetische Felder entstehen, falls die Ladungen in Bewegung gesetzt und infolgedessen elektromagnetische Wellen freigesetzt werden.

Der 3D-Touchsensor verwendet ein elektrisches Feld, um Bewegungen auf und über der Referenzelektrode PCB zu erfassen. Die Elektroden auf dem 3D- Touchsensor sind weitaus kleiner als die Wellenlängen. Infolge dieser Geometrie sind die magnetischen Felder vernachlässigbar, sodass man ein näherungsweise elektrostatisches Feld erhält. Dieses Feld wird zur Erfassung von Objekte eingesetzt, welche sich über dem 3D-Touchsensor befinden (Inc., 2016b).

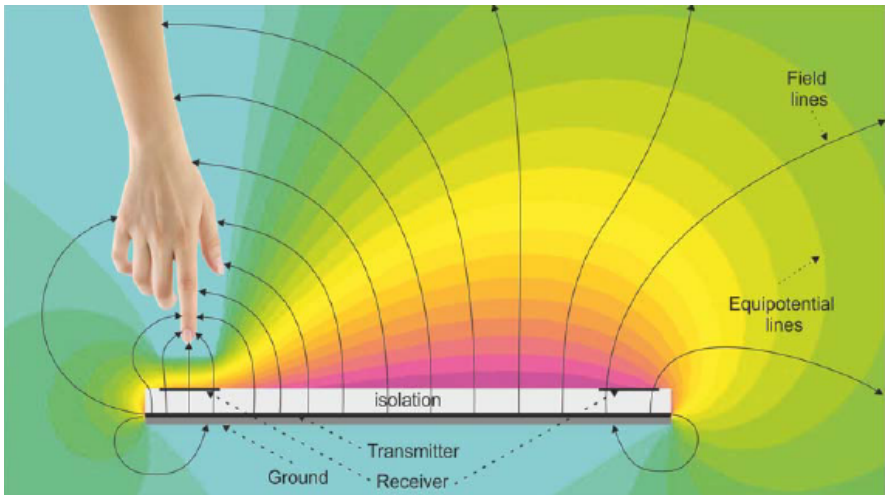

Abbildung 4.3: Gestörtes elektrisches Feld des 3D-Touchsensors (Inc., 2016b)

Abbildung 4.3 stellt die Änderung des elektrischen Feldes bei Einführung der Hand in das Feld dar. Das elektrische Feld wird an der Stelle der Hand gekrümmt. Durch die Leitfähigkeit des menschlichen Körpers werden die Feldlinien angezogen und geerdet. Dies hat zur folge, dass die elektrische Feldstärke an diesem Punkt sinkt.

Mittels der Elektroden auf dem 3D-Touchsensor werden die Änderung des elektrischen Feldes und die Position der Störung ermittelt. Aus diesen Informationen können die Positionen der Hand ermittelt werden. Des Weiteren können die Informationen als Eingabemuster für die Handgesten gesammelt werden. Diese Eingabemuster fungieren anschließend als die Eingaben für maschinelle Lernverfahren.

MGC3130-Komponente

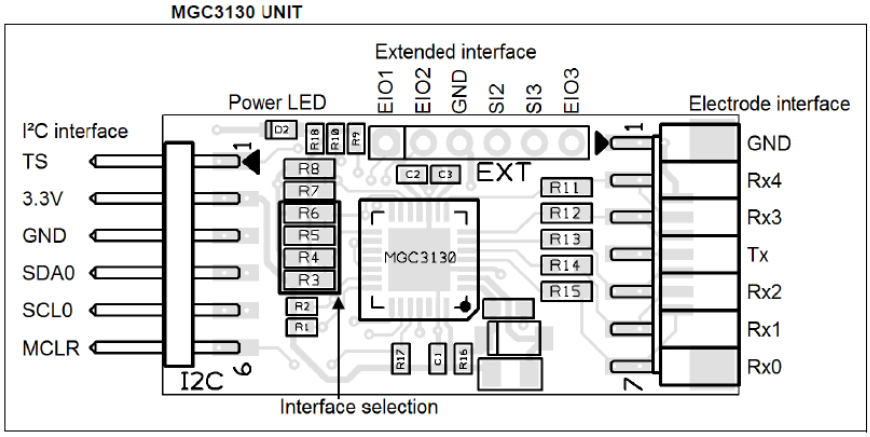

Abbildung 4.4: MGC3130-Komponente (Inc., 2016a)

In Abbildung 4.4 ist die MGC3130-Komponente des 3D-Touchsensors dargestellt. Der wichtigste Teil des MGC3130 ist der 3D-Tracking und Gesten-Controller, welcher das Transmitter-Signal kreiert, um das elektrische Feld zu erzeugen. Ferner steuert es die Kommunikation zwischen den Geräten, empfängt die Signale und verarbeitet sie auf der SPU. Die Digital Signal Processing Unit(SPU) ist in den Controller integriert, steuert die Hardware-Blöcke und verarbeitet die Algorithmen. überdies filtert sie die Sensordaten und gibt kontinuierlich Positionsinformationen.

Die MGC3130-Einheit besitzt zwei Schnittstellen. Die Elektroden-Schnittstelle empfängt die fünf Elektroden-Signale von der Referenzelektrode PCB und sendet das Transmitter-Signal zu der Referenzelektrode PCB. Die fünf Elektroden-Eingänge sind über 10 $k\Omega$ Widerstände mit den R11, R12, R13, R14 und R15 Empfänger-Elektroden verbunden. Auf diese Weise werden hohe Frequenzen unterdrückt. Der Signalgenerator der MGC3130-Einheit ist mit dem Tx-Signal zur Transmitter-Elektrode verbunden. Über die I^2C Schnittstelle läuft die Verbindung und Kommunikation mit der I^2C to USB Bridge und dem Controller (Inc., 2016a, 24).

Referenzelektrode PCB

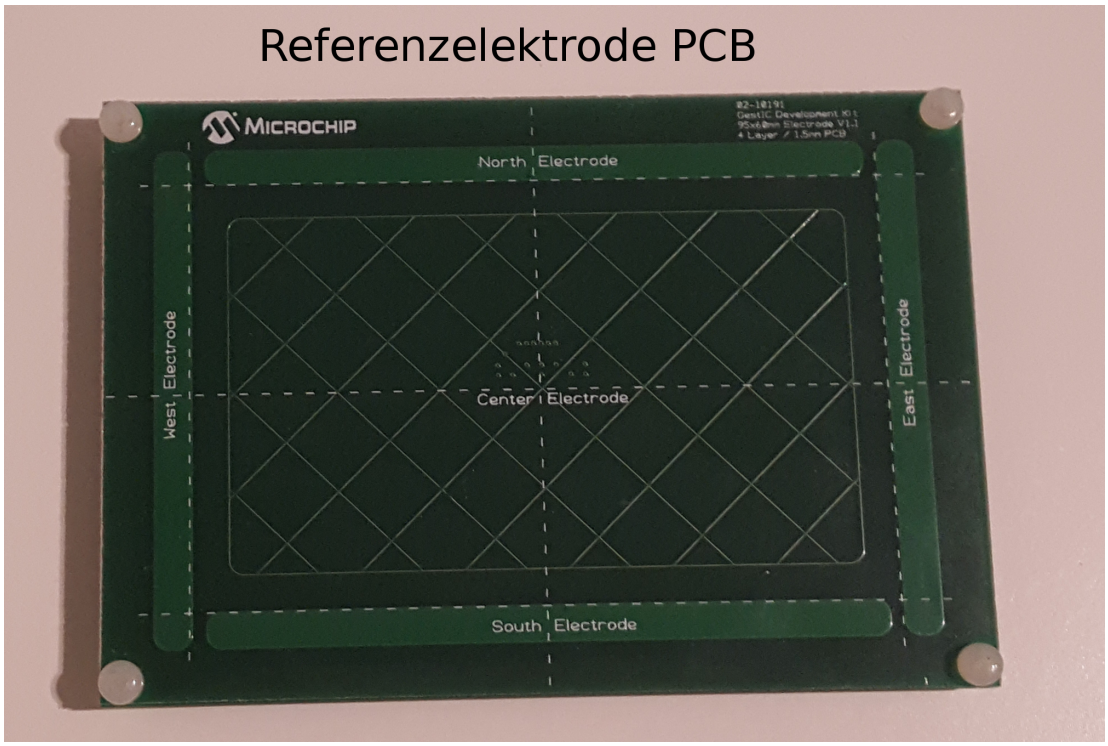

Abbildung 4.5: Referenzelektrode PCB

In Abbildung 4.5 ist die Referenzelektrode abgebildet, welche fünf Rx-Elektroden und eine Tx-Elektrode enthält und ist 95x60 mm groß ist. Die fünf Elektroden sind auf der Referenzelektrode in Nord, Süd, West, Ost und Zentrum aufgeteilt, wie in Abbildung 4.5 erkennbar.

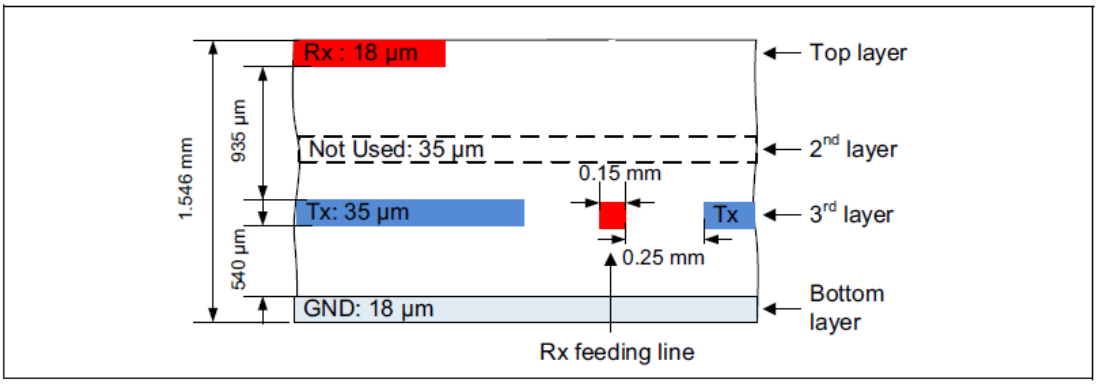

Abbildung 4.6: Ebenen der Referenzelektrode PCB (Inc., 2016a, 27)

In Abbildung 4.6 sind die vier Ebenen der Referenzelektrode dargestellt, aus welchen sie besteht. Auf der obersten Ebene befinden sich die fünf Elektroden, während die zweite Ebene leer ist. Auf der dritten Ebene ist die Tx-Elektrode lokalisiert, welche den Bereich des gesamten 3D-Touchsensors spannt. Auf der untersten Eben befindet sich die Erdung. Sie schützt die Elektroden vor externen Störungen und Einflüssen. Die Referenzelektrode besitzt eine Schnittstelle, mithilfe welcher die empfangenen Elektroden-Signale an die MGC3130-Komponenten gesandt werden. Die Schnittstellenverbindungen der Referenzelektrode zur MGC3130-Komponenten sind GND, Rx4, Rx3, Rx2, Rx1 und Rx0 (Inc., 2016a, 27).

I²C to USB Bridge

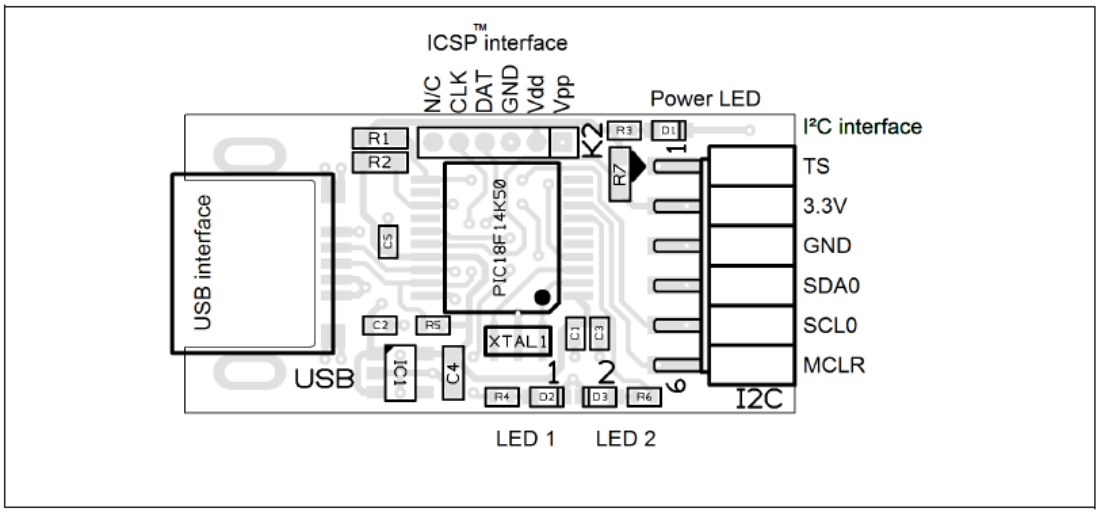

Abbildung 4.7: I²C to USB Bridge-Komponente (Inc., 2016a, 28)

In Abbildung 4.7 ist die I²C to USB Bridge Komponente des 3D-Touchsensors abgebildet. Es steuert die Datenübertragung und Kommunikation, über eine mini USB-Verbindungen zwischen einem Endgerät und der MGC3130-Komponente.

Die I²C to USB Bridge enthält zwei Schnittstellen. Die eine Schnittstelle führt über eine mini USB-Verbindung zu einem Endgerät(PC), während die andere Schnittelle zum MGC3130-Komponenten führt. Sämtliche Komponenten werden über die USB-Verbindung mit Strom versorgt. Die I²C to USB Bridge-Komponente weist sechs Pins auf, mit deren Hilfe der Datenverkehr zur MGC3130-Komponente geregelt wird bzw. stattfindet(Inc., 2016a, 28).

4.2 Gesten

Im folgenden Abschnitt werden die Handgesten vorgestellt, welche zu klassifizieren sind. Diese sind in drei Datensätze aufgeteilt, wobei jeder Datensatz Eingabemuster für vier Handgesten enthält, welche klassifiziert werden.

Die Datensätze werden als Trainings-, Validations- und Testdaten für das neuronale Netz verwendet. Die Eingabemuster wurden automatischen generiert. Als Lernverfahren wird das überwachte Lernen eingesetzt, infolgedessen jeder Datensatz die Eingabemuster und die zugehörige Ausgabewerte der Handgesten umfasst. Für jeden Datensatz ist ein Szenario mit verschiedenen Handgesten erstellt, welches jeweils den Ablauf der Handgesten visualisiert. Die Handgesten sind in statische und dynamische Handgesten unterteilt.

4.2.1 Szenario 1

Datensatz 1 enthält Eingabemuster und die zugehörigen Ausgabewerte für vier statische Handgesten, welche Gesten präsentieren, die über eine bestimmte Zeitspanne konstant bleiben. Für jeden Zeitpunkt wird eine vollständig separate Klassifizierung durchgeführt.

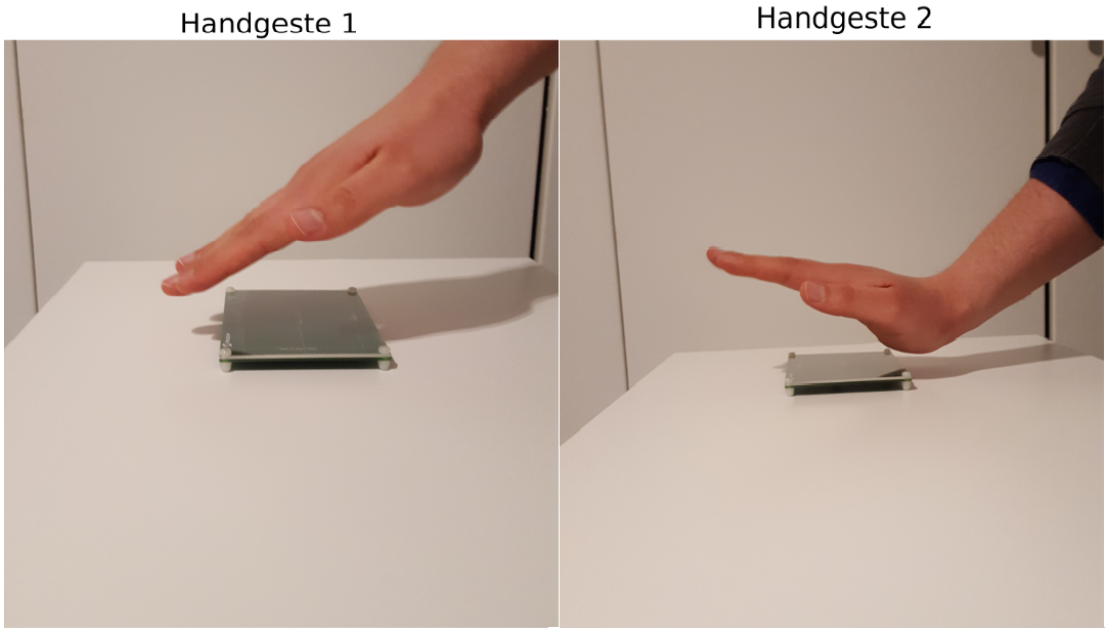

Abbildung 4.8: Statische Handgeste 1 und 2

In Abbildung 4.8 ist die erste statische Handgeste aus Szenario 1 dargestellt. Bei dieser Handgeste ist die Hand über dem 3D-Touchsensor an den Fingerspitzen nach unten in die Richtung der Nordelektrode gebeugt.

In Abbildung 4.8 ist die zweite statische Handgeste zu sehen. Hier wird die untere Fläche der Hand nach hinten in die Richtung der Südelektrode gebeugt.

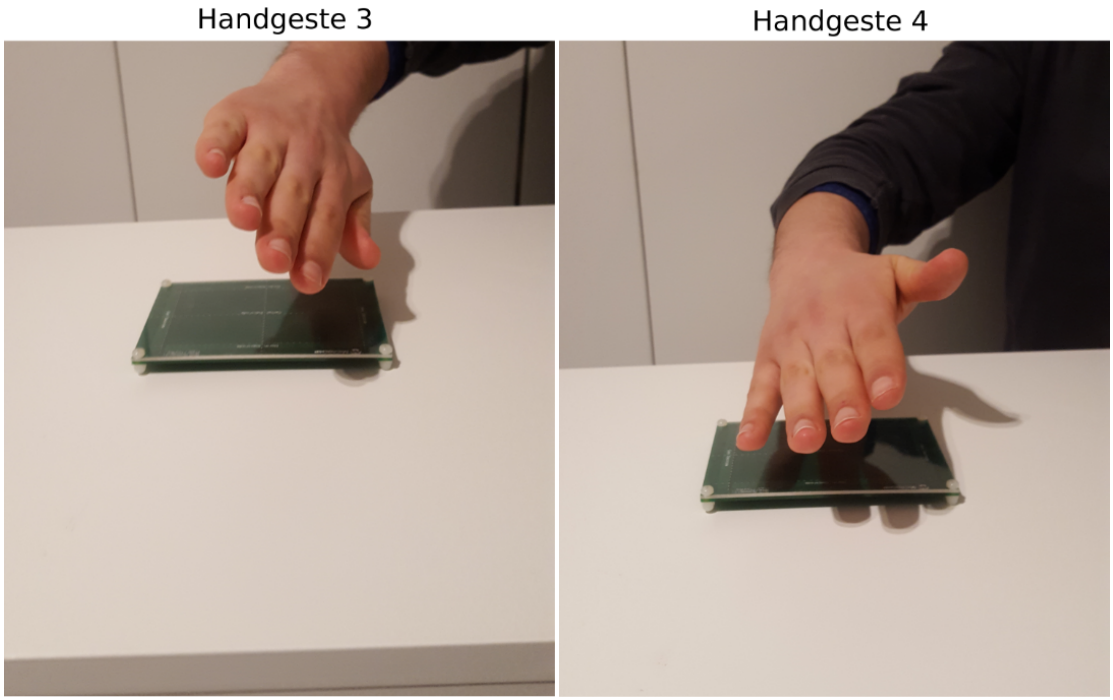

Abbildung 4.9: Statische Handgeste 3 und 4

Bei der dritten statischen Handgeste, siehe Abbildung 4.9, ist die Hand nach links in Richtung der Westelektrode geneigt.

Zuletzt ist die vierte statische Handgeste in Abbildung 4.9 abgebildet. Hierbei ist die Hand nach rechts in Richtung der Ostelektrode geneigt.

4.2.2 Szenario 2

In diesem Szenario werden einfache dynamische Handgesten getätigt. Diese werden als einfach bezeichnet, da die Bewegungen keine Komplexität aufweisen und die Handgesten während der Ausführung deutlich voneinander unterscheidbar sind.

Eine dynamische Handgeste setzt sich aus mehreren Eingabemustern zusammen. Folglich bestehen die Handgesten aus diesem Szenario aus mehreren Sequenzen von Eingabemustern.

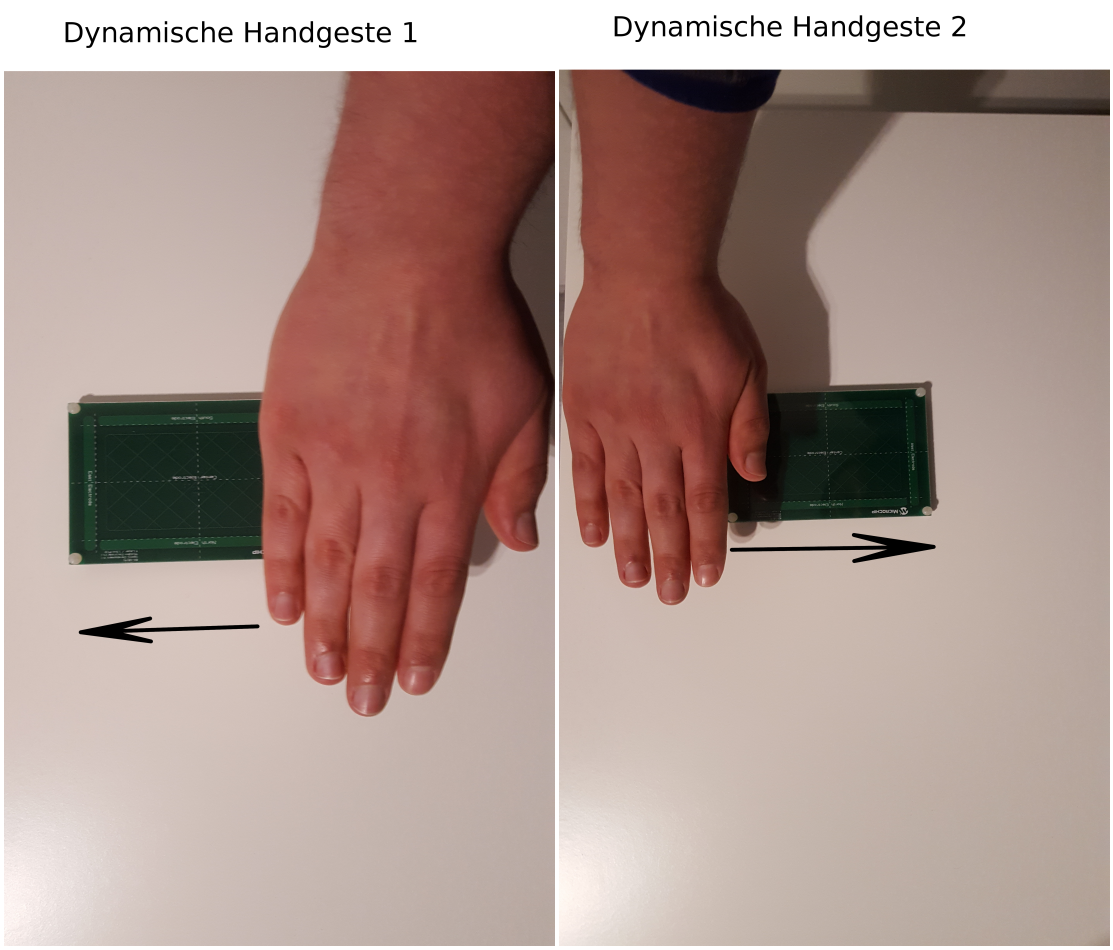

Abbildung 4.10: Einfache dynamische Handgeste 1 und 2

In Abbildung 4.10 ist die Ausführung der ersten dynamischen Handgeste zu sehen. Die Hand wird mit offener Handfläche, ausgehend von der Westelektrode zur Ostelektrode, bewegt. Die Bewegung erfolgt in einem Zug ohne Unterbrechung.

In Abbildung 4.10 ist die Ausführung der zweiten dynamischen Handgeste abgebildet. In diesem Fall wird die Hand, ebenfalls mit offener Handfläche, von der Ostelektrode zur Westelektrode bewegt.

Beide dynamischen Handgesten demonstrieren die gleichen Abläufe, unterscheiden sich jedoch hinsichtlich der Start- und Endposition.

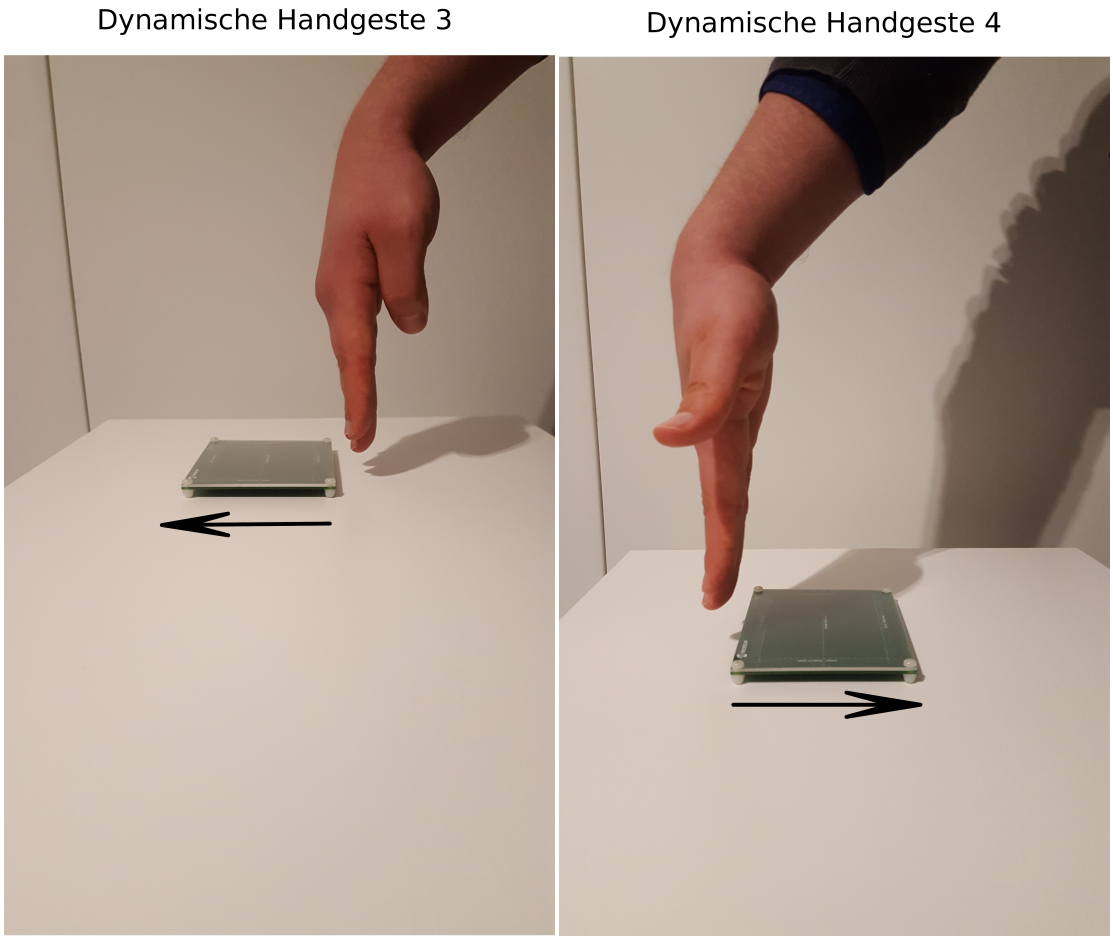

Abbildung 4.11: Einfache dynamische Handgeste 3 und 4

Im Rahmen der dritten dynamischen Handgeste in Abbildung 4.11 wird die Hand, mit den Fingerspitzen nach unten zeigend, von der Südelektrode zur Nordelektrode bewegt.

Hingegen ähnelt die vierte dynamische Handgeste in Abbildung 4.11 der Haltung der Hand der dritten dynamischen Handgeste. Diesbezüglich wird die Hand von der Nordelektrode zur Südelektrode geführt.

Beide dynamischen Handgesten werden ohne Unterbrechung ausgeführt.

4.2.3 Szenario 3

In Szenario drei werden dynamische Handgesten getätigt. Die Handgesten aus diesem Szenario divergieren vom zweiten Szenario im Hinblick auf Gesten, Dauer und Ausführung. Insoweit sind die Gesten, die Dauer sowie die Ausführungssequenz doppelt so lang wie die dynamischen Handgesten aus dem zweiten Szenario. Die dynamischen Handgesten aus diesem Szenario sind nicht leicht voneinander zu unterscheiden, da infolge einer schnellen Bewegung der Hand, Abtastzeitpunkte und die zugehörigen Positionen verpasst werden können. Dies führt das Risiko mit sich, dass Fehlklassifizierungen getätigt werden. Demzufolge sind sie komplexer und schwieriger zu erfassen.

Nachfolgend werden die Bewegungen der dynamischen Handgesten aus diesem Szenario anhand von Bildern veranschaulicht.

Abbildung 4.12: Komplexe dynamische Handgeste 1 und 2

In Abbildung 4.12 ist die Bewegung während der Ausführung der ersten dynamischen Handgeste aus diesem Szenario zu erkennen. Hierbei wird beginnend von der Westelektrode des 3D-Touchsensors die Hand mit offener Handfläche in die Richtung der Ostelektrode bewegt. Anschließend wird, ohne die Hand in einer Ruheposition zu lassen, diese zurück zur Westelektrode bewegt.

Die zweite dynamische Handgeste aus diesem Szenario ist in Abbildung 4.12 dargestellt. Der Ablauf der Handgeste gleicht der ersten dynamischen Handgeste aus Abbildung 4.12, mit der einzigen Ausnahme, dass sich die Start- und Endpositionen ändern. Die Hand wird von der Ostelektrode ausgehend zur Westelektrode und erneut zurück zur Ostelektrode bewegt.

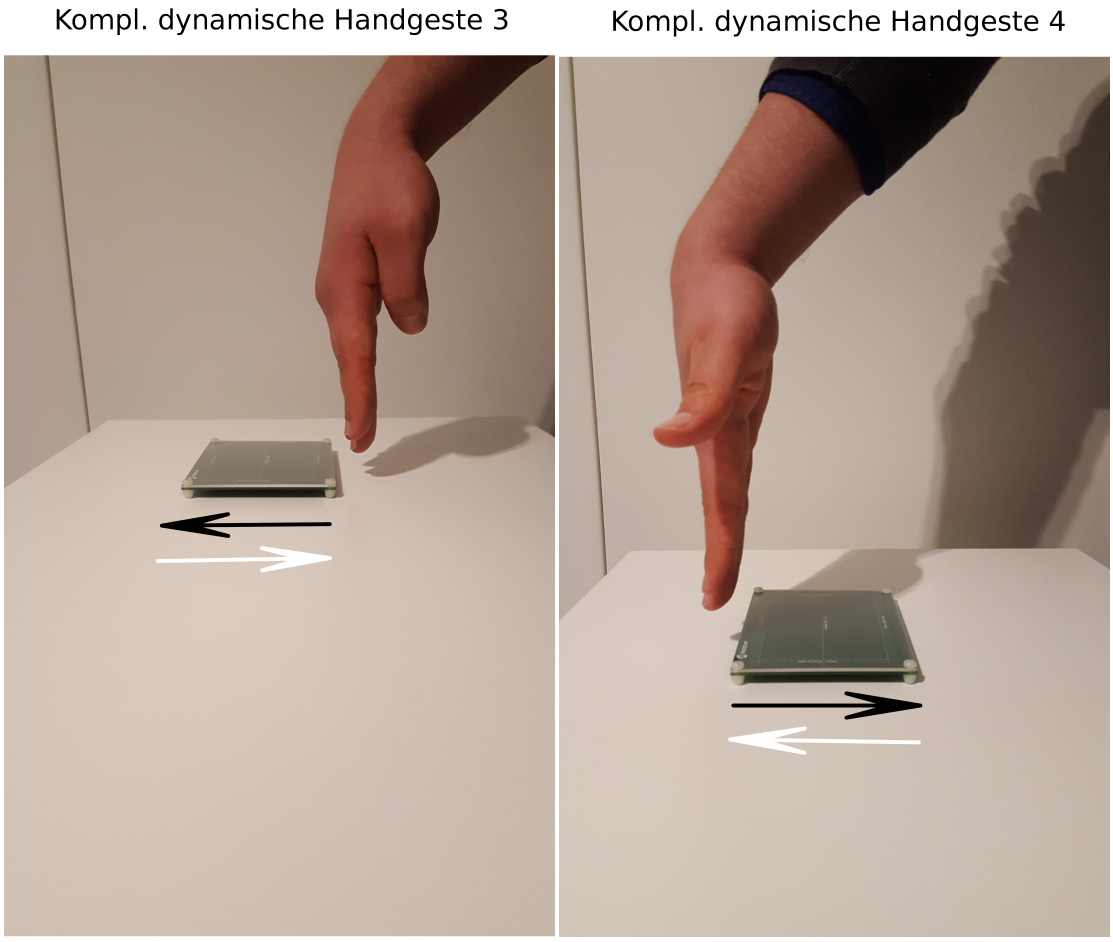

Abbildung 4.13: Komplexe dynamische Handgeste 3 und 4

Auf ähnliche Weise erfolgen die Abläufe der dritten und vierten dynamischen Handgesten aus diesem Szenario. Sie unterscheiden sich lediglich im Hinblick auf die Start- und Endpositionen der Hand.

In Abbildung 4.13 ist die Ausführung der dritten Handgeste dargestellt. Die Hand wird mit den Fingerspitzen auf den 3D-Touchsensor zeigend von der Südelektrode zur Nordelektrode bewegt und wieder zurück zur Südelektrode, ohne dass eine Bewegungspause erfolgt.

In Abbildung 4.13 ist die Bewegung der Hand bei der vierten Handgeste dieses Szenarios erkennbar. Die Hand wird wie in der dritten Handgeste gehalten. Dabei wird sie von der Nordelektrode zur Südelektrode und erneut zurück zur Nordelektrode bewegt.

4.3 Signalvorverarbeitung

In diesem Abschnitt der Analyse werden Verfahren vorgestellt, mithilfe welcher die Elektrodensignale aus dem 3D-Touchsensor verarbeitet werden. Die Eingabemuster der Datensätze setzen sich aus den fünf Elektrodensignalen(Nord-,Süd-,West-,Ost- und Zentrumelektrode) des 3D-Touchsensors zusammen. Fehlerhafte Daten in den Datensätzen beeinträchtigen die Klassifikationsleistung von maschinellen Lernverfahren. Aus diesem Grund müssen die Elektrodensignale in den Datensätzen der drei Szenarien vorverarbeitet werden. Die MGC3130-Komponente des 3D-Touchsensors bearbeitet die Elektrodensignale größtenteils bereits vor, aufgrund dessen die eigene Signalvorverarbeitung erleichtert und reduziert wird.

4.3.1 Normalisierung

Wie bereits in Unterabschnitt 2.1.3 erwähnt, ist die Normalisierung von Daten bei maschinellen Lernverfahren von erheblicher Relevanz. Die Wertebereiche der Elektrodensignale des 3D-Touchsensors können sehr stark voneinander variieren. Mithin müssen die Daten auf einen einheitlichen Wertebereich skaliert werden. Nachfolgend werden die Amplituden-, die Min-Max- und die z-score Normalisierung, vorgestellt, mittels dessen die Elektrodensignale skaliert werden.

Amplitudennormalisierung

Mit der Amplitudennormalisierung werden die Elektrodensignale auf einen Wertebereich skaliert, in welchem die Werte nahe beieinanderliegen.

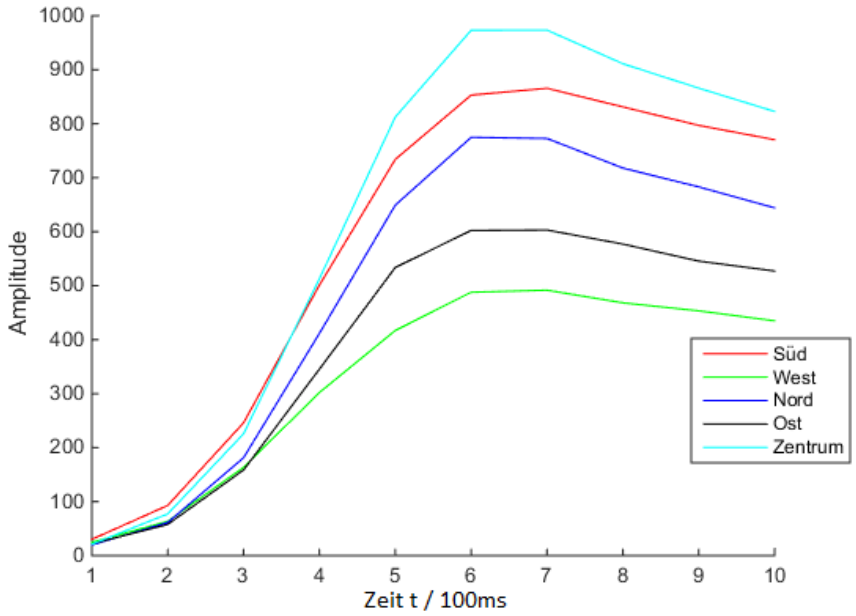

Abbildung 4.14: Unbearbeitete Elektrodensignale des 3D-Touchsensors

In Abbildung 4.14 sind die Signalverläufe dargestellt, in deren Rahmen die Wertebereiche der einzelnen Elektrodensignale erheblich voneinander variieren. Zunächst wird für jedes Elektrodensignal der Mittelwert(Scharmittelwert) an einem bestimmten Zeitpunkt berechnet.

$$\overline{y}(i) = \sum_{m=1}^{M} \frac{S_m(i)}{M} \tag{4.1}$$

Dabei ist entspricht

- M der Anzahl der Signale,
- $S_m(i)$ dem Signalwert zum Zeitpunkt i und
- $\overline{y}(i)$ dem Scharmittelwert der M-Signale an einem bestimmten Zeitpunkt i.

Anschließend werden mit der folgenden Formel die Elektrodensignale skaliert, sodass die Wertebereiche verkleinert werden.

$$S_{Norm}(i) = \frac{S_m(i)}{\overline{y}(i)} \tag{4.2}$$

Dabei ist

- $S_m(i)$ der Signalwert zum Zeitpunkt i und
- $S_{Norm}(i)$ das normierte Signal zum Zeitpunkt i.

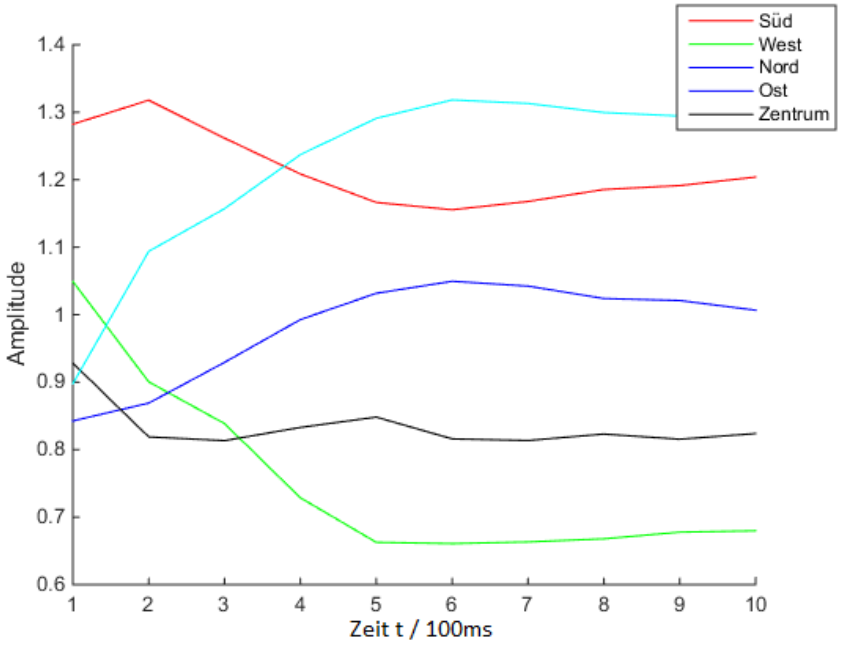

Abbildung 4.15: Skalierte Elektrodensignale des 3D-Touchsensors

In Abbildung 4.15 sind die skalierten Signalverläufe aus Abbildung 4.14 nach Anwendung der Amplitudennormalisierung abgebildet. Es ist eine signifikante Verkleinerung der Differenzen und des Wertebereich der Elektrodensignale erkennbar.

Min-Max-Normalisierung

Bei der Min-Max-Normalisierung werden die Daten auf einen bestimmten Wertebereich skaliert.

$$y = (y_{max} - y_{min}) \cdot \frac{x - x_{min}}{x_{max} - x_{minm}} + y_{min} \tag{4.3}$$

Dabei ist

- y der normalisierte Wert,
- y_{max} die Obergrenze nach der Skalierung,
- y_{min} die Untergrenze nach der Skalierung,
- x_{max} die Obergrenze der eigentlichen Daten,
- x_{min} die Untergrenze der eigentlichen Daten und
- x der aktuelle Wert.

Z-score-Normalisierung

Die z-score-Normalisierung verwendet die Standardabweichung und die Varianz des Datensatzes zum Zweck der Normalisierung eines Wertes.

$$Z = \frac{X - \mu}{\sigma} \tag{4.4}$$

Dabei ist

- Z der normalisierte Wert,
- X die Zahl, welche normalisiert wird,
- μ Varianz des Datensatzes und
- σ die Standardabweichung.

Fazit

Durch die Normalisierungsverfahren wird die Verarbeitungszeit von Algorithmen verbessert. Des Weiteren ist es für maschinelle Lernverfahren bedeutend, dass die Daten sich in einem gemeinsamen Wertebereich befinden. Dies vereinfacht die Klassifizierung bei maschinellen Lernverfahren.

Ein Nachteil der Min-Max-Normalisierung ist, dass Ausreißer die Normalisierung verfälschen und falsche Klassifizierungsergebnisse hervorrufen. Demgegenüber ist ein Nachteil der z-score-Normalisierung, dass bereits Daten existieren müssen (Runkler, 2015).

4.3.2 Reduktion von Instanzen

In Unterabschnitt 2.1.1 werden Verfahren zur Identifikation von Ausreißern genannt sowie der Einfluss auf die Ergebnisse erläutert. Darüber hinaus wird die Notwendigkeit der Datenreduzierung beschrieben. Nachstehend werden einige dieser Verfahren präsentiert.

Glätten mit dem Gleitenden Mittelwert (Smoothing with moving average)

Der gleitende Mittelwert (Constantinos) wirkt auf Signale wie ein Tiefpassfilter. Der Algorithmus wird auf die Rohdaten angewandt. Im Wege der Glättung werden Ausreißer und Rauschen in den Signalen gedämpft, infolgedessen das Ergebnis einem verbesserten Signal/Rausch- Verhältnis entspricht. Im Folgenden wird der gleitende Mittelwert ungerader Ordnung vorgestellt.

$$m = 2 \cdot n + 1 \tag{4.5}$$

$$y(k)_s = \sum_{i=0}^{n} \frac{y_{k+i}}{m} \tag{4.6}$$

Dabei ist

- m die ungerade Ordnung, eine ungerade Anzahl von aufeinanderfolgenden Datenpunkten,
- $y(k)_s$ der gemittelte Mittelwert,
- n aufeinanderfolgende Datenpunkte,
- y_{k+i} Wert des Signals zum Zeitpunkt k-i.

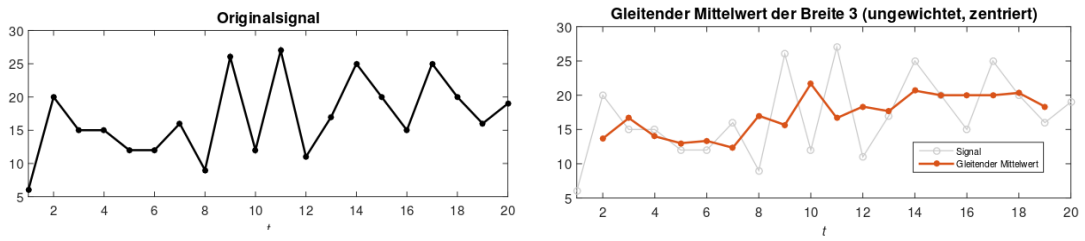

Abbildung 4.16: Signal mit Rauschanteilen links und Signal nach Anwendung des Gleitenden Mittelwertes rechts (Wikipedia, 2016a)

In Abbildung 4.16 ist im linken Bild ein Signal erkennbar, welches Störsignale enthält. Auf dem rechten Bild ist das Signal mit dem gleitenden Mittelwert-Verfahren geglättet worden, infolgedessen die Ausreißer am Signal gedämpft worden sind.

Zunächst wird ein Fenster, die Anzahl an Abtastwerten eines Signals bestimmt, über welchen der Mittelwert gebildet wird. Dies geschieht mittels der Gleichung 4.5. Im Anschluss daran wird der Mittelwert kraft der Gleichung 4.6 berechnet. Dieser Mittelwert wird an die $\frac{m+1}{2}$ Stelle eingetragen. Darauffolgend wird das Fenster um einen Abtastwert weiter verschoben und der Mittelwert aus diesem Fenster wird berechnet. Das Ergebnis wird daraufhin an die $1 + \frac{m+1}{2}$ des Signals eingetragen. Dies wird iterativ fortgesetzt, bis das Ende des Signals erreicht ist. Die Stärke des Filters kann durch eine größere Filterbreite oder durch wiederholte Anwendung des Algorithmus auf ein Signal verstärkt werden.

Fazit

Der Vorteil des Algorithmus ist, dass es sehr einfach ist und demzufolge der Aufwand gering ist. Dennoch besteht ein Nachteil darin, dass sämtliche Werte außerhalb des Fensters vernachlässigt werden. Als Folge dessen können Information verloren gehen.

Stratified-Sampling

Das Stratified-Sampling (Wikipedia, 2016e) ist ein Verfahren zur Datenreduktion. Bei diesem Verfahren wird aus einem großen Datensatz ein äquivalenter kleiner Datensatz gebildet.

Hierbei werden die Elemente aus dem originären Datensatz vorerst in gleiche Gruppen aufgeteilt. Hiernach werden zufällig und mit gleicher Wahrscheinlichkeit Elemente aus den Gruppen gewählt. Hierbei erfolgt die Auswahl der Elemente anhand eines Zufallsgenerators. Ausschlaggebend ist, dass jedes Element der Gruppen die gleiche Wahrscheinlichkeit besitzt, gezogen

zu werden. Falls es zu doppelten Ziehung eines Elements kommt, wird es ignoriert. Folglich erhält man einen kleineren Datensatz, welcher dem ursprünglichen Datensatz äquivalent ist.

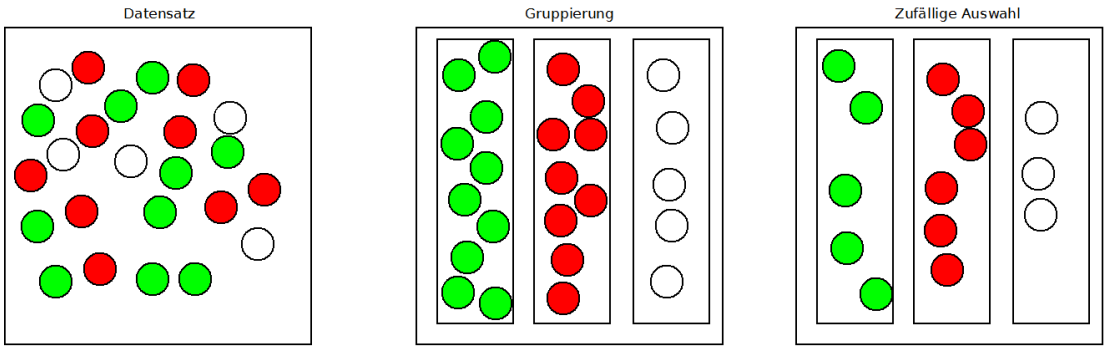

Abbildung 4.17: Stratified-Sampling

In Abbildung 4.17 ist links der große Datensatz abgebildet. Die Elemente des Datensatzes werden anfänglich in zugehörige Gruppen aufgeteilt, siehe mittleres Bild. Aus diese Gruppen werden zufällig sowie mit gleicher Wahrscheinlichkeit Elemente, mithin die Untermenge rechts in Abbildung 4.17, gebildet.

Fazit

Ein Vorteil des Stratified-Sampling ist, dass das Verfahren höchst simple umzusetzen ist. Ein weiterer Vorteil besteht darin, dass aufgrund der Gruppierung die Elemente der Gruppen mit gleichen Anteilen in dem neuen Datensatz enthalten sind. Hingegen ist ein Nachteil des Stratified-Sampling die Bestimmung der Anzahl der Gruppen, da es ist nicht immer einfach ist, die Gruppen in Datensätzen zu erkennen und zu bilden.

Segmentierung

In der Arbeit von Gustavo Monte (Gustavo, 2008) wird ein Signalverarbeitungsverfahren vorgestellt, in dessen Zusammenhang Signalverläufe determiniert und der weitere Verlauf geschätzt werden. Dies geschieht im Wege der Identifikation von relevanten Abtastwerten. Hierfür wird das Signal überabgetatstet. Ausschlaggebende Abtastwerte und die zugehörigen Abtastzeitpunkte werden identifiziert und gespeichert. Anschließend werden diese Abtastwerte acht vordefinierten Signalverläufen zugeordnet. Die gewonnen Informationen können bei der weiteren Analyse verwendet werden.

Nicht jeder Abtastwert eines Signales weist die gleiche Wichtigkeit und einen äquivalenten Informationsgehalt für ein Signal auf. Mittels der linearen Interpolation werden Abtastwerte, welche keine wichtigen Informationen enthalten, interpoliert.

Algorithm LIN1
initialization:
k = 2
n = 1
etotal = 0 ; accumulative error
mark(n) = x(n) ; tagged samples vector
timepos(n) = n ; time index vector
Parameters:
ε_{max} ;max accumulative error
ε_i ; max interpolation error
N ; max segment size
Procedure:
while *LOOP FOREVER* **do**
  error = abs ($\frac{x(n)+x(n+k)}{2} - x(\frac{n+k}{2})$) ;
  etotal = etotal + error ;
  if *error* < ε_i *AND etotal* < ε_{max} *AND k* < *N* **then**
    k = k + 2;
  else
    n = $n + \frac{k}{2}$;
    mark(n) = x(n) ;
    timepos(n) = n ;
    k = 2 ;
    etotal = 0 ;
  end
end

Algorithm 1: Segmentierungs-Algorithmus (Gustavo, 2008)

In algorithm 1 ist der Pseudocode des Algorithmus dargestellt. Zunächst wird zwischen zwei Abtastwerten interpoliert. Hieraus wird die Differenz zwischen dem eigentlichen Abtastwert und dem interpolierten Wert berechnet. Die Differenz entspricht dann dem Fehler, welcher solange akkumuliert wird, bis er einen bestimmten Schwellenwert erreicht hat. Bei der Erreichung dieses Schwellwertes werden der aktuelle Abtastwert sowie der zugehörige Abtastzeitpunkt

markiert und gespeichert. Der Algorithmus startet von dem letzten Punkt an, weiter zu interpolieren. Im Falle mangelnder Erzielung des Schwellenwertes wird zwischen dem nächsten Abtastpunkt interpoliert. Beendet ist der Algorithmus, wenn der letzte Abtastwert erreicht ist. Falls der Schwellenwert während der gesamten Interpolation nicht überschritten wird, wird der letzte Abtastwert markiert.

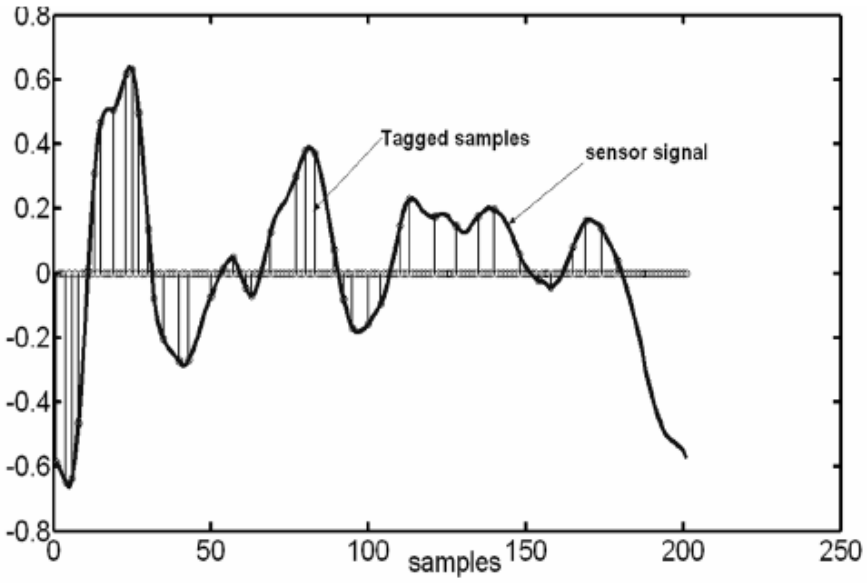

Abbildung 4.18: Signal mit markierten Abtastwerten nach Anwendung des Segmentierungs-Algorithmus

In Abbildung 4.18 sind ein Signalverlauf sowie die markierten Abtastwerte und Abtastpunkte nach Anwendung des Algorithmus dargestellt. Erkennbar ist, dass die markanten Punkte des Signals ordnungsgemäß erfasst werden.

Aufgrund der Segmentierung des Signals ist der Algorithmus für die Reduzierung von Abtastwerten eines Signals gut geeignet. Folglich können unbedeutende Abtastwerte eines Signals aus einem Datensatz entfernt werden, was gleichzeitig den Datensatz reduziert. Das Signal kann mithilfe der markierten Abtastwerte und Abtastzeitpunkte vollständig rekonstruiert werden. Des Weiteren wird in der Arbeit ein Verfahren vorgestellt, mittels dessen die Form des Signals bestimmt werden kann. Zu diesem Zweck werden die zuvor berechneten Abtastwerte und Abtastzeitpunkte verwendet.

Fazit

Ein Vorteil des Algorithmus ist, dass er auf sehr simple Weise zu implementieren ist. Des Weiteren werden keine komplexen Rechnungen getätigt, so dass der Segmentierungs-Algorithmus in Echtzeit in Mikrocontroller implementiert werden kann. Ein Nachteil ist, dass die Anwendung des Segmentierungs-Algorithmus im Falle rauschbehafteter Signale zu Fehlern oder Abweichungen führen kann.

4.4 Aufbau der Neuronalen Netz

In diesem Abschnitt der Analyse wird die Vorgehensweise beim Entwurf neuronaler Netze erläutert. Es werden Aspekte genannt, welche eine signifikante Rolle bei der Datensatzvorbereitung spielen. Weiterhin werden einige Verfahren dargeboten, kraft derer die Anzahl an Schichten und Neuronen festgelegt werden.

4.4.1 Datensatzaufbereitung

Die richtige Gestaltung der Datensätze wie die Trainings-,Test- und Validationsdaten üben einen großen Einfluss auf die Klassifikationsleistungen von neuronalen Netzen aus. In (Masters, 1993) werden einige Aspekte genannt, welche bei der Erstellung eines guten Datensatzes behilflich sind.

Die Datensätze neuronaler Netze müssen jede zu klassifizierende Klasse enthalten. Fehlen Eingabemuster einer Klasse in den Datensätzen, so werden die neuronalen Netze diese Muster nicht lernen. Zudem ist es belangvoll, dass die Eingabemuster dieser Klassen zahlreiche Variationen enthalten und vielfältig sind, damit eine bessere Generalisierung erzielt wird. Des Weiteren ist es bedeutend, dass die Datensätze verhältnismäßige Eingabemuster aufweisen, welche zufälliges Rauschen enthalten, damit die reale Anwendung simuliert werden kann. Falls die Datensätze über eine zu hohe Anzahl an rauschbehafteten Eingabemuster verfügen, werden die Klassifikationsleistungen der neuronalen Netze verringert.

Zudem ist darauf zu achten, dass die Trainingsdaten keine versteckten systematischen Fehler implizieren. Ferner können bei der Gestaltung der Datensätze Eingabemuster einer Klasse häufiger in Erscheinung treten, falls diese Klasse bei der Anwendung wiederholt auftaucht. Die Trainingsdaten haben eine großen Einfluss auf die Klassifikationsleistung der neuronalen Netze, infolgedessen es wichtig ist, diese gut zu selektieren und aufzubereiten.

Neuronale Netze besitzen in der Regel eine große Anzahl an Parametern. Je größer ein neuronales Netz ist, desto mehr Parameter wird es aufweisen. Aus diesem Grund wird empfohlen,

die Menge der Trainingsdaten zu erhöhen (Masters, 1993). Falls die Trainingsdaten zu klein gewählt werden, steigt die Wahrscheinlichkeit einer Überanpassung(Overfitting) beim Trainieren. Bei der Überanpassung lernt das neuronale Netz sämtliche Einzelheiten der Datensätze, ebenfalls Merkmale, welche bei der Klassifikation keine Rolle spielen. Folglich werden die Trainingsdaten sehr gut erkannt, obgleich das neuronale Netz nicht zu generalisieren fähig ist. Als Faustregel wird für den Quotienten aus Trainingsdaten zu Parameterzahl ein Faktor vom 2- bis 10-fachen empfohlen.

Bei unzureichenden Trainingsergebnissen wird vorgeschlagen, die Trainingsdaten mit den Testdaten zu mischen und das neuronale Netz erneut zu trainieren. Auf diese Weise wird versucht, hinreichend Variationen in den Trainingsdaten zu erhalten. Des Weiteren hat die Auswahl der Hyperparameter(Anzahl der verdeckten Schichten, Anzahl der Neuronen, Aktivierungsfunktion, Lernverfahren usw.) einen erheblichen Einfluss auf die Klassifikationsleistungen der neuronalen Netze. Mit den Trainingsdaten werden die Verbindungsgewichte der Neuronen eingestellt. Die Validationsdaten werden zum Zweck der Minimierung einer Überanpassung verwendet. Die Testdaten werden angewandt, um zu prüfen, inwieweit das neuronale Netz nach dem Training generalisieren und die Eingabemuster richtig klassifizieren kann.

4.4.2 Anzahl an verdeckten Schichten und Neuronen

Verdeckte Schichten

Eine der elementarsten Anforderungen im Rahmen des Entwurfs neuronaler Netze ist die Bestimmung der optimalen Anzahl an verdeckten Schichten und Neuronen in den verdeckten Schichten. Das Ziel ist es, die Menge an verdeckten Schichten und Neuronen in den verdeckten Schichten zu determinieren, in welcher die Trainingszeit verkürzt und der Fehler in der Ausgabe verringert wird. Zu diesem Thema existieren eine Zahl an wissenschaftlichen Artikeln und Büchern, siehe (Bengio, 2012; Masters, 1993; Alan J u. a., 2015).

Es hat sich herausgestellt, dass es kein Verfahren gibt, anhand dessen für jedes Problem die optimale Anzahl an Schichten und Neuronen bestimmt werden kann. Zudem beeinflussen zahlreiche Faktoren die Netzkonfiguration.

Für die Lösung von linear separierbaren Funktionen sind keine verdeckten Schichten vonnöten. Neuronale Netze sind in der Lage, einzig durch eine Eingabe- und Ausgabeschicht linear separierbare Funktionen richtig zu lösen (Masters, 1993). Ein neuronales Netz mit einer verdeckten Schicht und einer endlichen Anzahl an Neuronen, kann jede stetige Funktion approximieren (George, 1989). Sind die Probleme komplexerer Art, können zwei verdeckte Schichten verwen-

det werden (Karsoliya, 2012). Es gibt keine theoretischen Gründe, mehr als zwei verdeckte Schichten zu nutzen (Masters, 1993). Eine Vorgehensweise bei komplexeren Anforderungen ist es, das neuronale Netz um mehrere verdeckte Schichten zu erweitern, sodass die Neuronen auf die Schichten verteilt werden, wobei die erste verdeckte Schicht die meisten Neuronen aufweist. Je weiter die folgenden Schichten von der ersten verdeckten Schicht entfernt sind, desto weniger Neuronen sollten sie enthalten.

Neuronen

Eine der wesentlichen Herausforderungen beim Entwurf neuronaler Netze ist die Bestimmung der Anzahl an Neuronen in den verdeckten Schichten. Der Einsatz einer zu geringen Anzahl an Neuronen kann zu einer Unteranpassung(Underfitting) führen, infolgedessen das neuronale Netz nicht lernt. Unteranpassung bedeutet, dass das neuronale Netz zu wenig Neuronen besitzt, um die Informationen während des Trainings richtig zu verarbeiten. Demgegenüber kann eine zu hohe Menge an Neuronen in der verdeckten Schicht in einer Überanpassung(Overfitting) resultieren. Die Testdaten werden in diesem Fall fehlerhaft klassifiziert. Obendrein wird die Trainingszeit erhöht.

Nachfolgend werden einige bewertete Verfahren und sogenannte Regeln veranschaulicht, mittels welcher die Anzahl an Neuronen in den verdeckten Schichten bestimmt bzw. abgeschätzt werden.

Geometrische Pyramiden Regel

Bei der **geometrischen Pyramiden Regel** (Masters, 1993) wird die Anzahl an Neuronen in den verdeckten Schichten anhand der folgenden Formel bestimmt.

$$r = \sqrt[3]{\frac{n}{m}} \tag{4.7}$$

$$NHID_1 = m \cdot r^2 \tag{4.8}$$

$$NHID_2 = m \cdot r \tag{4.9}$$

Dabei ist

- n die Anzahl an Eingabeneuronen,
- m die Anzahl an Ausgabeneuronen,

- $NHID_1$ die Anzahl an Neuronen in der ersten verdeckten Schicht und
- $NHID_2$ die Anzahl an Neuronen in der zweiten verdeckten Schicht.

Demzufolge nimmt die Struktur eines neuronalen Netzes die Form einer Pyramide ein. Bei diesem Verfahren besitzen die Eingabeschichten die meisten Neuronen. Schichten, welche weiter von der Eingabeschicht entfernt sind, weisen weniger Neuronen auf.

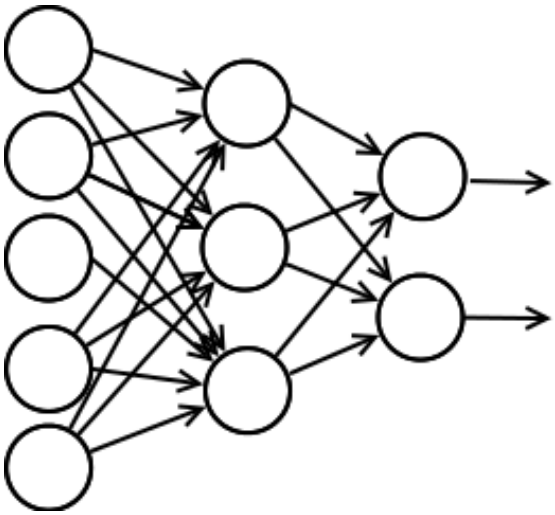

Abbildung 4.19: Geometrische Pyramide

Folglich ergibt dies für ein neuronales Netz mit fünf Eingabeneuronen und zwei Ausgabeneuronen eine verdeckte Schicht mit drei Neuronen, siehe Abbildung 4.19.

Fazit

Dieses Verfahren zur Bestimmung der Anzahl an Neuronen in verdeckten Schichten eignet sich für den Fall einfacher Anforderungen, welche zahlreiche Eingabe- und Ausgabeneuronen enthalten. Gleichzeitig ist ein Nachteile dieses Verfahrens, dass es eine grobe Annäherung darstellt und es im Falle komplexerer Anforderungen mit einer geringen Anzahl an Eingabe- und Ausgabeneuronen nicht verwendet werden kann.

Versuch und Irrtum Verfahren

Beim **Versuch und Irrtum Verfahren** (Alan J u. a., 2015) wird die optimale Anzahl an Neuronen in den verdeckten Schichten durch Annäherung bestimmt. Zunächst wird mit einer

sehr kleinen Anzahl an Neuronen, z.B. mit 2 Neuronen, in einer verdeckten Schicht begonnen. Dieses neuronale Netz wird trainiert und getestet. Die Leistung des neuronalen Netzes, mithin der Fehler der Ausgabe, wird bewertet. Falls der Fehler zu groß ist, wird ein weiteres Neuron hinzugefügt und das neuronale Netz erneut trainiert und getestet. Bei Bedarf kann ebenso eine zusätzliche verdeckte Schicht ergänzt werden. Dies wird solange wiederholt bis die gewünschte Klassifikationsleistung erreicht ist oder keine Verbesserungen bewerkstelligt werden können.

Fazit

Der Vorteil dieses Verfahrens ist, dass man bei richtiger Anwendung eine näherungsweise optimale Netzkonfiguration erhält. Hingegen ist das Verfahren insoweit nachteilig, dass es sehr zeitaufwendig ist. Aufgrund des iterativen Vorgangs wird das neuronale Netz jedes Mal angepasst, trainiert und getestet, was zugleich sehr viel Zeit beansprucht. Überdies kann bei Unaufmerksamkeiten eine Überanpassung eintreten.

Daumenregel

In (Alan J u. a., 2015; Karsoliya, 2012) werden weitere Verfahren und Regeln dargeboten, anhand welcher die optimale Anzahl an Neuronen in den verdeckten Schichten determiniert werden kann.

Einige **Regel**, die beim Entwurf neuronaler Netze getroffen werden, sind,

- dass die Anzahl an Neuronen in den verdeckten Schichten zwischen der Anzahl an Neuronen in der Eingabeschicht und Ausgabeschicht liegen (Blum, 1992),
- die Anzahl der Neuronen in den verdeckten Schichten nicht mehr als die doppelte Anzahl an Neuronen in der Eingabeschicht entspricht (Swingler, 1996; Berry und S.Linof, 2004),
- die Anzahl der Neuronen in den verdeckten Schichten zwei Drittel der Anzahl an Neuronen in der Eingabeschicht entsprechen. Falls dies nicht hinreichend ist, können die Neuronen in der verdeckten Schicht zusätzlich um die Anzahl von Ausgabeneuronen ergänzt werden (Karsoliya, 2012).

Fazit

Diese Regeln sind leicht zu beachten und führen im Falle einiger Anwendungen zu guten Ergebnissen. Weiterhin wird kein Aufwand zur Befolgung der jeweiligen Regel investiert. Ein nennenswerter Nachteil, welcher mit den oben genannten Regeln einhergeht, ist die

Abhängigkeit der Neuronen in den verdeckten Schichten von der Anzahl an Neuronen in den Eingabe- und Ausgabeschichten. Meistens hängt die Anzahl an Neuronen in den verdeckten Schichten zusätzlich von den Trainingsdatensätzen, Netzarchitekturen sowie Trainings- und Aktivierungsfunktionen ab.

Pruning-Algorithmen

Ein weiteres Verfahren ist der **Pruning-Algorithmus** (Alan J u. a., 2015). Bei diesem Verfahren wird ein übergroßes neuronales Netz iterativ gekürzt. Zu Beginn wird das neuronale Netz bewusst mit zu vielen Neuronen in den verdeckten Schichten entworfen, bevor dieses neuronale Netz trainiert wird. Im Anschluss an das Training werden die Verbindungsgewichte der Neuronen untersucht. Dabei werden die Neuronen entfernt, deren Verbindungsgewichte annähernd Null sind, da sie keinen Einfluss auf die Klassifikationsleistung haben. Somit werden iterativ Neuronen aus den verdeckten Schichten entfernt, welche keinen Einfluss auf die Klassifikationsleistung aufweisen. Dies wird solange wiederholt bis die optimale Netzkonfiguration erlangt wird.

Fazit

Der Vorteil von Pruning-Algorithmen ist, dass im Ergebnis ein neuronales Netz entsteht, welches eine optimale Klassifikationsleistung aufweist. Dennoch weisen Pruning-Algorithmen einige bedeutende Nachteile auf, u.a. die Feststellung des übergroßen neuronalen Netzes. Das neuronale Netz könnte theoretisch zu klein gewählt werden, das zur Entstehung mangelhafter Klassifikationsleistungen beiträgt. Daneben ist das Verfahren zur Auffindung von Neuronen, welche entfernt werden müssen, zeitaufwendig. Zusätzlich dauern Pruning-Algorithmen sehr lange, wie etwa im Falle einer der meist bekannten Pruning-Algorithmen, Optimal Brain Surgeon (OBS), welcher 14 Stunden erfordert, um ein neuronales Netz mit 50 Neuronen in der verdeckten Schicht auf 10 Neuronen zu kürzen (Alan J u. a., 2015).

5 Realisierung und Auswertung

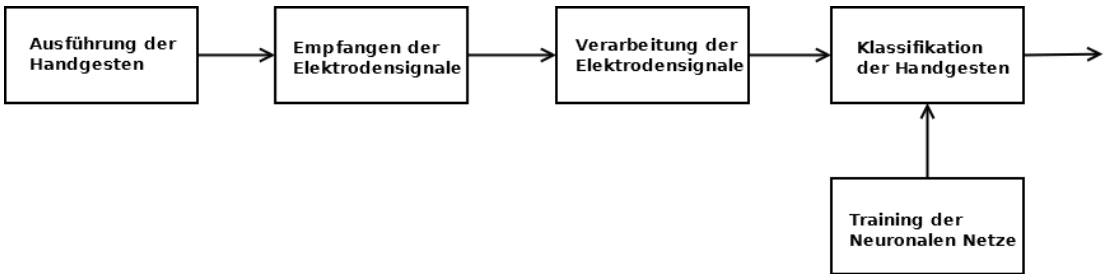

Abbildung 5.1: Handgesten-Identifikationssystem

In diesem Kapitel wird die Realisierung der Identifikation der Handgesten aus den drei Szenarien präsentiert. In Abbildung 5.1 ist das Handgesten-Identifikationssystem dargestellt. Zunächst werden die Handgesten über den 3D-Touchsensor ausgeführt. Die Elektrodensignale werden empfangen und verarbeitet. Schließlich werden die verarbeiteten Elektrodensignale als Eingaben in die jeweiligen neuronalen Netze geleitet. Die neuronalen Netze wurden zuvor anhand der Trainingsdaten trainiert.

In diesem Kapitel wird zunächst die Signalvorverarbeitung beschrieben, mittels welcher die Elektrodensignale des 3D-Touchsensors bearbeitet werden. Im Anschluss werden die Realisierungen der einzelnen Szenarien dargestellt. Dabei werden die neuronale Netze aufgebaut und die Architektur entwickelt. Für jedes Szenario werden mehrere neuronale Netze entwickelt, während in der Auswertung das neuronale Netz mit der höchsten Klassifikationsleistung bestimmt wird. Die Klassifikationsleistung wird anhand der mittleren quadratischen Abweichung bestimmt(MSE).

Die Szenarien 2 und 3 unterscheiden sich lediglich im Hinblick auf ihren Aufbau. Die Anzahl der verdeckten Schichten und Neuronen in den verdeckten Schichten variieren in den beiden Szenarien.

Die Auswertung wird anhand der Trainings-, Validations- und Testdatensätze getätigt. Zusätzlich werden 50 Handgesten ausgeführt, anhand welcher die Erkennungsraten ermittelt werden. Abschließend wird ein Fazit gezogen.

5.1 Signalvorverarbeitung

In diesem Abschnitt werden die angewendeten Verfahren zur Verarbeitung der Elektrodensignale des 3D- Touchsensors beschrieben. Die Signalverarbeitung ist in allen Szenarien identisch, da die Signale aus der gleichen Quelle, dem 3D-Touchsensor, stammen. Die Elektrodensignale des 3D-Touchsensors werden durch die SPI in der MGC3130-Einheit zum größten Teil vorverarbeitetet, so dass die Signale nicht rauschbehaftet sind. Dementsprechend kommen keine Verfahren zum Zuge, mittels welcher Rauschen an den Signalen gedämpft werden.

5.1.1 Normalisierung

Anhand der Signalverläufe des 3D-Touchsensors werden die Signalverarbeitungsschritte erläutert.

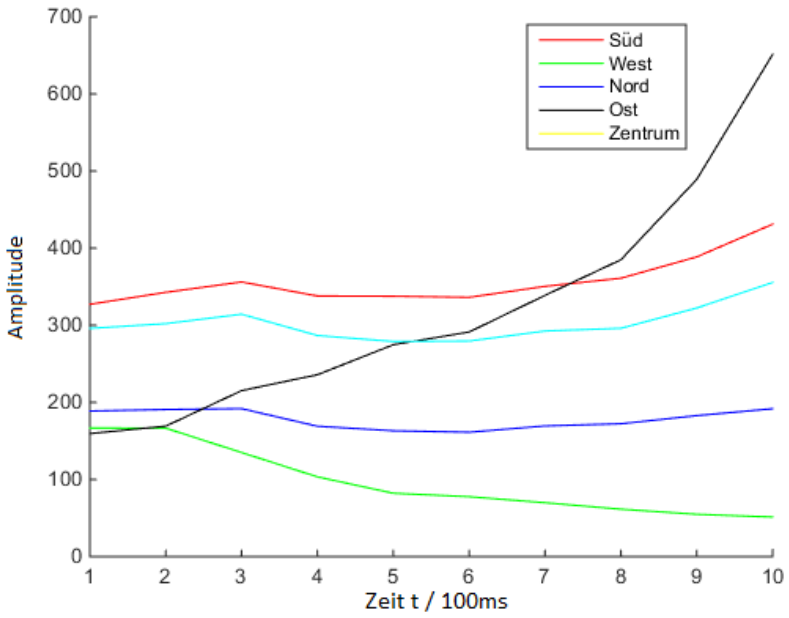

Abbildung 5.2: Unverarbeitete Elektrodensignale des 3D-Touchsensors

In Abbildung 5.2 sind die unverarbeiteten Elektrodensignale dargestellt. Eine Signalvorverarbeitung wird durchgeführt, da die Wertebereiche der Elektrodensignale sehr stark voneinander variieren. Zusätzlich erfordern der Backpropagation-Lernalgorithmus, die Tangens-hyperbolicus-

und Sigmoid-Aktivierungsfunktionen Eingabewerte im Bereich zwischen -1 und 1 bzw. zwischen 0 und 1. Demzufolge ist eine Skalierung der Elektrodensignale auf einen Wertebereich von -1 bis 1 bzw. 0 bis 1 notwendig.

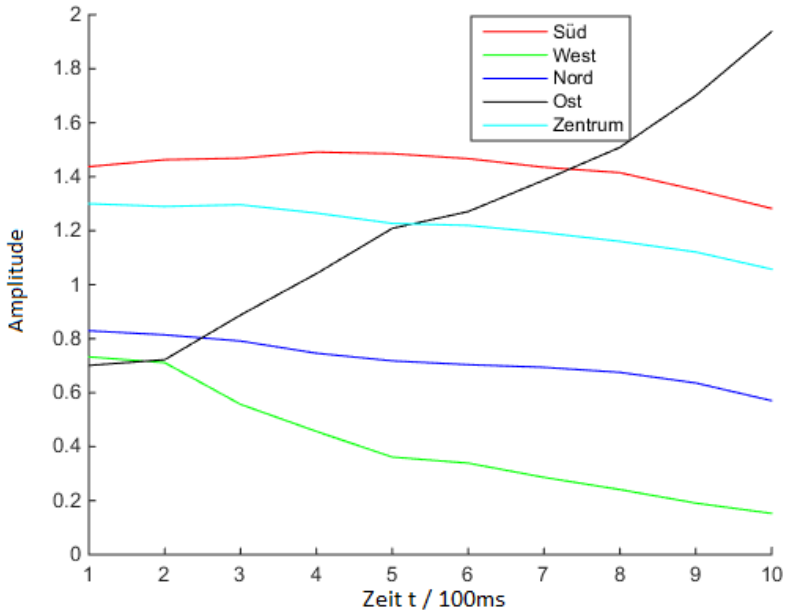

Abbildung 5.3: Elektrodensignale nach der Anwendung der Amplitudennormalisierung

In Abbildung 5.3 sind die Elektrodensignale aus Abbildung 5.2 mit der Amplitudennormalisierung aus Abschnitt 4.3.1 auf einen kleineren Wertebereich skaliert, infolgedessen die Differenzen zwischen den Elektrodensignalen kleiner sind.

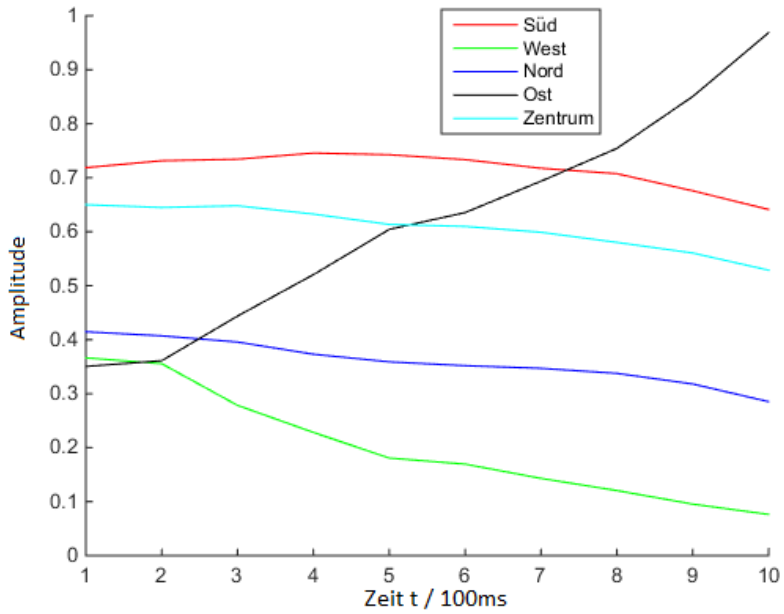

Abbildung 5.4: Elektrodensignale nach der Anwendung der Min-Max-Normalisierung

In Abbildung 5.4 sind die Elektrodensignale aus Abbildung 5.3 mit dem Min-Max-Normalisierungsverfahren aus Abschnitt 4.3.1 auf einen Wertebereich zwischen 0 und 1 skaliert. Mithin erhalten die Neuronen in der Eingabeschicht die normalisierten Elektrodensignale als Eingabewerte.

Die unmittelbare Anwendung der Min-Max-Normalisierung auf die Elektrodensignale ist nicht möglich, da die Obergrenzen der Elektrodensignale zu stark variieren und die tatsächliche Obergrenze selten erreicht wird.

5.1.2 Datenreduktion

Die Reduktion der Datensätze der drei Szenarien wird mit den gleichen Verfahren realisiert. Zur Datenreduktion wird das Statified-random-Sampling-Verfahren aus Abschnitt 4.3.2 eingesetzt, da die Handgesten eindeutig in Gruppen eingeteilt werden können.

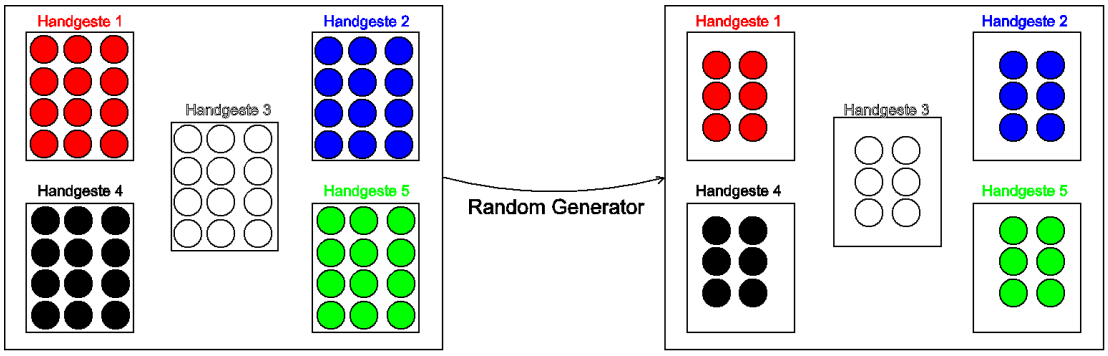

Abbildung 5.5: Datenreduktion durch das Stratified-Sampling-Verfahren

In Abbildung 5.5 ist die Datenreduktion der Datensätze mit dem Stratified-random-Sampling-Verfahren veranschaulicht dargestellt. Infolge der direkte Einordnung der Eingabemuster nach den Handgesten ist bei der Erstellung der äquivalenten kleineren Datensätze keine zusätzliche Einordnung vonnöten. Die ursprünglichen Datensätze werden durch die neuen äquivalenten Datensätze ersetzt.

Aufgrund der Skalierung der Elektrodensignale und der Reduktion der Datensätze sind neue äquivalente Datensätze entstanden. Mithilfe der Größe der Datensätze wird versucht, Überanpassung zu vermeiden. Ferner enthalten die Datensätze hinreichend Variationen in den Eingabemustern, welche mittels Generalisierung beeinflusst werden. Zusätzlich sind die Daten derart normalisiert, dass sie als Eingabewerte für neuronale Netze geeignet sind.

5.1.3 Datensatzaufbereitung

Die Datensätze der drei Szenarien sind unter der Einhaltung der Aspekte aus Unterabschnitt 4.4.1 erstellt. Die Qualität der Datensätze bildet einen bedeutenden Faktor für die Klassifikationsleistung der neuronalen Netze, sodass insbesondere Wert auf die Erstellung der Datensätze gelegt wird.

In den Datensätzen der Szenarien sind die verschiedenen Eingabemuster der Handgesten in gleicher Anzahl vertreten. Überdies wird bei der Erstellung darauf Acht gegeben, dass die Eingabemuster der Handgesten Variationen aufweisen, um die Generalisierung zu erhöhen.

Die SPI-Einheit der MGC3130-Komponente dämpft die Aufgetreten Raucheffekte, infolgedessen kein Rauschen in den Eingabemustern enthalten ist. Um die Realanwendung während des Trainingsprozesses der neuronalen Netze zu simulieren, enthalten die Datensätze eine geringe Anzahl an unvollständigen Sequenzen der Handgesten. Da zu viele Störungen die

Klassifikationsleistung beeinträchtigen, wird auf das Verhältnis von guten Handgesten und Handgesten mit Störungen geachtet.

Ein weiterer wichtiger Faktor zur Steigerung der Klassifikationsleistung und der Generalisierung der neuronalen Netze ist die Größe der Datensätze. Neuronale Netze mit zahlreichen verdeckten Schichten und Neuronen weisen eine große Anzahl an Hyperparameter auf, sodass die Datensätze so groß wie möglich gehalten werden, um einer Überanpassung vorzubeugen.

5.2 Szenario 1

In Unterabschnitt 4.2.1 werden die vier statischen Handgesten aus diesem Szenario vorgestellt.

Eingabemuster	Ausgabemuster	Trainingsdaten	Validationsdaten	Testdaten
400	400	240	80	80

Tabelle 5.1: Datensatz aus Szenario 1

In Tabelle 5.1 wird der Datensatz aus diesem Szenario dargestellt, welcher aus insgesamt 400 Eingabe- und Ausgabemustern besteht. Dabei repräsentieren je 100 Eingabemuster eine Handgeste. Zugleich ist der Datensatz in Trainings-,Validations- und Testdaten aufgeteilt.

Die Eingabemuster setzen sich aus einem 8x1-Vektor zusammen. Die Komponenten des 8x1-Vektors sind

- die Südelektrode,
- die Westelektrode,
- die Nordelektrode,
- die Ostelektrode,
- die Zentrumelektrode,
- die x-Koordinate,
- die y-Koordinate und
- die z-Koordinate.

Ein Ausgabemuster ist ein Ganzzahlwert, welcher die klassifizierte Klasse repräsentiert. Dabei werden die Handgesten wie folgt klassifiziert,

- eine 1 in der Ausgabe repräsentiert die Handgeste 1,
- eine 2 in der Ausgabe repräsentiert die Handgeste 2,
- eine 3 in der Ausgabe repräsentiert die Handgeste 3 und
- eine 4 in der Ausgabe repräsentiert die Handgeste 4.

Mittels der verwendeten Eingabemuster werden die Merkmale der Handgesten eindeutig repräsentiert.

5.2.1 Aufbau und Architektur der Feedforward-Neuronalen Netze

Die Netzkonfiguration der neuronalen Netze wird anhand der Regeln aus Abschnitt 4.4.2, des Versuch-und-Irrtum Verfahrens aus Abschnitt 4.4.2 und des Pyramiden-Algorithmus aus Abschnitt 4.4.2 bestimmt. Dabei wird unternommen, die optimale Netzkonfiguration zu determinieren, mit welcher die Klassifikationsleistung, die Differenz zwischen der Netzausgabe und der erwarteten Ausgabe, am höchsten ist. Die Neuronen in der Eingabe- und Ausgabeschicht sind feste Größen. Sie wurden anhand der Merkmale der Eingabemuster und der zu klassifizierenden Klassen bestimmt.

Als Netztyp wird für jede Netzkonfiguration ein **Feedforward-Netz** angewandt, da in diesem Szenario statische Handgesten klassifiziert werden und somit die neuronalen Netze nicht zustandsbehaftet sind. Das Resilient-Backpropagation-Algorithmus wird als Lernverfahren für die neuronalen Netze appliziert. Beim dem Resilient-Backpropagation handelt es sich um ein Lernverfahren, in dessen Rahmen die Gewichtsänderungen von den Vorzeichen des Gradienten abhängen (Zell, 1994).

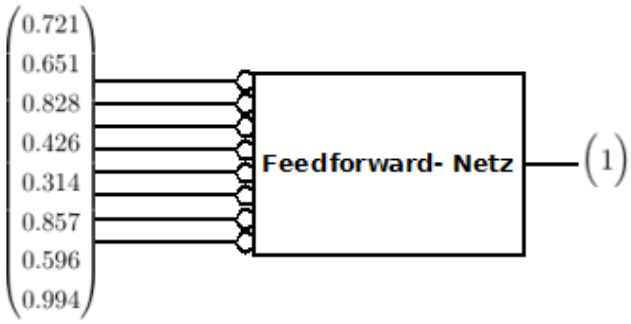

Abbildung 5.6: Eingabe- und Ausgabesequenz der Feedforward-Netze

In Abbildung 5.6 ist die Eingabe- und Ausgabesequenz der Feedforward-Netze illustriert. Eine statische Handgeste besteht aus einem Eingabe- und Ausgabemuster. Zu jedem Eingabemuster wird die Klassifizierung der Handgeste durchgeführt.

Netz	Anzahl verdeckte Schichten	Neuronen in der Eingabeschicht	Neuronen in den verdeckten Schichten	Neuronen in der Ausgabeschicht
Netz 1	1	8	3	1
Netz 2	1	8	6	1
Netz 3	1	8	4	1
Netz 4	1	8	14	1
Netz 5	1	8	5	1
Netz 6	2	8	[4 2]	1

Tabelle 5.2: Netzkonfiguration der Feedforward-Netze

In Tabelle 5.2 sind die resultierenden Netzkonfigurationen dargestellt.

5.2.2 Auswertung

In diesem Abschnitt werden die neuronalen Netze aus Tabelle 5.2 ausgewertet. Die Netzkonfigurationen resultieren aus der Anwendung der Verfahren aus Unterabschnitt 4.4.2. Gesucht wird die Netzkonfiguration, in welcher die Klassifikationsleistung der statischen Handgesten am höchsten ist.

Die Auswertung der neuronalen Netze erfolgt im Wege des Vergleichs der Klassifikationsleistungen. Des Weiteren werden 50 Handgesten getätigt und anhand dieser die Erkennungsraten ermittelt.

Netz	Erkennungsrate	Leistung (MSE)	Iterationen
Netz 1	100 %	0.171	101
Netz 2	100 %	0.00378	100
Netz 3	100 %	0.00634	81
Netz 4	100 %	0.000894	197
Netz 5	100 %	0.000447	197
Netz 6	100 %	8.18*e^-08	40

Tabelle 5.3: Klassifikationsleistungen der Feedforward-Netze

In Tabelle 5.3 sind die Resultate nach dem Trainieren der neuronalen Netze aus 5.2.1 verbildlicht. Ersichtlich ist, dass sämtliche neuronale Netze eine Erkennungsrate von 100% aufweisen und das die Klassifikationsleistungen sehr gut sind. Fernerhin ist zu erkennen, dass das Netz 6 die beste Klassifikationsleistung demonstriert. Dies beruht auf der Tatsache, dass mittels der Verwendung von mehrerer verdeckter Schichten komplexere Aufgaben gelöst werden. Infolge des Einsatzes mehrerer Schichten steigt die Trainingszeit durch die Rückpropagierung, bedingt durch die Anzahl an Neuronen und der zusätzlichen Schichten. Bei größeren neuronalen Netzen ist abzuwägen, ob eine verdeckte Schicht mit mehreren Neuronen zu bevorzugen ist, um die Länge der Trainingsdauer nicht zu sehr auszuweiten.

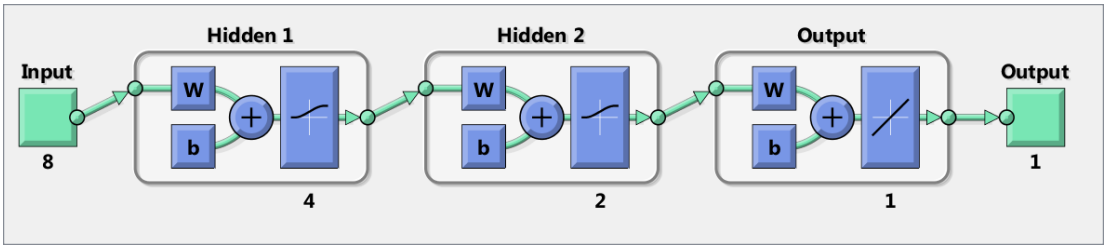

Abbildung 5.7: Netz 6 mit der optimalen Netzkonfiguration (Matlab Darstellung)

In Abbildung 5.7 ist das neuronale Netz 6 dargestellt, welches die optimalste Netzkonfiguration aufweist. Es besteht aus

- 8 Neuronen in der Eingabeschicht,
- 4 Neuronen in der ersten verdeckten Schicht,
- 2 Neuronen in der zweiten verdeckten Schicht,
- 1 Neuron in der Ausgabeschicht,
- die zwei verdeckten Schichten weisen Sigmoid-Aktivierungsfunktionen auf,
- die Ausgabeschicht hat eine lineare Aktivierungsfunktion und
- als Lernverfahren kommt der Resilient-Backpropagation-Algorithmus zum Zuge.

5.2.3 Fazit

Die Auswertung der neuronalen Netze zur Identifikation von Handgesten zeigt hohe Erkennungsraten auf. Dies ist aus der **??** zu entnehmen. Die eingesetzten neuronalen Netze haben eine hohe Klassifikationsleistung erreicht, da die mittleren quadratischen Abweichungen sehr niedrig sind. Hinzukommend wurde das neuronale Netz nach dem Training mit reellen Handgesten getestet, wobei sämtliche Handgesten zu 100% richtig erkannt wurden. Aus diesen Resultaten ist zu schließen, dass die Datensätze gut waren, zugleich die Handgesten gut repräsentierten. Des Weiteren sind die acht Merkmale der Eingabemuster richtig gewählt. Die Regeln zur Bestimmung der Anzahl an verdeckten Schichten und die Neuronen der verdeckten Schichten haben sich bei dieser Klassifikation als eine zuverlässige Herangehensweise bestätigt. Es hat sich gezeigt, dass eine verdeckte Schicht zur Erlangung einer hohen Klassifikationsleistung genügt. Der Grund für die hohe Klassifikationsleistung kann unter Umständen auf die Einfachheit der zu klassifizierenden Handgesten zurückzuführen sein. Infolge der genauen und simplen Unterscheidung der Handgesten haben die neuronalen Netze keine Probleme, diese zu klassifizieren.

5.3 Szenario 2

In diesem Abschnitt wird die Realisierung der vier dynamischen Handgesten aus Unterabschnitt 4.2.2 vorgestellt.

Eingabemuster	Ausgabemuster	Trainingdaten	Validationsdaten	Testdaten
2000	2000	1200	400	400

Tabelle 5.4: Datensatz Szenario 2

In Tabelle 5.4 ist der Datensatz aus diesem Szenario dargestellt. Der Datensatz enthält insgesamt 2000 Eingabe- und Ausgabemuster. Dabei repräsentieren je 500 Eingabemuster eine Handgeste. Der Datensatz ist zudem in Trainings-, Validations- und Testdaten aufgeteilt.

Die Eingabemuster setzen sich aus einem 5x1-Vektor zusammen. Die Komponenten des 5x1-Vektors bestehen aus den Werten,

- der Südelektrode,
- der Westelektrode,
- der Nordelektrode,
- der Ostelektrode und
- der Zentrumelektrode.

Ausgabemuster	Handgeste
[0 0 1]	Handgeste 1
[0 1 0]	Handgeste 2
[0 1 1]	Handgeste 3
[1 0 0]	Handgeste 4
[0 0 0]	Keine Handgeste

Tabelle 5.5: Ausgabemuster Szenario 2

In Tabelle 5.5 sind die Ausgabemuster aus diesem Szenario illustriert, welche sich aus einem 3x1-Vektor zusammensetzen. Die Ausgabemuster verkörpern die zur klassifizierenden Handgesten.

5.3.1 Aufbau und Architektur der Rekurrenten Neuronalen Netze

Die Anzahl der Neuronen und den verdeckten Schichten in den neuronalen Netzen, welche in diesem Szenario entwickelt werden, wird mit dem Versuch-und-Irrtum Verfahren aus Abschnitt 4.4.2 bestimmt. Mittels des Versuch-und-Irrtum-Verfahrens wird zugleich nach der optimalen Netzkonfiguration geforscht. Die Regeln aus Abschnitt 4.4.2 werden nicht angewandt, da sie zu einer kleinen Anzahl an Neuronen in den verdeckten Schichten führen. Insoweit resultiert eine kleine Anzahl an Neuronen in einem zustandsbehafteten neuronalen Netz in einer geringen Klassifikationsleistung und führt möglicherweise zur Unteranpassung.

Die Anzahl an Neuronen in den Eingabe- und Ausgabeschichten sind fest und entsprechen den Merkmalen sowie den zu klassifizierenden dynamischen Handgesten. Die neuronalen Netze müssen zustandsbehaftet sein, infolgedessen **Rekurrente Neuronale Netze** eingesetzt werden. Als Lernverfahren findet der Bayesian-Regularization-Algorithmus Anwendung. Der Bayesian-Regularization-Algorithmus stellt die Gewichte nach dem Levenberg-Marquardt-Algorithmus ein. Ferner minimiert dieser die Fehler in der Ausgabe, indem die Gewichte in der Weise gewählt werden, dass eine gute Generalisierung stattfindet (regularization backpropagation).

Abbildung 5.8: Eingabe- und Ausgabesequenzen der Rekurrenten Neuronalen Netze

In Abbildung 5.8 sind die Eingabe- und Ausgabesequenzen der Rekurrenten Neuronalen Netze wiedergegeben. Eine dynamische Handgeste setzt sich aus einer Sequenz von fünf Eingabemustern zusammen. Dabei werden die Eingabemuster nacheinander als Eingabewerte in die Eingabeschicht der neuronalen Netze geführt. Erst wenn die letzte Sequenz der Eingabemuster in das neuronale Netz geführt wird, wird die klassifizierte Klasse in der Ausgabe des neuronalen Netzes ausgegeben. Die Ausgabe des neuronalen Netzes gibt zu jedem Eingabemuster welches nicht die letzte ist, als Ausgabe keine Handgeste aus.

Netz	Verdeckte Schichten	Neuronen in der Eingabeschicht	Neuronen in den verdeckten Schichten	Neuronen in der Ausgabeschicht	Trainings Alg.
Netz 1	1	5	30	3	BR
Netz 2	2	5	[10 5]	3	BR
Netz 3	2	5	[15 7]	3	BR

Tabelle 5.6: Netzkonfiguration der Rekurrenten Neuronalen Netzen

In Tabelle 5.6 sind die Netzkonfigurationen, welche sich nach der Anwendung des Versuch-und-Irrtum Verfahrens aus Kapitel Abschnitt 4.4.2 ergeben, abgebildet. In dem folgenden Abschnitt werden die Klassifikationsleistungen der Rekurrenten Neuronalen Netze ausgewertet.

5.3.2 Auswertung

In Abschnitt 5.3.1 wurden als Folge des Einsatzes des Versuch-und-Irrtums-Verfahrens drei Rekurrente Neuronale Netze mit unterschiedlichen Netzkonfigurationen entwickelt. Das Ziel bestand darin, eine Netzkonfiguration aufzuspüren, anhand welcher die Klassifikationsleistung der dynamischen Handgesten am höchsten ist. Die Anzahl der Neuronen der Eingabe- und Ausgabeschicht sind dabei unveränderbare Größen. Mittels des Versuch-und-Irrtum-Verfahrens wurde die Anzahl der verdeckten Schichten und Neuronen in den verdeckten Schichten angepasst.

Für die Auswertung der Rekurrenten Neuronalen Netze werden die resultierenden Klassifikationsleistungen nach dem Training verglichen. Darüber hinaus werden 50 dynamische Handgesten aus Unterabschnitt 4.2.2 getätigt und die Erkennungsrate ermittelt.

Netz	Erkennungsrate	Leistung (mse)	Iterationen
Netz 1	100 %	0.000368	207
Netz 2	100 %	0.000675	92
Netz 3	100 %	0.000226	78

Tabelle 5.7: Klassifikationsleistungen der dynamischen neuronalen Netze

In Tabelle 5.5 sind die Erkennungsraten, Klassifikationsleistungen und Iterationen der entwickelten Rekurrenten Neuronalen Netze dargestellt. Die Rekurrenten Neuronalen Netze

demonstrieren sehr hohe Klassifikationsleistungen, da die Differenzen zwischen den tatsächlichen Ausgaben und den erwarteten Ausgaben sehr gering sind. Infolgedessen sind ebenfalls die Erkennungsraten der Rekurrenten Neuronalen Netze sehr hoch, so dass sämtliche getätigten dynamischen Handgesten erkannt werden.

Die Auswertung belegt, dass mit einer verdeckten Schicht und zahlreichen Neuronen in der verdeckten Schicht eine hohe Klassifikationsleistung erreicht wird, aufgrund dessen eine verdeckte Schicht genügt. Ferner ist zu erkennen, dass neuronale Netze mit zwei verdeckten Schichten und wenig Neuronen in den verdeckten Schichten weniger Iterationen benötigen und folglich der Trainingsprozess weniger Zeit beansprucht.

Die Auswertung belegt, dass die Klassifikationsleistung steigt, je mehr Neuronen in den verdeckten Schichten enthalten sind. Überdies wird ersichtlich, dass je größer die Anzahl der Neuronen in den verdeckten Schichten ist, desto mehr Iterationen beim Training erforderlich sind, was wiederum zu einer längeren Trainingsdauer führen kann. Die Auswertung weist ferner nach, dass das Netz 3 die höchste Klassifikationsleistung erreicht und damit die optimalste Netzkonfiguration aufweist.

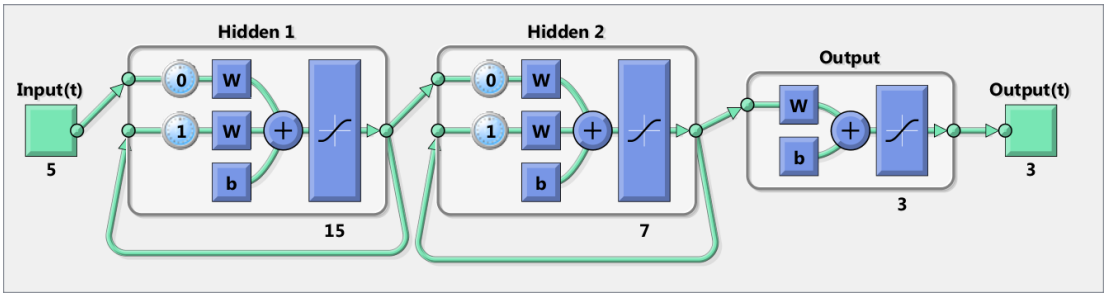

Abbildung 5.9: Netz 3 mit der optimalen Netzkonfiguration (Matlab Darstellung)

In Abbildung 5.9 ist das Rekurrente Neuronale Netz 3 dargestellt, welches die optimalste Netzkonfiguration enthält. Es besteht aus

- 5 Neuronen in der Eingabeschicht,
- 15 Neuronen in der ersten verdeckten Schicht,
- 7 Neuronen in der zweiten verdeckten Schicht,
- 3 Neuronen in der Ausgabeschicht,

- alle drei Schichten weisen eine Tangens-hyperbolicus-Aktivierungsfunktion auf und
- als Lernverfahren ist der Bayesian-regularization-Algorithmus angewandt worden.

5.3.3 Fazit

Im Verlauf der Entwicklung der Rekurrenten Neuronalen Netze zur Klassifikation von dynamischen Handgesten aus Szenario 2 hat sich die Bestimmung der optimalen Netzkonfiguration als schwierig erwiesen. Die Anwendung der Regeln aus Abschnitt 4.4.2 zur Determinierung der optimalen Netzkonfiguration führt zur Unteranpassung der Rekurrenten Neuronalen Netze, da aufgrund einer zu geringen Zahl an Neuronen in der verdeckten Schicht die Informationsverarbeitung nicht hinreicht. Mithilfe des Versuch-und-Irrtum-Verfahrens aus Abschnitt 4.4.2 wurden Netzkonfigurationen ermittelt, welche hohe Klassifikationsleistungen aufweisen. Dabei wurde iterativ die Anzahl der verdeckten Schichten und Neuronen soweit erhöht bis eine akzeptable Menge gefunden wurde, bei der die Klassifikationsleistung hoch ist. Rekurrente Neuronale Netze, welche eine höhere Anzahl an Neuronen in den verdeckten Schichten aufweisen, zeigen eine höhere Klassifikationsleistung. Zugleich ist die Trainingsdauer von Rekurrenten Neuronalen Netzen mit vielen Neuronen länger. Insgesamt wurden im Wege der Anwendung des Versuch-und-Irrtum Verfahrens Netzkonfigurationen aufgespürt, anhand welcher die Klassifikation der dynamischen Handgesten ein hohes Niveau aufweist und eine sehr gute Erkennungsrate erzielt wird.

5.4 Szenario 3

Die dynamischen Handgesten dieses Szenarios wurden in Unterabschnitt 4.2.3 präsentiert. Im Folgenden wird die Realisierung der Identifikation der dynamischen Handgesten aus diesem Szenario erläutert.

Eingabemuster	Ausgabemuster	Trainingdaten	Validationsdaten	Testdaten
2000	2000	1200	400	400

Tabelle 5.8: Datensatz Szenario 3

In Tabelle 5.8 ist der Datensatz aus diesem Szenario zu erkennen, welcher insgesamt 2000 Eingabe- und Ausgabemuster beinhaltet. Je 500 Eingabemuster repräsentieren dabei eine Handgeste. Der Datensatz ist in Trainings-, Validations- und Testdaten aufgeteilt.

Die Eingabemuster setzen sich aus einem 5x1-Vektor zusammen, wobei die Komponenten des 5x1-Vektors aus den nachfolgenden Werten bestehen,

- der Südelektrode,
- der Westelektrode,
- der Nordelektrode,
- der Ostelektrode und
- der Zentrumelektrode.

Ausgabemuster	Handgeste
[0 0 1]	Handgeste 1
[0 1 0]	Handgeste 2
[0 1 1]	Handgeste 3
[1 0 0]	Handgeste 4
[0 0 0]	Keine Handgeste

Tabelle 5.9: Ausgabemuster aus Szenario 3

In Tabelle 5.9 sind die Ausgabemuster und die zugehörigen Handgesten dargestellt. Die Ausgabemuster setzen sich aus einem 3x1-Vektor zusammen.

5.4.1 Aufbau und Architektur der Rekurrenten Neuronalen Netze

Die Anzahl an Neuronen und verdeckten Schichten wird mit dem Versuch-und-Irrtum-Verfahren aus Unterabschnitt 4.4.2 bestimmt. Die Regeln aus Unterabschnitt 4.4.2, zur Bestimmung der Neuronen in den verdeckten Schichten, werden nicht angewendet, da sie zu einer kleinen Anzahl an Neuronen führen und somit die Informationen nicht entsprechend verarbeiten. Eine kleine Anzahl an Neuronen in den verdeckten Schichten kann, im Falle der Komplexität der dynamischen Handgesten, zur Unteranpassung führen. Die Anzahl an Neuronen in der Eingabe- und Ausgabeschicht sind durch die Merkmale der Eingabemuster und der zu klassifizierenden Handgesten vorgegeben.

Rekurrente Neuronale Netze werden als Netztyp verwendet. Überdies findet der Bayesian-regularization-Algorithmus als Lernverfahren Anwendung.

Abbildung 5.10: Eingabe- und Ausgabesequenzen der Rekurrenten Neuronalen Netzen aus Szenario 3

In Abbildung 5.10 sind die Eingabe- und Ausgabesequenzen der Rekurrenten Neuronalen Netzen abgebildet. Die dynamischen Handgesten setzen sich aus einer Sequenz von fünf Eingabemustern zusammen, wobei pro Eingabemuster ein Ausgabemuster ausgegeben wird. Erst wenn das fünfte Eingabemuster als Netzeingabe eingegeben worden ist, wird die klassifizierte Handgeste in der Ausgabe des neuronalen Netzes ausgegeben.

Netz	Verdeckte Schichten	Neuronen in der Eingabeschicht	Neuronen in den verdeckten Schichten	Neuronen in der Ausgabeschicht	Trainings Alg.
Netz 1	1	5	18	3	BR
Netz 2	1	5	30	3	BR
Netz 3	2	5	[15 5]	3	BR

Tabelle 5.10: Netzkonfigurationen nach Anwendung des Versuch-und-Irrtum-Verfahrens

In Tabelle 5.10 sind die Netzkonfigurationen dargestellt, die sich nach dem Versuch-und-Irrtum-Verfahren ergeben haben.

5.4.2 Auswertung

Mit dem Versuch-und-Irrtum-Verfahren aus Abschnitt 4.4.2 wurden in Unterabschnitt 5.4.1 drei Rekurrente Neuronale Netz mit hohen Klassifikationsleistungen entwickelt. Die Rekurrenten Neuronalen Netze haben voneinander verschiedene Netzkonfigurationen.

Das Ziel war es, ein Rekurrentes Neuronalen Netz mit optimaler Netzkonfiguration zu finden, bei welcher die Klassifikationsleistung im Rahmen der Klassifizierung der dynamischen Handgesten aus Unterabschnitt 4.2.3 am höchsten ist.

Die Neuronen in der Eingabe- und Ausgabeschicht sind dabei unveränderlich. Sie repräsentieren die Merkmale der Handgesten bzw. die zu klassifizierenden Handgesten. Die Anzahl der

verdeckten Schichten und der Neuronen in den verdeckten Schichten wurde mit dem Versuch-und-Irrtum-Verfahren ermittelt.

Bei der Auswertung der Rekurrenten Neuronalen Netze werden die Klassifikationsleistungen nach dem Training verglichen. Des Weiteren werden 50 Handgesten getätigt und die Erkennungsrate ermittelt.

Netz	Erkennungsrate	Leistung(mse)	Iterationen
Netz 1	100 %	0.000757	489
Netz 2	100 %	0.000979	69
Netz 3	100 %	0.000538	162

Tabelle 5.11: Klassifikationsleistung der Rekurrenten Neuronalen Netze aus Szenario 3

In Tabelle 5.11 sind die drei Rekurrenten Neuronalen Netzen mit den Erkennungsraten, Klassifikationsleistungen und den Iterationen dargestellt. Die Erkennungsrate beträgt bei allen drei Netzen 100 %. Die Klassifikationsleistungen der neuronalen Netze sind sehr hoch, da die mittleren quadratischen Abweichungen sehr niedrig sind. Während der Entwicklung der Rekurrenten Neuronalen Netze mit dem Versuch-und-Irrtum-Verfahren wurden bewusst Netzkonfigurationen ausgewählt, welche sehr gute Klassifikationsleistungen aufweisen.

Die Klassifikationsleistung von Netz 2 ist im Vergleich zu den anderen beiden Netzen geringer. Des Weiteren besitzt Netz 2 eine verdeckte Schicht, welche im Vergleich zu den anderen beiden Rekurrenten Neuronalen Netzen die meisten Neuronen besitzt. Im Vergleich dazu erreicht Netz 1 die Zweitbeste Klassifikationsleistung, obwohl es die geringste Anzahl an Neuronen in der verdeckten Schicht aufweist. Demgegenüber erzielt Netz 3 mit zwei verdeckten Schichten die beste Klassifikationsleistung.

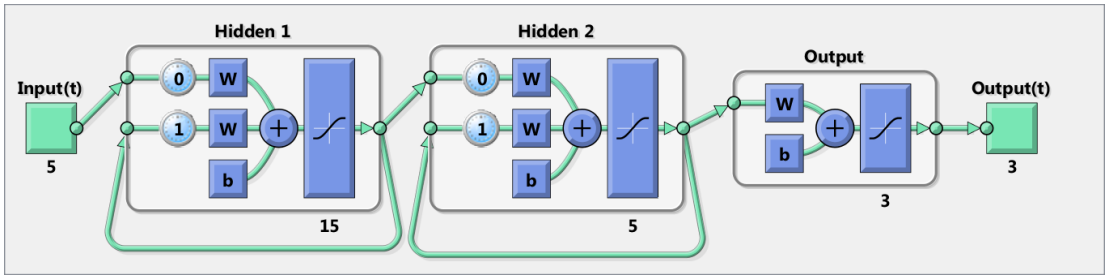

Abbildung 5.11: Netz 3 mit der optimalsten Netzkonfiguration (Matlab Darstellung)

In Abbildung 5.11 ist das Rekurrente Neuronale Netz 3 mit der besten Klassifikationsleistung und der optimalsten Netzkonfiguration dargestellt. Es besteht aus

- 5 Neuronen in der Eingabeschicht,
- 15 Neuronen in der ersten verdeckten Schicht,
- 5 Neuronen in der zweiten verdeckten Schicht,
- 3 Neuronen in der Ausgabeschicht,
- alle Schichten weisen Tangens-hyperbolicus-Aktivierungsfunktionen auf und
- als Lernverfahren wird der Bayesian-regularization-Algorithmus angewandt.

5.4.3 Fazit

In diesem Abschnitt der Realisierung wurden drei Rekurrente Neuronale Netze zur Klassifikation von dynamischen Handgesten aus Unterabschnitt 4.2.3 entwickelt. Der Zweck bestand darin, eine optimale Netzkonfiguration zu ermitteln, mithilfe welcher die höchste Klassifikationsleistung erzielt werden konnte. Dabei wurden die Anzahl der verdeckten Schichten, die Neuronen in den verdeckten Schichten und das Lernverfahren mittels des Versuch-und-Irrtum-Verfahrens ermittelt.

Die Suche nach der optimalsten Netzkonfiguration hat sich als schwierig und zeitaufwendig herausgestellt. Die optimale Netzkonfiguration hängt von einigen Faktoren, wie die Auswahl der Hyperparameter und die Aufbereitung des Datensatzes, ab.

Aus den Auswertungen in Tabelle 5.11 ist zu entnehmen, dass das Netz 3, mit zwei verdeckten Schichten, die optimalste Netzkonfiguration sowie die höchste Klassifikationsleistung aufweist. Wie aus der Auswertung zu entnehmen, ist es möglich, mit nur einer verdeckten Schicht das Problem der Klassifikation der dynamischen Handgesten zu lösen. Dabei demonstriert Netz 1 mit wenig Neuronen eine bessere Klassifikationsleistung als Netz 2 mit mehr Neuronen. Je mehr verdeckte Schichten und Neuronen verwendet werden, desto länger dauert der Trainingsprozess. Folglich ist Netz 1 zu bevorzugen.

6 Ausblick und Fazit

Im Rahmen dieser Arbeit wurden drei Szenarien erstellt, in deren Rahmen Handgesten anhand eines 3D- Touchsensors klassifiziert worden sind. Dabei enthält Szenario 1 statische Handgesten, Szenario 2 hingegen einfache dynamische Handgesten sowie Szenario 3 komplexe dynamische Handgesten.

Als Klassifikatoren wurden Feedforward Neuronale Netze und Rekurrente Neuronale Netze eingesetzt. Die Zielsetzung war, neuronale Netze mit optimalen Netzkonfigurationen und damit einhergehenden hohen Klassifikationsleistungen zu entwickeln.

Zunächst wurden Datenvorverarbeitungsverfahren beschrieben, mithilfe welcher Daten und Signale vorverarbeitet werden. Anschließend wurden die Grundlagen von neuronalen Netzen erläutert. Dabei wurde insbesondere Wert auf Feedforward Neuronale Netze sowie Rekurrente Neuronale Netze gelegt.

Eine weitere elementare Problemstellung dieser Arbeit ist die Netzkonfiguration der neuronalen Netze. Insoweit wurden im Analyseteil dieser Arbeit Verfahren und bewährte Regeln aufgelistet, anhand welcher eine optimale Netzkonfiguration erstellt worden ist. Zu betonen ist, dass die Auswahl der richtigen Hyperparameter und die Trainingsdaten einen großen Einfluss auf die Klassifikationsleistung der neuronalen Netze ausüben.

Für sämtliche Szenarien wurden mehrere neuronale Netze mit verschiedenen Netzkonfigurationen entwickelt und anhand dieser die Klassifikationsleistung bewertet.

In Szenario 1 wurden die Regeln aus Abschnitt 4.4.2 genutzt, um die Netzkonfigurationen der Feedforward- Netze zubestimmen. Es hat sich herausgestellt, dass unter Verwendung der Regeln aus Abschnitt 4.4.2, Feedforward-Netze erstellt wurden, welche hohe Klassifikationsleistungen aufweisen. Demzufolge kann angenommen werden, dass die Regeln für Feedforward-Netze gut geeignet sind.

In Szenario 2 und 3 wurden Rekurrente Neuronale Netze zur Klassifikation der dynamischen Handgesten eingesetzt, da die dynamischen Handgesten aus Sequenzen bestehen und die neuronalen Netze somit zustandsbehaftet sein müssen. Für die Determinierung der optimalen

Netzkonfigurationen wurde das Versuch-und-Irrtum-Verfahren aus Abschnitt 4.4.2 angewendet. Dabei wurde mit dem Bayesian-regularization-Algorithmus ein Trainingsalgorithmus ermittelt, anhand dessen die Klassifikationsleistungen der Rekurrenten Neuronalen Netze sehr hoch sind. Eine mögliche Annahme ist, dass die Datensätze beider Szenarien zu klein gewählt wurden, sodass alternative Lernverfahren zu nicht guten Klassifikationsleistungen geführt haben.

Es hat sich erwiesen, dass sich neuronale Netze zur Identifikation von Handgesten gut eignen und mittels dieser hohe Klassifikationsleistungen erlangt werden können. Fernerhin hat sich gezeigt, dass eine verdeckte Schicht genügt, um gute Klassifikationsleistungen zu erzielen.

Das Trainieren und die Zusammenstellung der richtigen Netzkonfigurationen hat sich als schwer und zeitraubend offenbart. Oftmals ist es nicht möglich, auf Anhieb die optimale Netzkonfiguration zu finden.

Eine Alternative zu den verwendeten Rekurrenten Neuronalen Netzen sind die Long-short-term-memory-Netze. Mit ihnen ist es möglich, eine weitaus größere Anzahl an Sequenzen zu verarbeiten. Darüber hinaus werden Probleme wie der Vanishing-gradient-Problem gelöst, zu deren Entstehung es in Rekurrenten Neuronalen Netzen kommen kann. Die Long-short-term-memory-Netze sind mächtig und können demnach komplexere Aufgaben lösen, infolgedessen neuronale Netze mit mehreren Schichten entwickelt werden können. Ein Ausblick könnte die Weiterentwicklung durch Einsatz von Long-Short-term-memory-Netzen darstellen.

Literaturverzeichnis

[Adam u. a. 2016] Adam, Gbison ; Chris, Nicholson ; Josh, Patterson: *A Beginner's Guide to Recurrent Networks and LSTMs*. 2016. – URL erhältlichaufhttps://deeplearning4j.org/lstm. – [Online; Stand 15. November 2016]

[Alan J u. a. 2015] Alan J, Thomas ; Militos, Petridis ; Simon, D W. ; Saeed Malekshahi, Gheytassi ; Robert, E M.: On Predicting the Optimal Number of Hidden Nodes. (2015)

[regularization backpropagation] backpropagation, Bayesian regularization: *Bayesian regularization backpropagation*. – URL https://de.mathworks.com/help/nnet/ref/trainbr.html. – Eingesehen am 13.11.2016

[Bengio 2012] Bengio, Yoshua: Practical Recommendations for Gradient-Based Training of Deep Architectures. (2012)

[Berry und S.Linof 2004] Berry, Michael J. ; S.Linof, Gordon: *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management Second Edition*. Indianapolis, Indiana , USA : Wiley Publishing, Inc, 2004. – ISBN 0-471-47064-

[Blum 1992] Blum, Adam: *Neural Networks in C++: An Object-oriented Framework for Building Connectionist Systems*. New York, NY, USA : John Wiley & Sons, Inc., 1992. – ISBN 0-471-53847-7

[Constantinos] Constantinos, Efstathiou: *Signal Smoothing Algorithmus*. – URL http://195.134.76.37/applets/AppletSmooth/Appl_Smooth2.html. – Eingesehen am 11.11.2016

[García u. a. 2014] García, S. ; Luengo, J. ; Herrera, F.: *Data Preprocessing in Data Mining*. Springer International Publishing, 2014 (Intelligent Systems Reference Library). – URL https://books.google.de/books?id=SbFkBAAAQBAJ. – ISBN 9783319102474

[George 1989] George, Cybenko: Approximation by Superpositions of Sigmoidal Funtion. In: *Mathematics of Control, Signals, and Systems* 2 (1989), S. 303–314

[Guenter und Karl F. 2010] Guenter, Daniel R. ; Karl F., Wender: *Neuronale Netze Eine EinfÃ¼hrung in die Grundlagen, Anwendung und Datenauswertung*. Hogrefe, vorm. Verlag Hans Huber, 2010. – URL http://neuronalesnetz.de/einleitung.html. – Eingesehen am 14.11.2016. – ISBN 9783456848815

[Gustavo 2008] Gustavo, Monte: Sensor Signal Preprocessing Techniques for Analysis and Prediction. (2008). ISBN 978-1-4244-1766-7

[Hochreiter und Schmidhuber 1997] Hochreiter, S. ; Schmidhuber, J.: Long Short-Term Memory. In: *Neural Computation* 9 (1997), Nov, Nr. 8, S. 1735–1780. – ISSN 0899-7667

[Inc. 2016a] Inc., 2012-2015 Microchip T.: *MGC3030 Hillstar Development Kit Users Guide*. November 2016. – Eingesehen am 11.11.2016

[Inc. 2016b] Inc., 2012-2015 Microchip T.: *MGC3030/3130 3D Tracking and Gesture Controller Data Sheet*. http://ww1.microchip.com/downloads/en/DeviceDoc/40001667D.pdf. November 2016. – Eingesehen am 11.11.2016

[J. Arturo u. a. 2010] J. Arturo, Olvera-Lopez ; J. Ariel, Carrasco-Ochoa ; J. Francisco, Martinez-Trinidad ; Josef, Kittler: A review of instance selection methods. In: *Springer Science + Buisness Media B.V.* (2010), May, S. 133–143

[Karsoliya 2012] Karsoliya, Sauranh: Approximating Number of Hidden layer neurons in Multiple Hidden LayerBPNN Architecture. (2012)

[Kotsiantis u. a. 2006] Kotsiantis, S. B. ; Kanellopoulos, D. ; Pintelas, P. E.: Data Preprocessing for Supervised Leaning. In: *International journal of computer science* 1 (2006), Nr. 2, S. 111–117. – ISSN 1306-4428

[Kriesel 2007] Kriesel, David: *Ein kleiner Überblick über Neuronale Netze*. URL erhältlichaufhttp://www.dkriesel.com, 2007

[Masters 1993] Masters, Timothy: *Practical Neural Network Recipes in C++*. Academic Press, Inc., 1993

[Matlab 2016] Matlab, Neuron M.: *Multilayer Neural Network Architecture, Matlab Neural Network*. 2016. – URL https://de.mathworks.com/help/nnet/ug/multilayer-neural-network-architecture.html. – [Online; Stand 14. November 2016]

[Olah 2015] Olah, Christopher: *Understanding LSTM Networks*. 2015. – URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/. – [Online; Stand 15. November 2016]

[Runkler 2015] Runkler, T.A.: *Data Mining: Modelle und Algorithmen intelligenter Datenanalyse*. Springer Fachmedien Wiesbaden, 2015 (Computational Intelligence). – URL https://books.google.de/books?id=FLkANAEACAAJ. – ISBN 9783834816948

[Swingler 1996] Swingler, K: *Applying Neural Networks, A Practical Guide*. London, UK : Academic Press., 1996. – ISBN 9780126791709

[Verma u. a. 2013] Verma, N. K. ; Agrawal, A. K. ; Sevakula, R. K. ; Prakash, D. ; Salour, A.: Improved signal preprocessing techniques for machine fault diagnosis. In: *2013 IEEE 8th International Conference on Industrial and Information Systems*, Dec 2013, S. 403–408. – ISSN 2164-7011

[Wikipedia 2016a] Wikipedia: *Gleitender Mittelwert — Wikipedia, Die freie EnzyklopÃ¤die*. 2016. – URL https://de.wikipedia.org/w/index.php?title=Gleitender_Mittelwert&oldid=158546577. – [Online; Stand 16. November 2016]

[Wikipedia 2016b] Wikipedia: *KÃ¼nstliches Neuron — Wikipedia, Die freie EnzyklopÃ¤die*. 2016. – URL https://de.wikipedia.org/w/index.php?title=K%C3%BCnstliches_Neuron&oldid=153614177. – [Online; Stand 14. November 2016]

[Wikipedia 2016c] Wikipedia: *Long short-term memory — Wikipedia, The Free Encyclopedia*. 2016. – URL https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=749138439. – [Online; accessed 12-November-2016]

[Wikipedia 2016d] Wikipedia: *Nervenzelle — Wikipedia, Die freie EnzyklopÃ¤die*. 2016. – URL https://de.wikipedia.org/w/index.php?title=Nervenzelle&oldid=159239905. – [Online; Stand 15. November 2016]

[Wikipedia 2016e] Wikipedia: *Stratified sampling — Wikipedia, The Free Encyclopedia*. 2016. – URL https://en.wikipedia.org/w/index.php?title=Stratified_sampling&oldid=747437594. – [Online; accessed 2-November-2016]

[Zell 1994] Zell, Andreas: *Simulation neuronaler Netze*. MÃ¼nchen, M, Deutschland : R. Oldenbourg Verlag MÃ¼nchen Wien, 1994. – ISBN 3-486-24350-0

[Zhang u. a. 2015] Zhang, H. ; Lin, H. ; Li, Y.: Impacts of Feature Normalization on Optical and SAR Data Fusion for Land Use/Land Cover Classification. In: *IEEE Geoscience and Remote Sensing Letters* 12 (2015), May, Nr. 5, S. 1061–1065. – ISSN 1545-598X

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 22. November 2016 Orhan Aykac