Schriftenreihe des
Instituts für Mikrowellentechnik

15

Marco Altmann

Lernende Verfahren zur Gesten- und Kochklassifikation mittels Radarsensorik

Institut für Mikrowellentechnik
Universität Ulm

Lernende Verfahren zur Gesten- und Kochklassifikation mittels Radarsensorik

universität ulm

Lernende Verfahren zur Gesten- und Kochklassifikation mittels Radarsensorik

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS (Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der Universität Ulm

von

Marco Altmann

aus Heilbronn

Gutachter:	Prof. Dr.-Ing. Christian Waldschmidt
	Prof. Dr.-Ing. Thomas Walter
	Prof. Dr.-Ing. Peter Ott
Amtierende Dekanin:	Prof. Dr. phil. habil. Anke Huckauf

Ulm, 13. Oktober 2022

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Dissertation, Universität Ulm,
Fakultät für Ingenieurwissenschaften, Informatik und Psychologie, 2022

Impressum
Universität Ulm
Institut für Mikrowellentechnik
Prof. Dr.-Ing. Christian Waldschmidt
Albert-Einstein-Allee 41
89081 Ulm
https://www.uni-ulm.de/mwt

Eine Übersicht über alle Bände der Schriftenreihe finden Sie unter
https://www.uni-ulm.de/mwtschriften

ISBN 978-3-948303-32-7 ISBN 978-3-948303-33-4 (E-Book)

Vorwort

Diese Dissertation ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter im Forschungsprojekt MikroSens entstanden. Die beteiligten Hochschulen im ZAFH-Projekt sind die Hochschulen Ulm, Pforzheim und Heilbronn. Ein weiterer Partner ist die Universität Ulm und im Speziellen das Institut für Mikrowellentechnik. Meine Tätigkeit im Projekt fand an der Hochschule Heilbronn über die komplette Projektlaufzeit vom 01.01.2016 bis 31.03.2021 statt. Das Projekt MikroSens wurde zu gleichen Teilen vom Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg sowie dem Europäischen Fonds für regionale Entwicklung gefördert.

Mein Dank gilt allen Professoren und Kollegen des Projekts MikroSens, die zum Erfolg dieser Arbeit beigetragen haben. Ein ganz besonderer Dank gilt Prof. Dr.-Ing. Christian Waldschmidt, der diese Dissertation an der Universität Ulm betreut. Ebenso gilt ein ganz besonderer Dank meinem Betreuer Prof. Dr.-Ing. Peter Ott an der Hochschule Heilbronn. Die Anregungen, wertvollen Diskussionen und tatkräftige Unterstützung aller MikroSens Beteiligten haben maßgeblich zum Erfolg dieser Arbeit beigetragen.

Weiterhin möchte ich mich bei meinen direkten Kollegen an der Hochschule Heilbronn für die angenehme Arbeitsatmosphäre und Unterstützung bedanken. Besonders hervorzuheben sind die Kollegen und Freunde Dmitrii Kozlov, Andreas Müller und Jens Klaski. Mein Dank gilt auch Prof. Dr.-Ing. Nicolaj Stache für die Unterstützung im Bereich Künstliche Intelligenz.

Zuletzt gilt mein Dank meinen Eltern, meiner Familie und meinen Freunden, die mich während meiner Promotionszeit unterstützt und motiviert haben.

Leonberg, im Oktober 2022

Kurzfassung

In dieser Dissertation werden lernende Verfahren zur Gesten- und Kochklassifikation mittels Radarsensorik vorgestellt. Die Verfahren umfassen zum einen die Klassifikation der Radardaten mithilfe verschiedener tiefer neuronaler Netzwerke. Zum anderen wird ein kognitives Radar vorgestellt, bei dem das Radar adaptiv auf die Szene reagieren und die Messparameter entsprechend anpassen kann. Die Verfahren werden anhand der Anwendung Kochsensor evaluiert. Der Kochsensor ist ein 120 GHz FMCW-Radar, das oberhalb eines Kochfelds angebracht ist und die aktive Kochposition, verschiedene Steuergesten und das Kochen selbst klassifiziert. Auf diese Weise wird ein Überkochschutz realisiert.

Die in dieser Arbeit erforschten Verfahren zeigen verschiedene Klassifikationsmöglichkeiten anhand von Range-Doppler-Matrix-Daten auf. Zunächst erfolgt die Klassifikation mit einer einzelnen Messung (Einzelbild-Klassifikation). Dieses Verfahren zeigt speziell bei dynamischen Klassen, wie den Gesten oder dem Kochen, Schwächen. Deutlich bessere Ergebnisse zeigt die Sequenz-Klassifikation, bei der eine Sequenz von Messungen mithilfe netzwerkinterner Speicher ausgewertet wird. Ein ähnlich gutes Ergebnis wird erreicht, wenn bei der Einzelbild-Klassifikation zusätzliche Sensormodalitäten zum Trainieren des Klassifikationsnetzwerks herangezogen werden. Bei diesem sogenannten Cross Learning werden die zusätzlichen Sensoren, wie eine Wärmebildkamera, allerdings nur zum Trainieren und nicht zur Klassifikation in der Anwendung benötigt. Ein weiteres lernendes Verfahren ist das Reinforcement Learning, das dem Radar ein kognitives Verhalten erlaubt. Das Radar kann adaptiv auf die Szene reagieren und die adaptiven Messparameter optimal wählen.

Eine wichtige Herangehensweise für lernende Verfahren ist die Wiederverwendung bereits trainierter Netzwerke für ein Transfer Learning. Vortrainierte Netzwerke können aus vorangegangenen Trainings stammen oder mittels unüberwachtem multimodalem Autoencoder-Training erzeugt werden. In dieser Arbeit wird vorgestellt, wie das Transfer Learning für Radarsensordaten genutzt werden kann.

Inhaltsverzeichnis

1 Einleitung

Seit einigen Jahren nehmen lernenden Verfahren eine zunehmend wichtige Rolle bei der Verarbeitung von Daten aller Art ein. Lernende Verfahren und im Speziellen tiefe neuronale Netzwerke (Deep Learning) sind Stand der Technik in der visuellen Objekterkennung, Spracherkennung, Texterkennung und in vielen anderen Bereichen wie in der Analyse von sozialen Medien, Kaufverhalten, Klima, Geologie, Medizin oder Genetik [1]–[3]. Im Bereich der Radarsensorik sind neuronale Netzwerke bisher weniger stark verbreitet, wobei die Einsatzmöglichkeiten und die Anzahl der Publikationen von Jahr zu Jahr zunehmen [4]. Einige Beispiele sind die Gestenerkennung [5]–[8], die Klassifikation menschlicher Aktivitäten [9]–[13] und das Zählen von Personen [14]. Andere Anwendungen sind beispielsweise die Automatische Zielerkennung (ATR) mit einem Synthetic Aperture Radar (SAR) [15]–[19] oder die Klassifikation von Drohnen [20]. Neuronale Netzwerke sind außerdem für das biologisch inspirierte Konzept des kognitiven Radars interessant [21], [22]. Eine Anwendung des kognitiven Radars aus der Literatur ist die Interferenzvermeidung [23], [24].

Moderne Radarsensorik, wie sie etwa in der Automobilindustrie zum Einsatz kommt, basiert meist auf sogenannten Frequency-Modulated Continuous Wave (FMCW)-Radarsensoren [25]–[29]. Auch die Anwendung im industriellen Umfeld ist zunehmend von Bedeutung [17], [30]–[34]. Die Leistungsfähigkeit, Lernfähigkeit und Flexibilität neuronaler Netzwerke kann im Allgemeinen zur Verbesserung der Detektion und Klassifikation eingesetzt werden.

Seit 2012 sind neuronale Netzwerke bzw. Deep Learning in der Bildverarbeitung sehr populär, da ein neuronales Netz die jährliche ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [35] mit deutlichem Vorsprung gewonnen hat. Dies ist ein Bildverarbeitungswettbewerb, bei dem etwa 1,2 Millionen Bilder je einer von 1000 verschiedenen Klassen zugeordnet werden müssen. 2012 wurde dieser Wettbewerb erstmals von einem neuronalen Netzwerk, dem sogenannten *AlexNet*, benannt nach dessen Entwickler Alex Krizhevsky, mit deutlichem Abstand gewonnen. Das neuronale Netzwerk hat eine Fehlerquote

von 15,3% [36] erreicht während der zweite Platz nur eine Fehlerquote von 26,2% erreicht hat. Mit diesem Ergebnis wurde ein Meilenstein in der Bildverarbeitung gesetzt und alle darauf folgenden Wettbewerbe wurden von neuronalen Netzwerken gewonnen.

Die Popularität neuronaler Netzwerke wird weiterhin durch eine stetig steigende Rechenleistung vorangetrieben. Für das sogenannte Training neuronaler Netzwerke ist eine große Rechenleistung erforderlich, die etwa von Grafikprozessoren (GPUs) bereitgestellt werden kann. Das aus heutiger Sicht relativ einfache *AlexNet* mit 650.000 Neuronen und 60 Millionen Parametern wurde etwa sechs Tage lang auf zwei GTX580 GPUs trainiert [36]. Dabei war hauptsächlich der Speicher von 3GB pro GPU limitierend. Ohne GPUs hätte das Training mehrere Monate benötigt. Heutige GPUs sind im Vergleich zu 2012 um ein Vielfaches leistungsfähiger und ein Speicher von 8-24 GB pro GPU ist üblich. Damit können wesentlich größere und leistungsfähigere neuronale Netzwerke trainiert werden.

1.1 Methodik und Ziele der Arbeit

In dieser Arbeit werden verschiedene Ansätze zur Gesten- und Kochklassifikation mittels Radarsensorik untersucht. Dies umfasst einerseits die Interpretation beziehungsweise Klassifikation der Radardaten mithilfe verschiedener tiefer neuronaler Netzwerke. Andererseits soll die Radarsensorik adaptiv auf die Szene reagieren und die Parametrisierung so anpassen, dass die Szene optimal klassifiziert werden kann. Die Radardaten liegen in Form einer Range-Doppler-Matrix (RDM) vor. Das heißt, es werden Entfernungen und relative Geschwindigkeiten gemessen. Die Grundlagen werden in Kapitel 2.1 erläutert. Abbildung 1.1 zeigt eine Übersicht der Ansätze dieser Arbeit.

Im Kapitel 4 wird eine einzelne RDM mittels eines Convolutional Neural Network (CNN) verarbeitet und klassifiziert. Ein CNN ist ein neuronales Netzwerk, das hauptsächlich aus Convolutional (Conv)-Schichten (siehe Kapitel 2.2.1) aufgebaut ist. Der Einfluss von weiteren Schichten, wie etwa Pooling-Schichten, wird untersucht. Die Regulierungstechniken Dropout und Batch Normalization werden experimentell untersucht. Das Ziel ist eine Netzwerk-Architektur mit einem guten Verhältnis zwischen der Anzahl an Parametern des Netzwerks und der Genauigkeit der Klassifikation. Die Netzwerk-Inferenz soll auf einem einge-

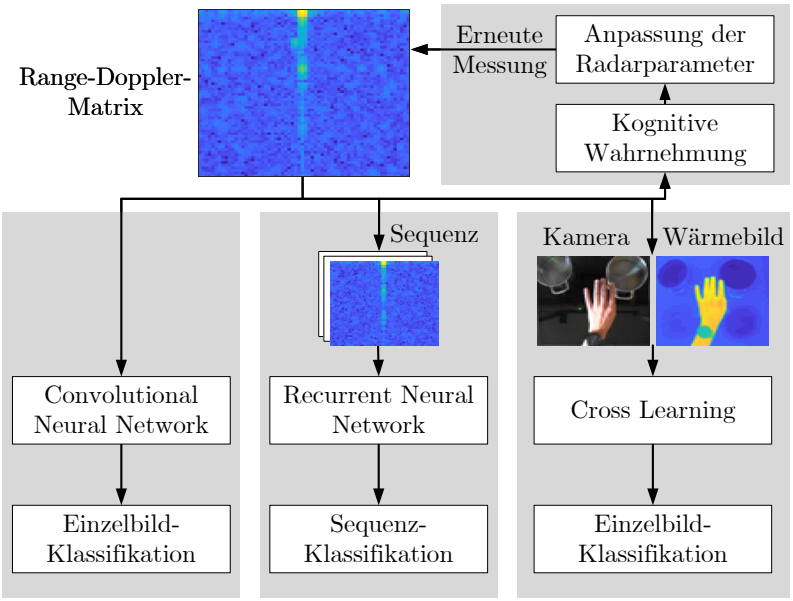


Abbildung 1.1: Übersicht der verschiedenen Ansätze zur Szenenerkennung mittels einer RDM. Neben der Klassifikation wird die adaptive Parametrisierung des Radars untersucht.

betteten System mit eingeschränkter Rechenkapazität implementiert werden können.

Eine Sequenz von RDM bietet Vorteile bei der Klassifizierung dynamischer Prozesse. Dazu zählen beispielsweise Gesten oder Kochprozesse. Bei der Sequenz-Klassifikation werden nicht nur einzelne RDM klassifiziert, sondern Informationen aus vorherigen RDM werden im Netzwerk gespeichert und für die aktuelle Klassifikation berücksichtigt. Ein solches Netzwerk wird auch Recurrent Neural Network genannt. Das im Netzwerk interne Speichern erfolgt beispielsweise mittels sogenannter Long Short-Term Memory (LSTM)-Schichten. In diesem Kapitel wird außerdem auf die Vorteile des sogenannten *Transfer Learning*

eingegangen. Dabei sind Teile des Netzwerks bereits vortrainiert und werden wiederverwendet.

Zur weiteren Verbesserung der Einzelbild-Klassifikation wird in Kapitel 5 ein *Cross Learning* Ansatz untersucht. Dabei werden zusätzliche Sensordaten, wie etwa Kamera- oder Wärmebilddaten, für das Training eines neuronalen Netzwerks verwendet. Die Eigenschaften und Vorteile der verschiedenen Sensoren können auf diese Weise kombiniert werden. Bei der Klassifikation in der Anwendung (Inferenz) wird auf die zusätzlichen Sensoren verzichtet. Der Ansatz bietet außerdem die Möglichkeit, mittels unüberwachtem Lernen ein vortrainiertes Netzwerks für ein *Transfer Learning* zu generieren.

In Kapitel 6 erfolgt die adaptive Parametrisierung des Radars auf Grundlage einer Constant False Alarm Rate (CFAR)-Detektion. Falls die Ziele in der Szene nicht detektiert werden, werden die Radarparameter entsprechend angepasst und es wird erneut gemessen. Das Ziel ist dabei zum einen, alle Ziele einer Szene mit möglichst wenigen Messungen zu detektieren. Zum anderen können auf diese Weise die optimalen Radarparameter für eine Szene bestimmt werden. Die Radarparameter werden hinsichtlich einer ressourcenschonenden (Energiebedarf, Abwärme) und schnellstmöglichen Messung optimiert.

Zusammenfassend sind die Kernziele dieser Arbeit:

- Die Klassifikation einzelner RDM mittels möglichst effizienter neuronaler Netzwerke zur eingebetteten Inferenz.
- Die Sequenz-Klassifikation einer dynamischen Szene, die mittels einer RDM-Sequenz abgebildet wird.
- Das Cross Learning verschiedener Sensormodalitäten zur Kombination der Vorteile verschiedener Sensoren bei unimodaler Inferenz.
- Die adaptive Parametrisierung kognitiver FMCW-Radar-Messungen zur ressourcenschonenden Szenen-Detektion.

Teile dieser Arbeit sind bereits in Zeitschriften und Konferenzen veröffentlicht [37]–[40].

1.2 Anwendung Kochsensor

Als dynamische Szene wird in dieser Dissertation ein Kochsensor inklusive Gestensteuerung betrachtet. Die Hauptfunktion ist ein Überkochschutz für Flüssigkeiten wie etwa Milch auf einem Herd. Ein Überkochen oder Anbrennen kann vermieden werden, indem rechtzeitig die Heizleistung reduziert wird. Der Herd besteht aus insgesamt vier Kochfeldern (2x2 Anordnung). Die aktiven Kochfelder werden vom Radar detektiert. Die Gestensteuerung erlaubt das Ein- und Ausschalten, sowie das Hoch- und Herunterregeln der Heizleistung. Der Herd ist in Abbildung 1.2 dargestellt. Die Sensoren befinden sich etwa 0,7 m oberhalb des Kochfelds. Dies entspricht der typischen Höhe einer Dunstabzugshaube. Insgesamt kommen drei Sensoren zum Einsatz, welche in Kapitel 2.3 vorgestellt werden. Der Hauptsensor ist ein 120 GHz FMCW-Radar. Für das *Cross Learning* in Kapitel 5 werden die Zusatzsensoren Wärmebildkamera und Farbkamera verwendet. Die drei Sensordaten werden synchronisiert mit 30 Messungen beziehungsweise Bildern pro Sekunde aufgenommen.

Das 120 GHz FMCW-Radar wurde im Rahmen des ZAFH-Projekts MikroSens entwickelt. Am ZAFH-Projekt sind die Universität Ulm, sowie die Hochschulen Heilbronn, Pforzheim und Ulm beteiligt. Das Vorhaben wird vom Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg, sowie dem Europäischen Fonds für regionale Entwicklung gefördert. Die Radarplattform wird im Kapitel 2.3.1 vorgestellt. Die leistungsfähige Plattform ermöglicht eine eingebettete Signalverarbeitung zur Berechnung der RDM. Die Implementierung von Algorithmen zur CFAR-Detektion oder einfacher CNNs zur Inferenz ist ebenfalls möglich. Die Parameter der FMCW-Radar-Messungen sind konfigurierbar.

Die Entfernungsmessung des Radars bietet die Möglichkeit, Füllstände in einem Topf zu messen. Ein drohendes Überkochen (steigender Füllstand) kann erkannt werden. Mithilfe des Dopplereffekts beziehungsweise der Geschwindigkeitsmessung können charakteristische Signaturen erfasst werden, die beim Kochen einer Flüssigkeit auftreten. Die charakteristischen Signaturen entstehen durch aufsteigende Gasbläschen und die sprudelnde Oberfläche beim Kochen.

Im industriellen Umfeld kann ein solcher Ansatz auf die Überwachung von Prozessen in Chemieanlagen oder Bioreaktoren übertragen werden. Verschiedene Prozessmesstechniken sind etwa bei der Herstellung von Biodiesel [41] oder dem Brauen von Bier [42] relevant. Das Radar kann als eine weitere Prozess-

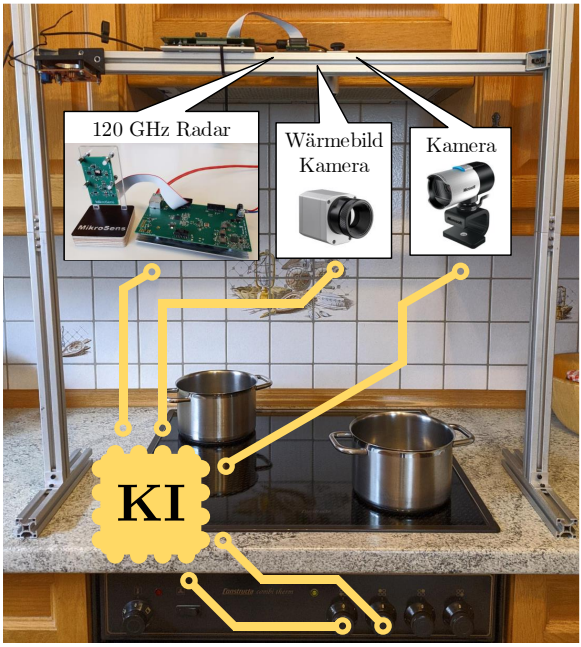


Abbildung 1.2: Aufbau der Anwendung Kochsensor. Es handelt sich um ein 2x2 Kochfeld. Auf Höhe der Dunstabzugshaube sind die drei Sensoren angebracht. Die Vorrichtung zur Ansteuerung der Heizleistung ist nicht dargestellt.

messtechnik ergänzt werden. Lernende Verfahren können bei der Klassifikation relevanter Prozesseigenschaften helfen.

Für die Anwendung sind zehn Klassen für eine Multi-Label-Klassifikation (siehe Kapitel 2.2.6) definiert. Bei der Multi-Label-Klassifikation kann jede Klasse unabhängig voneinander wahr oder falsch sein. Das bietet den Vorteil, dass alle möglichen Szenarien in der Szene mit möglichst wenigen Klassen abgebildet werden können. Etwa bei den vier Kochpositionen sind auf diese Weise nur vier Klassen notwendig anstatt jede mögliche Kombination als separate Klasse abzubilden. Die zehn Klassen sind folgendermaßen definiert:

- Drei Gesten für die Steuerung des Herds:
  - *Kreis-Geste* zum Ein- und Ausschalten.
  - *Fingerreiben* zur Verringerung der Heizleistung.
  - *Handneigung* zur Erhöhung der Heizleistung.
- Generelle *(Hand-) Bewegungen* ohne bestimmte Geste oder Funktion. Hierunter fallen beispielsweise Umrühren, das Hinzufügen von Zutaten, Würzen, das Bewegen/Umstellen von Töpfen und andere Bewegungen, die in einer Kochszene auftreten können.
- Vier Kochpositionen entsprechend der 2x2 Kochfelder:
  - *Hinten Links*
  - *Hinten Rechts*
  - *Vorne Links*
  - *Vorne Rechts*
- *Gefüllter Topf* gibt an, ob ein Topf mit etwas gefüllt ist. Der Topfinhalt kocht dabei nicht. Ohne Füllung muss ein Topf beispielsweise nicht überwacht werden. Die entsprechende Kochposition sollte ausgeschaltet sein.
- *Kochen* ist wahr, wenn der Topfinhalt sprudelnd kocht. Ein Überkochen könnte bald eintreten. Die Kochposition sollte gegebenenfalls abgeschaltet werden.

Die zehn Klassen werden im Kapitel 3 ausführlich mit Beispielbildern der verschiedenen Sensoren dargestellt. Die drei Gesten zur Steuerung wurden auf Basis von [5] gewählt. Die Gesten sollen einerseits eine gute Klassifikationsgenauigkeit aufweisen und andererseits möglichst nicht mit den generellen *(Hand-) Bewegungen* bei Kochvorgängen verwechselt werden.

In der Anwendung soll die Klassifikation ausschließlich mittels Radar erfolgen (unimodale Klassifikation). Die Gründe für diese Entscheidung sind folgende:

- Ein Radar ist robust in rauen Umgebungen mit Regen, Schnee oder Nebel in Automobilanwendungen [25]. Durch etwa Wasserdampf können beim Kochen vergleichbare Bedingungen auftreten. Die optischen Kameras können beschlagen und nicht mehr funktionieren.

- Radare sind bezogen auf Datenschutz und Privatsphäre weniger kritisch als Kameras zu sehen. In privaten Küchen sollten Radare größere Akzeptanz finden als Kameras.

- Hochauflösende Wärmebildkameras sind im Vergleich zu Radaren oder Farbkameras sehr teuer. Aus Kostengründen soll nur ein Radar verwendet werden.

- Ein Radar ist fähig, die zehn Klassen der Kochanwendung robust zu klassifizieren (siehe Kapitel 5.6).

2 Grundlagen

In diesem Kapitel werden die nötigen Grundlagen der Radarsensorik und des maschinellen Lernens vorgestellt. Anschließend werden die in dieser Arbeit verwendeten Sensoren vorgestellt.

2.1 Radargrundlagen

Die in dieser Arbeit verwendeten Radarsensoren sind sogenannte frequenzmodulierte Dauerstrichradare (FMCW-Radare). Diese weit verbreiteten Radare ermöglichen das gleichzeitige Messen von Abstand und Geschwindigkeit verschiedener Ziele [25], [30]. Das Modulationsverfahren verwendet eine sich zeitlich linear verändernde Sendefrequenz

$$f_{Tx} = f_0 + B\frac{t}{T}, \quad t \in [0, T]\,. \tag{2.1}$$

Die Sendefrequenz hat die Form einer Rampe mit der Startfrequenz f_0. Die Rampensteigung ist der Quotient aus dem Frequenzhub, beziehungsweise der Bandbreite B und der Rampendauer T. Deshalb wird auch von Frequenzrampen gesprochen (siehe Abbildung 2.1). Die Mittenfrequenz

$$f_\mathrm{c} = f_0 + \frac{B}{2}. \tag{2.2}$$

wird oft auch als Trägerfrequenz (carrier frequency) bezeichnet.

Das Empfangssignal ist um τ zeitverzögert. Die Zeitverzögerung

$$\tau = 2\frac{r}{c_0} \tag{2.3}$$

ist von der Entfernung r des reflektierenden Ziels abhängig und berücksichtigt den Hin- und Rückweg des Signals. Zusätzlich ist die Zeitverzögerung vom Übertragungsmedium abhängig. Die Ausbreitungsgeschwindigkeit in Luft entspricht

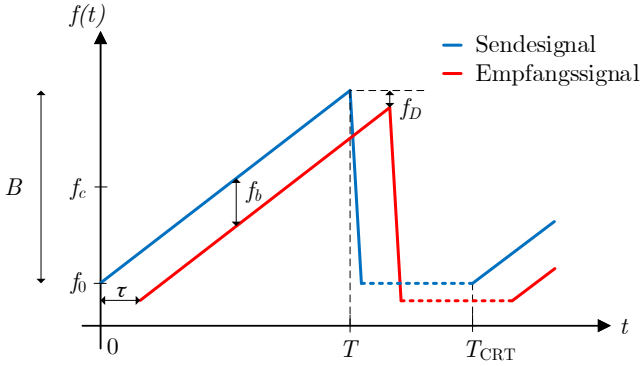


Abbildung 2.1: FMCW bzw. Chirp-Sequence-Frequenzrampe. Das Empfangssignal ist zum Sendesignal verschoben. Die zeitliche Verschiebung ist vom Hin- und Rückweg des Signals abhängig. Eine Frequenzverschiebung kann infolge einer Dopplerverschiebung auftreten.

näherungsweise der Lichtgeschwindigkeit im Vakuum c_0.

Weiterhin tritt bei relativer Bewegung zwischen Radar und Ziel eine Dopplerverschiebung auf. Mithilfe der Dopplerfrequenz

$$f_D = \frac{2v}{\lambda} \cos\varphi = \frac{2f_c v}{c_0} \cos\varphi \tag{2.4}$$

kann die radiale Geschwindigkeit v von Zielen bestimmt werden. Sie ist abhängig von der Wellenlänge $\lambda = \frac{c}{f}$ und dem Winkel φ zwischen Radar und Ziel. Die Dopplerfrequenz kann sowohl positiv als auch negativ sein. Das Vorzeichen berücksichtigt die Richtung der Geschwindigkeit relativ zum Radar.

Das Empfangssignal kann unter Berücksichtigung von Gleichung 2.3 und 2.4 folgendermaßen beschrieben werden:

$$f_{Rx} = f_0 + B\frac{t-\tau}{T} - f_D. \tag{2.5}$$

Durch das Mischen des Sende- und Empfangssignals im Empfänger kann das sogenannte Zwischenfrequenzsignal f_b (auch *beat frequency* genannt) bestimmt werden:

$$f_b = f_{Tx} - f_{Rx} = B\frac{\tau}{T} + f_D = \frac{2B}{c_0 T}r + \frac{2f_c}{c_0}v\cos\varphi. \tag{2.6}$$

Das Zwischenfrequenzsignal f_b enthält einen entfernungs- und einen geschwindigkeitsabhängigen Teil. Zur Entkopplung beider Teile wird das sogenannte Chirp-Sequence-Modulationsverfahren [30], [43] verwendet. Dabei werden sich wiederholende Frequenzrampen mit großer Rampensteigung und einer kurzen Wiederholdauer T_CRT verwendet. Eine große Rampensteigung wird mittels großer Bandbreite B und kurzer Dauer T erreicht. Die Chirp-Wiederholdauer T_CRT ist in der Regel im Bereich unter 100 µs. Da die Dopplerfrequenz nicht von der Rampensteigung B/T abhängig ist, bleibt die Dopplerfrequenz in einem niedrigen Frequenzbereich, während die entfernungsabhängigen Frequenzen im Spektrum nach oben wandern. Bei ausreichend großer Rampensteigung hat dies den Vorteil, dass die entfernungs- und geschwindigkeitsabhängigen Frequenzkomponenten (siehe Gleichung 2.6) entkoppelt werden.

Bei der Betrachtung einer einzelnen Rampe kann bei einer ausreichend schnellen Chirp-Sequence-Modulation die Dopplerfrequenz vernachlässigt werden. Die Entfernung

$$r = \frac{c_0 T}{2B} f_b \tag{2.7}$$

kann direkt durch die Messung des Zwischenfrequenzsignals f_b im Basisband bestimmt werden. Dazu wird das gemischte Basisbandsignal mit einem Analog-zu-Digital-Wandler (ADC) digitalisiert. Aus der Fast Fourier Transformation (FFT) des digitalisierten Zeitsignals ergibt sich das Frequenzspektrum, aus dem auf die Entfernungen von Zielen geschlossen werden kann.

Zur Geschwindigkeitsmessung kann eine Sequenz von L Rampen beobachtet werden. Dazu wird eine zweite FFT entlang der Sequenz von Rampen berechnet. Die Berechnung der beiden FFTs in Entfernungs- und Geschwindigkeitsrichtung wird auch als Zweidimensionale Fast Fourier-Transformation (2D-FFT) bezeichnet. Das Prinzip ist in Abbildung 2.2 dargestellt. Als Ergebnis folgt die sogenannte RDM, welche in der Literatur auch als Range-Doppler-Map bezeichnet wird. Die zweidimensionale Darstellung ermöglicht das einfache

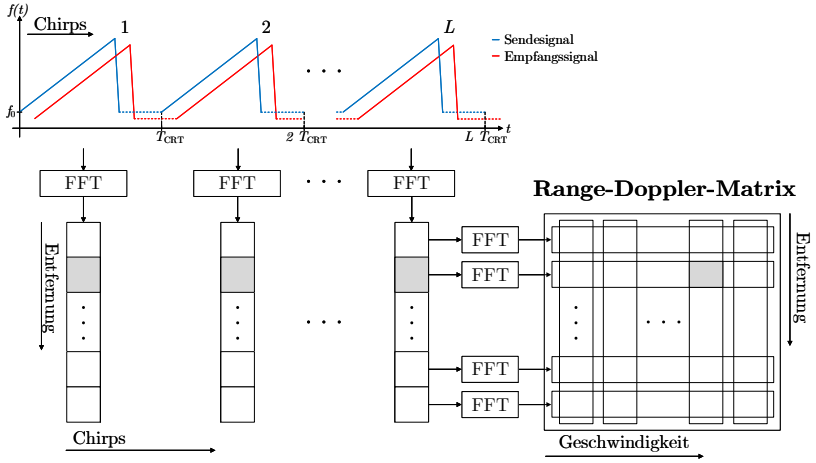


Abbildung 2.2: 2D-FFT zur Berechnung der RDM aus einer Chirp-Sequence-Messung (nach [43]). Die erste FFT wird in Entfernungsrichtung für jede Rampe berechnet. Die zweite FFT wird in Geschwindigkeitsrichtung entlang der Sequenz von Rampen (Chirps) berechnet.

Ablesen der Entfernung und Geschwindigkeit eines Ziels mit ausreichendem Signal-zu-Rausch-Verhältnis (SNR).

Die Geschwindigkeitsauflösung

$$\Delta v = \frac{c_0}{2 f_c T_{\mathrm{CRT}} L} \tag{2.8}$$

ist maßgeblich von der Beobachtungsdauer $T_{\mathrm{CRT}} L$ abhängig. Eine gute Geschwindigkeitsauflösung kann daher durch das Senden vieler Rampen in großem zeitlichen Abstand erreicht werden. In der Praxis ist dies allerdings nicht grenzenlos möglich, da die Messzeit oder der Speicher anwendungsbedingt begrenzt sein kann. Zusätzlich gibt es eine Abhängigkeit von der Bandbreite B, welche die Mittenfrequenz f_c nach Gleichung 2.2 beeinflusst. Im Gegensatz zur Beobachtungsdauer ist dieser Effekt oft zu vernachlässigen.

Die Entfernungsauflösung Δr kann durch Einsetzen der theoretisch möglichen

Auflösung des Zwischenfrequenzsignals $\Delta f_\mathrm{b} = \frac{1}{T}$ in Gleichung 2.7 angenähert werden [30]:

$$\Delta r = \frac{c_0}{2B}. \tag{2.9}$$

Mit dieser Entfernungsauflösung ist bei Radaren die Zieltrennfähigkeit gemeint. Das ist der Mindestabstand zweier Punktziele, damit diese noch als zwei separate Ziele detektiert werden können. In der Praxis kommen weitere Störfaktoren wie etwa Rauschen, Nichtlinearitäten der Rampen oder ausgedehnte Ziele statt Punktziele hinzu. Weiterhin ist die Entfernungsauflösung durch die Digitalisierung bzw. Diskretisierung von der Abtastfrequenz f_s des ADCs unter Einhaltung des Abtasttheorems abhängig. Dies limitiert die maximale eindeutig messbare Entfernung

$$r_\mathrm{max} = \frac{c_0 T}{4B} f_s, \quad f_s > 2 f_{b_{max}}. \tag{2.10}$$

Die Größe einer sogenannten Entfernungszelle

$$\Delta r_{Zelle} = \frac{c_0 T}{2B} \frac{f_s}{N_{FFT}} \tag{2.11}$$

in der RDM (Abbildung 2.2) hängt außerdem von der Größe N_FFT der verwendeten FFT (N-Punkt-FFT) ab. Genauer gesagt wird aufgrund der Symmetrie des reellwertigen Basisbandsignals nur das einseitige Spektrum mit den Koeffizienten bis $N_\mathrm{FFT}/2$ betrachtet. Im Vergleich zu Gleichung 2.10 kürzt sich der Faktor 4 im Nenner zu 2.

Die Größe einer Geschwindigkeitszelle entspricht der Geschwindigkeitsauflösung in Gleichung 2.8, falls die Größe der zweiten FFT entsprechend der Anzahl der Rampen L gewählt wird. Die maximale eindeutig messbare Geschwindigkeit in positiver und negativer Richtung beträgt

$$|v_\mathrm{max}| = \frac{c_0}{4 f_c T_\mathrm{CRT}}. \tag{2.12}$$

Die gesamte Geschwindigkeitsdynamik von maximal positiver zu maximal negativer Geschwindigkeit beträgt damit $2|v_\mathrm{max}|$.

Die für diese Arbeit wichtigen Radarparameter und die mithilfe der Gleichungen berechneten Größen sind in Tabelle 2.1 am Beispiel eines 120 GHz

Tabelle 2.1: Radarparameter und die daraus resultierenden Größen mit Beispielwerten eines 120 GHz Radars.

Symbol	Parameter	Wert
f_0	Startfrequenz	120 GHz
f_c	Trägerfrequenz (Mittenfrequenz)	122,5 GHz
B	Bandbreite	5 GHz
T	Chirp-Dauer	23,48 µs
T_{CRT}	Chirp-Wiederholdauer	54 µs
L	Anzahl an Rampen	64
N_{FFT}	Größe der Entfernungs-FFT	2048
f_s	ADC Abstastfrequenz	100 MHz
Δr_{Zelle}	Größe einer Entfernungszelle	0,0344 m
Δv	Größe einer Geschwindigkeitszelle	0,3543 m/s
r_{max}	Maximale Entfernung	35,22 m
v_{max}	Maximale Geschwindigkeit	±11,34 m/s

Radars zusammengefasst. Bei dem biologisch inspirierten Prinzip des kognitiven Radars [21], [22] sind einige der Radarparameter adaptiv und werden vom Radar selbst situationsbedingt angepasst. In dieser Arbeit (Kapitel 6) sind das die Bandbreite B, die Chirp-Dauer T, die Chirp-Wiederholdauer T_{CRT} und die Anzahl an Rampen L.

Die Zieldetektion, welche zwischen Zielen und einem unvermeidbaren Rauschen unterscheiden muss, kann beispielsweise mit einem CFAR-Algorithmus erfolgen [44]. Dabei wird der Rauschpegel anhand benachbarter Zellen (lokal) geschätzt und ein Schwellwert berechnet. Signalamplituden oberhalb dieses Schwellwerts werden als Ziele detektiert. Für die zweidimensionale RDM gibt es spezielle CFAR-Algorithmen [45], die Entfernung und Geschwindigkeit gemeinsam für die Rauschpegelschätzung und Schwellwertberechnung berücksichtigen. Ein Parameter des CFAR-Algorithmus ist die Falschalarmwahrscheinlichkeit.

Bei der Verarbeitung der RDM durch ein neuronales Netzwerk ist kein CFAR-Algorithmus erforderlich. Die RDM wird als eine Art ‚Bild' betrachtet und verarbeitet. Dabei stellt jedes Element der RDM ein Pixel dar. Im Kapitel 6 wird ein CFAR-Algorithmus mit adaptiver Falschalarmwahrscheinlichkeit zur Berechnung der Belohnung in der kognitiven Architektur verwendet.

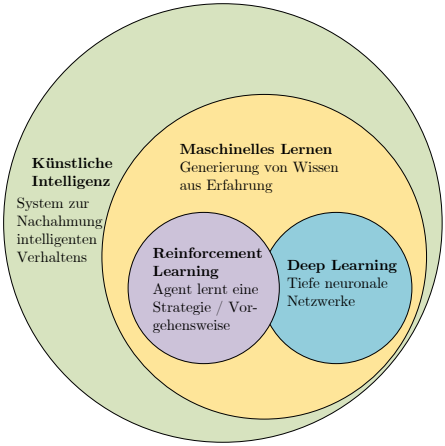
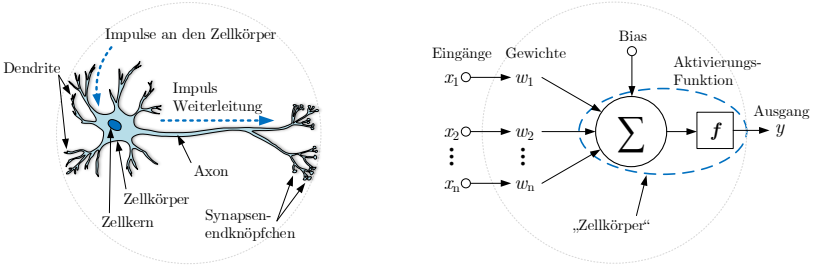


Abbildung 2.3: Einordnung der Begriffe Künstliche Intelligenz, Maschinelles Lernen, Deep Learning und Reinforcement Learning (nach [46]). Die Kombination aus Deep Learning und Reinforcement Learning wird Deep Reinforcement Learning genannt [47].

2.2 Grundlagen des maschinellen Lernens

Mit dem Begriff Künstliche Intelligenz (KI) wird ein Teilgebiet der Informatik bezeichnet, das sich mit der Nachahmung intelligenten Verhaltens befasst [48]–[50]. Das maschinelle Lernen ist ein Teilbereich der KI. Maschinelles Lernen befasst sich mit der Generierung von Wissen aus Erfahrung beziehungsweise dem selbstständigen Lernen aus Daten [51], [52]. Die algorithmischen Ansätze können sehr unterschiedlich sein. Einige Beispiele sind etwa Endscheidungsbäume, Support Vektor Maschinen (SVM) [53] oder Nächste-Nachbarn-Klassifikatoren [54]. Ein moderner Ansatz sind sogenannte künstliche neuronale Netzwerke. Aus diesem Ansatz ist der Begriff Deep Learning [2], [46], [55], [56] hervorgegangen, der für ‚tiefe' neuronale Netzwerke steht. Der Übergang, ab welchem ein neuronales Netzwerk als tief gilt, ist fließend. Daher lässt der Begriff Deep Learning in der Regel keine Aussage über die Komplexität und Tiefe des Netzwerks zu.



(a) Biologische Nervenzelle **(b)** Modell eines Neurons

Abbildung 2.4: Aufbau einer biologischen Nervenzelle und das daraus abgeleitete Modell eines Neurons (nach [59]).

Ein weiterer Teilbereich des maschinellen Lernens ist das verstärkende Lernen (Reinforcement Learning) [47], [57], [58]. Dabei lernt ein sogenannter Agent eine Strategie beziehungsweise eine Vorgehensweise für bestimmte Situationen. Die Vorgehensweise wird mittels eines Belohnungssystems bewertet und verbessert. Bei der Kombination aus Deep Learning und Reinforcement Learning wird die Vorgehensweise mittels eines tiefen neuronalen Netzwerks abgebildet und trainiert. Diese Kombination wird auch Deep Reinforcement Learning genannt [47]. Die Einordnung der Begriffe ist in Abbildung 2.3 dargestellt.

Das mathematische Modell eines neuronalen Netzwerks, beziehungsweise eines einzelnen Neurons, ist von dem Aufbau einer biologischen Nervenzelle abgeleitet (siehe Abbildung 2.4). Ein stark vereinfachtes Modell des Neurons bildet die gewichtete Summe aus den Eingängen und addiert das Bias (eigene Gewichtung). Anschließend folgt eine nichtlineare Aktivierungsfunktion, die je nach Funktion über die Aktivierung entscheidet oder den Ausgabewert skaliert. In einem neuronalen Netzwerk werden diese Neuronen schichtweise (in sogenannten Layern) angeordnet. Die Eingangs- und Ausgangssignale werden von Schicht zu Schicht weitergegeben. Ein vollständig verbundenes Netzwerk (Fully Conntected (FC)), bei dem alle Neuronen aus einer Schicht mit allen aus der nächsten Schicht verbunden sind, ist in Abbildung 2.5 dargestellt. Bei diesem Beispiel sind die Neuronen eindimensional (als Vektor) angeordnet. Bei

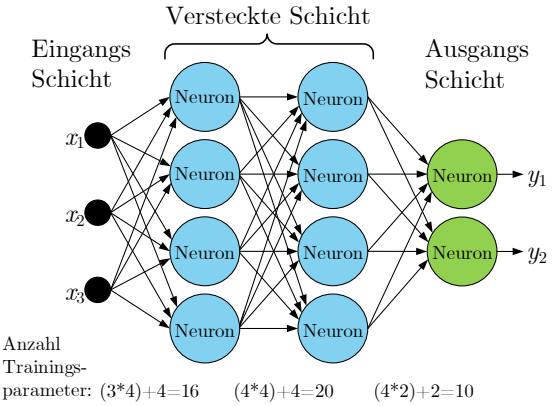


Abbildung 2.5: Beispiel eines neuronalen Netzwerks, das aus Schichten von Neuronen aufgebaut ist. Die Pfeilverbindungen stellen die (Übergangs-) Gewichtungen dar, die trainiert werden können. Neben den Übergangsgewichtungen hat jedes Neuron ein Bias, das ebenfalls trainiert werden kann. Die Anzahl trainierbarer Parameter entspricht der Anzahl trainierbarer Gewichtungen.

der Bildverarbeitung werden die Neuronen meist dreidimensional angeordnet. Die drei Dimensionen sind die Höhe H und Breite W des Bildes, sowie die Anzahl der Farbkanäle D als dritte Dimension. Eine dreidimensionale Schicht aus Neuronen wird daher auch als Volumen dargestellt (siehe Abbildung 2.6).

Bei dem Lernprozess des neuronalen Netzwerks, dem sogenannten Training, werden die Gewichtungen und das Bias angepasst. Das Training erfolgt mit einem Backpropagation-Algorithmus, der eine definierte Fehlerfunktion minimiert. Die Anzahl der Trainingsparameter setzt sich aus der Summe der Gewichte und dem Bias zusammen (siehe Abbildung 2.5). Das Training besteht in der Regel aus folgenden Schritten:

1. Die Eingangsdaten durchlaufen das Netzwerk vorwärts. Die Ausgangswerte werden berechnet.

2. Mithilfe einer definierten Fehlerfunktion wird aus den Ausgangswerten ein

Fehler berechnet. Beim sogenannten überwachten Lernen sind die wahren Ausgangswerte bekannt.

3. Der Fehlerwert durchläuft das Netzwerk rückwärts (Fehlerrückführung beziehungsweise Backpropagation). In jeder Schicht wird dabei ein Fehlergradient berechnet. Der Fehlergradient jedes Neurons entspricht dem Einfluss dieses Neurons auf den Gesamtfehler.

4. Entsprechend dem negativen Fehlergradient multipliziert mit einer Lernrate werden die Gewichtungen der Neuronen so angepasst, dass der Fehler minimal wird.

Dieses Vorgehen wird iterativ ausgeführt. Für neu initialisierte Netzwerke können sehr viele Iterationen notwendig sein, bis ein gewünschter Restfehler erreicht ist. Das Training kann durch Parallelisierung mithilfe von GPUs beschleunigt werden. Dabei werden die Eingangsdaten stapelweise verarbeitet. Ein solcher Stapel von Eingangsdaten wird *Batch* genannt. Die Batch-Größe n gibt an, wie viele Eingangsdaten parallel verarbeitet werden. Der oben genannte erste Trainingsschritt wird in der Regel für einen Batch parallel ausgeführt. Für die weiteren Schritte wird meist über den Batch gemittelt. Das Mitteln der Fehler und Fehlergradienten eines Batches führt insgesamt zu einem gleichmäßigerem Training und zur Unterdrückung von Ausreißern. Die Verarbeitung eines Batches entspricht einer Iteration. Die Verarbeitung des gesamten Trainingsdatensatzes wird Epoche genannt. Die Anzahl an Iterationen je Epoche hängt daher von der Trainingsdatensatzgröße und der Batch-Größe ab.

Eine weitere Möglichkeit zur Beschleunigung des Lernens ist die Verwendung bereits vortrainierter Netzwerke. Verschiedene Netzwerke der ILSVRC [35] stehen beispielsweise öffentlich zur Verfügung. Diese können ohne weiteres Training verwendet werden, um Bilddaten zu klassifizieren. Dabei stehen die 1000 vordefinierten Klassen zur Verfügung. Neuronale Netzwerke arbeiten so, dass die ersten Layer auf einfache Merkmale wie Farben oder Kanten (*low level features*) reagieren [60], [61]. Diese Merkmale sind oft universell für verschiedene Klassifikationsaufgaben geeignet. Erst in späteren Layern werden die einfachen Merkmale zu komplexeren Merkmalen (*high level features*) kombiniert. Bei einer Klassifikation von Kraftfahrzeugen ist ein solches komplexeres Merkmal beispielsweise ein Rad. Aufgrund dieser Funktionsweise neuronaler Netzwerke ist es möglich, auch nur Teile eines vortrainierten Netzwerks für eine völlig andere

Klassifikationsaufgabe zu verwenden. Dabei müssen dann nur einige Layer für die komplexeren Merkmale und die Klassifikation selbst trainiert werden. Dieses Vorgehen wird *Transfer Learning* genannt.

Nachfolgend werden einige wichtige Layer eines neuronalen Netzwerks vorgestellt. Allgemein betrachtet transformiert jeder Layer einen Eingang in einen Ausgang. Die dazu verwendete Funktion ist differenzierbar, damit die Fehlerrückführung und Gradientenberechnung erfolgen kann. Anschließend folgen einige Grundlagen zu den praktischen Herausforderungen neuronaler Netzwerke wie die Regulierung des Trainings, Overfitting, die Generalisierungslücke, die Definition der Fehlerfunktion und die Wahl des Trainingsalgorithmus mit einer geeigneten Lernrate.

2.2.1 Convolutional-Layer (Conv-Layer)

Der Conv-Layer ist eine Schicht, die mathematisch gesehen eine lokale diskrete Faltung durchführt. Im Gegensatz zum FC-Layer besteht beim Conv-Layer nur eine lokale Verbindungen zum vorherigen Layer (siehe Abbildung 2.6). Ein Neuron ist also nur mit einem kleinen Ausschnitt des vorherigen Layers verbunden. Dieser Ausschnitt wird auch *local receptive field* genannt. Der Ausschnitt ist durch die Filtergröße (räumliche Ausdehnung, z.B. 3x3) und die Tiefe des Eingangsvolumens definiert. Durch die Faltung wird das Filter schrittweise über das Eingangsvolumen bewegt. Je Filter entsteht eine *Volumenscheibe* des Ausgangsvolumens. Neuronen einer Volumenscheibe haben alle dieselben Gewichte, da dasselbe Filter repräsentiert wird. Jedes Neuron ist je nach räumlicher Position mit einem anderen Ausschnitt verbunden. Ein Filter, das beispielsweise auf eine Hell-Dunkel-Kante reagiert, deckt somit das gesamte Eingangsvolumen ab. Die Anzahl verschiedener Filter bestimmt die Tiefe des Ausgangs.

Ein Conv-Layer ist durch vier Hyperparameter definiert.

- C - Anzahl der Filter (auch Channels oder Kernels genannt)
- F - Räumliche Filtergröße (Spatial Size)
- S - Die Schrittweite, um die der Filter verschoben wird (Stride)
- P - Anzahl der Nullen, die am Rand ergänzt werden (Zero Padding)

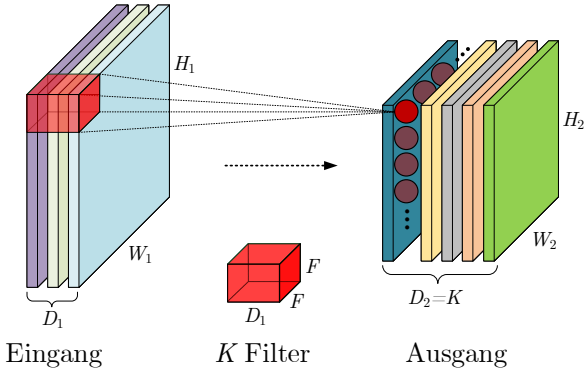


Abbildung 2.6: Beispiel eines Convolutional-Layers. Jedes Neuron ist nur mit einem kleinen Ausschnitt des Eingangs in der Größe des Filters verbunden. Bildlich gesprochen wird das Filter schrittweise über den Eingang bewegt. Die Anzahl der Filter entspricht der Tiefe des Ausgangs (Anzahl der Volumenscheiben).

Das Eingangsvolumen $W_1 H_1 D_1$ wird mit folgenden Formeln in ein Ausgangsvolumen $W_2 H_2 D_2$ transformiert:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \tag{2.13}$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 \tag{2.14}$$

$$D_2 = C. \tag{2.15}$$

Die Anzahl der Neuronen ist gleich dem Ausgangsvolumen $W_2 H_2 D_2$. Jedes Neuron ist mit dem Ausschnitt $F^2 D_1$ verbunden. Unter Berücksichtigung des Bias gibt es $F^2 D_1 + 1$ Trainingsparameter je Neuron. Das Besondere an dem Conv-Layer ist nun, dass die Trainingsparameter eines Filters über eine Volumenscheibe identisch sind. Deshalb beträgt die Anzahl der Trainingsparameter

$(F^2 D_1 + 1)C$ anstelle von $(F^2 D_1 + 1)W_2 H_2 D_2$. Die Anzahl der Filter ist deutlich geringer als die Anzahl an Neuronen. Damit kann der Rechenaufwand und Speicherplatzbedarf im Vergleich zu einem FC-Layer signifikant reduziert werden. [62] geht sogar soweit, das Netzwerk ausschließlich aus Conv-Layern aufzubauen. Dieser Extremfall wird Fully Convolutional Network (FCN) genannt.

2.2.2 Rekurrente Layer (LSTM-Layer)

Ein rekurrenter Layer ist ein Layer mit Rückkopplung. Ein Netzwerk mit rekurrentem Layer wird auch rekurrentes Netzwerk genannt. Ein State of the Art Layer mit Rückkopplung ist der LSTM-Layer. Die Rückkopplung ist dabei eine Art interner Speicher. LSTM wird alltäglich zur Sprach- und Texterkennung, sowie zur Übersetzung verwendet [63]. Bekannte Vertreter sind Alexa, Siri, Google oder sonstige smarte Sprachassistenten und Übersetzer. Erstmals veröffentlicht wurde LSTM bereits 1997 [64]. Seither wurde die grundlegende Idee immer weiter verbessert. 1999 wurde beispielsweise ein *Forget Gate* [65] ergänzt. Dieses ermöglicht es dem Netzwerk, das Vergessen (teilweises Löschen des internen Speichers) zu lernen. Vorteilhaft ist das besonders für das kontinuierliche Klassifizieren, wie es bei der Kochanwendung benötigt wird.

Ein Vergleich verschiedener Varianten von sogenannten LSTM-Zellen gibt [63]. Die State of the Art Variante in Abbildung 2.7 wird in dieser Arbeit verwendet. Die grundsätzliche Struktur besteht aus vier Gattern (englisch *gates*). Jedes Gatter entspricht einem internen Layer mit trainierbaren Gewichtungen und Bias. In der Standardvariante handelt es sich jeweils um einen FC-Layer. Anschließend folgt eine Aktivierungsfunktion. Typischerweise wird die *Sigmoid* oder *Tangens hyperbolicus (tanh)* Funktion verwendet (Vergleich Abbildung 2.10). Die vier Gatter sind:

- Vergessen-Gatter: Welche Speicherstellen sollen vergessen werden? Die Sigmoidfunktion liefert Werte zwischen 0 und 1. Eine 0 heißt entsprechend, dass diese Speicherstelle gelöscht werden soll, da diese Speicherstelle mit 0 multipliziert und sozusagen zurückgesetzt wird.

- Eingangsgatter: Welche Speicherstellen sollen ein Update erhalten? (Sigmoidfunktion)

- Update-Gatter: Welche Werte sollen in den Speicher geschrieben werden?

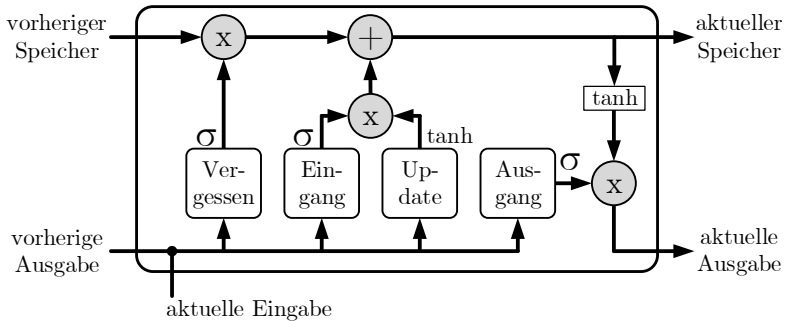


Abbildung 2.7: Eine LSTM-Zelle nach [63] ist aus vier Gattern aufgebaut. Jedes Gatter entspricht einem trainierbaren Layer. Jedem Gatter wird eine Verkettung aus vorheriger Ausgabe und der aktuellen Eingabe zugeführt. Die spezielle Struktur der Gatter erlaubt das Löschen, Updaten und die Ausgabe des internen Speichers. Die Ausgabe wird aus einer Kombination aus Speicherinhalt, vorheriger Ausgabe und aktueller Eingabe berechnet.

Hier wird typischerweise die tanh-Funktion mit Werten zwischen -1 und +1 verwendet. Es können entsprechend positive und negative Werte in den Speicher addiert werden.

- Ausgangsgatter: Welche Speicherstellen sollen ausgegeben werden? (Sigmoidfunktion)

Die trainierbaren Parameter jedes Gatters beinhalten neben den Gewichtungen und Bias für die aktuelle Eingabe zusätzlich die rekurrenten Gewichtungen und Bias für die vorherige Ausgabe. Im jedem Gatter wird also die vorherige Ausgabe und die aktuelle Eingabe berücksichtigt, indem beide verkettet werden. Das Ergebnis der jeweiligen Gatter wird anschließend entsprechend Abbildung 2.7 weitergeleitet und mit punktweiser Addition oder Multiplikation zusammengeführt. Das Vergessen-Gatter löscht teilweise den internen Speicher. Das Eingangs- und Update-Gatter aktualisieren den Speicher. Die aktuelle Ausgabe ergibt sich schließlich aus der punktweisen Multiplikation von Ausgangsgatter und dem bereits aktualisierten Speicher nach einer tanh-Aktivierungsfunktion.

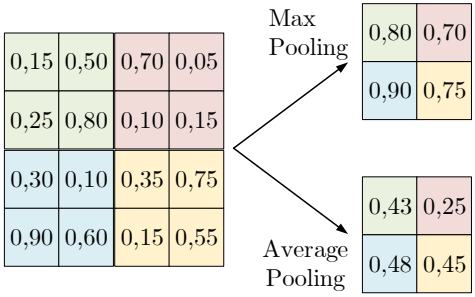


Abbildung 2.8: Beispiel eines Maximalwert- und Mittelwert-Poolings mit den Hyperparametern $F = 2$ und $S = 2$. Die räumliche Größe wird dabei halbiert. Je nach Operation wird der Maximalwert oder Mittelwert aus dem entsprechenden Bereich (farblich dargestellt) übernommen.

2.2.3 Pooling-Layer

Ein Pooling-Layer reduziert die räumlichen Dimensionen W und H und entspricht damit einer Art *Downsampling*. Weit verbreitete Varianten sind das Maximalwert-Pooling und das Mittelwert-Pooling [66]. Die Hyperparameter sind die Filtergröße F und Schrittweite S. Im Gegensatz zum Conv-Layer wird die Tiefe D des Volumens nicht verändert ($D_1 = D_2$). Ansonsten sind die Gleichungen 2.13 und 2.14 auch für Pooling-Layer gültig.

In Abbildung 2.8 ist das Maximalwert-Pooling und das Mittelwert-Pooling dargestellt. In diesem Beispiel mit der Filtergröße $F = 2$ wird beim Maximalwert-Pooling der Maximalwert aus einem 2x2 Bereich gewählt. Bei dem Mittelwert-Pooling wird der Mittelwert aus den vier Werten des Bereichs berechnet. Das Maximalwert-Pooling leitet im Prinzip Ausreißer weiter. Aufgrund der biologischen Inspiration neuronaler Netzwerke ist es aber durchaus gewünscht, dass auf große Reize reagiert wird und diese nicht durch eine Mittelwertbildung unterdrückt werden. Der Einfluss von Pooling-Layern wird in Kapitel 4.1 untersucht.

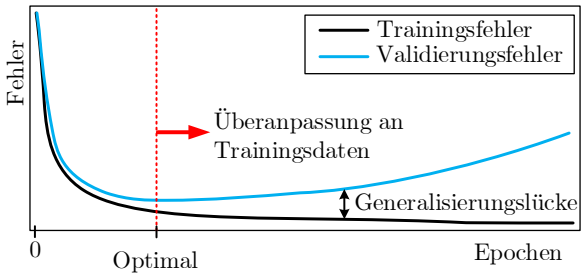


Abbildung 2.9: Ein typisches Problem beim Training eines neuronalen Netzwerks ist die Überanpassung (Overfitting). Dabei ist das Netzwerk so sehr an die Trainingsdaten angepasst, dass die Validierung mit anderen Daten schlecht ist. Dieses Problem ist typischerweise durch das Auseinanderlaufen des Trainings- und Validierungsfehler zu erkennen.

2.2.4 Dropout und Batch Normalization

Dieser Typ von Layer wird zur Regulierung (Regularization) des Trainings verwendet. Das Training soll damit gleichmäßiger und stabiler (Vermeidung von Ausreißern) werden. Allerdings wird das Training in der Regel langsamer. Die Verwendung solcher Layer kann, aber muss nicht vorteilhaft sein. Vielmehr kommt es darauf an, ob das Training beispielsweise eine Überanpassung (Overfitting) [67]–[69] oder sonstige Probleme aufweist. Der Effekt einer Überpassung ist in Abbildung 2.9 dargestellt. Bei der Überanpassung ist das neuronale Netzwerk zu sehr an die Trainingsdaten angepasst. Dadurch geht die Generalisierung verloren und der Validierungsfehler nimmt zu. Der Abstand der beiden Kurven wird auch Generalisierungslücke genannt.

Zur Verringerung der Generalisierungslücke kann der Dropout-Layer [70] verwendet werden. Dieser deaktiviert zufällig einen gewissen Prozentsatz an Neuronen während des Trainings. Die deaktivierten Neuronen werden für jeden Batch erneut zufällig ausgewählt. Auf diese Weise ist ein zufälliger Teil des Netzwerks inaktiv. Die grundsätzliche Idee dahinter ist, dass dadurch eine Redundanz entsteht und die Generalisierung verbessert wird.

Der Batch Normalization-Layer [71] normalisiert den Ausgang eines Layers vor

der Aktivierungsfunktion. Die Normalisierung erfolgt für jeden Batch separat. Die Ausgangswerte sollen ein Mittelwert von 0 und eine Varianz von 1 haben. Die Vorteile sind ein schnelleres und besseres Training. Die Regulierung mittels Dropout Layer soll durch die Batch Normalization überflüssig sein. In der Praxis ist dies allerdings nicht immer der Fall. Außerdem ist der Batch Normalization-Layer für die Sigmoid-Aktivierungsfunktion entworfen worden und die genannten Vorteile sind von der Wahl der passenden Aktivierungsfunktion abhängig.

2.2.5 Aktivierungsfunktionen

Ein Aktivierungslayer folgt in der Regel nach jedem Layer, der Neuronen enthält (siehe Abbildung 2.4). Die Neuronen werden entsprechend der gewählten Aktivierungsfunktion aktiviert beziehungsweise skaliert. Aufgrund der Übersichtlichkeit werden die Aktivierungslayer oft nicht aufgelistet oder dargestellt. Auch in dieser Arbeit wird auf die Darstellung verzichtet, aber die verwendete Funktion wird im zugehörigen Text genannt. Einige Aktivierungsfunktionen sind in Abbildung 2.10 dargestellt. Prinzipiell ist jede beliebige Funktion anwendbar. Meist wird jedoch die Rectified Linear Unit (ReLU) [72] oder eine leicht veränderte Version mit beispielsweise einer Begrenzung (*Clipped* ReLU) verwendet. Negative Ausgangswerte werden von den meisten Funktionen unterdrückt. Eine Ausnahme ist die *Leaky* ReLU.

Eine weitere wichtige Aktivierungsfunktion, welche in der Regel erst ganz am Ende eines Netzwerks verwendet wird, ist die Softmax-Funktion. Die Ausgangswerte (auch scores s_j genannt) werden als einfach interpretierbare prozentuale Wahrscheinlichkeiten P_j ausgegeben. Die Summe der Wahrscheinlichkeiten wird zu 1 normiert. Deshalb wird die Softmax-Funktion meist zur Klassifikation am Ende des Netzwerks verwendet. Die Berechnung für K Klassen erfolgt mittels

$$P_j(s) = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}}. \tag{2.16}$$

2.2.6 Netzwerk-Ausgabe und Fehlerfunktionen

Die Wahl einer geeigneten Fehlerfunktion ist essentiell für den Lernerfolg und die Leistungsfähigkeit (z.B. Klassifikationsgenauigkeit) eines neuronalen Netzwerks. Die Fehlerfunktion ist von der Art des Netzwerks sowie dem generellen

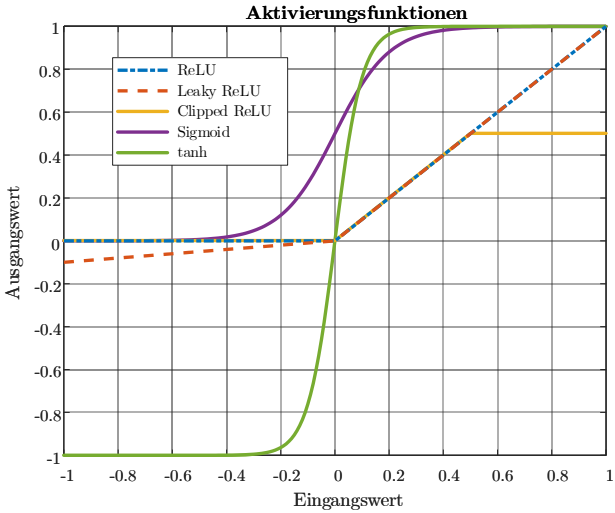


Abbildung 2.10: Darstellung einiger relevanter Aktivierungsfunkionen, die in dieser Arbeit verwendet werden. Mit einer Aktivierungsfunktion kann der Ausgabewert eines Neurons angepasst und gegebenenfalls begrenzt werden.

Verwendungszweck des Netzwerks abhängig. Der einfachste Fall ist eine binäre Entscheidung, beispielsweise zwischen Ja und Nein. Als Fehlerfunktion wird meist die Differenz zwischen Netzwerkausgabe und wahrem Wert berechnet. Die Differenz kann wahlweise quadriert oder logarithmiert werden.

Die Entscheidung zwischen mehreren Klassen K ist eine sogenannte Multi-Class-Klassifikation mit sich gegenseitig ausschließenden Klassen. Das heißt, nur eine der K Klassen kann gleichzeitig wahr sein. Beim überwachten Training ist die grundlegende Wahrheit (*Ground Truth*) bekannt und nur eine Klasse wahr. Die wahre Klasse wird in der Regel mit 1-aus-n-Code (*One-Hot-Encoding*) codiert. Bei zehn möglichen Klassen ist das beispielsweise ein Vektor der Länge zehn mit nur einem Element ungleich 0. Die Anzahl der Neuronen im letzten FC-Layer entspricht der Anzahl der Klassen. Am Ausgang wird für die Multi-

Class-Klassifikation meist die Softmax-Funktion (Gleichung 2.16) verwendet. Als Fehlerfunktion wird der sogenannte *Cross-Entropy-Fehler* (Kreuzentropie) [73] verwendet. Der Fehler wird in der Literatur oft auch *loss* genannt.

Anstelle einer Entscheidung oder Wahrscheinlichkeit für definierte Klassen können neuronale Netzwerke auch numerische Werte ausgeben. In der Literatur wird dies Regression genannt. Typische Beispiele sind etwa die Ausgabe einer Temperatur in Grad Celsius. Die grundsätzliche Struktur der Netzwerke ist dabei kaum anders und unterscheidet sich meist nur in den letzten Schichten und der Fehlerfunktion. Die Anzahl der Neuronen im letzten FC-Layer entspricht der Anzahl der gewünschten Ausgabewerte. Der Wertebereich für die Ausgabe eines Neurons kann mit der *Clipped* ReLU Aktivierungsfunktion skaliert und begrenzt werden. Für die Fehlerfunktion wird meist eine Form der mittleren quadratischen Abweichung (Mean Square Error (MSE)) verwendet.

In dieser Arbeit wird eine Mischung aus Klassifikation und Regression verwendet, die Multi-Label-Klassifikation genannt wird [74]. Der Unterschied zur Multi-Class-Klassifikation ist, dass die Klassen sich nicht gegenseitig ausschließen. Daher können mehrere Klassen gleichzeitig wahr sein. Ebenso ist es möglich, dass keine Klasse wahr ist. Praktisch umgesetzt wird dies als Regression mit einer Begrenzung des Ausgabe-Wertebereichs zwischen 0 und 1. Die Ausgabe entspricht dann der Wahrscheinlichkeit, dass die Klasse wahr ist. Für K Klassen werden damit K unabhängige Wahrscheinlichkeiten ausgegeben. Jedes Neuron kann als unabhängiger binärer Klassifikator betrachtet werden. Mit einer entsprechenden Entscheidungsschwelle (*threshold*) trifft jedes Neuron eine binäre Entscheidung. Je nach Anwendung und Definition der Klassen kann es passieren, dass die binären Klassifikatoren nicht vollständig unabhängig voneinander sind.

Für die Multi-Label-Klassifikation sind spezielle Fehlerfunktionen und Vorgehensweisen entwickelt worden. Einen Überblick gibt [74]. Eine Art Softmax-Funktion für die Multi-Label-Klassifikation ist als *Sparsemax* in [75] beschrieben. Je nach Anwendung kann es vorteilhaft sein, die Klassen zu Gruppieren und zunächst die übergeordnete Gruppe zu ermitteln. Anschließend wird innerhalb der Gruppe die genaue Klasse bestimmt. In [76] ist ein solcher Ansatz mit verketteten Klassifikatoren beschrieben. In dieser Arbeit werden verschiedene Fehlerfunktionen untersucht und verglichen.

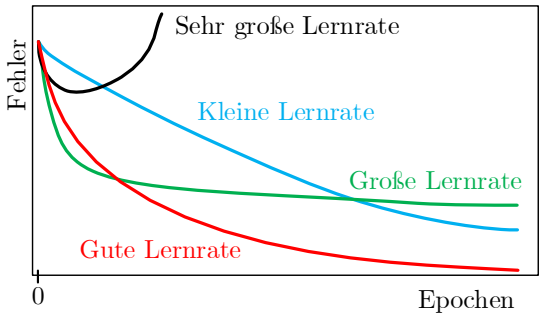


Abbildung 2.11: Die Lernrate hat einen großen Einfluss auf den Trainingsverlauf. Eine ungeeignete Lernrate ist typischerweise an einem gar nicht, zu früh oder zu spät konvergierendem Fehler zu erkennen. Bei modernen Trainingsalgorithmen kann sich die Lernrate auch adaptiv im Trainingsverlauf verändern.

2.2.7 Training mittels Fehlerrückführung

Das Training neuronaler Netzwerke mittels Fehlerrückführung (Backpropagation) basiert auf dem Gradientenverfahren. Einen Überblick der State of the Art Gradientenverfahren für neuronale Netzwerke geben [77], [78]. In dieser Arbeit wird der Trainingsalgorithmus Adaptive Momentum Estimation (Adam) [79] verwendet. Adam bietet im Gegensatz zu anderen Trainingsalgorithmen [77] einige praktische Vorteile, die im Folgenden erläutert werden. Der größte Vorteil ist, dass Adam während des Trainings adaptiv die Lernrate anpasst. Die Adaption basiert einerseits auf dem Durchschnitt der quadrierten Gradienten und andererseits auf den vorherigen Gradienten (Momentum) [79]. Aufgrund der Adaption ist nur eine initiale Lernrate erforderlich, wodurch der Einfluss der initial gewählten Lernrate im Laufe des Trainings abnimmt. Der praktische Aufwand für eine Optimierung der Lernrate kann damit reduziert werden.

Die Wahl einer geeigneten Lernrate ist wie in Abbildung 2.11 dargestellt essentiell. Eine zu kleine Lernrate kann das Training unnötig in die Länge ziehen und ein minimaler Fehler wird gegebenenfalls nie erreicht. Bei einer zu

großen Lernrate können einzelne Updates der Gewichtungen einen zu großen Einfluss haben. Dies führt gegebenenfalls zu einer starken Überanpassung an etwa Ausreißer. Der adaptive Trainingsalgorithmus Adam benötigt nur eine initiale Lernrate, welche typischerweise bei 10^{-3} bis 10^{-6} liegt. Die Lernrate sollte immer unter Berücksichtigung der Batch-Größe n festgelegt werden. Ist die Batch-Größe größer, werden Ausreißer durch das Mitteln über den Batch weitestgehend unterdrückt. Daher kann die Lernrate mit steigender Batch-Größe ebenfalls größer gewählt werden. Weiterhin kann ein Lernraten-Faktor für jeden Layer individuell festgelegt werden. Damit können bestimmte Layer stärker oder weniger stark trainiert werden. Dies ist besonders beim *Transfer Learning* wichtig. Die vortrainierten Layer können beispielsweise komplett eingefroren werden, indem der Lernraten-Faktor für diese Layer zu 0 gesetzt wird.

Für das Training rekurrenter Netzwerke sind spezielle Gradientenverfahren notwendig, da die Fehlerrückführung auch durch die zeitliche Sequenz erfolgen muss. Der Ansatz wird *Backpropagation through time* genannt und ist in [80] beschrieben. Als Trainingsparameter kommt hier noch die Möglichkeit hinzu, den internen Speicher des LSTM-Layers zurückzusetzen. Das Zurücksetzen kann beispielsweise nach jeder Iteration, nach jeder Epoche oder auch nie erfolgen.

2.3 Vorstellung der verwendeten Sensoren und Software

Nachfolgend werden der 120 GHz Radarsensor sowie die verwendeten Kameras vorgestellt. Als Software kommt MATLAB® 2020b zum Einsatz.

2.3.1 ZAFH MikroSens 120 GHz Sensor

Der Radarsensor basiert auf dem TRX-120 Monolithic Microwave Integrated Circuit (MMIC) von Silicon Radar. Der Chip ist in Abbildung 2.12 links in der Mitte des Frontends zu sehen. Auf der rechten Seite ist das Backend zu sehen. Das Field Programmable Gate Array (FPGA) basierte Backend erlaubt eine sehr schnelle Signalverarbeitung, da etwa die 2D-FFT in programmierbarer Hardware umgesetzt ist. Einfache CNNs zur Inferenz können auf den zentralen Rechenkernen implementiert werden. Die typischen Parameter des Sensors sind in Tabelle 2.1 aufgelistet. Parameter wie die FFT-Größe, die Bandbreite, die Chirp-Dauer, die Anzahl der Rampen und die Chirp-Wiederholdauer sind

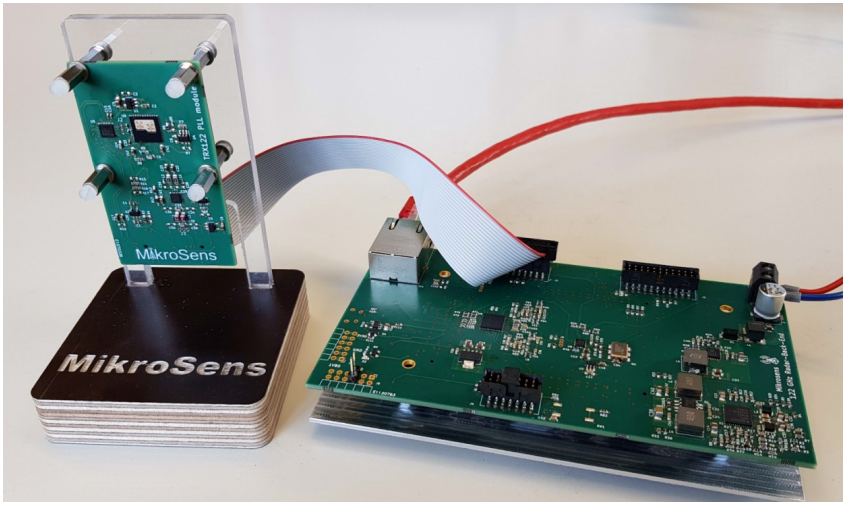


Abbildung 2.12: MikroSens 120 GHz Radar: Auf der linken Platine ist der TRX-120 MMIC mit integrierten Antennen zu sehen. Auf der rechten Platine befindet sich ein FPGA zur Signalverarbeitung sowie die Netzwerkschnittstelle zu einem Rechner mit MATLAB®. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

konfigurierbar. Das Radar kann Entfernungen und Geschwindigkeiten messen. Der Sensor erlaubt keine Winkelschätzung, da nur je eine Sende- und Empfangsantenne vorhanden ist. Der Abstrahlwinkel der Antennen beträgt sowohl horizontal als auch vertikal etwa ±30°(-6dB).

In dieser Arbeit wird durchgängig der 120 GHz Sensor verwendet. Bereits veröffentlichte Teile dieser Dissertation basieren auf einem 77 GHz Sensor und einem anderen Datensatz ohne Gestenklassifikation und nur einem einzelnen Kochfeld [37]–[39]. Damit die Ergebnisse der verschiedenen Ansätze verglichen werden können, werden in dieser Arbeit die Experimente mit dem 120 GHz Sensor und einem einheitlichen Datensatz wiederholt. Die Zahlenwerte können aufgrund des neuen Datensatzes mit zusätzlichen Klassen von den bereits veröffentlichten Ergebnissen [37]–[39] abweichen. Die grundsätzlichen Ergebnisse konnten mit dem 120 GHz Sensor und dem neuen Datensatz reproduziert werden.

2.3.2 Farbkamera und Wärmebildkamera

Die verwendete Farbkamera (RGB-Kamera) ist eine Microsoft LifeCam Studio HD Webcam. Die Auflösung beträgt 1920x1080 Pixel. Die maximale Framerate beträgt 30 Hz.

Die Wärmebildkamera ist eine Optris PI 640 mit einer Auflösung von 640x480 Pixel und einer Framerate von bis zu 125 Hz. Messungen sind im Temperaturbereich von −20 °C bis 900 °C möglich. Für die Kochanwendung wird der Temperaturmessbereich von 0 °C bis 125 °C verwendet. Ein Cerankochfeld wird zwar bis zu etwa 600°C heiß, aber das zumeist auf Wasser basierende Kochgut wird in der Regel nicht wesentlich heißer als 100 °C. Die Wärmebilder werden mit 125 °C normiert. Die Temperatur kann daher pixelweise ausgelesen werden, indem der normierte Pixelwert mit 125 multipliziert wird.

2.3.3 Software

Die Software dieser Dissertation ist in MATLAB® 2020b umgesetzt. Die Software umfasst folgende Komponenten:

- Synchronisierte Erfassung und gegebenenfalls Abspeichern aller drei Sensordaten.
- Training und Test verschiedener neuronaler Netzwerke.
- Echtzeitanwendung für den Kochsensor mit Klassifikation und automatischer Abschaltung im Falle von Überkochen. Zusätzlich eine Bedienoberfläche für die manuelle Steuerung.

Sofern keine Angaben zu Parametern genannt sind, werden die MATLAB-Standardparameter für die jeweiligen Funktionen und Objekte (Software-Klassen) verwendet.

Bereits veröffentlichte Teile dieser Dissertation wurden im Tensorflow [81] Framework umgesetzt, da MATLAB® die Deep Learning Toolbox erst im Laufe dieser Arbeit entsprechend aufgebaut hat. Alle notwendigen Tools sind erst seit MATLAB® Release 2019a verfügbar. Die Ergebnisse sind mit denen des Tensorflow Frameworks, abgesehen von statistischen Schwankungen, identisch.

3 Datensatz

In diesem Kapitel werden der Datensatz und die definierten Klassen der Kochanwendung vorgestellt. Die verschiedenen Ansätze der Arbeit werden anhand dieses einheitlichen Datensatzes verglichen. Dazu werden abschließend die Details zur Fehlerberechnung und die Bewertung der Leistungsfähigkeit der Netzwerke erläutert.

Der Datensatz enthält insgesamt 57.600 Stichproben. Da jeweils eine RDM und zwei Bilder (Farb- und Wärmebild) je Stichprobe aufgenommen werden, enthält der Datensatz insgesamt 172.800 Bilder. Die RDM wird ebenfalls als eine Art ‚Bild' betrachtet. Dabei stellt jedes Element der RDM ein Pixel dar. Aufgrund der Sequenz-Klassifikation in Kapitel 4.2 werden die Stichproben in Sequenzen der Länge 20 aufgezeichnet. Der Datensatz enthält damit 2.880 Sequenzen. Die Framerate der synchronisierten Sensoren beträgt 30 Bilder pro Sekunde. Die Beobachtungszeit einer Sequenz beträgt 667 ms. Innerhalb dieser Beobachtungszeit kann eine Geste gut abgebildet und klassifiziert werden [5].

Für die Experimente wird der Datensatz einmalig zufällig in einen Trainings- und Testdatensatz geteilt. Die Teilung erfolgt anhand ganzer Sequenzen, damit einzelne Sequenzen nicht geteilt werden. Durch die Teilung wird sichergestellt, dass es keine Überschneidungen der Trainings- und Testdaten gibt [82], [83]. Die prozentuale Aufteilung hat dabei, abhängig von der Anwendung, einen großen Einfluss auf das Ergebnis [82]. Ein zu großer Trainingsdatensatz kann zu einer Überanpassung [67]–[69] an die Trainingsdaten führen (siehe Kapitel 2.2.4). Weiterhin ist der Testdatensatz ausreichend groß zu wählen, sodass ein Vergleich der Fehlerrate (siehe Kapitel 3.3) eine Aussagekraft über absolute Fehler zulässt. Bei einem zu kleinen Testdatensatz könnten die Unterschiede der Fehlerrate nur einzelne oder Bruchteile von Stichproben betreffen.

Die prozentuale Aufteilung in dieser Arbeit ist mit 52% Trainings- und 48% Testdaten gewählt. Der Trainingsdatensatz enthält 1.500 Sequenzen und der Testdatensatz entsprechend 1.380 Sequenzen. Anders ausgedrückt sind das 30.000 Stichproben im Trainings- und 27.600 Stichproben im Testdatensatz.

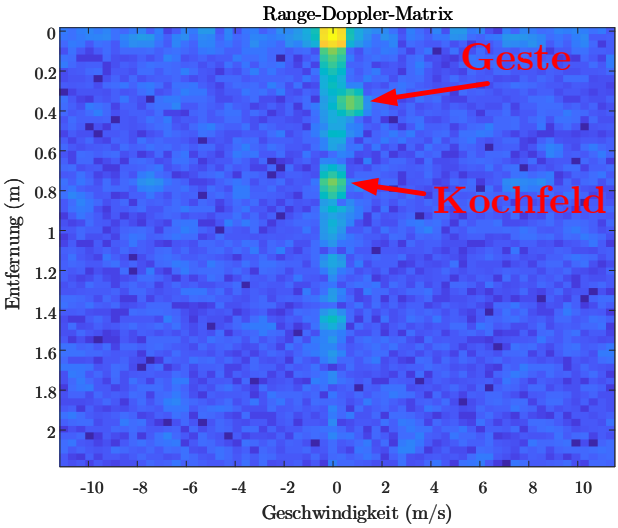


Abbildung 3.1: Beispiel einer Range-Doppler-Matrix (RDM) aus der Kochanwendung. In etwa 0,7 m Entfernung ist die Kochfeldoberfläche messbar. Eine Geste ist mit entsprechenden Geschwindigkeiten zwischen Radar und Kochfeld zu erkennen.

Der Datensatz ist am selben Herd mit zwei Töpfen desselben Typs aufgenommen worden. Die Gesten wurden von insgesamt vier verschiedenen Personen aufgezeichnet. Dabei wurden die Gesten links- und rechtshändig ausgeführt. In Abbildung 3.1 ist eine typische RDM aus der Kochanwendung dargestellt. Die Kochfeldoberfläche ist etwa 0,7 m von dem Radarsensor entfernt messbar. Die 0,7 m entsprechen etwa der Höhe einer Dunstabzugshaube. Der Radarsensor ist wenige Grad schräg montiert, sodass die Entfernung und auch die Radar Cross Section (RCS) je nach Kochposition leicht unterschiedlich sind. Der Radarquerschnitt (RCS) beschreibt die Reflektivität eines Objekts. Dies erlaubt die Detektion der Kochposition, obwohl diese nicht etwa mittels einer Winkelschätzung direkt messbar ist. Zwischen dem Radarsensor und dem Kochfeld kann die

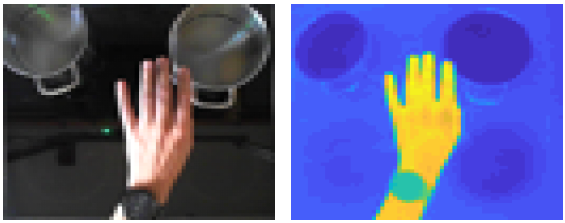
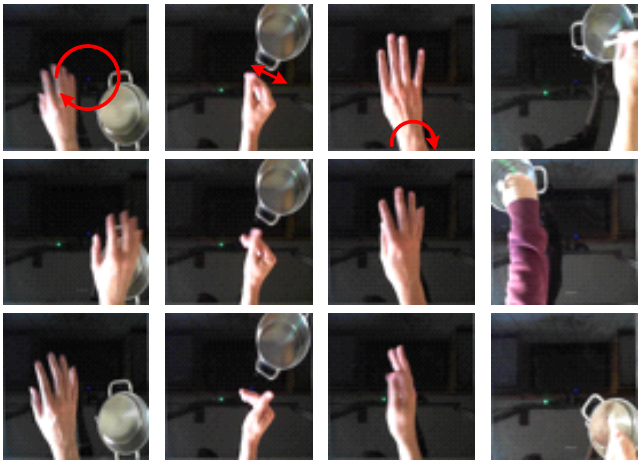


Abbildung 3.2: Beispiele der Kamera- und Wärmekamerabild aus der Kochanwendung. Die beiden nebeneinander montierten Kameras zeigen bestmöglichst denselben Bildausschnitt. Im Wärmebild ist zu erkennen, dass die Kochfelder in Relation zur Körpertemperatur kalt beziehungsweise ausgeschaltet sind.

Entfernung und Geschwindigkeit einer Geste erkannt werden. Ein Topf würde unmittelbar über dem Kochfeld erscheinen. Die Entfernung ist vom Füllstand abhängig. Beim Kochen treten in der Entfernung des Topfs charakteristische Geschwindigkeiten der sprudelnden Flüssigkeit auf.

Die Radarparameter sind wie in Tabelle 2.1 gewählt. Entsprechend der Anzahl an Rampen gibt es 64 Geschwindigkeitszellen. In Entfernungsrichtung werden nur die ersten 64 von 2048 Entfernungszellen betrachtet, da eine maximale Entfernung von 2,2 m für die Anwendung ausreichend ist. Im Bereich größer als 0,7 m können sogenannte Geisterziele auftreten, die durch Mehrwegeausbreitung entstehen. Dabei handelt es sich im eigentlichen Sinne um ein Störsignal. Für die Klassifikation mittels eines neuronalen Netzwerks können diese Informationen dennoch hilfreich sein, falls die Geisterziele charakteristisch für eine bestimmte Klasse sind. Im Rahmen dieser Arbeit wird die RDM als eine Art Bild mit 64x64 Pixel betrachtet. Das neuronale Netzwerk weiß nicht, dass das Bild eine Entfernungs- und Geschwindigkeitsmessungen darstellt.

Die Farbkamera und Wärmebildkamera sind neben dem Radarsensor in 0,7 m Höhe montiert. Die Bilder entsprechen daher einer Draufsicht. Ein Beispiel ist in Abbildung 3.2 gegeben. Die Bilder werden jeweils so zugeschnitten, dass das 2x2 Kochfeld als Ganzes abgebildet wird. Die Auflösung wird entsprechend der Radarbilder auf 64x64 Pixel reduziert. Die beiden Kamerabilder stimmen nicht pixelweise überein, da die Kameras nebeneinander montiert sind und



(a) Kreis-Geste (b) Fingerreiben (c) Handneigung (d) Handbewegung

Abbildung 3.3: (a)-(c) Beispielbilder der definierten Gesten, die zur Steuerung der Kochanwendung verwendet werden. (d) Typische Bewegungen bei Kochvorgängen wie Umrühren, die keine Geste zur Steuerung darstellen. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

unterschiedliche Objektive (Sichtfeld, Vergrößerung) und native Auflösungen verwenden.

3.1 Definierte Klassen

Wie bereits in Kapitel 1.2 beschrieben, bildet der Datensatz die Anwendung Kochsensor ab. Neben einer Kocherkennung werden die Kochposition auf einem 2x2 Kochfeld detektiert und verschiedene Gesten zur Steuerung klassifiziert. In Abbildung 3.3 sind Kamerabilder der Gesten dargestellt. Die drei Gesten

Abbildung 3.4: Beispielbilder verschiedener Kochpositionen auf dem 2x2 Kochfeld. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

zur Steuerung wurden auf Basis von [5] gewählt. Die Gesten sollen einerseits eine gute Klassifikationsgenauigkeit aufweisen und andererseits möglichst nicht mit anderen Bewegungen bei Kochvorgängen verwechselt werden. Typische Bewegungen bei Kochvorgängen sind etwa Umrühren, das Hinzufügen von Zutaten, Würzen oder das Umstellen von Töpfen. Aus diesem Grund werden die *Kreis-Geste*, *Fingerreiben* und *Handneigung* für die Steuerungsgesten ausgewählt. Eine weitere Klasse wird für *generelle (Hand)-Bewegungen* ohne bestimmte Geste definiert. Die *Kreis-Geste* und generelle *Handbewegungen* sind je nach ausführender Person eher schnellere Bewegungen mit großer Amplitude der ganzen Hand einschließlich des Arms. Bei den beiden Gesten *Fingerreiben* und *Handneigung* werden hingegen eher nur Finger beziehungsweise das Handgelenk bewegt. Die Bewegungen sind langsamer und mit kleiner Amplitude. Der Arm steht in der Regel still. Tendenziell sind diese Gesten schwieriger zu klassifizieren.

Zur Detektion der Kochpositionen sind vier Klassen definiert. Beispielbilder sind in Abbildung 3.4 dargestellt. Jede der Klassen repräsentiert eine der Kochpositionen *Hinten Links*, *Hinten Rechts*, *Vorne Links* und *Vorne Rechts*. Im Datensatz sind keine bis maximal zwei Kochpositionen gleichzeitig belegt. Die Multi-Label-Klassifikation bietet hier den großen Vorteil, dass alle möglichen Kombinationen von Kochpositionen durch die vier Klassen abgebildet werden können. Bei einem Multi-Class-Klassifikationsansatz wären für Szenen mit

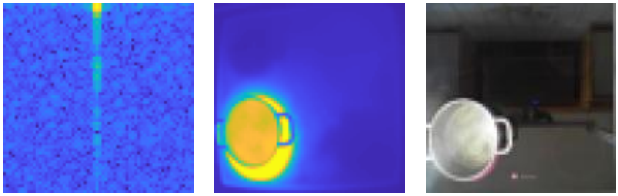


Abbildung 3.5: Beispielbilder aller drei Sensoren von der Klasse *Kochen*. Links: Range-Doppler-Matrix, Mitte: Wärmebild, Rechts: Kamerabild. Die drei Sensorbilder werden synchronisiert aufgezeichnet und haben individuelle Vor- und Nachteile. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

keinem bis maximal zwei Kochpositionen schon elf Klassen notwendig. Wie die Experimente in der Praxis gezeigt haben, ist durch die Multi-Label-Klassifikation sogar eine Szene mit drei oder vier belegten Kochpositionen detektierbar, obwohl der Trainingsdatensatz keine solche Szenen enthält.

Die Klasse *Kochen* nimmt eine besondere Rolle in der Anwendung ein, da der Überkochschutz maßgeblich auf dieser Klasse basiert. In Abbildung 3.5 sind Beispielbilder der Klasse *Kochen* aller drei Sensoren dargestellt. In der RDM ist das Kochen kaum ersichtlich. Jedoch treten für das Kochen charakteristische Geschwindigkeiten in der Entfernung des Füllstands auf. Diese sind typischerweise kleiner als 1 m/s. Im Gegensatz zur ruhenden Wasseroberfläche nimmt die RCS bei sprudelnd kochender Oberfläche ab. Diese Merkmale sind für eine Klassifikation mit einem neuronalen Netzwerk ausreichend. Im Wärmebild ist das Kochen mittels der direkten Temperaturmessung am einfachsten zu erkennen. Durch die Begrenzung des Temperaturmessbereich auf 125°C ist das Cerankochfeld selbst in Sättigung. Der Kochtopf sowie das Kochgut bei etwa 100°C heben sich davon ab. Im Farbkamerabild äußert sich das Kochen maßgeblich durch Wasserdampf. Wasserdampf tritt allerdings schon weit vor dem sprudelnden Kochen auf und ist kein alleiniger Indikator für das Kochen. Mit zunehmender

Zeit und keiner ausreichender Ablüftung kann im Kamerabild ausschließlich Wasserdampf zu sehen sein.

Die Klasse *Kochen* ist so definiert, dass entweder die Klasse *Kochen* oder die Klasse *Gefüllter Topf* wahr ist. Streng genommen setzt ein Kochen auch eine gewisse Füllung im Topf voraus. Allerdings führt das Kochen typischerweise zu einer Veränderung der Füllstandsmessung, da durch die zufälligen Oberflächenbewegungen die Radarwellen zufällig gestreut werden. Dies führt zu einem sich ständig ändernden Füllstand und einem abnehmenden RCS, welcher die Messung zusätzlich erschwert. Diese Eigenschaft führt dazu, dass die Klassen nicht unabhängig voneinander sind. Die Klasse *Kochen* würde oft mit der Klasse *Gefüllter Topf* verwechselt werden. Beziehungsweise ein Netzwerk würde einen nur gefüllten Topf fälschlicherweise meist auch mit *Kochen* klassifizieren. Durch die sich gegenseitig ausschließende Definition der Klassen ist der Übergang von *Gefüllter Topf* zu *Kochen* deutlicher abgegrenzt und Verwechslungen werden vermieden. In der Praxis führt dies zu einem besserem und weniger fehleranfälligem Überkochschutz. Mit folgender Definition kann immer nur eine der beiden Klassen gleichzeitig wahr sein:

- *Gefüllter Topf* ist wahr, wenn ein Topf gefüllt ist und der Topfinhalt nicht kocht.
- *Kochen* ist wahr, wenn ein Topf gefüllt ist und der Topfinhalt kocht.

Aufgrund der Multi-Label-Klassifikation kann eine Stichprobe keine bis mehrere wahre Klassen aufweisen. Die 57.600 Stichproben weisen insgesamt 110.000 wahre Klassen auf. Pro Stichprobe sind damit durchschnittlich 1,9 wahre Klassen enthalten. Die Verteilung der Klassen ist in Abbildung 3.6 dargestellt. Die Anzahl an Gesten ist verhältnismäßig gering, da die Gesten wesentlicher aufwändiger aufzuzeichnen sind. Zusätzlich sind bei den Gesten teilweise Kochpositionen belegt und ein Kochvorgang im Hintergrund vorhanden. Ebenso setzt die Klasse *Kochen* mindestens eine belegte Kochposition voraus. Die Klassen der Kochpositionen sind daher verhältnismäßig überrepräsentiert. Aus diesen Gründen wird bei der Bewertung der Leistungsfähig des Netzwerks jede Klasse als separater binärer Klassifikator betrachtet. Der Klassifikationsfehler wird für jeden Klassifikator, beziehungsweise jede Klasse, separat angegeben. Die genannten Beispiele zeigen, dass die definierten Klassen nicht vollständig unabhängig voneinander sind, da eine Klasse eine andere voraussetzen kann.

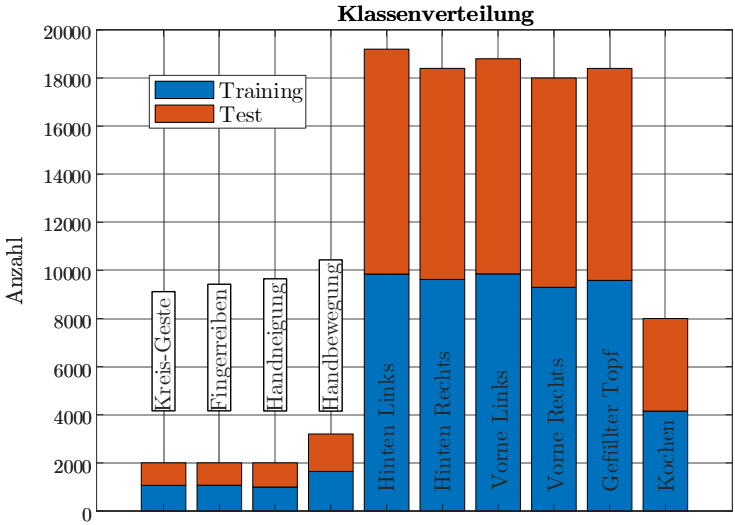


Abbildung 3.6: Verteilung der Klassen im Datensatz. Eine einzelne Stichprobe kann mehrere wahre Klassen aufweisen. Die Gesten sind verhältnismäßig unterrepräsentiert, da sie aufwändiger aufzuzeichnen sind. Eine Klasse kann eine andere voraussetzen. Beispielsweise erfordert *Kochen* mindestens eine belegte Kochposition. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

3.2 Ground Truth und Fehlerberechnung (Loss)

Für die Multi-Label-Klassifikation mit zehn Klassen entspricht die grundlegende Wahrheit (*Ground Truth*) einem Vektor der Länge zehn. Jede Stelle im Vektor entspricht einer der definierten Klassen. Der Vektor wird mit Nullen initialisiert. Die für die Stichprobe zutreffenden Klassen werden entsprechend zu 1 gesetzt. Dies wird auch *Hot-Encoding* genannt. Ein solcher Vektor y ist in Abbildung 3.7 dargestellt.

Entsprechend dem *Ground Truth*-Vektor y muss auch die Ausgabe des neuronalen Netzwerks $\hat{y}$ ein Vektor der Länge zehn sein. Der letzte Layer ist daher ein FC-Layer mit zehn Neuronen. Als Aktivierungsfunktion wird eine *Clipped* ReLU verwendet. Als Begrenzung wird 1 gewählt, sodass der Wertebereich der Ausgabe je Neuron zwischen 0 und 1 liegt. Der Wert kann als Wahrscheinlichkeit, dass diese Klasse zutrifft, interpretiert werden. Für die binäre Entscheidung wird ein Schwellwert (*threshold*) von 0,5 gewählt.

Der Fehler (*loss*) wird mit einem benutzerdefinierten Regressions-Layer (siehe 2.2.6) berechnet. Dies ermöglicht die Fehlerberechnung mittels einer beliebigen Funktion. Drei einfache Beispiele sind in Abbildung 3.7 dargestellt. Der Fehlerbetrag ist die Differenz von Netzwerkausgang und *Ground Truth* als Betrag. Beim quadratischen Fehler werden größere Abweichungen stärker gewichtet als kleinere. Bei der Schwellwertmethode werden nur Fehlerbeträge über einer festgelegten Schwelle berücksichtigt und zu 1 gesetzt. Der Fehler ist sozusagen binär. Dieses Vorgehen ist eher zur Bewertung der Leistungsfähigkeit (Klassifikationsgenauigkeit) des Netzwerks üblich (siehe nachfolgendes Kapitel) und wird nicht als Fehler für das Training (Backpropagation) verwendet. Die Frage, welche Fehlerfunktion die besten Ergebnisse liefert, kann pauschal nicht beantwortet werden. Dies hängt vom Datensatz, dem neuronalen Netzwerk und dem Trainingsalgorithmus ab.

Damit der Fehler unabhängig von der gewählten Batch-Größe n ist, wird der Fehler über den Batch gemittelt. Zusätzlich wird über die Anzahl der Klassen K gemittelt. Dies überführt die obigen Fehler zu den statistischen Fehlern Mean Absoulte Error (MAE), MSE und Root Mean Square Error (RMSE) [84]. In

	Kreis-Geste	Fingerreiben	Handneigung	Handbewegung	Hinten Links	Hinten Rechts	Vorne Links	Vorne Rechts	Gefüllter Topf	Kochen	Summe	
Ground Truth	0	1	0	0	0	1	0	0	1	0	3	y
Netzwerk Ausgang	0,1	1	0,3	0,1	0,2	0,9	0	0,3	0,4	0,2	3,5	ŷ
Fehler Betrag	0,1	0	0,3	0,1	0,2	0,1	0	0,3	0,6	0,2	1,9	|ŷ-y|
Fehler Quadrat	0,01	0	0,09	0,01	0,04	0,01	0	0,09	0,36	0,04	0,65	$(\hat{y}-y)^2$
Schwellw. 0,5 Fehler	0	0	0	0	0	0	0	0	1	0	1	|ŷ-y|>0,5

Abbildung 3.7: Ground-Truth-Vektor und Ausgabevektor des Netzwerks für zehn Klassen bei der Multi-Label-Klassifikation. Typische Fehlerberechnung sind der Fehlerbetrag, der quadratische Fehler oder ein Schwellwert-Fehler.

dieser Arbeit wird hauptsächlich der

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{K}(\hat{y}_{i,j} - y_{i,j})^2}{nK}} \tag{3.1}$$

und verwandte Formen verwendet. Die genaue Funktion wird in den entsprechenden Kapiteln angegeben. Ein Beispiel ist der Half Mean Square Error (HMSE)

$$HMSE = \frac{\sum_{i=1}^{n}\sum_{j=1}^{K}(\hat{y}_{i,j} - y_{i,j})^2}{2n}, \tag{3.2}$$

der speziell in dieser Arbeit bessere Ergebnisse als der RMSE liefert. Entsprechende Untersuchungen werden in Kapitel 5.2 durchgeführt. Der Hintergrund ist die Eigenschaft des *Ground Truth*-Vektors y, durchschnittlich zwei wahre

Klassen zu enthalten. Aus diesem Grund wird der MSE nur halbiert statt durch die Anzahl der Klassen K zu teilen. Auf die Wurzel wird verzichtet.

3.3 Bewertung der Leistungsfähigkeit des Netzwerks

Eine Übersicht verschiedener Gütekriterien ist in [74] gegeben. Bei der Multi-Label-Klassifikation wird für jede Klasse separat die sogenannte Wahrheitsmatrix (Konfusionsmatrix) berechnet. Diese ist in Abbildung 3.8 für einen binären Klassifikator dargestellt. In dieser Arbeit entspricht jede Klasse einem binären Klassifikator. Die Wahrheitsmatrix enthält die Kennzahlen Richtig Positiv (RP), Richtig Negativ (RN), Falsch Positiv (FP) und Falsch Negativ (FN). Mit diesen Kennzahlen kann die Korrektklassifikationsrate (Genauigkeit, engl. accuracy)

$$\frac{RP + RN}{RP + RN + FP + FN} \tag{3.3}$$

berechnet werden. Diese wird oft in Prozent angegeben. Ebenso kann das Komplement in Form der Falschklassifikationsrate

$$\frac{FP + FN}{RP + RN + FP + FN} \tag{3.4}$$

berechnet werden. Korrektklassifikationsrate und Falschklassifikationsrate addieren sich zu 1 bzw. 100%. Die Summe $RP + RN + FP + FN$ im Nenner entspricht der Anzahl an Stichproben im Testdatensatz, da auch jedes Nicht-Vorkommen (Klasse Negativ) klassifiziert wird. Unter Berücksichtigung der Klassenverteilung in Abbildung 3.6 macht diese Bewertung speziell bei den Gesten wenig Sinn. Im Testdatensatz kommt eine Geste nur ungefähr 1.000 Mal vor. Dem stehen 27.600 Stichproben gegenüber. Würde das Netzwerke beispielsweise ausschließlich negativ klassifizieren und damit keine einzige Geste erkennen, wäre die Genauigkeit für eine Geste nach Gleichung 3.3 dennoch 96,4%.

Aus diesem Grund werden die Gütekriterien Sensitivität (auch Trefferquote, engl. recall)

$$\frac{RP}{RP + FN} \tag{3.5}$$

	Positiv klassifiziert	Negativ klassifiziert
Klasse Positiv	Richtig Positiv (RP)	Falsch Negativ (FN)
Klasse Negativ	Falsch Positiv (FP)	Richtig Negativ (RN)

Abbildung 3.8: Wahrheitsmatrix (Konfusionsmatrix) für einen binären Klassifikator mit den Kennzahlen RP, RN, FP und FN.

und positiver Vorhersagewert (auch Relevanz, engl. precision)

$$\frac{RP}{RP + FP} \tag{3.6}$$

betrachtet. Die Sensitivität gibt die Wahrscheinlichkeit an, dass eine tatsächlich vorkommende Klasse erkannt und klassifiziert wird. Der positive Vorhersagewert gibt die Wahrscheinlichkeit an, dass eine positiv klassifizierte Klasse auch tatsächlich vorhanden ist. Diese Gütekriterien beziehen sich nun auf die Summe tatsächlich im Datensatz vorkommenden Klassen $RP + FN$ beziehungsweise der vorkommenden FP. Der verhältnismäßig große Anteil der RN fällt weg. Die Gütekriterien je Klasse sind damit unabhängig von der Klassenverteilung und können miteinander verglichen werden.

Da die Gütekriterien bei den meisten Experimenten über 90% liegen, werden die entsprechenden Komplemente betrachtet. Diese sind in der Regel intuitiver vergleichbar. Das Komplement der Sensitivität ist die Falsch-Negativ-Rate

$$\frac{FN}{RP + FN}. \tag{3.7}$$

Das Komplement des positiven Vorhersagewerts ist die Falscherkennungsrate

$$\frac{FP}{RP + FP}. \tag{3.8}$$

Zur besseren Übersichtlichkeit bei der Darstellung der Ergebnisse werden Gleichung 3.7 und 3.8 in Form der Fehlerrate

$$\frac{FP + FN}{RP + FP + FN} \tag{3.9}$$

zusammengefasst. In dieser Arbeit wird die Fehlerrate für jede Klasse separat angegeben. Für die durchschnittliche Fehlerrate werden die RP, FP und FN über K Klassen aufsummiert und die Fehlerrate nach Gleichung 3.9 berechnet.

In MATLAB® ist die Standardleistungskennzahl für eine Regression der RMSE. Da die Multi-Label-Klassifikation in MATLAB® unter diese Kategorie fällt, kann während des Trainings der RMSE ausgegeben werden. Dies ist zwar keine qualitative Bewertung der Leistungsfähigkeit für die Anwendung, aber sie gibt einen ersten Anhaltspunkt für den Erfolg des Trainings schon während des Trainings.

3.4 Zusammenfassung

In diesem Kapitel wurde der Datensatz und die zehn definierten Klassen der Kochanwendung vorgestellt. Der Datensatz mit insgesamt 57.600 Stichproben wurde an einem Herd mit 2x2 Kochfeldern aufgenommen. Neben den RDM-Radardaten wurden synchronisierte Farb- und Wärmebilder aufgenommen. Der Datensatz wurde sequenzweise in 52% Trainings- und 48% Testdaten aufgeteilt. Die Gesten wurden von insgesamt vier verschiedenen Personen aufgezeichnet. Die Gesten wurden so gewählt, dass einerseits eine gute Klassifikationsgenauigkeit gegeben ist und andererseits möglichst keine Verwechslungen mit anderen Bewegungen bei Kochvorgängen auftreten. Die Klassenverteilung zeigt eine Unterrepräsentation der Gesten, da diese wesentlicher aufwändiger aufzuzeichnen sind als die anderen Klassen.

Da in der Anwendung mehrere Klassen gleichzeitig wahr sein können, wird eine Multi-Label-Klassifikation verwendet. Dabei kann jede Klasse als separater binärer Klassifikator betrachtet werden. Die Fehlerberechnung erfolgt mithilfe des HMSE (Gleichung 3.2). Für jede Klasse wird in den Auswertungen die Fehlerrate (Gleichung 3.9) angegeben, die den verhältnismäßig großen Anteil der RN ausblendet. Die Gütekriterien je Klasse sind damit unabhängig von der Klassenverteilung und können miteinander verglichen werden. Für die durch-

schnittliche Fehlerrate werden die RP, FP und FN über K Klassen aufsummiert und die Fehlerrate berechnet.

4 Range-Doppler-Matrix-Klassifikation

Die RDM-Klassifikation erfolgt zunächst anhand einzelner Messungen. Diese sogenannte Einzelbild-Klassifikation erfolgt mittels eines CNN. Die Ergebnisse wurden bereits am Beispiel eines 77 GHz Radars für eine Kocherkennung veröffentlicht [37]. Für die Dissertation wird der Datensatz des 120 GHz Radars inklusive Gesten- und Kochpositionsbestimmung verwendet. Mit diesen Erweiterungen sind die Anforderungen an das Netzwerk wesentlich höher.

In einem nächsten Schritt werden die dynamischen Gesten und der dynamische Kochprozess anhand einer Sequenz von Messungen klassifiziert. Dazu wird ein rekurrentes Netzwerk mit LSTM-Layer verwendet. Die ursprünglichen Experimente wurden ebenfalls am Beispiel eines 77 GHz Radars durchgeführt und in [38] veröffentlicht. Neben einer generell besseren Korrektklassifikationsrate können dynamische Übergänge, wie sie etwa von Nicht-Kochen zu Kochen auftreten, besser und vor allem stabiler klassifiziert werden.

4.1 Einzelbild-Klassifikation

Das Ziel der Einzelbild-Klassifikation ist ein gutes Verhältnis zwischen der Genauigkeit der Klassifikation und der Anzahl an Parametern des Netzwerks. Der Hintergrund ist eine Implementierung der Netzwerk-Inferenz auf einem eingebetteten System mit beschränkter Rechenkapazität. Bei einfachen Netzwerken nimmt die Genauigkeit typischerweise mit der Anzahl der Parameter, wie in Abbildung 4.1 dargestellt, zu. Dieser Verlauf ist nicht linear. Ab einem gewissen Punkt kann die Genauigkeit trotz zunehmender Parameter kaum verbessert werden. Weiterhin ist es möglich, dass ein Netzwerk mit sehr vielen Parametern nicht mehr effizient trainiert werden kann oder dass es zu einer Überanpassung kommt (siehe Kapitel 2.2.4). In einem solchen Fall ist die Genauigkeit trotz mehr Parametern schlechter. Diese Probleme können teilweise mithilfe spezieller Layer wie Dropout oder Batch Normalization gelöst werden. Wie in Kapitel 2.2.1 beschrieben, haben Conv-Layer generell weniger Parameter als etwa FC-Layer.

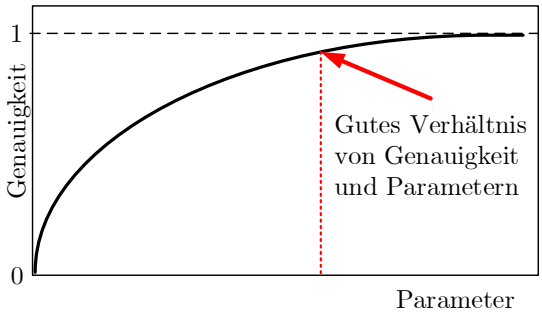


Abbildung 4.1: Prinzipielles Verhältnis der Genauigkeit der Klassifikation zu der Anzahl an Parametern eines neuronalen Netzwerks. Aufgrund des nichtlinearen Zusammenhangs und der Gefahr einer Überanpassung sind mehr Parameter nicht zwangsläufig von Vorteil. Vielmehr ist es wichtig, ein optimales Verhältnis zu finden.

Deshalb sind FCN in dieser Hinsicht sehr attraktiv. Die Anzahl der Parameter kann zusätzlich mit einem Pooling-Layer reduziert werden.

Die genannten Punkte werden im Rahmen dieses Kapitels untersucht. Der Einfluss von Pooling, Dropout und Batch Normalization Layer wird experimentell für die Kochanwendung ermittelt. Ein FCN wird in Kapitel 5 betrachtet und verglichen.

4.1.1 Netzwerk-Architektur

Das Netzwerk in [37] hat eine Multi-Class-Klassifikation mit den Klassen Kochen, Handbewegung, kein Topf, leerer Topf, gefüllter Topf und Umrühren verwendet. Es wurde nur ein einzelnes Kochfeld betrachtet. Für diese Arbeit wird die Multi-Label-Klassifikation, wie in Kapitel 3.1 beschrieben, verwendet. Die alten Klassen aus [37] werden damit abgedeckt und um die Kochpositionserkennung und spezifische Gesten erweitert.

Die Netzwerk-Architektur ist in Abbildung 4.2 dargestellt. Die Conv-Layer sind so gewählt, dass ein rezeptives Feld von 11x11 in der RDM abgedeckt

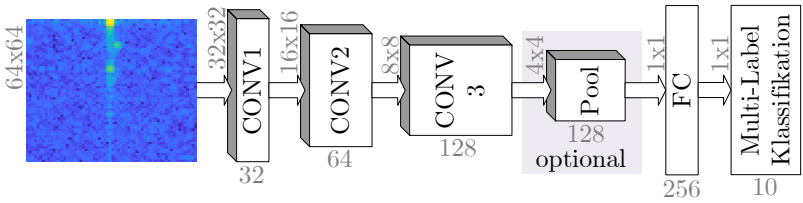


Abbildung 4.2: Architektur des Einzelbild-Netzwerks. Links neben den Layern ist jeweils die räumliche Größe angegeben. Die Tiefe, beziehungsweise die Anzahl der Filter, ist jeweils darunter dargestellt. Der Pooling-Layer ist in dieser Architektur optional.

wird. Typische Ziele in der RDM sind etwa über 5x5 Zellen ausgedehnt (siehe Abbildung 3.1). Entsprechend dem CFAR-Verfahren werden einige weitere Zellen um das Ziel berücksichtigt. Ein rezeptives Feld von 11x11 kann erreicht werden, indem mehrere 3x3 Conv-Layer (mit $F = 3$, $S = 2$) direkt hintereinander angeordnet werden. [85] zeigt, dass dadurch die Anzahl der Parameter und der Rechenaufwand im Gegensatz zu einem einzigen Conv-Layer mit größerem Filter reduziert werden kann. Zusätzlich werden bessere Ergebnisse erreicht. Je 3x3 Conv-Layer wird die räumliche Größe halbiert und die Tiefe verdoppelt, indem die Anzahl der Filter verdoppelt wird.

Der Pooling-Layer ist optional und reduziert die räumliche Größe um den Faktor 2^2. Dem FC-Layer folgt die Multi-Label-Klassifikation. Die Conv- und FC-Layer verwenden die ReLU Aktivierungsfunktion. Falls Batch Normalization verwendet wird, erfolgt die Normalisierung jeweils vor der Aktivierungsfunktion. Dropout wird bei Verwendung in Conv2, Conv3 und FC mit 50% deaktivierten Neuronen eingesetzt. In Conv1 ist Dropout nicht sinnvoll, da sonst zufällige Teile des RDM Eingangsbildes ignoriert werden.

Diese Architektur hat sich entsprechend Abbildung 4.1 als gutes Verhältnis von möglicher Genauigkeit zu der Anzahl an Parametern herausgestellt. Zusätzliche Conv- oder FC-Layer haben die Genauigkeit in [37] nicht maßgeblich verbessert. Für die nun erweitere Aufgabenstellung mit Gesten- und Kochpositionserkennung sind zusätzliche Layer vorteilhaft. Ein solches Netzwerk wird in Kapitel 5 gezeigt.

Die Anzahl an Parametern beträgt 2.192.650 ohne den Pooling-Layer. Mit dem räumlichen Downsampling des Pooling-Layers sind es 619.786 Parameter. Der Pooling-Layer mit einer Filtergröße von zwei reduziert die Parameter um etwa Faktor vier (räumliches Downsamling 2^2). Der Speicherbedarf bei der Inferenz liegt bei 6,2 MB beziehungsweise 1,4 MB. Das sehr kompakte Netzwerk ist damit auch auf einem eingebetteten System ausreichend schnell für 30 Bilder pro Sekunde ausführbar.

4.1.2 Training

Das Netzwerk zur Einzelbild-Klassifikation wird von Grund auf trainiert, da kein vortrainiertes Netzwerk vorliegt. Die Gewichtungen der Layer werden mit der Xavier-Glorot-Methode [86] initialisiert. Im Wesentlichen handelt es sich um eine Normalverteilung, deren Grenzen vom Layer-Typ und den jeweiligen Hyperparametern abhängen. Als Trainingsalgorithmus wird Adam verwendet. Als gute Lernrate nach Abbildung 2.11 hat sich 10^{-5} in Kombination mit einer Batch-Größe von 256 herausgestellt. Das Training läuft für 500 Epochen. Der Trainingsdatensatz wird nach jeder Epoche durchmischt, sodass sich jeder Batch jedes Mal aus anderen Stichproben zusammensetzt. Als Fehlerfunktion wird der HMSE nach Gleichung 3.2 verwendet.

Die Trainingsverläufe für Experimente mit und ohne Batch Normalization und Dropout sind in Abbildung 4.3 dargestellt. Die Ergebnisse sind in Tabelle 4.1 aufgeführt. Der Trainingsverlauf ohne Batch Normalization und Dropout ist am wenigsten gleichmäßig, da kein Layer zur Regulierung des Trainings verwendet wird. Eine Generalisierungslücke nach Abbildung 2.9 ist vorhanden und deutet auf eine Überanpassung hin. Mit dem Einsatz von Batch Normalization wird der Trainingsverlauf gleichmäßiger. Die Ergebnisse werden leicht besser. Die Generalisierungslücke kann allerdings nicht geschlossen werden.

Mit dem Einsatz von Dropout und ohne Batch Normalization ist der Trainingsverlauf wieder weniger gleichmäßig. Die Generalisierungslücke scheint genau umgekehrt zu sein. Der Trainingsfehler ist nun größer als der Testfehler. Dies ist typisch für den Einsatz von Dropout. Bei der Berechnung des Trainingsfehlers ist durch Dropout ein Teil der Neuronen deaktiviert. Der Fehler nimmt zu. Für den Testfehler wird hingegen das gesamte Netzwerk ohne deaktivierte Neuronen verwendet. Der Fehler ist typischerweise kleiner. Mit Dropout sind

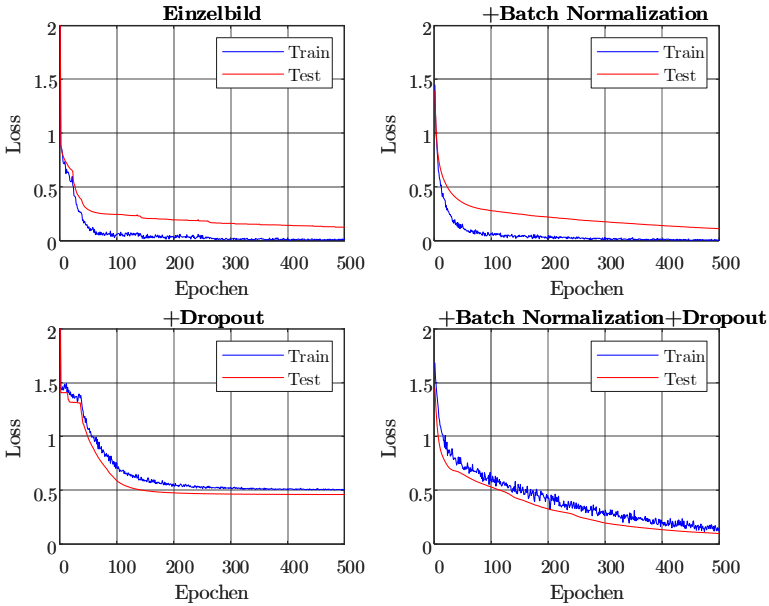


Abbildung 4.3: Verlauf der Trainings mit verschiedenen Trainingsbedingungen wie teilweise zusätzlicher Batch Normalization und/oder Dropout. Mit Batch Normalization ist das Training aufgrund der Regulierung gleichmäßiger. Die Generalisierungslücke kann mittels Dropout reduziert werden.

die Kurven also nur bedingt vergleichbar. Ob der Einsatz von Dropout sinnvoll ist, kann schließlich am Ergebnis erkannt werden. In diesem Fall führt Dropout zu einer schlechteren Genauigkeit, da das Training ab einem Fehler von etwa 0,5 stagniert.

Als letztes wird Batch Normalization in Kombination mit Dropout verwendet. Die Effekte beider Regulierungen sind im Trainingsverlauf erkennbar. Die Batch Normalization führt zu einem gleichmäßigem Trainingsverlauf. Dropout in Verbindung mit Batch Normalization schließt weitgehend die Generalisierungslücke. Auch hier ist Dropout-typisch der Trainingsfehler etwas größer als der Testfehler. Die Kombination aus beiden Regulierungsschichten führt letztendlich zu dem besten Ergebnis (siehe Tabelle 4.1).

4.1.3 Ergebnisse

Die Ergebnisse der Einzelbild-Klassifikation sind in Tabelle 4.1 für das Netzwerk ohne Pooling-Layer und in Tabelle 4.2 für das Netzwerk mit Pooling-Layer dargestellt. Die Ergebnisse ohne Pooling-Layer sind generell besser. Bereits in [37] lautet das Ergebnis, dass der Maximalwert-Pooling-Layer für RDM Daten keinen Vorteil außer der Parameterreduktion bringt. Dieses Ergebnis wird hier bestätigt. Der Grund liegt sehr wahrscheinlich bei der bereits sehr geringen räumlichen Größe von 64x64 der RDM. Werden stattdessen beispielsweise Kamerabilder betrachtet, sind deutlich höhere Auflösungen von mehreren Megapixel üblich. Da ist ein Downsampling sinnvoll und notwendig.

Bei dem Netzwerk ohne Maximalwert-Pooling-Layer wird der kleinste durchschnittliche Fehler über alle Klassen mit der Kombination aus Batch Normalization und Dropout erreicht. Besonders die Klassen *Kochen* und *Gefüllter Topf* werden verhältnismäßig gut klassifiziert. Das Netzwerk wurde ursprünglich für diese Kocherkennung entworfen und funktioniert entsprechend gut. Bei den Gesten werden die deutlicheren Bewegungen aus Hand und Arm, wie in der Kreis-Geste, besser klassifiziert als die feinen Bewegungen etwa beim *Fingerreiben*. Die Kochfeldbelegung funktioniert *Vorne Links* am zuverlässigsten. Dies kann auf die Ausrichtung des Radars zurückgeführt werden, welcher absichtlich etwas schräg auf die Kochfelder in Richtung vorne links ausgerichtet ist.

Die Ergebnisse mit dem Maximalwert-Pooling-Layer sind im direkten Vergleich jeweils schlechter. Ohne Regulierung werden manche Klassen gar nicht

Tabelle 4.1: Vergleich der Ergebnisse der Einzelbild-Klassifikation teilweise mit Batch Normalization (+Batch) und Dropout (+Drop). Das beste Ergebnis wird ohne Pooling-Layer und dem kombinierten Einsatz von Batch Normalization und Dropout erreicht.

	Einzelbild	+Batch	+Drop	+Batch+Drop
RMSE	0,4994	0,4611	0,6450	0,3841
∅-Fehler	15,71%	13,76%	25,25%	9,52%
Kreis-Geste	14,57%	11,03%	44,94%	7,37%
Fingerreiben	26,96%	26,74%	47,17%	22,67%
Handneigung	35,68%	36,33%	91,81%	28,68%
Handbewegung	13,67%	7,91%	34,75%	5,95%
Hinten Links	22,13%	18,72%	34,72%	14,89%
Hinten Rechts	22,81%	23,02%	37,41%	15,36%
Vorne Links	6,88%	6,28%	10,76%	4,00%
Vorne Rechts	18,02%	15,82%	25,35%	9,41%
Gefüllter Topf	7,13%	4,66%	7,56%	2,98%
Kochen	8,44%	4,62%	8,22%	3,10%

Tabelle 4.2: Vergleich der Ergebnisse mit zusätzlichem Maximalwert-Pooling-Layer.

	MaxPooling	+Batch	+Drop	+Batch+Drop
RMSE	0,7101	0,5549	1,3841	0,7740
∅-Fehler	26,67%	19,05%	100,00%	32,37%
Kreis-Geste	100,00%	18,77%	100,00%	100,00%
Fingerreiben	100,00%	32,15%	100,00%	100,00%
Handneigung	100,00%	44,13%	100,00%	100,00%
Handbewegung	5,84%	15,03%	100,00%	100,00%
Hinten Links	10,49%	26,33%	100,00%	34,52%
Hinten Rechts	6,74%	30,05%	100,00%	49,05%
Vorne Links	3,56%	8,93%	100,00%	13,78%
Vorne Rechts	6,27%	20,46%	100,00%	26,32%
Gefüllter Topf	100,00%	7,94%	100,00%	14,91%
Kochen	3,28%	8,68%	100,00%	14,79%

erkannt. Das betrifft vor allem die Gesten, die nur feine Bewegungen aufweisen. Diese Details können durch das Pooling verloren gehen. Die Kochfeldbelegung wird erstaunlich gut erkannt. Dies kann mit der prinzipiellen Ähnlichkeit des Maximalwert-Pooling nach Abbildung 2.8 und der 2x2 Kochfelderkennung begründet werden. In beiden Fällen wird aus einem 2x2 Raster der Maximalwert verwendet und die restlichen Werte verworfen. Batch Normalization verbessert das Ergebnis insgesamt. Die Ergebnisse einzelner Klassen werden aber auch durchaus verschlechtert, etwa bei der Kochfeldbelegung. Durch die Normalisierung werden die Unterschiede zwischen den Klassen geringer, da Ausreißer unterdrückt werden. Dies betrifft gleichermaßen Ausreißer im negativen als auch im positiven Sinne.

Die Kombination aus Maximalwert-Pooling-Layer und Dropout führt zu keinem erfolgreichen Training. Daran haben auch andere Hyperparameter wie Lernrate oder Batch-Größe, sowie Wiederholungen des Experiments nichts geändert. Bei näherer Betrachtung bestehen die Ausgabe des Netzwerks und die Aktivierungen einzelner Layer ausschließlich aus Nullen. In der Literatur wird in einem solchen Fall auch von einem *toten Netzwerk* oder dem *Dying ReLU*-Problem gesprochen [87]. Die Ursachen liegen meist in einer ungünstigen Kombination aus Netzwerk-Architektur, Aktivierungsfunktion und Initialisierung. Nach wenigen Trainingsiterationen stirbt das Netzwerk und gibt ausschließlich Nullen aus. Die Fehlerrate nach Gleichung 3.9 beträgt entsprechend 100%. Hier sei noch einmal angemerkt, dass die Korrektklassifikationsrate nach Gleichung 3.3 aufgrund der vielen RN dennoch etwa 81% betragen würde, obwohl nur Nullen ausgegeben werden. Dies unterstreicht, warum die Korrektklassifikationsrate wenig Aussagekraft für die Anwendung in dieser Arbeit hat.

Der negative Effekt des Dropout-Layers in Verbindung mit dem Maximalwert-Pooling-Layer setzt sich auch mit zusätzlicher Batch Normalization fort. Die Gesten haben eine Fehlerrate von 100% und auch die übrigen Klassen zeigen eine höhere Fehlerrate. Zusammenfassend sollte kein Dropout mit dem optionalen Maximalwert-Pooling-Layer verwendet werden.

Das beste Ergebnis mit einer durchschnittlichen Fehlerrate von 9,52% wird ohne Maximalwert-Pooling-Layer und dem kombinierten Einsatz von Batch Normalization und Dropout erreicht.

4.2 Sequenz-Klassifikation

Eine Weiterentwicklung der Einzelbild-Klassifikation ist die sogenannte Sequenz-Klassifikation. Dabei wird eine zeitliche Sequenz von je 20 RDM betrachtet. Die Idee ist, dynamische Prozesse, wie das Kochen oder die Gesten, mit einer Sequenz mehrerer Messungen abzubilden. Spezielle Netzwerke mit internen Speichermöglichkeiten können die Dynamik dieser Prozesse lernen, indem Informationen aus vorherigen RDM im Netzwerk gespeichert und für die aktuelle Klassifikation berücksichtigt werden. Das im Netzwerk interne Speichern erfolgt mittels sogenannter LSTM-Layer (siehe Kapitel 2.2.2). Die bereits veröffentlichten Experimente in [38] befassen sich unter anderem mit dem manuellen Löschen des internen LSTM-Speichers zu verschiedenen Trainingszeitpunkten. Untersucht wird das Löschen nach jedem Batch, jeder Epoche und kein Löschen. Das Ergebnis ist, dass der interne Speicher nicht gelöscht werden sollte. Vielmehr soll das Netzwerk selbstständig lernen, welche Informationen aus dem internen Speicher gelöscht werden. Dies ist durch das Vergessen-Gatter [65] möglich. In dieser Dissertation wird das manuelle Löschen nicht verwendet und nicht nochmals untersucht.

Für das Lernen der dynamischen Prozesse ist es wichtig, dass die chronologisch richtige Reihenfolge einer Sequenz erhalten bleibt. In [5] werden die ausgewählten Gesten nach etwa zehn Einzelbildern erkannt. Das entspricht bei der Framerate von 40 Hz einem Beobachtungszeitraum von 250 ms bis zur Erkennung einer Geste. Mit längerer Beobachtungszeit nimmt die Sicherheit zu.

Für das Training in dieser Dissertation wird eine Sequenzlänge von 20 gewählt. Bei der maximalen Framerate von 30 Hz entspricht das einer Beobachtungszeit von 667 ms. Neben der Gestenklassifikation ist die längere Beobachtungszeit auch für die Kocherkennung von Vorteil. Ein Übergang von Nicht-Kochen zu Kochen nimmt Zeit in Anspruch. Für die Inferenz in der Anwendung spielt die Sequenzlänge keine Rolle, beziehungsweise beträgt 1. Jedes Einzelbild wird direkt in Echtzeit vom Netzwerk verarbeitet, um Verzögerungen bei der Klassifikation zu vermeiden. Aus diesem Grund sind auch keine bilateralen LSTM-Layer möglich [88]. Bilateral heißt, dass die Sequenz in beide Richtungen durchlaufen wird. Bei der Klassifikation werden also auch zukünftige Daten berücksichtigt. Für einen gespeicherten Datensatz ist dieser Ansatz möglich und sollte theoretisch auch bessere Ergebnisse liefern. In der Anwendung sind allerdings

keine zukünftigen Daten vorhanden. In dieser Arbeit wird daher der unilaterale LSTM-Layer verwendet, der ausschließlich vergangene Daten zur Klassifikation berücksichtigt.

4.2.1 Netzwerk-Architektur

Das Netzwerk ist eine Mischung aus dem bisherigen Einzelbild-CNN und einem LSTM-Layer. Der CNN-Teil (siehe Abbildung 4.2) dient der Merkmalsextraktion aus der RDM. Dieser Teil wird unabhängig voneinander für jeden Zeitschritt durchlaufen. Die trainierten Gewichtungen sind jeweils dieselben. Anstelle der direkten Multi-Label-Klassifikation nach dem FC-Layer, folgt der rekurrente LSTM-Layer. Dieser führt schließlich die Multi-Label-Klassifikation durch. Dabei basiert die Klassifikation, wie in Kapitel 2.2.2 beschrieben, auf der aktuellen und den vergangenen RDM. Die Informationen vergangener RDM werden im internen Speicher des LSTM-Layers gehalten. Die Netzwerk-Architektur ist in Abbildung 4.4 dargestellt.

Jeder Zeitschritt entspricht einer neuen RDM-Messung. Der Speicher wird in jedem Zeitschritt mit aktuellen Merkmalen des CNNs aktualisiert und zum nächsten Zeitschritt weitergegeben. Die Multi-Label-Klassifikation erfolgt für jeden Zeitschritt. Dies wird in MATLAB® Sequenz-Ausgang genannt. Es wäre auch möglich, nur am Ende einer Sequenz eine Ausgabe zu erzeugen. Dies würde allerdings eine Verzögerung bei der Klassifikation in der Anwendung bedeuten.

Für den LSTM-Layer muss eine Anzahl an Einheiten (englisch *units*) als Hyperparameter angegeben werden. Diese Anzahl entspricht jeweils der Anzahl an Neuronen der internen Layer (Vergleich Abbildung 2.7). Der interne Speicher ist ein Vektor mit der Länge der Anzahl der Einheiten. Der Ausgabevektor weist dieselbe Länge auf. Dies ermöglicht die punktweisen Operationen in der LSTM-Zelle. Bei den Experimenten haben sich 128 Einheiten für den LSTM-Layer als optimal herausgestellt. Dies entspricht etwa der Mitte zwischen dem vorherigen FC-Layer mit 256 Neuronen und den zehn Neuronen zur Multi-Label-Klassifikation. Weitere Layer zwischen LSTM und Klassifikation haben in den Experimenten keinen Vorteil gebracht, sondern die nur die Komplexität und die Anzahl an Parametern erhöht.

Für das Training dieses Netzwerks sind ebenfalls Layer zur Regulierung vorteilhaft. Neben Batch Normalization und Dropout in den CNN-Layern kann

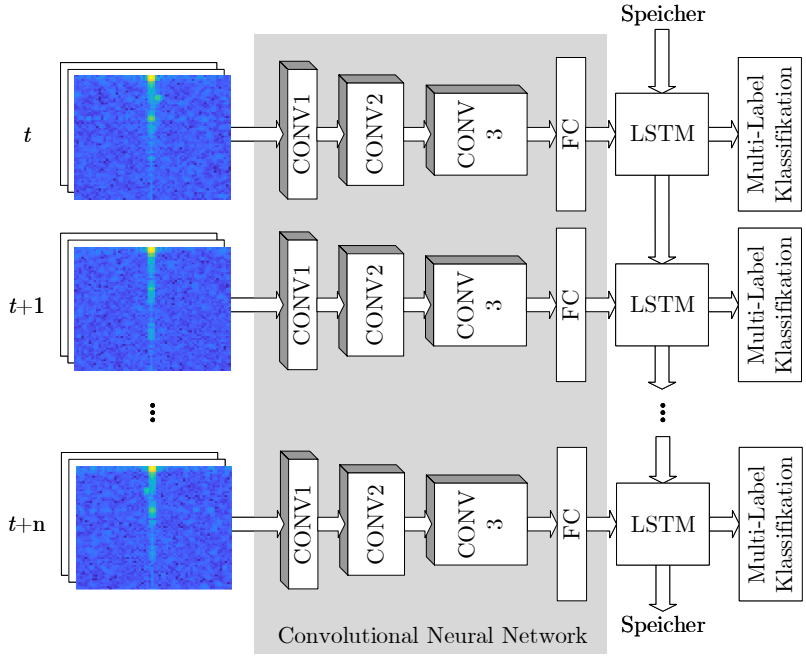


Abbildung 4.4: Architektur des CNN+LSTM Netzwerks. Das Einzelbild-CNN wird in jedem Zeitschritt wiederverwendet. Nach dieser Merkmalsextraktion folgt ein rekurrenter LSTM-Layer zur Multi-Label-Klassifikation. Die Klassifikation beruht damit nicht nur auf der aktuellen Messung, sondern zusätzlich auf gespeicherten Informationen vergangener Messungen. Abbildung nach [38], © 2019 EuMA.

auch eine spezielle Form von Dropout in dem LSTM-Layer verwendet werden. Das sogenannte rekurrente Dropout ist in [89] beschrieben und wird analog zum CNN-Teil mit 50% deaktivierten Neuronen verwendet.

Der CNN-Teil entspricht dem Netzwerk der Einzelbild-Klassifikation. Die in Kapitel 4.1 trainierten Netzwerke können nun als vortrainierte Netzwerke für ein *Transfer Learning* verwendet werden. Die Vorteile beim Training und die Ergebnisse werden nachfolgend vorgestellt.

4.2.2 Training

Im Vergleich zum Einzelbild-Training ist die Batch-Größe von 256 auf 64 reduziert. Pro Batch sind das dennoch mehr Bilder, da jede Sequenz 20 Einzelbilder enthält. Eine Batch-Größe von 128 übersteigt die 8 GB verfügbaren GPU-Speicher. Als passende Lernrate hat sich 10^{-5} herausgestellt. Das Training läuft für 1000 Epochen mit dem Trainingsalgorithmus Adam. Als Fehlerfunktion wird der HMSE nach Gleichung 3.2 verwendet. Alle Trainings verwenden die bewährte Kombination aus Batch Normalization und Dropout.

In Abbildung 4.5 ist der Trainingsverlauf mit und ohne *Transfer Learning* dargestellt. Ohne *Transfer Learning* wird das Netzwerk von Grund auf trainiert. Die Gewichtungen der Layer werden analog zur Einzelbild-Klassifikation mit der Xavier-Glorot-Methode [86] initialisiert. Beim *Transfer Learning* werden die bereits trainierten Layer der Einzelbild-Klassifikation verwendet. Die vier Layer sind in Abbildung 4.4 grau hinterlegt. Die Layer stammen vom besten Einzelbild-Netzwerk. Dies ist nach Tabelle 4.1 das Netzwerk ohne Maximalwert-Pooling-Layer und mit der Kombination aus Batch Normalization und Dropout.

In diesem Beispiel werden die Vorteile des *Transfer Learning* deutlich. Große Teile des Netzwerks sind bereits trainiert. Dadurch kann die Anzahl benötigter Trainingsiterationen beziehungsweise Epochen sehr deutlich reduziert werden. Der Fehler konvergiert sehr schnell zu einem minimalen Wert. Bereits nach etwa 100 Epochen könnte das Training abgeschlossen werden. Ohne *Transfer Learning* sind etwa 900 Epochen Training bis zu einem vergleichbaren Fehler notwendig. Der Fehlerwert ist beim Trainingsbeginn jeweils etwa gleich groß, da die Klassifikation aufgrund des untrainierten LSTM-Layers zunächst zufällig ist. Daran ändert auch die vortrainierte Merkmalsextraktion nichts. Dadurch kann es vorkommen, dass der große Fehler zu Trainingsbeginn die vortrainierten Layer

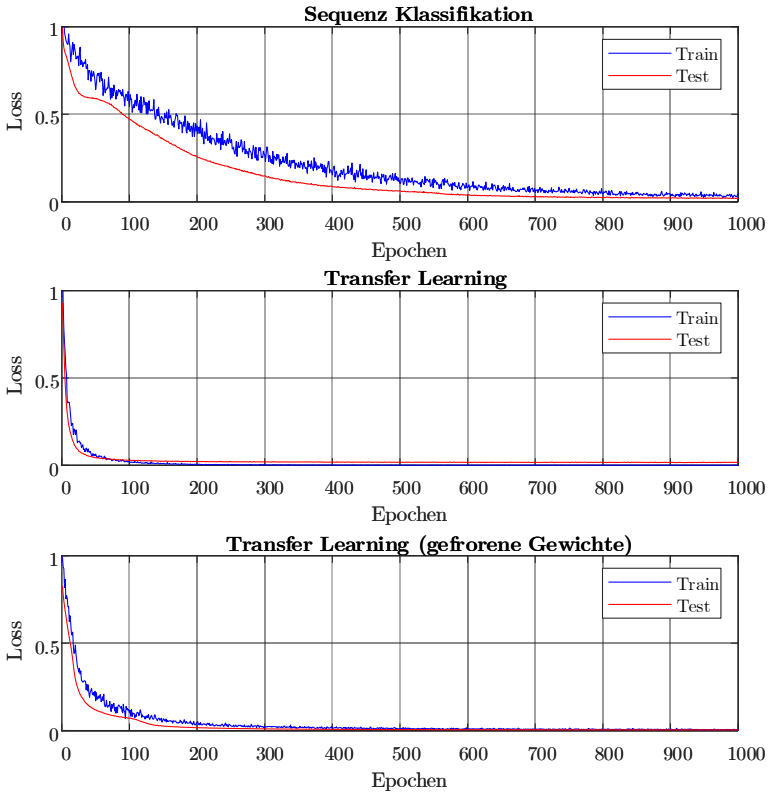


Abbildung 4.5: Verschiedene Trainings des LSTM-Netzwerks. Oben: Training mit neu initialisierten Gewichtungen, Mitte: *Transfer Learning* mit dem Einzelbild-CNN, Unten: *Transfer Learning* mit teilweise eingefrorenen Gewichtungen. Das *Transfer Learning* reduziert die benötigte Trainingszeit deutlich.

in den ersten Epochen stark verändert. Ein großer Fehler führt auch zu einem großen Fehlergradienten. Die vortrainierte Merkmalsextraktion kann dadurch sozusagen überschrieben werden und verloren gehen. Um dies zu verhindern kann ein Lernraten-Faktor für die vortrainierten Layer definiert werden. Ist dieser 0 sind die Gewichtungen vollständig eingefroren und können nicht verändert werden.

Bei einem *Transfer Learning* mit eingefrorenen Gewichtungen sind in diesem Beispiel etwa 200 Epochen für einen vergleichbaren Fehler notwendig. Eingefroren heißt, dass die Gewichtungen der vortrainierten Layer nicht trainiert und nicht verändert werden können. Der CNN-Teil zur Merkmalsextraktion bleibt unverändert. Da die Layer der Merkmalsextraktion nicht trainiert werden können, benötigt das Training des LSTM-Layers mehr Epochen. Alle notwendigen Anpassungen müssen im LSTM-Layer und der darauf folgenden Multi-Label-Klassifikation erfolgen. Der Vorteil ist, dass die bewährte Merkmalsextraktion erhalten bleibt.

Wie die Ergebnisse im nächsten Kapitel zeigen, kann die Fehlerrate durch das *Transfer Learning* weiter reduziert werden. Das Einfrieren der Gewichtungen benötigt etwas mehr Epochen zum Trainieren, liefert im Endeffekt aber bessere Ergebnisse (siehe Tabelle 4.3).

4.2.3 Ergebnisse

Die Ergebnisse der Sequenz-Klassifikation sind in Tabelle 4.3 dargestellt. Bei dem *Transfer Learning* gibt es die Möglichkeit, das vortrainierte Netzwerk einzufrieren, sodass nur der LSTM-Layer und die Multi-Label-Klassifikation trainiert wird.

Insgesamt betrachtet liefert die Sequenz-Klassifikation bessere Ergebnisse als die Einzelbild-Klassifikation. Dies ist auf mehrere Vorteile der Sequenz-Klassifikation zurückzuführen. In den meisten Stichproben ändern sich die Klassen innerhalb einer Sequenz nicht. Das LSTM-Netzwerk kann diese Eigenschaft erlernen. Die Klassifikation einer Sequenz erfolgt meist mit denselben Klassen für alle Zeitschritte. Die vergangenen Klassen werden in der LSTM-Zelle gespeichert. Wird in der aktuellen Messung kein Übergang zu einer anderen Klasse erkannt, wird die vorherige Klassifikation wiederholt. Hinzu kommt, dass die RDM-Messungen statistisches Rauschen aufweisen. Dieses Rauschen kann die

Tabelle 4.3: Vergleich der Ergebnisse der Einzelbild- und Sequenz-Klassifikation. Das beste Ergebnis wird mit einem *Transfer Learning* und eingefrorenen Gewichtungen erreicht.

	Einzelbild	Sequenz	Transfer Learning	Transfer Learning gefrorene Gewichte
RMSE	0,3841	0,2398	0,1688	0,0781
∅-Fehler	9,52%	3,73%	1,94%	0,41%
Kreis-Geste	7,37%	2,28%	1,52%	0,22%
Fingerreiben	22,67%	11,88%	5,20%	1,55%
Handneigung	28,68%	14,52%	15,53%	2,65%
Handbewegung	5,95%	7,86%	1,01%	0,30%
Hinten Links	14,89%	5,74%	2,85%	0,57%
Hinten Rechts	15,36%	4,60%	2,76%	0,60%
Vorne Links	4,00%	1,60%	0,55%	0,18%
Vorne Rechts	9,41%	2,04%	1,86%	0,40%
Gefüllter Topf	2,92%	2,88%	0,51%	0,08%
Kochen	3,10%	1,18%	0,83%	0,21%

LSTM-Zelle unterdrücken, da stets mehrere vergangene Messungen zusätzlich zur aktuellen Messung berücksichtigt werden. Statistisches Rauschen wird daher ausgemittelt. Neben der Messung an sich unterliegen auch die Klassen selbst einer hohen Varianz. Etwa beim Kochen ist das aufsteigen von Gasbläschen zufällig und nicht in jeder Einzelmessung sichtbar. Die Gesten können meist nicht mit einer einzelnen Messungen abgebildet werden, sondern die Geste als Ganzes ist nur über mehrere Messungen eindeutig erfassbar. Daher ist besonders für diese dynamischen Klassen die Sequenz-Klassifikation von großem Vorteil. Dieser Vorteil wird in den Ergebnissen widergespiegelt.

Neben der kürzeren Trainingszeit verbessert das *Transfer Learning* auch die Ergebnisse. Je komplexer ein neuronales Netzwerk wird, desto wichtiger ist eine geeignete Initialisierung und passende Trainingsparameter für ein erfolgreiches Training. Die beste Initialisierung ist ein speziell für diese Anwendung vortrainiertes Netzwerk. Die Merkmalsextraktoren der CNN-Layer sind dabei schon trainiert. Bei eingefrorenen Gewichtungen der vortrainierten Layer werden sogar nochmals bessere Ergebnisse erzielt, da die Merkmalsextraktoren vollständig

erhalten bleiben und nicht verändert werden.

Für die weitere Evaluation werden im nächsten Kapitel speziell Sequenzen mit Übergängen von Klassen in der Anwendung betrachtet und die Vorteile der Sequenz-Klassifikation in der Praxis erläutert.

4.3 Vergleich der Einzelbild- und Sequenz-Klassifikation in der Anwendung

Die Sequenz-Klassifikation liefert bessere Ergebnisse, da das LSTM-Netzwerk gelernt hat, dass sich Klassen nicht schlagartig ändern. Die Klassen der aktuellen Messungen sind mit hoher Wahrscheinlichkeit auch noch in der nächsten Messung wahr. Ein Übergang zu einer anderen Klassen benötigt einige Messungen Zeit. Ein solcher Übergang ist am Beispiel einer Geste in Abbildung 4.6 dargestellt. Bei der Sequenz-Klassifikation mit dem LSTM-Netzwerk werden etwa vier Messungen benötigt bis der Übergang von einer Geste zu einer anderen abgeschlossen ist. Unter Berücksichtigung der Entscheidungsschwelle von 0,5 ist bei der zweiten Messung noch die Klasse *Handneigung* wahr und ab der dritten Messung die Klasse *Fingerreiben*. Die Einzelbild-Klassifikation hingegen wechselt die wahre Klasse direkt in der zweiten Messung. Allerdings ist auch zu beobachten, dass die Klassen noch einige Messungen hin- und herspringen. Dabei handelt es sich letztendlich um Klassifikationsfehler. In der Anwendung sind diese instabilen Klassifikationen kritisch.

Das Verhalten des LSTM-Netzwerks könnte auch als Tiefpass beschrieben werden. Übergänge benötigen etwas Zeit, sind dafür aber stabiler. Kurzfristige Änderungen in einer einzelnen Messung werden unterdrückt. Dieses Tiefpassverhalten ist auf die Struktur einer LSTM-Zelle zurückzuführen (siehe Abbildung 2.7). Informationen aus der aktuellen Messung können nicht direkt ausgegeben werden. Jede Änderung muss zunächst in den LSTM-Speicher geschrieben werden. Bis die Veränderung einer Klasse im Speicher hinterlegt ist, können mehrere Messungen notwendig sein.

Ein Übergang von nicht Kochen zu Kochen ist in Abbildung 4.7 dargestellt. Bei der Sequenz-Klassifikation ist ein klarer Übergang erkennbar, obwohl die vierte Messung nicht eindeutig der Klasse *Kochen* zugeordnet werden kann. Bei der Einzelbild-Klassifikation führt die vierte Messung dazu, dass die Wahrscheinlichkeit für Kochen auf 0 fällt. Durch das Tiefpassverhalten der Sequenz-

Klassifikation fällt die Wahrscheinlichkeit nur kurzzeitig ab, aber bleibt oberhalb der Entscheidungsschwelle von 0,5. Die nicht eindeutige oder fehlerhafte vierte Messung wird weitestgehend unterdrückt. In der Anwendung ist ein solches Verhalten von Vorteil.

4.4 Zusammenfassung

In diesem Kapitel wurden die beiden ersten Kernziele dieser Arbeit erreicht. Zunächst erfolgt die Klassifikation einzelner RDM mittels möglichst effizienter neuronaler Netzwerke zur eingebetteten Inferenz. Dabei steht das Verhältnis zwischen Genauigkeit und der Anzahl an Parametern des neuronalen Netzwerks im Fokus. Das beste Ergebnis wird durch den kombinierten Einsatz von Batch Normalization und Dropout zur Regulierungen des Trainingsverlaufs erreicht. Dieses Netzwerk wird im weiteren Verlauf dieser Arbeit für das *Transfer Learning* verwendet.

Anschließend folgt die Sequenz-Klassifikation einer dynamischen Szene, die mittels einer RDM-Sequenz abgebildet wird. Dazu wird ein Netzwerk mit internen Speichermöglichkeiten (LSTM-Layer) verwendet. Der Teil zur Merkmalsextraktion kann mittels *Transfer Learning* von der bereits trainierten Einzelbild-Klassifikation übernommen werden. Da jede einzelne Messung direkt auf Grundlage der aktuellen Messung und der gespeicherten Information klassifiziert wird, entstehen in der Anwendung keine Verzögerungen. Ein ausführlicher Vergleich aller Klassifikationsmethoden dieser Arbeit folgt in Kapitel 5.6.

In diesem Kapitel wurden die Vorteile des *Transfer Learning* demonstriert. Neben einem deutlich schnelleren Training werden in den allermeisten Trainings bessere Ergebnisse erzielt, als wenn das Netzwerk frisch initialisiert trainiert wird. Weiterhin können die bereits trainierten Gewichtungen eingefroren werden, damit diese nicht weiter verändert werden können. Das Einfrieren der Gewichtungen ist dabei besonders zu Beginn des Trainings zu empfehlen, da sonst die bereits trainierten Gewichtungen überschrieben werden.

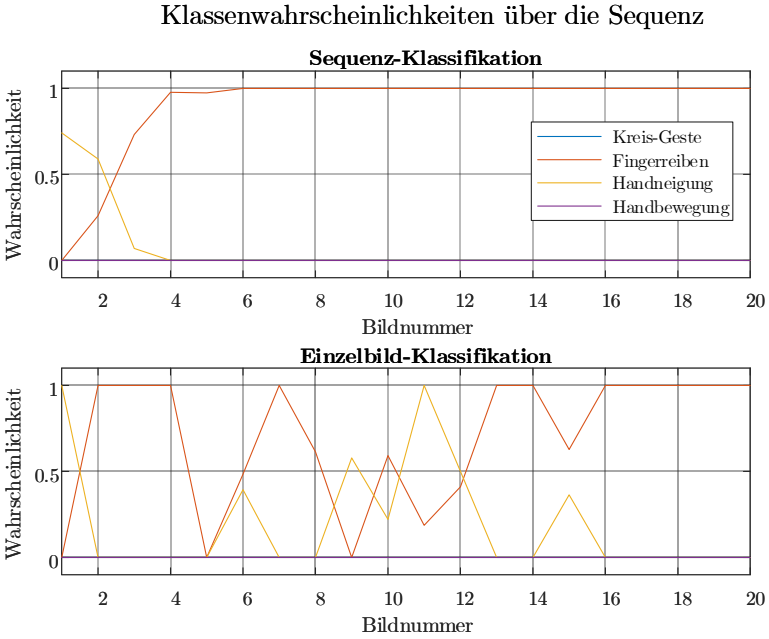


Abbildung 4.6: Vergleich der Wahrscheinlichkeiten der verschiedenen Gesten innerhalb einer Beispiel-Sequenz. Bei der Sequenz-Klassifikation gibt es einen einzigen und sauberen Übergang beim Wechsel der Gesten. Bei der Einzelbild-Klassifikation hingegen springt die klassifizierte Geste einige Male hin und her, da jede Messung separat betrachtet wird.

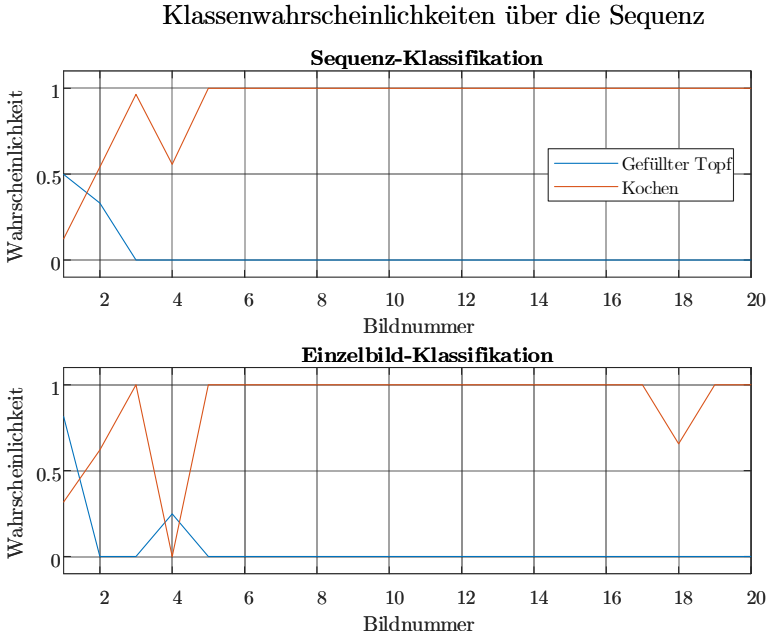


Abbildung 4.7: Vergleich der Wahrscheinlichkeiten der Klasse *Kochen* innerhalb einer Beispiel-Sequenz. Bei der Sequenz-Klassifikation gibt es einen einzigen und sauberen Übergang beim Wechsel von Nicht-Kochen zu Kochen. Bei der Einzelbild-Klassifikation hingegen springt der Zustand nochmals in die entgegengesetzte Richtung. Auch später in der Sequenz gibt es ein erneutes Abfallen der Wahrscheinlichkeit.

5 Cross Learning mittels zusätzlicher Sensordaten

Die Idee des *Cross Learning* mittels zusätzlicher Sensordaten ist aus verschiedenen Motivationsgründen entstanden. Ein zentraler Aspekt ist das Fehlen vortrainierter neuronaler Netzwerke für den Gebrauch in einer Radaranwendung. Die Vorteile vortrainierter Netzwerke werden bereits in Kapitel 4.2.3 gezeigt. Im Bereich von Kameradaten und der Bildklassifikation stehen leistungsfähige Netzwerke öffentlich zur Verfügung. Beispiele sind das ImageNet [35], das GoogLeNet [90] oder das ResNet [91]. Diese Netzwerke sind im Rahmen der ILSVRC entstanden. Die Netzwerke werden mit Millionen von Bildern trainiert. Kamerabilder sind dank Smartphones und sozialen Netzwerken nahezu unbegrenzt vorhanden. Limitierend ist vielmehr das korrekte Labeln der Daten.

Im Gegensatz zu Kameradaten sind Radardaten sehr unterschiedlich. Die RDM ist nur eine von vielen Darstellungsformen für ein einkanaliges Radar mit Chirp-Sequence-Modulationsverfahren. Die RDM ist maßgeblich von den gewählten Radarparametern abhängig (siehe Kapitel 2.1) und repräsentiert eine werte- und einheitenbasierte Messung. Radare mit mehreren Kanälen können zusätzlich zu der Entfernung und Geschwindigkeit der RDM auch Winkel messen [25]. Ein SAR erzeugt hochauflösende zweidimensionale Radarbilder [15], [30]. Aus diesen Gründen können die Daten von verschiedenen Radaren nicht wie bei Kameras beliebig gemischt werden. Ein neuronales Netzwerk kann kaum eine Verallgemeinerung aller Radar- und Datentypen erlernen. Vielmehr sind die Radardaten sehr spezifisch für ein bestimmtes Radar mit bestimmten Parametern in einer bestimmten Anwendung. Aus diesem Grund kann für die Kochanwendung auf kein öffentlich verfügbares vortrainiertes Netzwerk zurückgegriffen werden.

Bezogen auf die Kochanwendung wird in der Literatur bisher nur die Gestenklassifikation behandelt [5], [6], [92], [93]. Die Klassifikation von Kochvorgängen mittels Radar wird nach bestem Wissen erstmals in dieser Arbeit und den

zugehörigen Veröffentlichungen behandelt.

Eine Möglichkeit zur selbstständigen Erzeugung eines vortrainierten Netzwerks ist ein vorheriges unüberwachtes Lernen (unsupervised learning) [94], [95]. Anschließend erfolgt das überwachte Lernen mit den bekannten wahren Klassen der Kochanwendung. Ein solches Vorgehen wird auch *semi-supervised learning* genannt und kann mittels eines Autoencoders erfolgen [96].

Bei dem Ansatz mit einem Autoencoder kann ein weiterer Motivationsgrund für das *Cross Learning* eingeführt werden, indem das Autoencoder-Training mittels zusätzlicher Sensordaten multimodal erfolgt. Die Idee ist, die Vorteile der verschiedenen Sensoren zu kombinieren. Die Vorteile der zusätzlichen Sensoren sind bereits in Kapitel 3.1 am Beispiel des Kochens erläutert. Etwa die Wärmebildkamera ist aufgrund der Temperaturmessung ideal für die Kochklassifikation geeignet. In beiden Kameras ist die Kochposition gut zu erkennen. Die Kombination der Sensoren erfolgt durch ein multimodales Autoencoder-Training. Das Prinzip ist in Abbildung 5.1 dargestellt. Der Ziel-Sensor ist dabei der Sensor, welcher alleinig in der Anwendung verwendet werden soll. Die zusätzlichen Sensoren werden nur während des multimodalen Autoencoder-Trainings verwendet. Dabei werden die Ziel-Sensor-Daten zunächst durch den Encoder in eine komprimierte Wissensrepräsentation transformiert. Der Decoder transformiert die komprimierte Wissensrepräsentation wiederum in die Zusatz-Sensor-Daten. Anschließend kann in einem zweiten Trainingsschritt der zuvor trainierte Encoder mit der komprimierten Wissensrepräsentation als vortrainiertes Netzwerk verwendet werden. Die komprimierte Wissensrepräsentation enthält dabei relevante Merkmale (Features) des Ziel- und Zusatzsensors, da Zusatz-Sensor-Daten aus der komprimierten Wissensrepräsentation rekonstruiert werden können. Die zusätzlichen Sensoren sind in der Anwendung nicht notwendig.

In der Literatur sind multimodale Ansätze weit verbreitet [14], [97]–[107]. Im Gegensatz zu dieser Dissertation werden multimodale Ansätze meist nicht nur für ein vorangestelltes Training, sondern in der gesamten Anwendung verwendet. In der Anwendung führen mehrere Sensoren in der Regel zu besseren Ergebnissen und Redundanzen. In dieser Arbeit soll aber ausschließlich der Radarsensor zur Klassifikation in der Anwendung verwendet werden (unimodale Klassifikation).

Multimodale Ansätze in der Literatur decken vielseitige Anwendungen und Sensoren ab. Radarsensoren finden wenig Anwendung. Stattdessen werden zumeist Kameras, Tiefenkameras, Infrarotkameras (Nachtsicht), Text oder

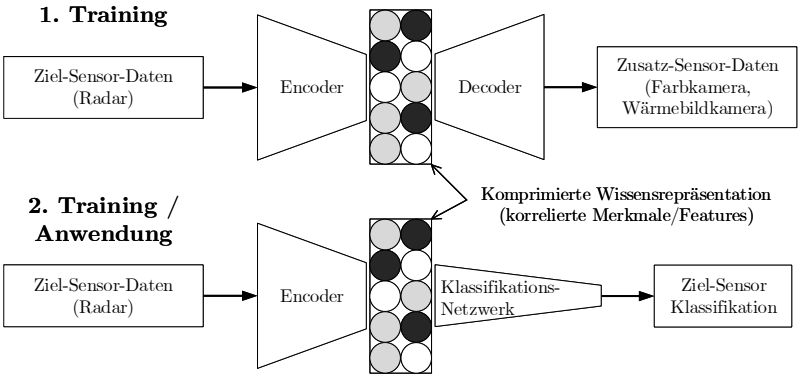


Abbildung 5.1: Ansatz des multimodalen *Cross Learnings* mit einem Autoencoder. Im ersten Schritt wird ein Autoencoder multimodal trainiert. Das heißt, Eingangs- und Ausgangsdaten haben verschiedene Modalitäten. Im zweiten Schritt wird der multimodal vortrainierte Encoder, beziehungsweise die komprimierte Wissensrepräsentation, zur weiteren Klassifikation mithilfe eines Klassifikationsnetzwerks verwendet. Zusätzliche Modalitäten sind in der Anwendung nicht notwendig, obwohl die komprimierte Wissensrepräsentation Merkmale aller Modalitäten aufweist. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

Audio verwendet. Eine Gestenerkennung mittels Video, Tiefenkamera und optischem Fluss wird in [97] gezeigt. Der Ansatz basiert auf einem 3D-CNN-Netzwerk pro Modalität und einer multimodalen Fehlerberechnung während des Trainings. Eine unimodale Klassifikation ist möglich. In [98] werden diese Modalitäten zur Gestenerkennung noch um eine Posenschätzung und Audio erweitert. Das Netzwerk funktioniert dabei nur multimodal, wobei einzelne Modalitäten zufällig und zeitweise fehlen können. Aufgrund der Ähnlichkeit zum klassischen Dropout-Layer wird dieser Ansatz *ModDrop* genannt. Neben Gesten werden in [99] und [100] vollständige menschliche Handlungen multimodal klassifiziert. In [101] und [102] werden Objekte und Personen mittels Kamera und Tiefenkamera detektiert.

Neben einer Klassifikation oder Detektion werden multimodale Ansätze auch zur Datenkonvertierung oder Datenerzeugung verwendet. In [103] werden die Modalitäten Bild und Text verknüpft, sodass zu einem Bild ein beschreibender Text generiert werden kann. In [104] wird aus einem Bild von Essen das zugehörige Rezept mit Zutaten und Kochanweisungen generiert. Diese beiden Beispiele können beispielsweise in einer Smartphone-App angewendet werden. Der Autoencoder-Ansatz dieser Dissertation kann ebenfalls zur Datenkonvertierung und Datenerzeugung verwendet werden. Im ersten Trainingsschritt nach Abbildung 5.1 wird ein solcher Autoencoder trainiert. Aus den Ziel-Sensor-Daten können nach dem Training die Zusatz-Sensor-Daten erzeugt werden. Die Erzeugung solcher Pseudodaten wird in der Literatur auch Halluzination genannt [105]–[107]. Für die Klassifikation in der Anwendung ist eine solche Halluzination nicht notwendig. Dennoch können halluzinierte Bilder beispielsweise für eine Benutzeroberfläche in der Kochanwendung interessant sein, da Kamerabilder von Anwendern einfacher zu interpretieren sind als eine RDM. Ein Beispiel wäre die Anzeige der Kochfeldbelegung mittels eines halluzinierten Bildes. Die Halluzination in der Kochanwendung wird in Kapitel 5.5 gezeigt.

Ein multimodaler Ansatz mit einem Autoencoder ist in [108] beschrieben. Die Modalitäten sind Video und Audio (Sprache). Die Videodaten unterstützen bei der Spracherkennung. In [109] werden ebenfalls Video- und Audiodaten verwendet. Die Anwendung ist ein Mapping von etwa Gesten zu bestimmten Tönen. Auf diese Weise kann mittels Gesten Musik gemacht werden. Das verwendete Netzwerk besteht aus gestapelten Autoencodern. Das heißt, der Ausgang eines Autoencoders dient als Eingang für einen nächsten Autoencoder und so weiter.

Ein multimodaler Ansatz mit einem Radar wird in [14] vorgestellt. Darin soll die Anzahl von Personen in einem Raum detektiert werden. Das verwendete 60 GHz Radar ist mehrkanalig und die Radardaten werden zu sogenannten *macro- und micro-Doppler range-angle images* vorverarbeitet. Die Daten enthalten also Informationen an welcher Stelle (Entfernung und Winkel) in einem Raum Bewegungen von Personen auftreten. Als zweite Modalität wird eine Farbkamera verwendet. Die Kamerabilder werden mittels eines neuronalen Netzwerks zu einer Art Heatmap vorverarbeitet. Die Heatmap zeigt an, an welcher Stelle im Raum sich Personen befinden. Die beiden Modalitäten werden also derart vorverarbeitet, sodass die relevanten Informationen an derselben

Stelle in den Bildern (räumlich gesehen) auftreten. Es folgt ein multimodales Training zur Klassifikation, wie viele Personen sich in einem Raum aufhalten (und bewegen).

Für das *Cross Learning* in dieser Dissertation erfolgt keine Vorverarbeitung der Sensordaten mit Ausnahme der 2D-FFT zur Berechnung der RDM (siehe Kapitel 2.1). Die korrelierenden Merkmale der RDM und der Kamerabilder müssen vom neuronalen Netzwerk selbst gefunden werden.

5.1 Netzwerk-Architektur

Die Netzwerk-Architektur des Autoencoders ist in Abbildung 5.2 dargestellt. Der Autoencoder besteht grundsätzlich aus zwei Teilen. Der Encoder transformiert die Eingangsdaten (Ziel-Sensor-Daten) in eine komprimierte Wissensrepräsentation. Der Decoder transformiert die komprimierte Wissensrepräsentation in die Ausgangsdaten (Zusatz-Sensor-Daten). Bei einem herkömmlichen Autoencoder sind die Eingangs- und Ausgangsdaten meist dieselben, beziehungsweise entsprechen derselben Modalität [96]. Die Eingangsdaten werden teilweise mittels Augmentation verändert, damit sich die Daten leicht unterscheiden. In [94] werden die Eingangsdaten absichtlich verrauscht. Der sogenannte *Denoising*-Autoencoder soll dieses Rauschen wieder entfernen.

Eine weitere weit verbreitete Variante der Ausgangsdaten ist die semantische Segmentierung [62]. Die semantische Segmentierung ist eine Form der Clusterung, bei der jedes Pixel des Eingangsbildes klassifiziert wird. Dazu erzeugt etwa ein Autoencoder ein Bild entsprechend der Größe des Eingangsbildes. Jedem Pixel im Ausgangsbild wird dabei eine Klasse zugeordnet. Zusammenhängende Bereiche derselben Klasse werden entsprechend als Cluster dargestellt. Ein sehr erfolgreicher Vertreter der semantischen Segmentierung ist das U-Net [110] aus der Biomedizintechnik. Das U-Net wird zur Segmentierung von Krebszellen im Gehirn, aber auch in vielen anderen nicht medizinischen Bereichen eingesetzt. Ein weiterer Forschungsschwerpunkt für die semantische Segmentierung ist das autonome Fahren. Eine Übersicht verschiedener Ansätze zur Segmentierung von unter anderem Fahrbahn, Fahrbahnmarkierung und anderen Verkehrsteilnehmern gibt [111].

Ein klassischer Ansatz für die Segmentierung und entsprechend auch für die Autoencoder-Architektur ist ein FCN [62]. Also ein Netzwerk, das maßgeblich

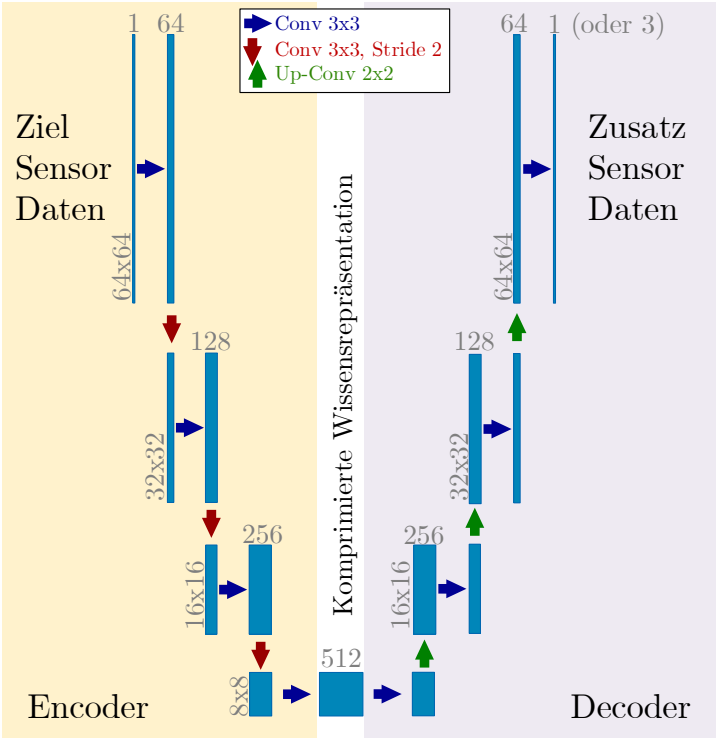


Abbildung 5.2: Die Autoencoder-Architektur besteht aus einem Encoder und einem Decoder. Dazwischen befindet sich eine komprimierte Wissensrepräsentation. Die blauen Boxen sind dreidimensionale Schichten aus Neuronen. Die räumliche Größe ist jeweils links und die Tiefe (Anzahl der Filter) darüber angegeben. Die Pfeile sind verschiedene Conv-Operationen, die in der Legende erläutert sind. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

aus Conv-Layer aufgebaut ist. Durch das *local receptive field* der Conv-Layer bleiben die lokalen Informationen erhalten. Auf FC-Layer wird verzichtet, da durch einen solchen Layer die räumliche Struktur verloren geht, die letztendlich am Ausgang wieder benötigt wird.

Die in dieser Dissertation verwendete Autoencoder-Architektur in Abbildung 5.2 basiert auf der U-Net-Architektur. Der U-förmige Aufbau ist gut erkennbar. Allerdings verwendet das U-Net normalerweise Maximalwert-Pooling-Layer zum Downsampling. Aufgrund der Erfahrungen in Kapitel 4.1 wird auf diese Layer verzichtet. Stattdessen werden Conv-Layer mit einer Schrittweite von 2 (Stride $S = 2$) zum räumlichen Downsampling verwendet. Vor jedem Downsampling wird die Anzahl der Filter C mittels eines Conv-Layer verdoppelt. Diese Grundstruktur aus Verdopplung der Filter und anschließendem Downsampling kann theoretisch so oft wiederholt werden, bis die räumliche Größe 1x1 beträgt. Die Anzahl der Wiederholungen dieser Struktur wird in der Literatur auch Encoder-Decoder-Tiefe genannt. In den Experimenten ist eine Encoder-Decoder-Tiefe von 3 als gutes Verhältnis zwischen der Genauigkeit zu der Anzahl an Parametern hervorgegangen. Ein Grund für die relativ geringe Tiefe ist hier wieder die von vorneherein kleine räumliche Größe der Daten von 64x64.

Auf der Decoder-Seite wird eine umgekehrte Grundstruktur aus Halbierung der Filter und anschließender *Up-Convolution* (auch Deconvolution [61], [112] oder transponierte Convolution genannt) verwendet. Diese Grundstruktur wird entsprechend der Encoder-Decoder-Tiefe wiederholt. Das Resultat ist der typische U-förmige Aufbau. Die *Up-Convolution*-Operation entspricht einem Upsampling [113].

Im originalen U-Net [110] gibt es in jeder Grundstruktur eine direkte Verbindungen zwischen Encoder und Decoder. Dies führt bei der semantischen Segmentierung zu wesentlich besseren Ergebnissen, da nicht alle Merkmale in der komprimierten Wissensrepräsentation enthalten sein müssen. Merkmale des Encoders können vielmehr auf jeder Stufe räumlicher Größe direkt an den Decoder übertragen werden. Für die Anwendung des *Cross Learning* in dieser Dissertation werden die direkten Verbindungen entfernt, da in dem zweiten Trainingsschritt nur der Encoder verwendet wird. Durch das Entfernen direkter Verbindungen ist sichergestellt, dass alle Merkmale in der komprimierten Wissensrepräsentation enthalten sind.

Als Aktivierungsfunktion wird die ReLU verwendet. Im letzten Layer wird

eine *Clipped* ReLU eingesetzt, die die Werte am Ausgang auf 0 bis 255 begrenzt. Dieser Wertebereich entspricht einem Bild mit einer typischen Farbtiefe von 8 Bit. Je nach Ausgangsmodalität wird die Tiefe (Anzahl der Kanäle) des Ausgangs angepasst. Die RDM und das Wärmebild weißen jeweils nur einen Kanal auf. Das Bild der Farbkamera besteht aus den drei Kanälen Rot, Grün und Blau (RGB).

Bei den Eingangs- und Ausgangsdaten kann es sich um verschiedene Modalitäten handeln. Dabei soll der Encoder eine Modalität komprimieren und mittels der komprimierten Wissensrepräsentation abbilden. Der Decoder erzeugt aus dieser Wissensrepräsentation eine beliebige Modalität. Die Ergebnisse des Autoencoder-Trainings werden in Kapitel 5.3 vorgestellt. Zur Visualisierung der Ergebnisse können Pseudodaten halluziniert werden. Einige Beispiele werden in Kapitel 5.5 gezeigt.

Die Netzwerk-Architektur zur Klassifikation ist in Abbildung 5.3 dargestellt. Der Encoder und die komprimierte Wissensrepräsentation werden vom Autoencoder übernommen und können vortrainiert für ein *Transfer Learning* verwendet werden. Anschließend folgt das Klassifikationsnetzwerk. Dieses besteht aus drei Wiederholungen eines Conv-Layers mit einer Schrittweite von 2 (Stride $S = 2$). Neben dem räumlichen Downsampling wird die Anzahl der Filter jeweils halbiert. Zuletzt folgt ein FC-Layer mit zehn Neuronen für die Multi-Label-Klassifikation. Bis auf diesen letzten Layer handelt es sich um ein FCN.

Für das Training des Klassifikationsnetzwerks werden verschiedene Möglichkeiten betrachtet. In Kapitel 5.2 wird das Netzwerk zunächst von Grund auf, also ohne vorheriges Autoencoder-Training, trainiert. Dieses Training erfolgt für alle Modalitäten separat, sodass die Leistungsfähigkeit der einzelnen Modalitäten bewertet werden kann. Anschließend folgt in Kapitel 5.4 das *Cross Learning* beziehungsweise das Cross-Modale *Transfer Learning*. Hier besteht die Möglichkeit, die Gewichtungen des vortrainierten Encoders einzufrieren.

5.2 Leistungsfähigkeit der einzelnen Modalitäten

Zur Bewertung der Leistungsfähigkeit der einzelnen Modalitäten wird das Netzwerk nach Abbildung 5.3 von Grund auf trainiert. Die Initialisierung erfolgt mittels der Xavier-Glorot-Methode [86]. Die Lernrate beträgt 10^{-5} in Kombination mit einer Batch-Größe von 256. Das Netzwerk wird für 200

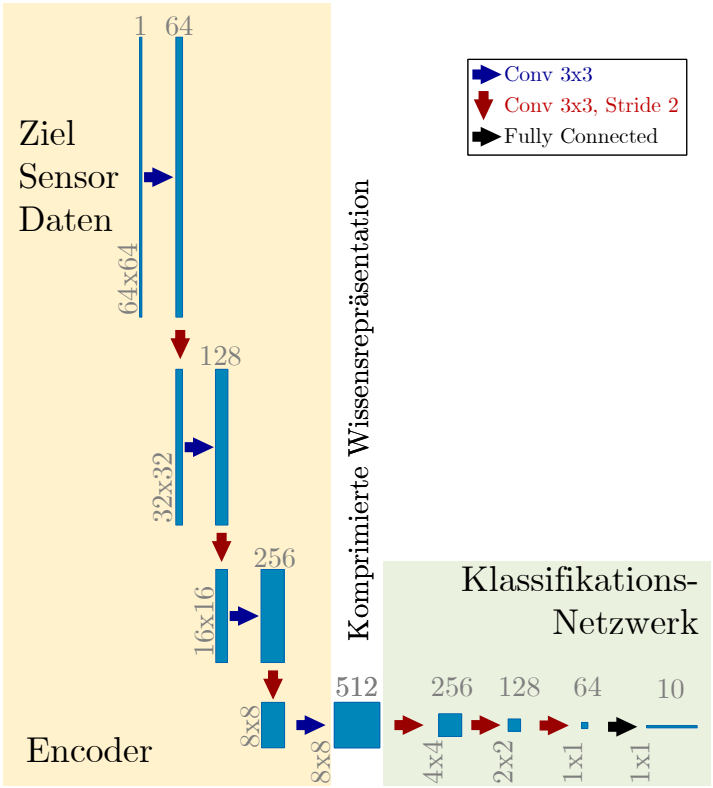


Abbildung 5.3: Klassifikationsnetzwerk, das den vortrainierten Encoder-Teil des Autoencoders wiederverwendet. Die blauen Boxen sind dreidimensionale Schichten aus Neuronen. Die räumliche Größe ist jeweils links und die Tiefe (Anzahl der Filter) darüber angegeben. Die Pfeile sind verschiedene Operationen, die in der Legende erläutert sind. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

Epochen trainiert. Als Trainingsalgorithmus wird Adam verwendet. Auf Batch Normalization und Dropout wird beim Training dieser FCNs verzichtet. In den Experimenten werden bessere Ergebnisse ohne diese Layer zur Regulierung erreicht.

Als Fehlerfunktion wird letztendlich der HMSE nach Gleichung 3.2 verwendet (siehe Kapitel 3.2). An dieser Stelle werden aber zunächst unterschiedliche Fehlerfunktionen in den Experimenten untersucht:

- Half Mean Square Error (HMSE) nach Gleichung 3.2.
- Root Mean Square Error (RMSE) nach Gleichung 3.1.
- Mean Square Error (MSE).
- Mean Absoulte Error (MAE) [84].
- Macro-Averaging F1-Score [74], [114].
- Micro-Averaging F1-Score [74], [114].

Der MAE ist typischerweise robuster gegenüber Ausreißern als einer der quadrierten Fehler wie etwa der RMSE [84]. Ausreißer werden aufgrund des Quadrierens stärker gewichtet. Bei der Multi-Label-Klassifikation werden Ausreißer allerdings generell mittels der *clipped* ReLU unterdrückt. Da Werte größer 1 abgeschnitten werden, liegt der Fehler pro Klassifikator immer zwischen 0 und 1. In diesem Wertebereich ist das Quadrieren sinnvoll, sodass größere Fehler stärker gewichtet werden. Aber ein einzelner großer Fehler kann nicht den gesamten Batch dominieren. Für den HMSE anstelle des RMSE spricht die Eigenschaft des *Ground Truth*-Vektors durchschnittlich zwei wahre Klassen zu enthalten. Daher wird auf die Wurzel verzichtet und der MSE halbiert. Zwischen MSE und HMSE gibt es außer dem Faktor 2 keinen Unterschied. Durch die durchschnittlich zwei wahren Klassen liefert der HMSE leicht bessere Ergebnisse.

Der F1-Score wird mittels Precision (Gleichung 3.6) und Recall (Gleichung 3.5) berechnet [74], [114]. Die 1 im Namen steht dabei für eine gleiche Gewichtung von Precision und Recall. Der F1-Score wird folgendermaßen berechnet:

$$F1\text{-}Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{5.1}$$

Tabelle 5.1: Leistungsfähigkeit der einzelnen Modalitäten Radar, Wärmebildkamera, RGB- und Graustufen-Kamera.

	Radar-RDM	Wärmebild	RGB	Graustufe
RMSE	0,2041	0,1922	0,2677	0,3262
∅-Fehler	2,79%	1,93%	3,74%	5,58%
Kreis-Geste	5,32%	25,53%	1,38%	2,34%
Fingerreiben	16,43%	28,03%	1,40%	2,58%
Handneigung	15,90%	50,30%	1,39%	3,08%
Handbewegung	2,69%	0,06%	0,00%	0,00%
Hinten Links	3,48%	0,00%	0,00%	0,00%
Hinten Rechts	3,62%	0,07%	0,00%	0,00%
Vorne Links	1,17%	0,03%	0,02%	0,04%
Vorne Rechts	2,09%	0,05%	0,00%	0,03%
Gefüllter Topf	1,04%	0,00%	0,02%	0,02%
Kochen	1,20%	0,00%	50,30%	74,49%

Das Macro- beziehungsweise Micro-Averaging bezieht sich auf die Durchschnittsberechnung der K Klassen. Beim Macro-Averaging werden Precision und Recall für jede Klasse separat berechnet. Anschließend werden die durchschnittliche Precision und Recall über K Klassen berechnet und schließlich der F1-Score. Auf diese Weise werden alle Klassen unabhängig von der Anzahl der Stichproben gleichermaßen gewichtet. Beim Micro-Averaging werden zunächst alle RP, FP und FN aufsummiert und einmalig für alle Klassen Precision und Recall und schließlich der F1-Score berechnet. Der Einfluss von Klassen mit wenigen Stichproben (wie die Gesten) ist daher verhältnismäßig gering.

Aufgrund der ungleichmäßigen Verteilung der Klassen im Datensatz der Kochanwendung ist der Macro-Averaging F1-Score zu bevorzugen. In der Praxis liefert das Macro-Averaging auch bessere Ergebnisse als das Micro-Averaging. Allerdings zeigen beide F1-Scores wesentlich schlechtere Ergebnisse im Vergleich zum HMSE. Ein Problem am F1-Score ist, dass zur Berechnung keine RN berücksichtigt werden. Jede Stichprobe enthält durchschnittlich nur etwa zwei RP und entsprechend acht RN. Diese Eigenschaft des Datensatzes erklärt die deutlich schlechteren Ergebnisse des F1-Scores als Fehlerfunktion im Training. Eine Optimierung hinsichtlich der RN ist wichtig für den Lernerfolg.

Die Ergebnisse des Trainings für die einzelnen Modalitäten Radar (RDM), Wärmebildkamera und Farbkamera (RGB-Kamera) sind in Tabelle 5.1 dargestellt. Als zusätzlicher Vergleich wird das drei-kanalige RGB-Bild in ein Graustufen-Bild umgewandelt. Die Wärmebildkamera liefert das insgesamt beste Ergebnis, wobei Schwächen bei der Gestenklassifikation vorhanden sind. Die Farbkamera zeigt Schwächen bei der Kocherkennung. Das Kochen ist auch als Mensch schwer im Kamerabild zu erkennen. Im Graustufen-Bild wird dieses Problem noch verstärkt. Hinzu kommt, dass die Kamera durch den entstehenden Wasserdampf nach kurzer Zeit beschlägt. Das Radar ist ein guter Allrounder und zeigt keine gravierenden Ausreißer in einer spezifischen Klasse.

Das Ergebnis der Modalität Radar wird im weiteren Verlauf als FCN Referenz-Netzwerk betrachtet, da die Klassifikation in der Anwendung ausschließlich mit dem Radar stattfinden soll. Verglichen mit dem Netzwerk zur Einzelbild-Klassifikation aus Kapitel 4.1 schneidet das FCN deutlich besser ab. Das FCN klassifiziert dabei auch nur Einzelbilder, ist aber mit fast drei Mal so vielen Layern deutlich komplexer aufgebaut. Der Sequenz-Klassifikation muss sich das FCN allerdings deutlich geschlagen geben. Die verschiedenen Klassifikationsmethoden werden im Kapitel 5.6 zusammengefasst und verglichen.

5.3 Autoencoder-Training

Beim Autoencoder-Training können alle möglichen Kombinationen aus verschiedenen Eingangs- und Ausgangsmodalitäten trainiert werden. Die Ergebnisse sind in Tabelle 5.2 dargestellt. Für die weitere Anwendung sind besonders die Ergebnisse mit der Radar-RDM als Eingangsmodalität interessant. In der Anwendung soll die Klassifikation nur mittels Radar erfolgen. Aus Gründen der Vollständigkeit werden aber auch Autoencoder mit anderen Modalitäten trainiert.

Die Initialisierung der Autoencoder erfolgt mittels der Xavier-Glorot-Methode [86]. In den Experimenten ist die Lernrate 10^{-4} in Kombination mit einer Batch-Größe von 256 als geeignet hervorgegangen. Das Netzwerk wird für 200 Epochen trainiert. Als Trainingsalgorithmus wird Adam verwendet. Als Fehlerfunktion wird hier nicht der HMSE verwendet, da keine Multi-Label-Klassifikation erfolgt. Am Ausgang des Autoencoders wird ein Bild erzeugt. Als Fehlerfunktion wird

Tabelle 5.2: Ergebnisse des Autoencoder-Trainings aller Kombinationsmöglichkeiten der verschiedenen Modalitäten.

Eingangsmodalität	Ausgangsmodalität	RMSE
Radar-RDM	Wärmebild	88,91
Radar-RDM	RGB-Bild	717,92
Radar-RDM	Graustufen-Bild	378,00
Radar-RDM	Wärme- und RGB-Bild	564,62
Radar-RDM	Radar-RDM	135,36
Wärmebild	Radar-RDM	784,05
Wärmebild	RGB-Bild	770,58
Wärmebild	Wärmebild	39,69
RGB-Bild	Radar-RDM	782,99
RGB-Bild	Wärmebild	63,57
RGB-Bild	RGB-Bild	310,94

daher der

$$RMSE_{Autoencoder} = \sqrt{\frac{\sum_{i=1}^{n}\sum_{p=1}^{HWC}(\hat{y}_{i,p} - y_{i,p})^2}{nHWC}} \tag{5.2}$$

verwendet. Dabei wird nicht über die Anzahl der Klassen K summiert und gemittelt, sondern über die Anzahl aller Pixel $H \cdot W \cdot C$. In Falle eines einkanaligen Bildes sind das $64 \cdot 64 = 4096$ Pixel. Der Wertebereich von y und $\hat{y}$ beträgt hier 0 bis 255 (8 Bit).

Die besten Ergebnisse werden mit einem Wärmebild als Ausgangsmodalität erzielt. Das Wärmebild kann theoretisch am besten rekonstruiert werden, da nur wenige unterschiedliche Temperaturen auftreten. Die meisten Pixel entsprechen der Raumtemperatur von etwa 21 °C. Bei den Gesten wird die Körpertemperatur von etwa 37 °C gemessen. Eine eingeschaltete Herdplatte wird aufgrund der Normierung mit dem Maximalwert von 125 °C gemessen. Das Kochgut kann unterschiedliche Temperaturen aufweisen. Beim Kochen sind es schließlich 100 °C.

Im RGB-Bild müssen zunächst drei Mal so viele Pixel rekonstruiert werden, da drei Farbkanäle vorhanden sind. Theoretisch sollte das keinen Einfluss auf

den RMSE haben, da nach Gleichung 5.2 mit der Anzahl der Kanäle gemittelt wird. Praktisch wird aber mit einem Graustufen-Bild ein besseres Ergebnis im Autoencoder-Training erzielt. Im nachfolgenden Kapitel wird verglichen, ob für das *Cross Learning* die fehlende Farbinformation oder ein besser trainierter Autoencoder vorteilhafter ist.

Als zusätzliches Experiment wird noch ein Autoencoder mit den beiden Modalitäten Wärmebild und RGB-Bild als Ausgangsmodalität trainiert. Das Ausgangsbild ist dann vier-kanalig. In diesem Experiment werden alle drei Modalitäten gleichzeitig verwendet. Die komprimierte Wissensrepräsentation enthält damit Merkmale aller Modalitäten. Der Nutzen für das *Cross Learning* wird im nächsten Kapitel gezeigt.

Zuletzt wird noch ein Autoencoder nach dem klassischem *semi-supervised learning* Ansatz betrachtet. Die Eingangs- und Ausgangsmodalität ist dabei identisch. Es werden keine zusätzlichen Sensordaten oder Modalitäten verwendet.

5.4 Cross-Modales Transfer Learning

In diesem Kapitel wird das Klassifikationsnetzwerk nach Abbildung 5.3 mittels dem cross-modalem *Transfer Learning* trainiert. Der Encoder-Teil ist multimodal vortrainiert. Es besteht die Möglichkeit, diese vortrainierten Gewichtungen des Encoders einzufrieren. Die multimodale Merkmalsextraktion kann damit vollständig erhalten bleiben. Der nicht vortrainierte Teil wird mittels der Xavier-Glorot-Methode [86] initialisiert. Als Trainingsalgorithmus wird Adam mit einer Lernrate von 10^{-5} mit einer Batch-Größe von 256 verwendet. Das Training läuft für 200 Epochen. Da ein großer Teil des Netzwerks bereits trainiert ist, werden zumeist deutlich weniger Epochen benötigt. Als Fehlerfunktion wird wie bei der Multi-Label-Klassifikation üblich der HMSE nach Gleichung 3.2 verwendet.

Die Ergebnisse sind in Tabelle 5.3 beziehungsweise in Tabelle 5.4 für eingefrorene Gewichtungen dargestellt. Die Klassifikation erfolgt stets alleinig mit der Radar-RDM. Die angegebene Modalität bezieht sich jeweils auf das multimodale Autoencoder-Training, dessen Encoder verwendet wird. Wie bereits in Kapitel 4.2.3 sind auch hier die Ergebnisse mit eingefrorenen Gewichtungen durchweg besser. Dabei tritt derselbe Effekt auf, dass die vortrainierten Merkmalsextraktoren zu Trainingsbeginn verloren gehen. Zu Trainingsbeginn ist der Fehler aufgrund des frisch initialisierten Netzwerkteils zunächst groß, da die

Tabelle 5.3: Ergebnisse des multimodalen *Cross Learnings*. Die Klassifikation erfolgt stets alleinig mit der Radar-RDM. Die Modalität, die zum *Cross Learning* verwendet wird, ist jeweils angegeben.

	Wärme	RGB	Graustufe	Wärme+RGB
RMSE	0,1201	0,2109	0,3638	0,2076
∅-Fehler	0,86%	2,41%	6,94%	2,33%
Kreis-Geste	2,87%	2,66%	3,19%	2,23%
Fingerreiben	9,67%	8,16%	100,00%	7,41%
Handneigung	8,25%	6,56%	100,00%	6,16%
Handbewegung	0,32%	0,13%	100,00%	0,00%
Hinten Links	1,09%	0,53%	0,57%	0,44%
Hinten Rechts	1,01%	0,65%	0,56%	0,51%
Vorne Links	0,13%	0,04%	0,08%	0,04%
Vorne Rechts	0,38%	0,32%	0,25%	0,13%
Gefüllter Topf	0,09%	0,07%	0,11%	0,07%
Kochen	0,16%	25,25%	0,08%	25,31%

Tabelle 5.4: Ergebnisse des multimodalen *Cross Learnings* mit eingefrorenen Gewichtungen des Encoders.

	Wärme	RGB	Graustufe	Wärme+RGB
RMSE	0,0988	0,1989	0,0987	0,1016
∅-Fehler	0,56%	2,18%	0,65%	0,65%
Kreis-Geste	1,38%	1,70%	2,55%	2,77%
Fingerreiben	7,20%	11,17%	8,49%	8,59%
Handneigung	7,06%	10,24%	7,95%	9,15%
Handbewegung	0,00%	0,45%	0,58%	0,19%
Hinten Links	0,66%	0,65%	0,59%	0,62%
Hinten Rechts	0,58%	0,46%	0,54%	0,52%
Vorne Links	0,02%	0,12%	0,10%	0,10%
Vorne Rechts	0,32%	0,29%	0,25%	0,21%
Gefüllter Topf	0,02%	8,82%	0,11%	0,10%
Kochen	0,00%	0,18%	0,21%	0,05%

Klassifikation zufällig ist. Der große Fehler führt zu einem großem Fehlergradienten, der die vortrainierte Merkmalsextraktion stark verändern kann. Die zuvor multimodal erlernte Merkmalsextraktion geht verloren. Um dies zu vermeiden werden die Gewichtungen eingefroren. Der Vorteil wird in den Ergebnissen aller Modalitäten widergespiegelt.

Das beste Ergebnis in Form der geringsten durchschnittlichen Fehlerrate ist das *Cross Learning* mit dem Wärmebild, dicht gefolgt vom Graustufenbild und der Kombination aus Wärme- und RGB-Bild. Speziell bei der Kochanwendung ist die Wärmebildkamera eine perfekte Ergänzung zum Radar. Bei der Klassifikation der Klasse *Kochen* kommt es praktisch zu keiner Fehlklassifikation mehr. Verglichen mit dem FCN Referenz-Netzwerk, das die identische Architektur aufweist und sich nur in der Art des Trainings unterscheidet, hat das *Cross Learning* die Fehlerrate für jede Klasse verbessert. Dies zeigt, dass neben der Architektur eines Netzwerks, das Trainingsverfahren beziehungsweise vortrainierte Netzwerke von großer Bedeutung sind. Besonders deutlich sind die Verbesserungen neben dem Kochen bei der Detektion der Kochposition. Da bieten die kamerabasierten Systeme deutliche Vorteile gegenüber einem Radar. Diese Vorteile werden mittels *Cross Learning* auf das Radar übertragen. Die meisten Fehler treten weiterhin bei der Gestenerkennung auf, da dies die schwierigsten Klassen mit einem relativ kleinen Datensatz sind. Eine Zusammenfassung aller Klassifikationsmethoden dieser Dissertation wird im Kapitel 5.6 gegeben.

Im Vergleich zwischen RGB- und Graustufenbild wird bei eingefrorenen Gewichtungen ein deutlich besseres Ergebnis mit dem Graustufen-Bild erreicht. Das bessere Autoencoder-Training ist für das *Cross Learning* also vorteilhafter als die Farbinformation. Bei Betrachtung der Beispielbilder in Abbildung 3.3, 3.4 und 3.5 wird auch klar, dass im Prinzip nur bei den Gesten überhaupt Farben auftreten. Das Schwarz und Silber des Kochfelds und der Töpfe ist in Graustufen quasi identisch mit dem Farbbild. Ohne eingefrorene Gewichtungen scheint das *Cross Learning* mit dem Graustufenbild fehlerhaft zu sein, da es das einzige Netzwerk ist, das schlechter als das FCN Referenz-Netzwerk ist. Das betrifft besonders die Klassen der Gesten. Eine Ursache kann die fehlende Farbinformation sein.

Die gleichzeitige Verwendung von Wärme- und RGB-Bild liefert keine besseren Ergebnisse als das Wärmebild allein. Dies ist einerseits auf ein schlechteres Ergebnis beim Autoencoder-Training zurückzuführen. Andererseits ist das Er-

gebnis des RGB-Bildes allein um etwa Faktor Vier schlechter als das Ergebnis des Wärmebildes. Das gemeinsame Ergebnis befindet sich entsprechend zwischendrin.

Die Unterschiede des *Cross Learning* mit dem Wärmebild, dem Graustufenbild und der Kombination aus Wärme- und RGB-Bild sind mit 0,09% sehr gering. Bezogen auf die Anzahl der Stichproben im Testdatensatz (27.600) ist das ein Unterschied von nur 25 Stichproben. Verglichen mit dem FCN Referenz-Netzwerk fällt der Unterschied mit 2,23% deutlicher aus. Durch das *Cross Learning* werden in 615 weiteren Stichproben Fehlklassifikationen vermieden.

5.5 Datenerzeugung durch Halluzination

Der Autoencoder in Abbildung 5.2 kann nach dem Training in Kapitel 5.3 zur Datenerzeugung verwendet werden. Dabei wird eine RDM eingespeist und am Ausgang des Decoders werden die entsprechenden Zusatz-Sensor-Daten erzeugt. Diese halluzinierten Pseudodaten können etwa für eine Anzeige der Kochfeldbelegung interessant sein, da Kamerabilder von Anwendern einfacher zu interpretieren sind als eine RDM. Ein Wärmebild visualisiert außerdem, welches Kochfeld heiß ist und kann damit vor möglichen Verbrennungen durch Berührung warnen. Weiterhin erlauben die halluzinierten Bilder eine Visualisierung der Ergebnisse des Autoencoder-Trainings aus Kapitel 5.3.

In der Literatur sind einige Anwendungsfälle für halluzinierte Daten in [105]–[107] gegeben. In [105] wird zu Bildbeschriftungen ein passendes Bild halluziniert. In [106] wird mithilfe eines Infrarot-Nachtsichtbildes ein RGB-Bild halluziniert. Dieses wird schließlich zur Gesichtserkennung verwendet. Die Halluzination eines RGB-Bildes aus dem Bild einer Tiefenkamera wird in [107] vorgestellt. Mithilfe des halluzinierten RGB-Bildes wird eine Objekterkennung verbessert.

In dieser Dissertation werden die halluzinierten Bilder nur als zusätzliche Visualisierung verwendet. Eine Klassifikation auf Grundlage der Halluzinationen wird nicht untersucht, wäre aber grundsätzlich denkbar. Einige Beispiele halluzinierter Daten sind in Abbildung 5.4 dargestellt. Links ist jeweils die real gemessene RDM abgebildet, die als Eingang für den Autoencoder verwendet wird. Daneben ist das halluzinierte Wärme- beziehungsweise Kamerabild dargestellt. In den beiden oberen Beispielen mit einem Topf sehen die RDM und das Kamerabild nahezu identisch aus. Im Wärmebild wird allerdings deutlich,

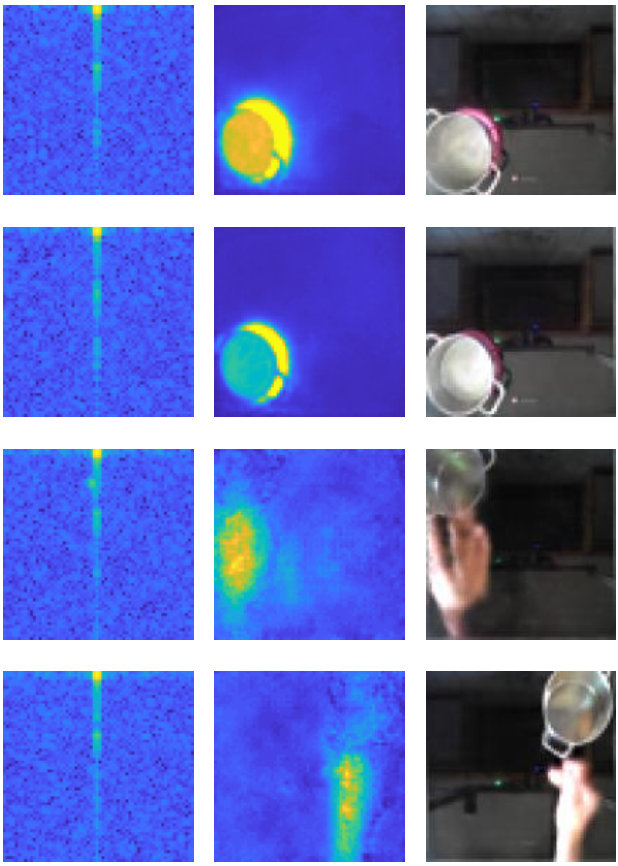


Abbildung 5.4: Beispiele halluzinierter Wärme- und Kamera-Bilder, die aus einer real gemessenen RDM erzeugt werden. Links: reale RDM, Mitte: halluziniertes Wärmebild, Rechts: halluziniertes Kamerabild. Erstveröffentlichung in [40], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/

dass der Topfinhalt einmal kocht und einmal nicht. Mittels dieser Visualisierung könnte etwa auf eine Verbrühungsgefahr hingewiesen werden.

Die beiden unteren Beispiele zeigen Gesten. Im Vergleich zu real aufgenommenen Bildern (Vergleich Abbildung 3.2 oder Abbildung 3.3) sind in den halluzinierten Bildern kaum Details vorhanden. Besonders in den Wärmebildern ist eine Hand nicht mehr als eine solche erkennbar, aber es ist ein Fleck mit entsprechender Körpertemperatur vorhanden. Der Detailverlust bei Gesten im Vergleich zu etwa den Kochpositionen ist auf zwei Gründe zurückzuführen. Zum einen können die Gesten in sehr vielen unterschiedlichen Variationen ausgeführt werden. Dies umfasst verschiedene Personen, Rechts- und Linkshänder sowie die Ausführung an verschiedenen Positionen zwischen Kochfeld und Sensoren. Hingegen sind bei den Kochpositionen nur kleinere Variationen in der Positionierung möglich. Als zweiter Grund sei auf den relativ kleinen Datensatz der Gesten hingewiesen.

Die Beispiele unterstreichen die Leistungsfähigkeit eines Autoencoders zur Encodierung und Decodierung von Bildern. Alle zur Decodierung notwendigen Informationen müssen prinzipbedingt in der komprimierten Wissensrepräsentation vorhanden sein. Damit wird deutlich, dass das cross-modale Autoencoder-Training beide Modalitäten in der komprimierten Wissensrepräsentation effizient verknüpft. Das *Transfer Learning* zur Klassifikation auf Basis dieser komprimierten Wissensrepräsentation bietet den entsprechenden Vorteil, aus mehreren Modalitäten gelernt zu haben, obwohl für die Klassifikation letztendlich nur eine Modalität verwendet wird. Wie die Experimente zeigen, führt dieses Vorgehen zu einer besseren Klassifikationsgenauigkeit.

5.6 Zusammenfassung und Vergleich aller Klassifikationsmethoden

In diesem Kapitel werden die Ergebnisse der verschiedenen Klassifikationsmethoden zusammengefasst. In Tabelle 5.5 sind die Ergebnisse der Einzelbild-Klassifikation (Kapitel 4.1), der Sequenz-Klassifikation (Kapitel 4.2), des FCN Referenz-Netzwerks (Kapitel 5.2) und schließlich des cross-modalem *Transfer Learning* (Kapitel 5.4) dargestellt. Bei letzterem wird das beste *Cross Learning* Netzwerk mit Wärmebildern und eingefrorenen Gewichtungen betrachtet.

Das insgesamt beste Ergebnis erzielt die Sequenz-Klassifikation auf Basis

Tabelle 5.5: Zusammenfassung aller Ergebnisse der RDM-Klassifikation. Dargestellt ist jeweils das beste Netzwerk der verschiedenen Ansätze.

	Einzelbild	Sequenz	FCN Ref.-Netzwerk	*Cross Learning* Wärmebild
RMSE	0,3841	0,0781	0,2041	0,0988
∅-Fehler	9,52%	0,41%	2,79%	0,56%
Kreis-Geste	7,37%	0,22%	5,32%	1,38%
Fingerreiben	22,67%	1,55%	16,43%	7,20%
Handneigung	28,68%	2,62%	15,90%	7,06%
Handbewegung	5,95%	0,30%	2,69%	0,00%
Hinten Links	14,89%	0,57%	3,48%	0,66%
Hinten Rechts	15,36%	0,60%	3,62%	0,58%
Vorne Links	4,00%	0,18%	1,17%	0,02%
Vorne Rechts	9,41%	0,40%	2,09%	0,32%
Gefüllter Topf	2,92%	0,08%	1,04%	0,02%
Kochen	3,10%	0,21%	1,20%	0,00%

einer CNN-Merkmalsextraktion gefolgt von einem rekurrenten LSTM-Layer. Dabei basiert die Klassifikation auf dem aktuellen und den vergangenen RDM. Die Informationen vergangener RDM werden im internen Speicher des LSTM-Layers gespeichert. Besonders für die dynamischen Klassen, wie den Gesten, ist die Sequenz-Klassifikation von großem Vorteil, da eine Geste nicht immer eindeutig mit einem Einzelbild abgebildet werden kann. Durch die zusätzlichen Informationen vergangener RDM kann die Klassifikation eindeutiger erfolgen und Verwechslungen mit anderen Gesten oder Klassen werden vermieden.

Bei den Klassen der Kochfeldbelegung sowie der Kocherkennung muss sich die Sequenz-Klassifikation dem *Cross Learning* geschlagen geben. Hierbei werden die Vorteile der zusätzlichen Sensordaten, wie dem Wärmebild, deutlich. Die kamerabasierten Systeme sind bestens zur Klassifikation der Kochfeldbelegung geeignet. Dies wird mithilfe des *Cross Learning* auf die RDM-Klassifikation übertragen. Besonders zur Kocherkennung ist die Wärmebildkamera durch die direkte Temperaturmessung der am besten geeignete Sensor. Die Sequenz-Klassifikation ist bei der Kocherkennung aber auch nur etwas schlechter, da das Kochen ebenfalls ein dynamischer Prozess ist, der mit einer Sequenz von RDM

besser abgebildet werden kann als mit einer Einzelmessung.

Die Einzelbild-Klassifikation wird deutlich vom *Cross Learning* übertroffen. Dahinter schlägt das FCN Referenz-Netzwerk das Netzwerk zur Einzelbild-Klassifikation aus Kapitel 4.1 ebenfalls mit Abstand. Das Referenz-Netzwerk weist etwa drei Mal so viele Layer auf. Da aber bis auf die Multi-Label-Klassifikation kein FC-Layer, sondern nur Conv-Layer verwendet werden, steigt die Anzahl an Parameter und der Speicherbedarf nur um etwa Faktor zwei statt drei. Hier sei noch einmal darauf hingewiesen, dass das Einzelbild-Netzwerk aus Kapitel 4.1 für einen anderen Datensatz mit weniger Klassen [37] und zur möglichst effizienten eingebetteten Inferenz entworfen wurde.

Die Vorteile der Sequenz-Klassifikation und des multimodalen *Cross Learning* könnten verknüpft werden, indem bei der Sequenz-Klassifikation als Merkmalsextraktion der multimodal vortrainierte Encoder verwendet wird. Auf weitere Experimente wird in dieser Dissertation allerdings verzichtet. Wie die Ergebnisse zeigen, besteht kaum noch Verbesserungspotential für Experimente mit diesem Datensatz. Die durchschnittliche Fehlerrate beträgt gerade noch 0,41%. Bezogen auf den Testdatensatz sind das 113 von 27.600 Stichproben. Für andere Datensätze oder Anwendungen könnte ein solcher Ansatz dennoch in Betracht gezogen werden.

Ein weiterer Ansatz zur multimodalen Sequenz-Klassifikation wäre die Implementierung eines LSTM-Autoencoders [115], [116] und anschließender Verwendung des rekurrenten Encoders mit LSTM-Speichermöglichkeiten zur Klassifikation.

6 Kognitive FMCW-Radar-Messungen

In den vorherigen Kapiteln sind die Radarparameter für alle Messungen konstant festgelegt. Die Parameter sind in Tabelle 2.1 aufgelistet. Die Größe der Entfernungs- und Geschwindigkeitszellen, sowie die maximale eindeutig messbare Entfernung und Geschwindigkeit sind damit ebenfalls festgelegt. In diesem Kapitel sind nun einige der Radarparameter adaptiv, sodass sich das Radar optimal an die zu klassifizierende Szene anpassen kann. Die adaptiven Parameter sind die:

- Bandbreite B
- Chirp-Dauer T
- Chirp-Wiederholdauer T_{CRT}
- Anzahl an Rampen der Chirp-Sequence L
- Falschalarmwahrscheinlichkeit der CFAR

Entsprechend der Gleichungen 2.8 und 2.11 können die resultierenden Größen der Geschwindigkeits- und Entfernungszellen in der RDM abhängig von den adaptiven Parametern berechnet werden. Die maximale eindeutig messbare Geschwindigkeit und Entfernung können mittels Gleichung 2.12 beziehungsweise Gleichung 2.10 berechnet werden. Für die adaptiven Parameter gelten hardwareseitige Randbedingungen. Weiterhin soll die Szene ressourcenschonend (Energiebedarf, Abwärme) und schnellstmöglich (wenige Messungen) erfasst werden.

Die Wertebereiche der adaptiven Parameter werden entsprechend der Möglichkeiten der realen Radar-Hardware gewählt. In den bereits veröffentlichten Ergebnissen in [39] wird das Texas Instruments® IWR1443 77 GHz Radar betrachtet. In der Veröffentlichung ist die Falschalarmwahrscheinlichkeit der CFAR nicht adaptiv. Es wird ein Szenario mit drei zufälligen Zielen betrachtet.

Tabelle 6.1: Diskrete Werte der adaptiven Radarparameter des 120 GHz Radars.

Symbol	Parameter	Diskrete Werte
B	Bandbreite	1 GHz, 2 GHz, 3 GHz, 4 GHz, 5 GHz
T	Chirp-Dauer	10 µs, 15 µs, 20 µs, 25 µs, 30 µs, 35 µs, 40 µs, 45 µs, 50 µs
L	Anzahl an Rampen	16, 32, 64
T_{CRT}	Chirp-Wiederholdauer	54 µs, 74 µs, 100 µs
CFAR	Falschalarmwahrscheinlichkeit	10^{-6}, 10^{-5}, 10^{-4}

In dieser Dissertation wird das 120 GHz MikroSens Radar verwendet (siehe Kapitel 2.3.1). Die Falschalarmwahrscheinlichkeit der CFAR wird als weiterer adaptiver Parameter hinzugefügt. Als erstes Experiment wird das Szenario mit drei zufälligen Zielen wiederholt. Im einem zweiten Experiment wird die Kochanwendung simuliert.

Der Wertebereich der adaptiven Parameter ist diskret. Die diskreten Werte sind in Tabelle 6.1 aufgelistet. In Summe gibt es 1215 Kombinationsmöglichkeiten. Aus den diskreten Werten können die Extremfälle, wie in Tabelle 6.2 und Tabelle 6.3 dargestellt, berechnet werden. Die kleinste maximal messbare Entfernung wird bei maximaler Bandbreite und minimaler Chirp-Dauer erreicht. Zwischen Bandbreite und Chirp-Dauer muss dabei ein Kompromiss gefunden werden, da die limitierte ADC-Abtastfrequenz und der verfügbare Speicher berücksichtigt werden müssen. Die Chirp-Dauer muss lang genug sein, um genügend Datenpunkte eines Chirps mit entsprechender Abtastfrequenz und unter Einhaltung des Abtasttheorems aufzuzeichnen. Aufgrund von Speicherlimits können 64 Entfernungszellen je Chirp gespeichert, beziehungsweise maximal eine 64-Punkt-FFT berechnet werden. Diese Limitierungen führen bei größerer maximal messbarer Entfernung zu einer Vergrößerung der Entfernungszellen und damit zu einer Verschlechterung der Auflösung. Hier gilt es also für die jeweilige Szene einen guten Kompromiss zu finden.

Bei der Geschwindigkeitsmessung wird die maximal messbare Geschwindigkeit maßgeblich durch die adaptive Chirp-Wiederholdauer beeinflusst. Die Geschwindigkeitsauflösung ist zusätzlich von der adaptiven Anzahl der Rampen abhängig

Tabelle 6.2: Extremfälle der maximal messbaren Entfernung des 120 GHz Radars auf Grundlage der diskreten Radarparameter.

Diskrete Werte	Maximale Entfernung	Größe Entfernungszelle
$B = 5\,\text{GHz}$ $T = 10\,\mu\text{s}$	0,9375 m	0,0146 m
$B = 1\,\text{GHz}$ $T = 50\,\mu\text{s}$	23,4375 m	0,3662 m

Tabelle 6.3: Extremfälle der maximal messbaren Geschwindigkeit des 120 GHz Radars auf Grundlage der diskreten Radarparameter.

Diskrete Werte	Maximale Geschwindigkeit	Größe Geschwindigkeitszelle
$L = 64$ $T_{\text{CRT}} = 100\,\mu\text{s}$	$\pm$6,1224 m/s	0,1913 m/s
$L = 16$ $T_{\text{CRT}} = 54\,\mu\text{s}$	$\pm$11,3379 m/s	1,4172 m/s

und wird grundsätzlich mit der Anzahl der Rampen verbessert. Damit nicht stets die maximale Anzahl an Rampen Verwendung findet, wird der Energiebedarf des Sensors berücksichtigt. Mehr Rampen führen zu einer größeren Einschaltdauer und damit zu einem höheren Energiebedarf und einer höherer Abwärme des Radarsensors. Neben der Energieeffizienz soll die zu klassifizierende Szene schnellstmöglich vollständig erfasst werden. Das heißt, die Anzahl benötigter Messungen zur Detektion aller Ziele in der Szene soll möglichst gering sein. Dies kann etwa durch eine geeignete Wahl der Radarparameter passend zur Szene erreicht werden. Mit einer minimalen Anzahl an Messungen wird die Szene schnellstmöglich vollständig erfasst und nochmals Energie gespart.

Eine schnellstmögliche und energieeffiziente Detektion aller Ziele in der Szene soll mittels einer adaptiven Parametrisierung, realisiert durch den Einsatz eines kognitiven FMCW-Radars, erreicht werden. Die Ziele in der Szene werden mit einem CFAR-Algorithmus in der RDM detektiert [45]. Die Falschalarmwahrscheinlichkeit ist dabei ebenfalls ein adaptiver Parameter. Anschließend folgt eine zweidimensionale Interpolation mithilfe benachbarter Entfernungs- und Geschwindigkeitszellen zur genaueren Bestimmung der Entfernung und

Geschwindigkeit des detektierten Ziels. Konkret wird der Schwerpunkt aller benachbarter Zellen oberhalb der CFAR-Schwelle berechnet. Schlussendlich zählt das Ziel als korrekt detektiert, falls die interpolierte Entfernung und Geschwindigkeit innerhalb einer festgelegten Toleranz zum wahren Wert liegen.

Das Kapitel kognitive FMCW-Radar-Messungen ist folgendermaßen aufgebaut. Im Abschnitt 6.1 werden zunächst die Literatur und die allgemeinen Grundlagen des kognitiven Radars zusammengefasst. Anschließend folgt in Abschnitt 6.2 die Vorstellung der Architektur für einen Reinforcement Learning Ansatz. Hierbei werden erneut die Netzwerke aus Kapitel 4.1 mittels *Transfer Learning* wiederverwendet. Als nächstes wird in Abschnitt 6.3 der Trainingsalgorithmus Proximal Policy Optimization (PPO) [117] vorgestellt. Zuletzt folgen die beiden Experimente mit der Drei-Ziel-Szene und der Kochanwendung.

6.1 Grundlagen des kognitiven Radars

Das Konzept des kognitiven Radars ist biologisch inspiriert. In der Tierwelt ist das Konzept in Form einer Echoortung etwa bei Fledermäusen oder Delphinen zu finden [118]. Die Echoortung dient den Tieren zur Orientierung und zum aktiven Aufspüren und Jagen von Beute [22], [118]. Das Verfolgen von Beute erfolgt dabei so, dass die ausgesendeten Schallwellen in der Dauer und Wiederholfrequenz an die Entfernung und Geschwindigkeit der Beute angepasst werden [119]. Diese Anpassung an die Beute ermöglicht es einer Fledermaus, sehr kleine Beute, wie etwa Insekten, selbst in der Dunkelheit zu jagen. Manche Arten von Delphinen sind in der Lage, kleine Fische in einer Entfernung bis zu 150 m in der rauen See zu orten [120]. Ähnlich wie neuronale Netzwerke mussten diese kognitiven Fähigkeiten der Tiere trainiert werden. Das biologische Training wird typischerweise Evolution genannt und nimmt sehr viel Zeit (Jahrtausende) in Anspruch.

Abgeleitet vom diesem Vorbild aus der Natur wurde das Konzept des kognitiven Radars entwickelt [21], [22], [121]. Praktische Anwendungen des kognitiven Radars sind beispielsweise die Interferenzvermeidung beim Automobil-Radar [24], die aktive Zielverfolgung zur Minimierung von Interferenz [23] oder ein Frequenzsprungverfahren zur elektronischen Gegenmaßnahme im Militärbereich [122]. Diese Beispiele basieren auf die ein oder andere Weise auf einer Erfassung oder Klassifikation der Umgebung, der sogenannten Wahrnehmung.

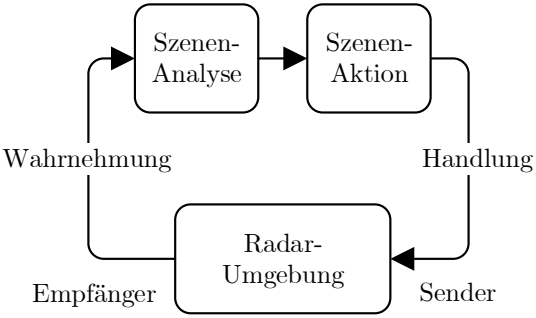


Abbildung 6.1: Das geschlossene Regelsystem aus Wahrnehmung und Handlung (Wahrnehmungs-Aktions-Schleife bzw. Perception-Action-Cycle nach [22]) bildet die Grundlage des kognitiven Radars.

Anhand dieser Wahrnehmung wird eine passende Handlungsstrategie entwickelt. Beim Radar kann eine bestimmte Handlung etwa mittels des Senders umgesetzt werden. Auf diese Weise schließt sich ein Kreis, da die Handlung wiederum einen Einfluss auf die Umgebung hat. Die Wahrnehmung erfolgt durch den Empfang der elektromagnetischen Radarwellen der Umgebung im Empfänger. Die Radarwellen können aktiv durch den Sender in die Umgebung eingebracht werden. Eine rein passive Nutzung vorhandener elektromagnetischen Wellen in der Umgebung ist ebenfalls möglich und wird passives kognitives Radar genannt [22], [123], [124].

Der geschlossene (Regel-) Kreis des kognitiven Radars wird in der Literatur Wahrnehmungs-Aktions-Schleife (Perception-Action-Cycle) [22], [121] genannt. Eine allgemeine Form für einen Radar ist in Abbildung 6.1 dargestellt. Die Handlung muss dabei nicht unbedingt einen direkten Einfluss auf die Umgebung haben, sondern kann beim Radar auch nur die eigene Wahrnehmung der Umgebung verändern. Beispiele mit direktem Einfluss wären etwa die bereits genannten Interferenzen, da die eigene Handlung einen Einfluss auf andere Radare in der Umgebung hat.

Die Radar-Umgebung kann prinzipiell eine reale oder eine simulierte Umgebung sein. Eine simulierte Umgebung muss entsprechend gut modelliert sein,

damit das System funktionieren kann. Das Gegenstück zur Umgebung ist die Szenen-Analyse und Szenen-Aktion. Diese beiden Blöcke werden oft zusammengefasst und als Agent bezeichnet. Für die Analyse kann etwa ein neuronales Netzwerk eingesetzt werden. Entsprechend der Analyse wird eine Szenen-Aktion ausgewählt. Das Ziel eines Agenten beziehungsweise seiner Handlungen ist typischerweise die Maximierung einer definierten Belohnung (engl. Reward). Die Bewertung der Handlung (Berechnung der Belohnung) erfolgt dabei oft durch die Radar-Umgebung selbst und wird als Teil der Wahrnehmung an die Szenen-Analyse zurückgegeben.

Mithilfe dieses geschlossenen Kreises kann ein Agent so trainiert werden, dass durch die Handlungen eine definierte Belohnung maximal wird. Ein solches Vorgehen ist in der Literatur weit verbreitet und wird insbesondere beim Einsatz neuronaler Netzwerke auch Reinforcement Learning (verstärkendes oder belohnendes Lernen) genannt [47], [57], [58]. Beispiele aus der Literatur sind sehr vielfältig und umfassen spielerische Anwendungen wie etwa Schach, Shogi und Go [125] oder Atari-Videospiele [126], aber auch verschiedene regelungstechnische Anwendungen wie das Ausbalancieren einer Stange auf einem Karren (Cartpole-Problem) oder dem Erlernen von Roboterbewegungen [127].

Die nachfolgenden Kapitel beschreiben die kognitive Architektur des Radars, welche beim Reinforcement Learning verwendet wird, im Detail. Anschließend wird der Trainingsalgorithmus PPO vorstellt. Abschließend werden die Experimente präsentiert und bewertet.

6.2 Kognitive Architektur

Die in dieser Dissertation verwendete kognitive Architektur ist von der allgemeinen Wahrnehmungs-Aktions-Schleife in Abbildung 6.1 abgeleitet. Die Szenen-Analyse und Szenen-Aktion wird durch einen trainierbaren Reinforcement Learning Agent umgesetzt. Das Ergebnis ist in Abbildung 6.2 dargestellt. Die für das Training notwendige Belohnung ist ergänzt und wird von der Radar-Umgebung erzeugt und an den Agenten übergeben. Bei der Beobachtung (engl. Observation) handelt es sich bei den Experimenten um eine RDM. Der Reinforcement Learning Agent besteht aus zwei separaten Netzwerken. Dieser Aufbau wird auch Actor-Critic-Architektur genannt [47], [128], [129]. Die Eingangsdaten der Netzwerke sind jeweils dieselbe RDM. Hierbei können die bisherigen Erkennt-

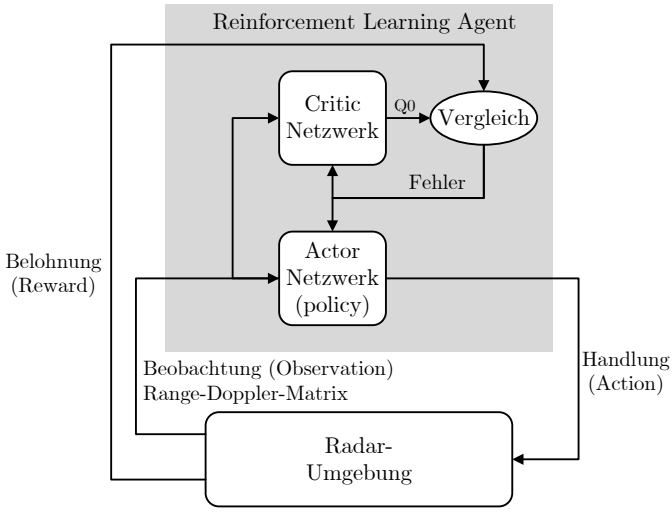


Abbildung 6.2: Die kognitive Architektur mit einem Reinforcement Learning Agent stellt eine Erweiterung der Wahrnehmungs-Aktions-Schleife dar. Der Agent setzt die Szenen-Analyse und Szenen-Aktion um. Die Wahrnehmung wird zum Trainieren des Agenten um eine Belohnung erweitert. Der Agent besteht aus zwei separaten Netzwerken. Das Actor-Netzwerk bestimmt die Handlung. Das Critic-Netzwerk beurteilt den Effekt der Handlung. Abbildung nach [39], © 2020 EuMA.

nisse zur Range-Doppler-Matrix-Klassifikation wiederverwendet werden. Ebenso ist ein *Transfer Learning* der Merkmalsextraktion mit den bereits trainierten Netzwerken aus den vorherigen Kapiteln möglich.

Das Actor-Netzwerk bestimmt auf Grundlage der Beobachtung die als nächstes auszuführende Handlung. Die Handlungen sind diskret (siehe Tabelle 6.1). Das Actor-Netzwerk ist also für die Vorgehensweise (engl. policy) zuständig. Das Critic-Netzwerk hingegen bestimmt den sogenannten Q0-Wert. Dies ist ein geschätzter Wert, wie viel Belohnung bei der aktuellen Beobachtung zu erwarten ist. Der Q0-Wert wird vereinfacht gesagt mit der tatsächlichen Belohnung verglichen, um einen Fehler für das Training zu berechnen. Mit diesem Fehler

werden letztendlich die beiden Netzwerke trainiert. Die Algorithmen und weitere Details werden im Kapitel 6.3 vorgestellt.

Die Wahrnehmungs-Aktions-Schleife wird iterativ durchlaufen. Ein solcher Durchlauf bis zum Erreichen einer Abbruchbedingung wird auch Episode genannt. Die Szene bleibt während einer Episode mit allen enthaltenen Zielen unverändert. Die Detektion aller Ziele ist eine Abbruchbedingung. Eine weitere Abbruchbedingung ist die maximale Anzahl an Messungen von zehn. Damit soll verhindert werden, dass bei einer ungünstigen Konstellation aus zufälligen Zielen und ungeeigneter Vorgehensweise des Agenten die Simulation vereinzelt sehr lange dauert. Gegebenenfalls können auch niemals alle Ziele detektiert werden und die Schleife würde nie enden. Ein Abbruch einer solchen Situation ist besonders für eine Parallelisierung der Simulationen wichtig. Wird eine der Abbruchbedingungen erfüllt, wird die aktuelle Schleife beziehungsweise Episode beendet. Für die nächste Episode werden die Ziele erneut zufällig in der Szene platziert.

6.2.1 Belohnung

Da das gesamte Training auf der Belohnung basiert, ist eine geeignete Definition der Belohnung essentiell für ein erfolgreiches Training und sinnvolle Handlungen des Agenten. Mithilfe der Belohnung kann die Vorgehensweise des Agenten in eine gewünschte Richtung gelenkt werden. Für eine schnellstmögliche und energieeffiziente Detektion aller Ziele in der Szene sollte also etwa die erfolgreiche Detektion eines Ziels belohnt werden. Viele Messungen hingegen sollten bestraft werden. Eine Bestrafung ist dabei als eine negative Belohnung zu verstehen.

Die Berechnung der Belohnung kann als die Summe verschiedener Teilbelohnungen definiert sein. In dieser Arbeit setzt sich die Belohnung aus folgenden Termen zusammen:

- $+1$ für jedes detektierte Ziel.
- $+50$ für die Detektion aller Ziele. Damit ist die gewünschte Aufgabe abgeschlossen.
- -3 für jedes nicht detektierte Ziel.
- $\frac{16-L}{64}$ (L: Anzahl Rampen der Chirp-Sequence)

- $-$*Anzahl an Messungen*. Mit zunehmender Anzahl der Messungen wird damit die Bestrafung vergrößert.

Die Definition und vor allem auch die Ausgewogenheit dieser Teilbelohnungen ist nicht trivial. Grundsätzlich sollten alle gewünschten Eigenschaften in der Definition repräsentiert sein. Ohne Belohnung beziehungsweise Bestrafung gibt es keinen Anlass für den Agenten in diese Richtung zu optimieren. Anschließend muss das richtige Maß der Teilbelohnungen zueinander gefunden werden. In diesem Beispiel ist etwa die Erfüllung der Aufgabe an sich, die Detektion aller Ziele, von größerer Wichtigkeit als die Energieeffizienz. Der Term für die Anzahl an Rampen je Chirp-Sequence-Messung ist so ausgelegt, dass mit 16 Rampen keine Bestrafung erfolgt. Mit 32 beziehungsweise 64 Rampen beträgt die Bestrafung entsprechend -0,25 und -0,75. Diese Bestrafung ist vergleichsweise gering. Die Bestrafung für Messungen nimmt hingegen linear mit der Anzahl an Messungen zu und kann maximal -10 betragen.

Die Belohnung für jedes detektierte Ziel ist mit $+1$ um Faktor 3 geringer als die Bestrafung für ein nicht detektiertes Ziel gewählt. Damit wird gewährleistet, dass im Falle von zwei detektierten und einem nicht detektierten Ziel weiterhin ein Bestrafung erfolgt.

Die Definition der Belohnung kann selbst als Optimierungsprozess betrachtet werden. Das Optimum der Terme kann iterativ gefunden werden und kann einige Zeit und Trainings mit verschiedenen Definitionen in Anspruch nehmen. In dieser Arbeit wird der Fokus auf die Detektion aller Ziele mit möglichst wenigen Messungen gelegt. Die Detektion aller Ziele wird daher am größten belohnt. Die Anzahl an Messungen und die nicht Detektion von Zielen wird am meisten bestraft.

Die Belohnung wird bei jeder Messung als Summe der Teilbelohnungen berechnet. Innerhalb einer Episode wird die Belohnung kumuliert. Mit der Belohnung (Reward) wird in der Regel die kumulierte Belohnung am Ende einer Episode gemeint. Beim Training wird typischerweise eine Maximierung der kumulierten Belohnung (auch langfristige Belohnung beziehungsweise long-term reward genannt) angestrebt. Damit wird dem Agenten ermöglicht, geringe oder gar negative Teilbelohnung in Kauf zu nehmen, um die (Haupt-) Aufgabe zu erfüllen. Im allgemeinen Sprachgebrauch und besonders im Schachspiel [125] wird das auch Bauernopfer genannt.

6.2.2 Radar-Umgebung

Die Radar-Umgebung wird in MATLAB® mithilfe der Phased Array System Toolbox simuliert. Das 120 GHz MikroSens Radar ist mit den Parametern nach Tabelle 2.1 beziehungsweise den adaptiven Parametern nach Tabelle 6.1 implementiert. Die Ziele sind als bewegliche Punktziele mit entsprechend abstandsabhängiger RCS implementiert. Die Bewegung der Ziele zwischen zwei aufeinanderfolgenden Messungen wird ebenfalls simuliert. Die Ausbreitung der Radarwellen wird im Freiraum angenommen. Ein typisches additives weißes gaußsches Rauschen wird dem Empfänger aufgeprägt. Aus den simulierten Radarsignalen wird schließlich die RDM berechnet, welche als Beobachtung für den Reinforcement Learning Agent dient. Beispiele der RDM beider Experimente sind in Abbildung 6.6 und 6.9 dargestellt. Das genaue Szenario der Radar-Umgebung wird in den jeweiligen Experimenten erläutert.

Die adaptiven Radarparameter haben wie Tabelle 6.2 und 6.3 zeigen einen großen Einfluss auf die Skalierung der RDM. Die Achsenskalierungen sollten deshalb stets beachtet werden. Manche Ziele können aufgrund mangelnder maximaler Entfernung überhaupt nicht in der RDM auftauchen. Weiterhin können bei mangelnder Auflösung zwei nahe beieinanderliegende Ziele in der RDM zu einem verschmieren und nicht unterschieden werden.

Die Größe der RDM beträgt immer 64x64. Die Anzahl der Entfernungszellen ist dabei stets 64. Die Anzahl der Geschwindigkeitszellen ist von der adaptiven Anzahl an Rampen abhängig. Da die verwendeten Netzwerke der kognitiven Architektur eine konstante Eingangsgröße voraussetzen, wird in Geschwindigkeitsrichtung gegebenenfalls Zero Padding (Anhängen von Nullen) bei der FFT verwendet.

Die Ziele werden innerhalb der markierten Zonen in Abbildung 6.6 beziehungsweise 6.9 zufällig platziert. Diese Platzierung erfolgt für jede Episode erneut zufällig. Innerhalb einer Episode bleibt die Geschwindigkeit eines Ziels konstant. Aufgrund der Bewegung der Ziele zwischen zwei aufeinanderfolgende Messungen kann sich die Entfernung während einer Episode verändern. Im Extremfall von maximal auftretender Geschwindigkeit sowie maximaler Chirp-Dauer und Anzahl, sind das $3\,\mathrm{m/s} \cdot 100\,\mu\mathrm{s} \cdot 64 \cdot 10 = 0{,}192\,\mathrm{m}$ innerhalb einer Episode. Dieser Effekt ist daher nicht zu vernachlässigen und muss mit simuliert werden.

Das Drei-Ziel-Szenario ist so gewählt, dass eine Messung mit ausreichend hoher maximaler Entfernung für Ziel 3 notwendig ist. Weiterhin ist eine hochauf-

lösende Messung im Nahbereich zur Unterscheidung von Ziel 1 und 2 notwendig. Typischerweise sind also nicht alle drei Ziele mittels einer einzigen Messung ausreichend genau detektierbar. Detektierbar heißt dabei, dass die interpolierte Entfernung und Geschwindigkeit innerhalb einer festgelegten Toleranz zum wahren Wert liegen muss.

Die im vorherigen Kapitel 6.2.1 vorgestellte Belohnung wird in der Radar-Umgebung berechnet. Dazu werden die drei CFAR-Detektionen mit den höchsten SNR mit den wahren Entfernungen und Geschwindigkeiten der Ziele verglichen. Unterhalb der maximalen Toleranz von 0,1 m beziehungsweise 0,1 m/s gilt das Ziel als detektiert und wird entsprechend gekennzeichnet. Für das weit entfernte Ziel 3 in dem Drei-Ziel-Szenario gilt die doppelte Toleranz. Ein Ziel muss innerhalb einer Episode nur einmalig detektiert und entsprechend gekennzeichnet werden. Dies ermöglicht die gewünschte Vorgehensweise des Agenten, mit verschiedenen Messungen innerhalb einer Episode verschiedene Ziele der Szene zu detektieren. Zuletzt werden von der Radar-Umgebung noch die Abbruchbedingungen überprüft und gegebenenfalls die Episode beendet.

6.2.3 Actor-Netzwerk

Der Actor beziehungsweise die Vorgehensweise (engl. policy) nach Abbildung 6.2 kann prinzipiell eine beliebige Funktion sein, die aus einer Beobachtung eine Handlung berechnet. Da die Definition oder Annäherung einer solchen Funktion alles andere als trivial ist, wird bei diesem Ansatz ein neuronales Netzwerk verwendet. Dieses kann mittels Daten die unbekannte Funktion erlernen und mit ausreichendem Training möglichst genau abbilden. Trainingsdaten sind hierbei prinzipiell unendlich vorhanden, da neue Daten beliebig oft simuliert werden können. Aus diesem Grund spielt die Radar-Umgebung selbst eine wichtige Rolle. Die Qualität der Trainingsdaten entspricht maßgeblich der Qualität der Radar-Umgebung.

Das Actor-Netzwerk ist in Abbildung 6.3 dargestellt. Als Eingang des Netzwerks wird eine 64x64 RDM in Form der Beobachtung verwendet. Das Eingangsbild wird zunächst von einem CNN verarbeitet. Die Architektur des CNN entspricht dem in Kapitel 4.1 verwendeten Netzwerk (Abbildung 4.2 ohne den Pooling-Layer). Damit wurden bereits gute Ergebnisse zur Interpretation von RDM gezeigt. Weiterhin ist durch die identische Architektur eine Wiederver-

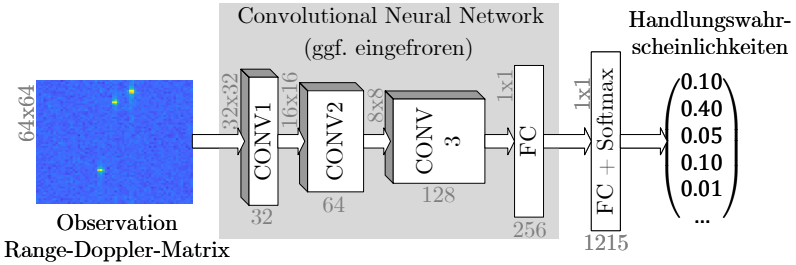


Abbildung 6.3: Das Actor-Netzwerk berechnet aus der Beobachtung eine Handlung. Das Netzwerk verwendet die CNN-Merkmalsextraktion der Einzelbild-Klassifikation wieder. Darauf folgt ein FC-Layer, der alle Kombinationsmöglichkeiten der adaptiven Parameter abbildet. Abbildung nach [39], © 2020 EuMA.

wendung, zumindest des CNN-Teils zur Merkmalsextraktion, durch ein *Transfer Learning* möglich. Nach bisherigen Erkenntnissen ist auch hier ein einfrieren der bereits trainierten Gewichtungen sinnvoll.

Nach der Merkmalsextraktion folgt ein noch untrainierter FC-Layer. Die Anzahl an Neuronen entspricht mit 1215 der Anzahl der diskreten Kombinationsmöglichkeiten der adaptiven Parameter. Hier handelt es sich um eine Multi-Class-Klassifikation mit sich gegenseitig ausschließenden Klassen, da immer nur eine Kombination der adaptiven Parameter als Handlung ausgeführt werden kann. Am Ausgang wird die Softmax-Funktion (Gleichung 2.16) verwendet und schließlich die Handlung mit der höchsten Wahrscheinlichkeit ausgeführt.

Während des Trainings wird das Netzwerk so trainiert, dass je nach Beobachtung jene Handlung ausgeführt wird, die zu maximaler langfristiger Belohnung führt. Anders ausgedrückt lernt das Netzwerk nicht unbedingt eine ideal parametrisierte Einzelmessung, sondern vielmehr eine sinnvolle Sequenz von Messungen zur Detektion aller Ziele und damit zur Erfüllung der Aufgabe. Obwohl in der Netzwerk-Architektur kein interner Speicher wie etwa ein LSTM-Layer verwendet wird, sind Informationen über zumindest die vergangene Handlung aufgrund der Wahrnehmungs-Aktions-Schleife vorhanden. Die vergangene Handlung hat schließlich einen direkten Einfluss auf die daraus resultierende Beobachtung.

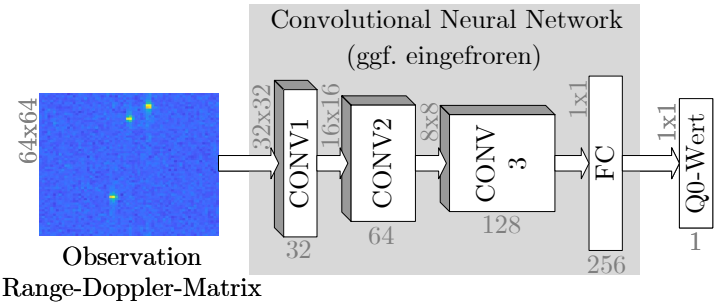


Abbildung 6.4: Das Critic-Netzwerk beurteilt auf Grundlage der Beobachtung den Effekt der Handlung. Die Beurteilung wird in Form des Q0-Werts ausgegeben. Zur Merkmalsextraktion wird das CNN-Netzwerk der Einzelbild-Klassifikation wiederverwendet. Abbildung nach [39], © 2020 EuMA.

6.2.4 Critic-Netzwerk

Das Critic-Netzwerk ist dem Actor-Netzwerk sehr ähnlich, hat aber wie Abbildung 6.2 zeigt, einen anderen Ausgang. Das Critic-Netzwerk berechnet aus der Beobachtung den Q0-Wert. Der Q0-Wert schätzt die zu erwartende Belohnung auf Grundlage der aktuellen Beobachtung und ist eine einzelne Zahl. Der Ausgang des Netzwerks ist daher ein einzelnes Neuron, das den Q0-Wert repräsentiert. Es handelt sich also im Gegensatz zum Actor-Netzwerk um kein Klassifikationsnetzwerk, sondern um ein Regressionsnetzwerk. Die Netzwerk-Architektur ist in Abbildung 6.4 dargestellt.

Die Eingangs-RDM wird zunächst von demselben CNN zur Merkmalsextraktion verarbeitet wie beim Actor-Netzwerk. Dieser Teil kann mittels *Transfer Learning* vortrainiert verwendet werden. Nach dem FC-Layer mit 256 Neuronen folgt nur noch ein einzelnes Neuron. Im Falle von eingefrorenen Gewichtungen des CNN-Teils müssen nur 256 Gewichtungen zwischen dem FC-Layer und dem einzelnen Neuron trainiert werden. Hinzu kommt das Bias des Q0-Wertes. Im Trainingsverlauf wird dieses Netzwerk so trainiert, dass die Differenz zwischen geschätzter Belohnung (Q0-Wert) und der Belohnung, welche von der Radar-Umgebung berechnet wird, minimal ist.

6.3 Training mittels Proximal Policy Optimization (PPO)

Das Training eines Reinforcement Learning Agenten ist maßgeblich von dessen Architektur abhängig. In dieser Arbeit wird die Actor-Critic-Architektur [47], [128], [129] verwendet. Dabei beurteilt der Critic die Vorgehensweise des Actors mittels des Q0-Werts. Der Trainingsalgorithmus für diese Art von Architektur wird daher auch *deep Q-Learning* [130]–[132] genannt. Vereinfacht gesagt wird, wie in Abbildung 6.2 dargestellt, die geschätzte kumulierte Belohnung (Q0-Wert) mit der tatsächlichen Belohnung verglichen und daraus ein Fehler berechnet. Dieser Fehler wird im Training für einen Backpropagation-Algorithmus verwendet (siehe Kapitel 2.2). Für die genaue Umsetzung gilt es viele Details zu beachten. Verschiedene Trainingsalgorithmen unterscheiden sich hauptsächlich darin, wie der Vergleich des Q0-Werts mit der Belohnung gestaltet ist und wie Trainingsdaten mithilfe der Wahrnehmungs-Aktions-Schleife erzeugt und verwendet werden.

Wie in Kapitel 6.2 erläutert, wird die Wahrnehmungs-Aktions-Schleife bis zu einer Abbruchbedingung iterativ durchlaufen. Ein Durchlauf entspricht einer Episode. Die dabei erzeugten Daten werden Erfahrung (engl. experience) genannt. Eine Erfahrung besteht in der Regel aus den Beobachtungen, den ausgeführten Handlungen und den resultierenden kumulierten Belohnungen innerhalb einer Episode. Eine solche Erfahrung könnte direkt dazu verwendet werden, die beiden Netzwerke zu trainieren. Ein anderer Ansatz ist, zunächst eine gewisse Anzahl an Erfahrungen als Trainingsdaten zu sammeln. Die Erfahrungen werden dann gesammelt in einem Batch zum Trainieren verwendet. Ein Vertreter dieses Ansatzes ist der State of the Art Algorithmus PPO [117], welcher 2017 von OpenAI vorgestellt wurde. In der Literatur wird die Batch-Größe des PPO-Algorithmus auch *PPO steps* genannt. Die Erzeugung der Erfahrungen kann und sollte parallelisiert ablaufen, da die Simulation der Wahrnehmungs-Aktions-Schleife, besonders aufgrund der Radar-Umgebung, viel Rechenzeit in Anspruch nimmt.

Eine weitere Besonderheit des PPO-Algorithmus ist, dass ein Batch von Erfahrungen stets mit derselben Vorgehensweise (mit demselben Actor) erzeugt wird. Der Batch von Erfahrungen wird dann nur einmalig für einen Trainingsschritt (Update) der Netzwerke verwendet. Die Erfahrungen werden anschließend verworfen. Neue Erfahrungen werden nachfolgend ausschließlich mit den geu-

pdateten Netzwerken erzeugt. Zusammen mit dem gemittelten Update über einen Batch an Erfahrungen wird dafür gesorgt, dass die Netzwerke stets nur in begrenztem Rahmen verändert werden können. Zur zusätzlichen Absicherung gibt es sogar einen Clipping-Faktor, der zu große Updates verhindert. Dieser Ansatz des PPO-Algorithmus soll das große Problem bei Reinforcement Learning Algorithmen verhindern, dass die Vorgehensweise sozusagen auf die schiefe Bahn gerät und nur noch schlechte Handlungen wählt. Die Vorteile begrenzter Updates werden in [133] (*Trust Region Policy Optimization*) dargestellt. Kurz gesagt ist das Training typischerweise gleichmäßiger und vermeidet Ausreißer. Ein großer Nachteil kleiner Updates ist ein langsameres Training. Dieser Nachteil kann aber mittels Rechenleistung beziehungsweise Parallelisierung kompensiert werden.

Der PPO-Fehler wird mithilfe der Generalized Advantage Estimation (GAE) [134] berechnet. Im Wesentlichen wird eine Episode rückwärts durchlaufen und die Differenz zwischen Q0-Wert und Belohnung für jede Iteration berechnet. Damit länger zurückliegende Handlungen weniger gewichtet werden, wird ein Diskontierungsfaktor (engl. discount factor) verwendet. Hinzu kommt ein weiterer Glättungsfaktor, der sogenannte GAE-Faktor. Mithilfe der GAE wird schließlich unter Berücksichtigung des Clipping-Faktors der PPO-Fehler berechnet. Zusätzlich wird beim Actor eine Entropie-Gewichtung eingeführt. Die Multi-Class-Klassifikation des Actors entspricht einer Wahrscheinlichkeitsverteilung aller Handlungen. Mithilfe der Entropie-Gewichtung kann das eindeutige Festlegen einer Handlung (hohe Entropie) als Fehler berücksichtigt werden. Auf diese Weise soll das Explorieren der (Radar-) Umgebung gefördert werden. Dies ist besonders hilfreich, falls der Agent in einem lokalen Minimum festhängt. Nur wenn durch Exploration alle Möglichkeiten der Umgebung erreicht werden können, kann die optimale Vorgehensweise gefunden werden.

Zum Trainieren des Agenten haben sich für die Radar-Umgebung folgende Trainingsparameter ergeben. Diese Parameter wurden in zahlreichen Trainings ermittelt und führen zum besten Ergebnis in den Experimenten.

- Diskontierungsfaktor: 0,99
- GAE-Faktor: 0,95
- Clipping-Faktor: 0,1

- Entropie-Gewichtung: 0,25
- Batch-Größe (*PPO steps*): 100
- Initiale Lernrate: 10^{-2} (Adam-Algorithmus)
- Epochen: 1 (Der Ansatz des PPO-Algorithmus ist, dass jeder Batch an Erfahrungen nur einmalig zum Trainieren verwendet wird.)

Das gewünschte Resultat des Trainings ist, dass dem Agenten eine schnellstmögliche und energieeffiziente Detektion aller Ziele in der Szene gelingt. Die Schnelle der Detektion aller Ziele kann im Trainingsverlauf anhand der Anzahl an Messungen je Episode dargestellt werden. Ein weiteres Maß zur Beurteilung des Agenten ist die kumulative Belohnung je Episode. Aufgrund des Explorationsverhaltens des Agenten und der zufällig erstellten Szenen können die Belohnung und Anzahl an Messungen je Episode sehr stark schwanken. Aus diesem Grund wird der Trainingsverlauf in den nachfolgenden Experimenten über 500 Episoden gemittelt (gleitender Durchschnitt) dargestellt. Damit ist der Trend des Trainings besser erkennbar.

Zur weiteren Beurteilung des Agenten können beispielsweise 500 Episoden simuliert werden ohne die Vorgehensweise zu ändern beziehungsweise zu trainieren. Die vom Agenten gewählten Parameter werden dabei aufgezeichnet und in Form eines Histogramms dargestellt. Als Beispiel sind in Abbildung 6.5 die Histogramme eines nicht trainierten Agenten dargestellt. Neben den Histogrammen der adaptiven Parameter Bandbreite, Chirp-Dauer, Chirp-Wiederholdauer, Anzahl an Rampen und Falschalarmwahrscheinlichkeit der CFAR ist auch das Histogramm der Anzahl an Messungen pro Episode dargestellt. Da der Agent frisch initialisiert und untrainiert ist, entsprechen die Histogramme der adaptiven Parameter in etwa einer Gleichverteilung. Die Anzahl an Messungen pro Episode beträgt zum Großteil zehn, da nach zehn Messungen die Abbruchbedingung der Radar-Umgebung greift. Zufälligerweise kann es dennoch passieren, dass der untrainierte Agent die passenden Parameter wählt und nach einigen Messungen alle Ziele detektiert.

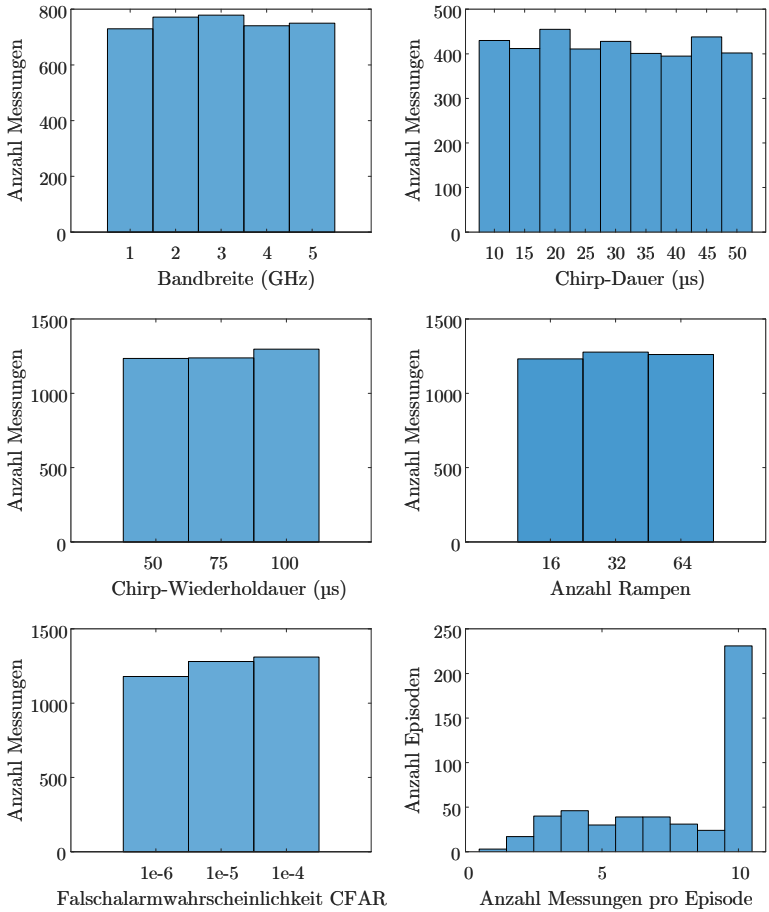


Abbildung 6.5: Die Histogramme der adaptiven Parameter eines frisch initialisierten Agenten vor dem Training zeigen in etwa eine Gleichverteilung. Die Anzahl an Messung entspricht größtenteils der Abbruchbedingung von maximal zehn Messungen pro Episode.

Tabelle 6.4: Mögliche Bereiche, in denen die Ziele im Drei-Ziel-Szenario zufällig generiert werden. Die Detektionstoleranz muss zur erfolgreichen CFAR-Detektion eingehalten werden.

Ziel	Möglicher Bereich	Detektionstoleranz
1	0,45 m .. 0,95 m	±0,1 m
	−1 m/s .. 1 m/s	±0,1 m/s
2	1 m .. 1,5 m	±0,1 m
	−1 m/s .. 1 m/s	±0,1 m/s
3	5 m .. 7 m	±0,2 m
	−3 m/s .. 3 m/s	±0,2 m/s

6.4 Experiment 1: Drei-Ziel-Szenario

Für ein erstes Experiment wird ein fiktives Drei-Ziel-Szenario erstellt. Die drei Ziele sind so gewählt, dass typischerweise nicht alle drei Ziele mittels einer einzigen Messung ausreichend genau detektierbar sind. Die Bereiche, in denen die Ziele zufällig für jede Episode generiert werden, sind in der RDM in Abbildung 6.6 dargestellt. Die Bereiche von Ziel 1 und 2 grenzen dabei direkt aneinander an. Damit diese Ziele getrennt voneinander detektiert werden können ist eine hochauflösende Messung im Nahbereich notwendig. Das dritte Ziel ist deutlich weiter entfernt und wird von einer Nahbereichsmessung nicht erfasst. Für das dritte Ziel sind daher andere Messparameter für eine höhere maximale Entfernung notwendig.

Tabelle 6.4 listet die möglichen Bereiche der Ziele auf. Zusätzlich ist die Detektionstoleranz angegeben. Die gemessene, interpolierte Entfernung und Geschwindigkeit muss zur erfolgreichen Detektion innerhalb dieser Toleranz zum wahren Wert liegen. Für das weiter entfernte Ziel 3 ist die Detektionstoleranz doppelt so groß gewählt, da mit größerer maximaler Entfernung die Größe einer Entfernungszelle zunimmt.

Das Drei-Ziel-Szenario ist absichtlich so designt, dass eine Einzelmessung in der Regel nicht ausreicht, um alle Ziele zu detektieren. Der Agent muss daher eine Sequenz von Messungen erlernen. Dazu wird das Prinzip der langfristigen Belohnung angewendet.

In Abbildung 6.7 ist der Trainingsverlauf für das Drei-Ziel-Szenario dargestellt.

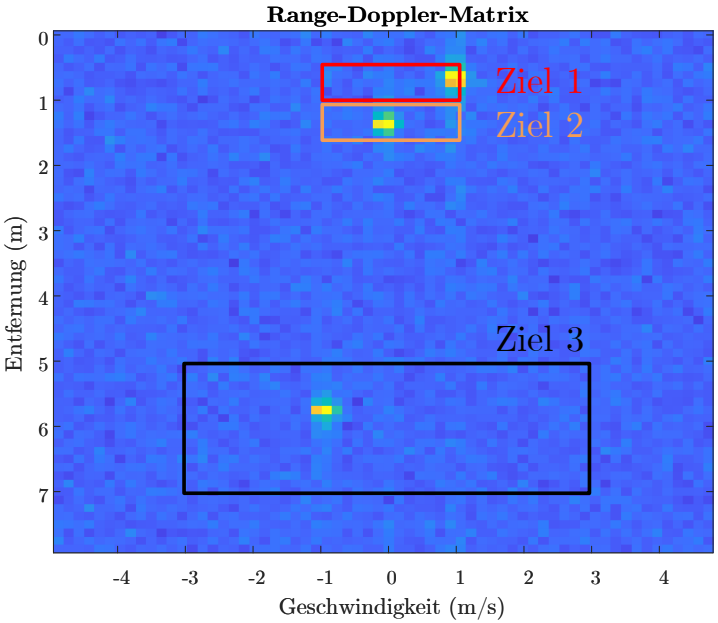
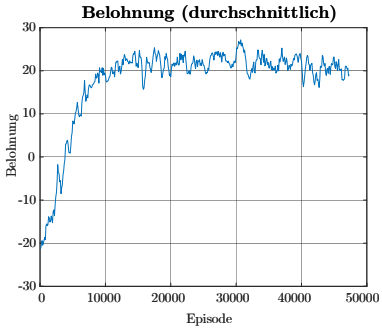
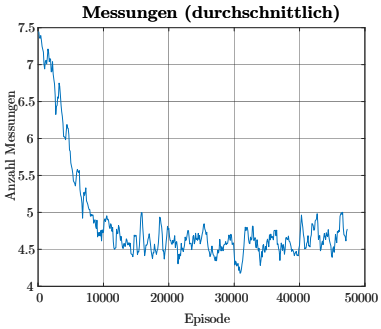


Abbildung 6.6: Simulierte RDM des Drei-Ziel-Szenarios. Die eingezeichneten Bereiche zeigen die möglichen Positionen der Ziele. Die Positionen werden für jede Episode zufällig gewählt. Abbildung nach [39], © 2020 EuMA.

Das Training wurde nach 47000 Episoden beendet. Der gleitende Durchschnitt der kumulativen Belohnung beträgt zu Beginn in etwa -20. Im Laufe des Trainings wird die Belohnung auf etwa +20 verbessert. Die Anzahl an Messungen nimmt dabei von etwa 7,5 zu 4,5 ab. Aufgrund des Designs der Belohnung ist eine hohe Belohnung im Prinzip gleichbedeutend mit einer niedrigen Anzahl an Messungen. Am Trainingsverlauf ist gut erkennbar, dass das Training nach etwa 10000 Episoden beendet werden könnte, da anschließend keine nennenswerte Verbesserungen mehr stattfindet. Aufgrund der Exploration gibt es dennoch bessere und schlechte Agenten. Ein besonders guter Agent stich etwa bei Episode



(a) Belohnung (Reward)



(b) Anzahl an Messungen

Abbildung 6.7: Gleitender Durchschnitt der kumulierten Belohnung und der Anzahl an Messungen im Trainingsverlauf des Drei-Ziel-Szenarios.

30700 heraus. Diese Agenten können explizit gespeichert und anschließend verwendet und weiter untersucht werden.

Der Trainingsverlauf der Anzahl an Messungen zeigt, dass der Idealfall von zwei Messungen pro Episode im gleitenden Durchschnitt nicht erreicht wird. Der gleitende Durchschnitt pendelt sich bei etwa 4,5 Messungen pro Episode ein. Es gibt also durchaus noch Verbesserungspotential für den Agenten. Zur genaueren Analyse wird der Agent mithilfe von Histogrammen untersucht.

Der beste Agent wird zur weiteren Beurteilung für 500 Episoden simuliert. Die entsprechenden Histogramme sind in Abbildung 6.8 dargestellt. Die Ergebnisse der Bandbreite und Chirp-Dauer deuten darauf hin, dass vom Agenten hauptsächlich zwei verschiedene Entfernungsmessungen gewählt werden. Die Nahbereichsmessung erfolgt mit einer Bandbreite von 5 GHz und einer Chirp-Dauer von 30 µs. Dies führt zu einer geringen maximal messbaren Entfernung bei einer guten Entfernungsauflösung. Auf diese Weise können die Ziele 1 und 2 unterschieden und detektiert werden. Die zweite Entfernungsmessung des Agenten erfolgt mit einer Bandbreite von 2 GHz und einer Chirp-Dauer von 40 µs. Diese Parameter führen nach Gleichung 2.10 zu einer maximal messbaren Entfernung von 9,375 m. Damit ist der mögliche Bereich des Ziels 3 abgedeckt

und Ziel 3 kann detektiert werden.

Für die Chirp-Wiederholdauer ist das Ergebnis des Agenten 100 µs. Dies führt nach Gleichungen 2.8 und 2.12 zu einer ausreichenden Geschwindigkeitsdynamik bei maximal möglicher Geschwindigkeitsauflösung. Einen geringeren Wert zu wählen verschlechtert die Geschwindigkeitsauflösung ohne einen Vorteil zu bieten. Diese Eigenschaft der Radar-Umgebung wird vom Agenten selbstständig erlernt.

Die Anzahl an Rampen beeinflusst nur die Geschwindigkeitsauflösung, welche mit der Anzahl an Rampen prinzipiell verbessert wird. Wie in Kapitel 6.2.1 beschrieben, ist der Term zur Bestrafung einer hohen Anzahl an Rampen pro Messung verhältnismäßig gering. Daher hat der Agent nur einen geringen Ansporn den Energiebedarf des Sensors hinsichtlich der Anzahl an Rampen zu optimieren. Zumeist wählt der Agent die maximale Anzahl an Rampen. Ein möglicher Grund für dieses Verhalten ist mit hoher Wahrscheinlichkeit das verwendete *Transfer Learning* beim Trainieren der Actor- und Critic-Netzwerke. Die besten Ergebnisse werden hier ebenfalls mit eingefrorenen Gewichtungen des CNN-Teils erreicht. Der CNN-Teil ist allerdings zur Merkmalsextraktion aus 64x64 RDM ausgelegt. Bei 16 oder 32 Rampen wird die RDM mittels Zero Padding (Anhängen von Nullen) hoch skaliert. Das kann zur Folge haben, dass bei weniger Rampen die Merkmalsextraktion keine guten Ergebnisse liefert. Der Agent meidet daher die Verwendung von weniger als 64 Rampen.

Bei der Falschalarmwahrscheinlichkeit der CFAR hat sich der Agent für den größten Wert 10^{-4} entschieden. Dies ist wenig überraschend, da die simulierte RDM im Prinzip nur additives weißes gaußsches Rauschen aufweist. Die Wahrscheinlichkeit ein nicht vorhandenes Falschziel im Rauschen zu detektieren ist damit sehr gering. Daher kann eine größere Falschalarmwahrscheinlichkeit toleriert werden. Der CFAR-Schwellwert kann damit eher niedrig gewählt werden. Das kommt schließlich der Detektion der drei Ziele zu Gute.

Im letzten Histogramm ist die Anzahl an Messungen pro Episode dargestellt. Wie erwartet werden größtenteils zwei Messungen pro Episode benötigt. Das sind eine Nahbereichsmessung und eine Messung mit höherer maximaler Entfernung für Ziel 3. Weitere Messungen sind teilweise nötig, die Anzahl nimmt allerdings kontinuierlich ab. Eine Einzelmessung tritt nur sehr selten auf, ist aber prinzipiell möglich, wenn Ziel 1 und 2 möglichst weit voneinander entfernt mit möglichst entgegengesetzten Geschwindigkeiten generiert werden. Häufiger hingegen tritt der Fall von zehn Messungen beziehungsweise dem Greifen der

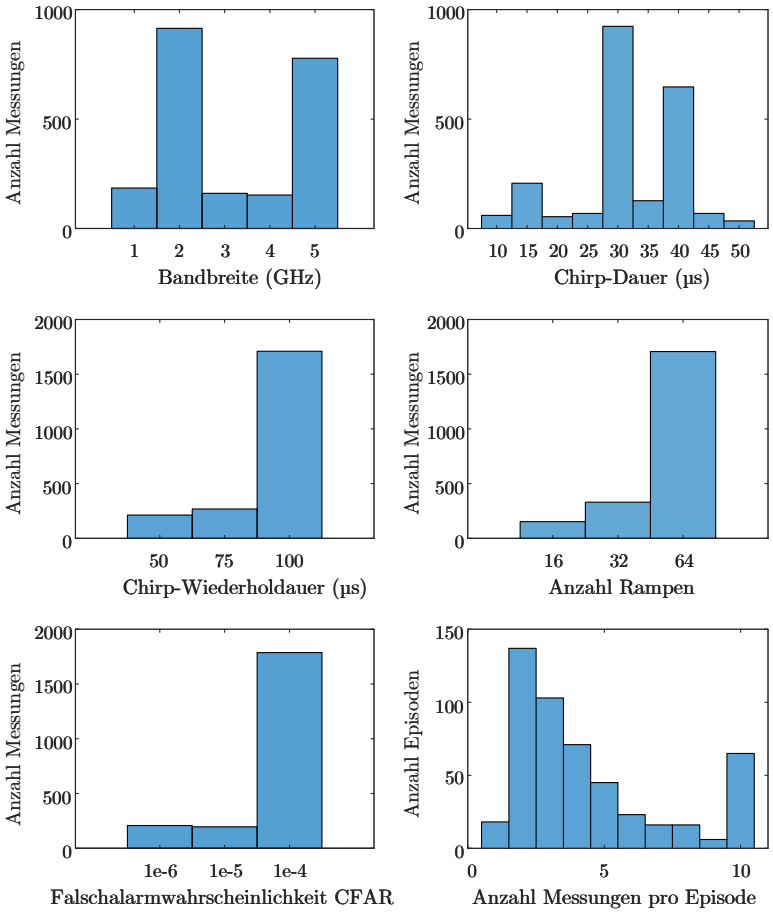


Abbildung 6.8: Die Histogramme der adaptiven Parameter im Drei-Ziel-Szenario nach dem Training zeigen, dass zumeist zwei Messungen benötigt werden. Eine Nahbereichsmessung mit 5 GHz und einer Chirp-Dauer von 30 µs, sowie eine Messung mit ausreichender maximaler Entfernung bei 2 GHz und einer Chirp-Dauer von 40 µs.

Abbruchbedingung ein. Mögliche Gründe sind das direkte Aneinandergrenzen von Ziel 1 und 2, sodass diese unabhängig von den adaptiven Parametern nicht getrennt voneinander detektiert werden können. Weiterhin sind auch Fehler in der Radar-Umgebung nicht ausgeschlossen. In manchen Episoden ist das SNR einzelner Ziele deutlich geringer als in anderen.

In diesem fiktiven Beispiel wurde die generelle Funktionsfähigkeit des kognitiven Radars zur adaptiven Parameterwahl demonstriert. Der Reinforcement Learning Agent ist in der Lage, eine Sequenz von Messungen mit nützlichen Parameterkombinationen zu erlernen, die zu einer langfristigen Belohnung führt. Im Endeffekt müssen die adaptiven Parameter sinnvoll kombiniert statt einzeln betrachtet werden. Auf diese Weise wurden zwei Parameterkombinationen und eine Sequenz von Messungen erlernt, mit denen alle Ziele in der Szene zumeist mit zwei aufeinanderfolgenden Messungen detektiert werden können.

6.5 Experiment 2: Kochanwendung

Das zweite Experiment mit kognitiven FMCW-Radar-Messungen simuliert das Hauptthema dieser Dissertation, die Kochanwendung. Die Ziele werden entsprechend der realen Kochanwendung, wie in Kapitel 3 beschrieben, modelliert. Dabei werden die real gemessenen Daten zur Modellierung berücksichtigt. Die wesentlichen Ziele sind zum einen die Gesten und zum anderen das Kochfeld beziehungsweise der Topf mit einer gegebenenfalls kochenden Flüssigkeit. Das Kochfeld wird entsprechend der realen Kochanwendung als Ziel in 0,7 m Entfernung modelliert. Je nach Vorhandensein und Füllstand eines Topfes kann die Entfernung zu diesem Ziel bis zu 0,1 m kürzer sein. Das Kochen wird durch eine sich für jede Messung ändernde Geschwindigkeit zwischen $-1\,\mathrm{m/s}$ und $1\,\mathrm{m/s}$ modelliert. Damit werden die charakteristischen und zufällig verteilten Geschwindigkeiten der sprudelnden Flüssigkeitsoberfläche berücksichtigt. Die Gesten treten zwischen Kochfeld und Radar mit einer typischen Geschwindigkeit von bis zu $\pm 3\,\mathrm{m/s}$ auf. Die Modellierung entspricht der *Kreis-Geste*, welche der Geste mit größter Amplitude und Geschwindigkeit entspricht. Die verschiedenen Gesten werden nicht modelliert. Die möglichen Bereiche der beiden Ziele werden in Tabelle 6.5 aufgelistet.

Eine simulierte RDM der Kochanwendung ist in Abbildung 6.9 dargestellt. Im Vergleich zu einer real gemessenen RDM (Vgl. Abbildung 3.1) ist in der

Tabelle 6.5: Mögliche Bereiche, in denen die Ziele in der Kochanwendung zufällig generiert werden. Die Detektionstoleranz muss zur erfolgreichen CFAR-Detektion eingehalten werden.

Ziel	Möglicher Bereich	Detektionstoleranz
1	0,1 m .. 0,5 m	±0,1 m
	−3 m/s .. 3 m/s	±0,1 m/s
2	0,6 m .. 0,7 m	±0,1 m
	−1 m/s .. 1 m/s	±0,1 m/s

simulierten Version weniger Rauschen vorhanden. Der Bereich unmittelbar am Radar und Nullgeschwindigkeit weist beim realen Radar den typischen Effekt der Antennenkopplungen auf, der diesen Bereich dominiert. In der Praxis kann diese statische Störung kompensiert werden. Für die Verarbeitung mit einem neuronalen Netzwerk sind statische Störungen allerdings nicht von Bedeutung. Dies wurde in der eigenen Veröffentlichung [37] mithilfe der Technik *Deep Dream* [60], [61] gezeigt. Daher wird auf die Modellierung statischer Störungen in der Radar-Umgebung der Kochanwendung verzichtet. Die in der Realität auftretenden Geisterziele aufgrund der Mehrwegeausbreitung werden ebenfalls nicht modelliert.

Die Radar-Umgebung der Kochanwendung wird für das Reinforcement Learning ausschließlich simuliert. Die Simulation erfolgt wie in Kapitel 6.2.2 beschrieben. Lediglich die möglichen Positionen der Ziele werden verändert und auf zwei reduziert (siehe Tabelle 6.5). Die Simulation ermöglicht eine Parallelisierung und somit ein verhältnismäßig schnelles Generieren von Trainingsdaten beziehungsweise Erfahrung. Die Generierung von Erfahrung in der realen Kochanwendung würde ein ständiges Umprogrammieren des 120 GHz Sensors erfordern. Speziell bei der Ausführung von Gesten könnte das nicht schnell genug während der Geste erfolgen. Bevor das Radar das nächste Mal messen könnte, wäre die Geste bereits vorbei. Ein ähnliches Problem gibt es auch beim Kochen, das ein dynamischer Prozess ist. Daten aus der realen Kochanwendung werden daher nicht zum Reinforcement Learning verwendet.

Im Gegensatz zum vorherigen Drei-Ziel-Szenario Experiment ist die Detektion der Ziele in der Kochanwendung mit einer Einzelmessung zu erreichen. Das Erlernen einer Messsequenz ist also nicht von Bedeutung beziehungsweise

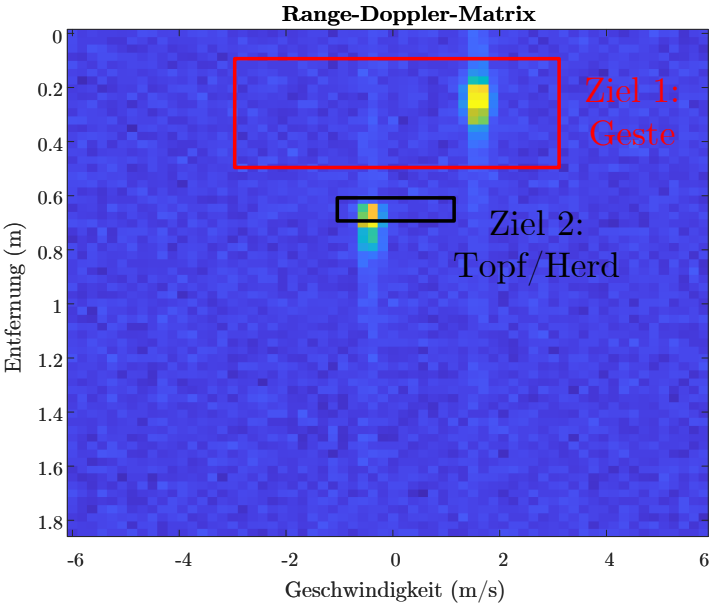
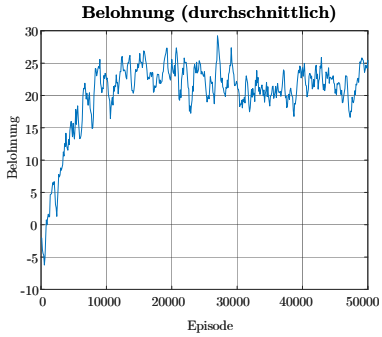


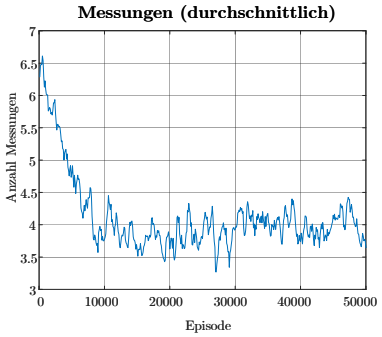
Abbildung 6.9: Simulierte RDM der Kochanwendung mit zwei Zielen. Ziel 1 entspricht einer Geste. Ziel 2 zeigt einen Topf beziehungsweise die Herdoberfläche. Das Kochen wird mittels Bewegungen in der Entfernung des Topfs simuliert.

überhaupt erforderlich. Mit diesem Experiment soll stattdessen gezeigt werden, dass der Reinforcement Learning Agent in der Lage ist, die optimalen Parameter für die Kochanwendung selbstständig zu finden. Im Idealfall sind die gelernten Parameter identisch mit den manuell gewählten Parametern der Kochanwendung.

Der Trainingsverlauf der Kochanwendung ist in Abbildung 6.10 dargestellt. Das Training lief für 50000 Episoden. Der gleitende Durchschnitt der kumulativen Belohnung beträgt zu Beginn in etwa -5. Im Laufe des Trainings kann die Belohnung auf bis zu +29 verbessert werden. Die Anzahl an Messungen nimmt im Trainingsverlauf von etwa 6,5 auf 3,5 ab. Auch hier ist der Effekt erkennbar,



(a) Belohnung (Reward)



(b) Anzahl an Messungen

Abbildung 6.10: Gleitender Durchschnitt der kumulierten Belohnung und der Anzahl an Messungen im Trainingsverlauf der Kochanwendung.

dass nach etwa 10000 Episoden keine generelle Verbesserungen mehr stattfinden. Im weiteren Verlauf pendelt der gleitende Durchschnitt der Belohnung um etwa 22. Ein Ausreißer bei etwa Episode 27000 führt zu dem besten Agent des Trainings. Zur genaueren Analyse wird dieser Agent für 500 Episoden simuliert und die Handlungen aufgezeichnet.

Die Histogramme der Handlungen sind in Abbildung 6.11 dargestellt. Die Ergebnisse der Bandbreite, Chirp-Dauer, Chirp-Wiederholdauer und der Anzahl an Rampen zeigen jeweils nur einen Spitzenwert für die benötigte Einzelmessung. Dabei handelt es sich um die vom Agenten für optimal befundenen Parameter für die Kochanwendung. Ein Vergleich mit den manuell gewählten Parametern in der Kochanwendung ist in Tabelle 6.6 gegeben. Bis auf die Chirp-Wiederholdauer sind die Parameter im Prinzip identisch beziehungsweise bestmöglich angenähert. Die Chirp-Wiederholdauer wurde allerdings manuell nicht optimal gewählt und der Agent hat den besseren Parameter gefunden. Wie Gleichung 2.8 und 2.12 zeigen, führen die 100 µs zu einer ausreichenden Geschwindigkeitsdynamik von $\pm 3\,\mathrm{m/s}$ für die Kochanwendung bei gleichzeitig maximal möglicher Geschwindigkeitsauflösung. Einen geringeren Wert zu wählen macht daher für den Agenten keinen Sinn, da die Erhöhung der Geschwindigkeitsdynamik eine

Tabelle 6.6: Vergleich der manuell gewählten Messparameter in der Kochanwendung und der vom Agenten gefundenen Parameter in der Simulation.

Parameter	Manuell gewählt	Agent
Bandbreite	5 GHz	5 GHz
Chirp-Dauer	20 µs	23,48 µs
Chirp-Wiederholdauer	100 µs	54 µs
Anzahl an Rampen	64	64

schlechtere Geschwindigkeitsauflösung zur Folge hat.

Bei der manuellen Wahl der Chirp-Wiederholdauer war die erforderliche Geschwindigkeitsdynamik für die Kochanwendung zu Beginn der Experimente nicht bekannt. Menschen sind theoretisch in der Lage, sehr schnelle Bewegungen auszuführen. Besonders bei den Gesten kann die Geschwindigkeitsdynamik von $\pm 3\,\mathrm{m/s}$ einfach überschritten werden. Aus diesem Grund wurde zunächst eine größere Geschwindigkeitsdynamik gewählt. Praktisch gesehen führt aber niemand die Gesten an einem Kochfeld so schnell aus. Die Chirp-Wiederholdauer wurde dann allerdings nicht an praktisch auftretende Geschwindigkeiten im Datensatz angepasst, da der Datensatz sonst erneut mit anderen Parametern hätte aufgenommen werden müssen.

Die Anzahl an Rampen wird wie in der Anwendung mit 64 gewählt. Der Grund für dieses Verhalten dürfte erneut beim *Transfer Learning* der Actor- und Critic-Netzwerke mit eingefrorenen Gewichtungen liegen. Der vortrainierte CNN-Teil ist zur Merkmalsextraktion aus 64x64 RDM ausgelegt und führt entsprechend nur mit 64 Rampen zu brauchbaren Ergebnissen. Der Agent meidet daher die Verwendung von weniger als 64 Rampen.

Die Falschalarmwahrscheinlichkeit der CFAR entspricht weitestgehend dem Ergebnis des vorherigen Experiments. Aufgrund der Simulation tritt auch hier nur additives weißes gaußsches Rauschen in der RDM auf (siehe Abbildung 6.9). In der Praxis weist die RDM der Kochanwendung viele Geisterziele aufgrund der Mehrwegeausbreitung auf (siehe Abbildung 3.1). Dies würde in der Praxis eine niedrigere Falschalarmwahrscheinlichkeit zur Unterdrückung der Geisterziele erfordern. Hingegen ist in der Simulation ohne Geisterziele ein niedriger CFAR-Schwellwert und damit eine größere Falschalarmwahrscheinlichkeit tolerierbar.

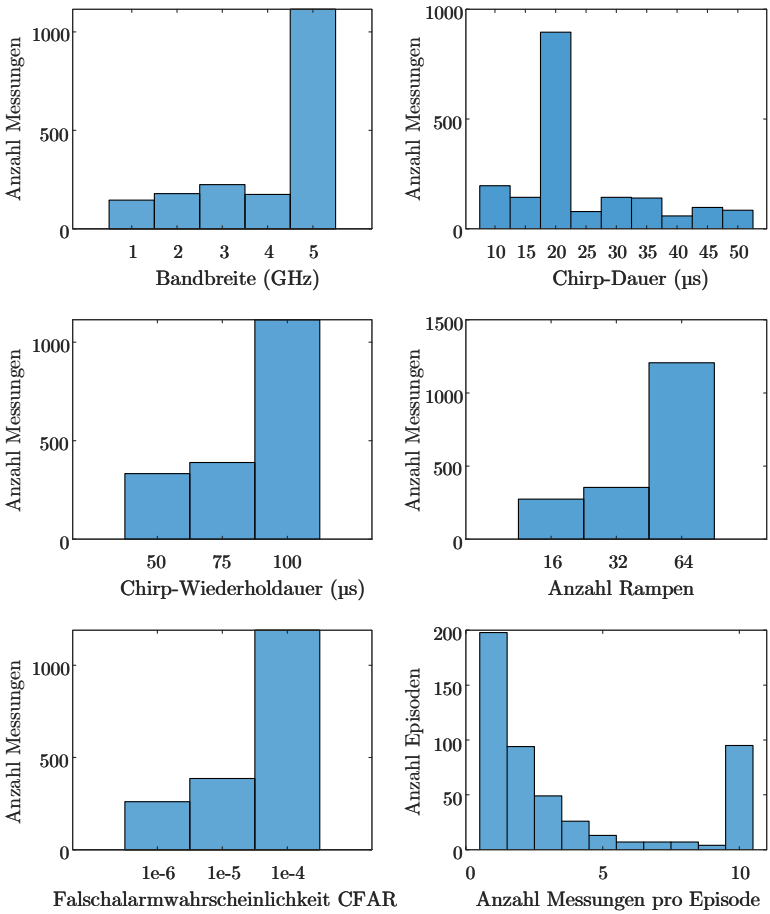


Abbildung 6.11: Die Histogramme der adaptiven Parameter in der Kochanwendung zeigen die vom Agenten erlernten Parameter. Die Parameter stimmen weitestgehend mit den manuell in der Anwendung gewählten Parametern überein. Wie erwartet wird zumeist nur eine Einzelmessung pro Episode benötigt.

Das Histogramm zu der Anzahl an Messungen pro Episode zeigt, dass am häufigsten nur eine Einzelmessung erforderlich ist. Anhand des Designs der Radar-Umgebung in diesem Experiment wird dies auch so erwartet. Weiterhin haben verschiedene Experimente in dieser Dissertation gezeigt, dass eine Einzelmessung auch in der praktischen Kochanwendung ausreichend ist (siehe Kapitel 5.6). Allerdings ist wie im ersten Experiment zum Drei-Ziel-Szenario der Effekt zu beobachten, dass der Fall von zehn Messungen beziehungsweise dem Greifen der Abbruchbedingung relativ häufig auftritt. Dies bestärkt die Theorie eines systematischen Fehlers in der Radar-Umgebung selbst. Auch hier ist in manchen Episoden das SNR einzelner Ziele deutlich geringer als in anderen und das Ziel kann unabhängig von den gewählten adaptiven Parametern nicht detektiert werden.

Zusammenfassend hat das zweite Experiment gezeigt, dass die Simulation und Praxis zu einer ähnlichen Parameterwahl und im Prinzip zu demselben Ergebnis führt. Vorausgesetzt die adaptiven Parameter werden passend gewählt, können die wesentlichen Bestandteile der Kochszene mit einer Einzelmessung erkannt werden. Dabei ist allerdings zu beachten, dass es sich bei der Simulation um eine reine CFAR-Detektion der Ziele handelt. In den praktischen Beispielen in dieser Dissertation steht die Klassifikation der Szene im Fokus. Ein nächster logischer Schritt wäre daher, die Belohnung des Reinforcement Learning Agenten an eine Klassifikationsgenauigkeit der Szene zu koppeln. Die korrekte Klassifikation wird sozusagen belohnt. Auf diese Weise wäre es das Ziel des Agenten, die langfristige Klassifikationsgenauigkeit zu maximieren. Diese Arbeit hat gezeigt, dass auch solche komplexe Anwendungen lernender Verfahren für Radarsensorik aussichtsreich sind.

6.6 Zusammenfassung: Kognitives Radar

In diesem Kapitel wurde das vierte Kernziel dieser Arbeit, die adaptive Parametrisierung kognitiver FMCW-Radar-Messungen zur ressourcenschonenden Szenen-Detektion, erreicht. Dazu wurden zwei Experimente durchgeführt, die die verschiedenen Aspekte des kognitiven Radars aufzeigen. Im Drei-Ziel-Szenario lernt der Reinforcement Learning Agent eine Sequenz von Messungen mit sinnvollen Parameterkombinationen, die zu einer langfristigen Belohnung führt. Das zweite Experiment ist die Kochanwendung. Der Reinforcement Learning Agent

erlernt die nahezu gleichen Parameter, wie sie in der Kochanwendung manuell gewählt wurden. Das kognitive Radar wurde als ein Reinforcement Learning Agent umgesetzt, der mit einer simulierten Radar-Umgebung interagiert. Der Reinforcement Learning Agent beinhaltet zwei neuronale Netzwerke in der Actor-Critic-Architektur, die mittels PPO-Algorithmus und *Transfer Learning* trainiert wurden. Dabei beurteilt der Critic die Vorgehensweise des Actors.

7 Zusammenfassung

Lernende Verfahren gewinnen im Bereich der Szenenerkennung und Klassifikation stetig an Bedeutung. Von der Leistungsfähigkeit neuronaler Netzwerke profitieren auch zunehmend Radarsensoren. In dieser Arbeit wurden verschiedene lernende Verfahren zur Szenenerkennung mit einem Radar realisiert. Die Verfahren wurden anhand der Beispielanwendung Kochsensor evaluiert und sukzessive weiter verbessert. Die Verfahren umfassen zum einen die Klassifikation der Radardaten mithilfe verschiedener tiefer neuronaler Netzwerke. Zum anderen wird ein Verfahren behandelt, bei dem das Radar adaptiv auf die Szene reagieren und die Messparameter entsprechend anpassen kann. In dieser Arbeit konnte gezeigt werden, dass mittels entsprechend angepasster lernender Verfahren sehr gute Ergebnisse auch für komplexe Anwendungen von Radarsensorik erzielt werden können.

Die Radardaten liegen in dieser Arbeit in Form einer Range-Doppler-Matrix (RDM) vor. Damit können Entfernungen und relative Geschwindigkeiten abgebildet werden. Als Datensatz wird die dynamische Szene eines Kochsensors inklusive Gestensteuerung betrachtet. Dessen Hauptfunktion ist ein Überkochschutz für Flüssigkeiten auf einem 2x2 Kochfeld. Die aktiven Kochfelder werden dabei detektiert und überwacht. Ein Überkochen oder Anbrennen wird durch rechtzeitiges Ausschalten oder durch Reduzierung der Heizleistung verhindert. Mithilfe der Gestensteuerung kann das Kochfeld manuell ein- und ausgeschaltet, sowie die Heizleistung eingestellt werden.

Als Radar wird ein 120 GHz FMCW-Radar verwendet, das im Projekt MikroSens entwickelt wurde. Für das Verfahren *Cross Learning* werden zusätzliche Sensordaten von einer Wärmebild- und Farbkamera erfasst. Die Sensoren befinden sich etwa 0,7 m oberhalb des Kochfelds auf Höhe einer Dunstabzugshaube. Der Datensatz bildet zehn verschiedene Klassen ab und beinhaltet insgesamt 57.600 Stichproben.

Die lernenden Verfahren dieser Dissertation zeigen verschiedene Möglichkeiten zur RDM-Klassifikation. Zunächst erfolgt die Klassifikation mit einer

einzelnen RDM (Einzelbild-Klassifikation). Dieses Verfahren zeigt speziell bei dynamischen Klassen, wie den Gesten oder dem Kochen, Schwächen. Deutlich bessere Ergebnisse zeigt die Sequenz-Klassifikation, bei der eine Sequenz von Messungen mithilfe netzwerkinterner Speicher ausgewertet wird. Ein ähnlich gutes Ergebnis wird erreicht, wenn bei der Einzelbild-Klassifikation zusätzliche Sensormodalitäten zum Trainieren des Klassifikationsnetzwerks herangezogen werden. Bei diesem sogenannten *Cross Learning* werden die zusätzlichen Sensoren allerdings nur zum Trainieren und nicht zur Klassifikation benötigt. Ein weiteres lernendes Verfahren ist das Reinforcement Learning, das einem Radar kognitives Verhalten erlaubt. Das Radar kann adaptiv auf die Szene reagieren und die adaptiven Messparameter optimal wählen.

Bei der Einzelbild-Klassifikation steht das Verhältnis zwischen Genauigkeit und der Anzahl an Parametern des neuronalen Netzwerks im Fokus. Dazu wird der Einfluss verschiedener Schichten wie Maximalwert-Pooling, Batch Normalization und Dropout untersucht. Das beste Ergebnis wird ohne die Verwendung eines Maximalwert-Pooling-Layers und dem kombinierten Einsatz von Batch Normalization und Dropout erreicht. Die durchschnittliche Fehlerrate beträgt 9,52%. Da es sich bei dem Maximalwert-Pooling-Layer um ein räumliches Downsampling handelt, ist der Einsatz bei kleinen Eingangsdaten, wie einer 64x64 RDM, nicht sinnvoll. Die Kombination aus Batch Normalization und Dropout führt zu einer guten Regulierungen des Trainingsverlaufs und schließt weitestgehend die Generalisierungslücke, indem eine Überanpassung vermieden wird. Das erste Kernziel dieser Arbeit, die Klassifikation einzelner RDM mittels möglichst effizienter neuronaler Netzwerke zur eingebetteten Inferenz, ist damit erreicht.

Bei der Sequenz-Klassifikation handelt es sich um eine signifikante Weiterentwicklung der Einzelbild-Klassifikation. Statt einem Einzelbild wird eine zeitliche Sequenz von RDM ausgewertet. Ein Netzwerk mit internen Speichermöglichkeiten (LSTM-Layer) speichert dazu die Informationen aus vergangenen RDM. Jede einzelne Messung kann bei der Ausführung direkt auf Grundlage der aktuellen Messung und der gespeicherten Information klassifiziert werden. Somit entstehen in der Anwendung keine Nachteile oder Verzögerungen. Mithilfe der Sequenz-Klassifikation kann die Dynamik des Datensatzes, wie das Kochen oder die Ausführung von Gesten, besser abgebildet und klassifiziert werden. Die durchschnittliche Fehlerrate beträgt 0,41% und ist damit das beste Ergebnis,

das in dieser Dissertation erreicht wird. Das zweite Kernziel dieser Arbeit, die Sequenz-Klassifikation einer dynamischen Szene, ist somit erreicht.

Der Teil zur Merkmalsextraktion im LSTM-Netzwerk kann von der bereits trainierten Einzelbild-Klassifikation übernommen werden. Bei diesem sogenannten *Transfer Learning* können die bereits trainierten Gewichtungen optional eingefroren werden, damit diese nicht weiter verändert werden können. Neben einem deutlich schnelleren Training werden mittels *Transfer Learning* in den allermeisten Trainings bessere Ergebnisse erzielt, als wenn das Netzwerk frisch initialisiert trainiert wird. Das Einfrieren der Gewichtungen ist dabei besonders zu Beginn des Trainings zu empfehlen, da sonst die bereits trainierten Gewichtungen überschrieben werden.

Das *Cross Learning* ist eine weitere Möglichkeit ein vortrainiertes Netzwerk für ein *Transfer Learning* zu erzeugen. Dazu wird zunächst ein Autoencoder unüberwacht trainiert. Unüberwacht heißt, dass die Klassen nicht bekannt sind, beziehungsweise keine Klassifikation erfolgt. Anschließend wird der Encoder-Teil des Autoencoders als vortrainiertes Netzwerk für ein überwachtes Lernen mit den bekannten Klassen der Kochanwendung verwendet. Das besondere an dem *Cross Learning*-Verfahren ist, dass das Autoencoder-Training mittels zusätzlicher Sensordaten multimodal erfolgt. Dabei werden die Vorteile verschiedener Sensoren kombiniert. Die Wärmebildkamera hat sich dabei als besonders hilfreich für die Kochanwendung herausgestellt. In der Anwendung wird weiterhin alleinig der Radarsensor verwendet. Die zusätzlichen Sensoren sind nur für das multimodale Training erforderlich. Im Gegensatz zur Sequenz-Klassifikation erfolgt die Klassifikation hier wieder nur mit einem Einzelbild. Die durchschnittliche Fehlerrate ist mit 0,56% aber auf dem Niveau der Sequenz-Klassifikation. Damit ist das dritte Kernziel dieser Arbeit, die Kombination verschiedener Sensormodalitäten zur unimodalen Inferenz, erreicht.

Das kognitive Radar optimiert die Szenenerkennung, indem adaptiv auf die erkannte Szene reagiert wird. Die Erkennung erfolgt mithilfe der Netzwerke, die auch zur Klassifikation verwendet wurden. Dabei kommt erneut das *Transfer Learning* zum Einsatz. Das Netzwerk entscheidet direkt über die Messparameter des Radars passend zur Szene. Die Aufgabe ist eine schnellstmögliche und energieeffiziente Detektion aller Ziele in der Szene. Die adaptiven Parameter sind die Bandbreite, Chirp-Dauer, Chirp-Wiederholdauer, die Anzahl an Rampen, sowie die Falschalarmwahrscheinlichkeit der CFAR. Das kognitive Radar ist

als Reinforcement Learning Agent, der mit einer simulierten Radar-Umgebung interagiert, umgesetzt. Der Reinforcement Learning Agent beinhaltet zwei neuronale Netzwerke in der Actor-Critic-Architektur, die mittels PPO-Algorithmus trainiert werden. Dabei beurteilt der Critic die Parameterwahl (Vorgehensweise) des Actors.

Es wurden zwei Experimente durchgeführt, die die verschiedenen Aspekte des kognitiven Radars aufzeigen. Im ersten Drei-Ziel-Szenario lernt der Reinforcement Learning Agent eine Sequenz von Messungen mit sinnvollen Parameterkombinationen, die zu einer langfristigen Belohnung führt. Dazu sind die drei Ziele so in der Szene platziert, dass eine Einzelmessung nicht ausreichend zur Detektion aller Ziele ist. Vielmehr sind zwei aufeinanderfolgende Messungen mit zwei unterschiedlichen Parameterkombinationen erforderlich. Diese Messsequenz und Parameterkombinationen wurden vom Reinforcement Learning Agenten erfolgreich erlernt. Das zweite Experiment ist die Kochanwendung. Die wesentlichen Bestandteile der Kochszene werden in der Radar-Umgebung simuliert. Das Ergebnis zeigt, dass der Reinforcement Learning Agent zu den nahezu gleichen Parametern kommt, wie sie in der Kochanwendung gewählt wurden. Das Reinforcement Learning ist also dazu geeignet, ohne Expertenwissen über Radarsensorik für eine konkrete Anwendung, eine optimale Parameterwahl der Sensorik einzustellen. Zusätzlich wird bestätigt, dass eine Einzelmessung zur Klassifikation der Kochszene, wie etwa beim *Cross Learning*, ausreichend ist. Das vierte Kernziel dieser Arbeit, die adaptive Parametrisierung kognitiver FM-CW-Radar-Messungen zur ressourcenschonenden Szenen-Detektion, ist damit erreicht.

Die Erkenntnisse dieser Arbeit wurden an einem Kochfeld als Demonstrator umgesetzt. Eine derartige Vorrichtung zur Überwachung eines Kochvorgangs mittels Radar wurde zum Patent angemeldet. Aufgrund der sukzessiven Weiterentwicklung konnten im Laufe der Dissertation weitere Funktionen, wie die Gestensteuerung, realisiert werden. Zuletzt haben auch die Netzwerke eine Genauigkeit erreicht, die eine robuste praktische Anwendung ermöglichen.

Verzeichnis der verwendeten Formelzeichen, Indizes und Abkürzungen

Die Formelzeichen und Indizes sind, soweit möglich, in Anlehnung an DIN 1304 gewählt, sodass Doppelnennungen weitgehend vermieden werden. Bei mehrfacher Verwendung eines Formelzeichens ergibt sich die Bedeutung eindeutig aus dem Zusammenhang. Übliche Laufindizes sowie elementare Operatoren und Funktionen werden hier nicht explizit aufgeführt. Die Formelzeichen sind nach DIN 1338 kursiv dargestellt. Vektoren und Matrizen sind fett gedruckt.

Formelzeichen

B . . . . . Bandbreite

C . . . . . Anzahl der Filter (Channels, Kernels)

c_0 . . . . . Lichtgeschwindigkeit im Vakuum

D . . . . . Volumentiefe (Depth)

F . . . . . Räumliche Filtergröße (Spatial Size)

f_0 . . . . . Startfrequenz

f_b . . . . . Zwischenfrequenzsignal (*beat frequency*)

f_c . . . . . Trägerfrequenz

f_s . . . . . Abstastfrequenz

H . . . . . Volumenhöhe (Height)

K . . . . . Anzahl an Klassen

L . . . . . Anzahl Rampen der Chirp-Sequence

λ . . . . . Wellenlänge

n . . . . . Batch-Größe

N_{FFT} . . . . . Größe der FFT (N-Punkt-FFT)

P . . . . . Anzahl Nullen des Zero Padding

r . . . . . Zielentfernung

Δr . . . . . Entfernungssauflösung

$r_{\max}$ . . . . . Maximale eindeutig messbare Entfernung

S . . . . . Schrittweite (Stride)

T . . . . . Chirp-Dauer

τ . . . . . Zeitverzögerung

T_{CRT} . . . . . Chirp-Wiederholdauer

v . . . . . Zielgeschwindigkeit

Δv . . . . . Geschwindigkeitsauflösung

$v_{\max}$ . . . . . Maximale eindeutig messbare Geschwindigkeit

W . . . . . Volumenbreite (Width)

$\hat{y}$ . . . . . Ausgabewert des Netzwerks

y . . . . . Ground Truth

Abkürzungen

2D-FFT	Zweidimensionale Fast Fourier-Transformation
Adam	Adaptive Momentum Estimation
ADC	Analog-zu-Digital-Wandler
CFAR	Constant False Alarm Rate
CNN	Convolutional Neural Network
Conv	Convolutional
FC	Fully Conntected
FCN	Fully Convolutional Network
FFT	Fast Fourier Transformation
FMCW	Frequency-Modulated Continuous Wave
FN	Falsch Negativ
FP	Falsch Positiv
FPGA	Field Programmable Gate Array
GAE	Generalized Advantage Estimation
GPUs	Grafikprozessoren
HMSE	Half Mean Square Error
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
KI	Künstliche Intelligenz
LSTM	Long Short-Term Memory
MAE	Mean Absoulte Error
MMIC	Monolithic Microwave Integrated Circuit

MSE	Mean Square Error
PPO	Proximal Policy Optimization
RCS	Radar Cross Section
RDM	Range-Doppler-Matrix
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RN	Richtig Negativ
RP	Richtig Positiv
SAR	Synthetic Aperture Radar
SNR	Signal-zu-Rausch-Verhältnis

Literaturverzeichnis

[1] Y. LeCun, Y. Bengio und G. Hinton, „Deep Learning," *Nature*, Band 521, Nr. 7553, S. 436–444, 2015.

[2] J. Schmidhuber, „Deep learning in neural networks: An overview," *Neural Networks*, Band 61, S. 85–117, 2015.

[3] L. Deng, „Deep Learning: Methods and Applications," *Foundations and Trends® in Signal Processing*, Band 7, Nr. 3-4, S. 197–387, 2014.

[4] A. Santra und S. Hazra, *Deep learning applications of short-range radars*. Boston: Artech House, 2020.

[5] S. Wang, J. Song, J. Lien, I. Poupyrev und O. Hilliges, „Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ACM, 2016.

[6] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja und I. Poupyrev, „Soli: ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics*, Band 35, Nr. 4, S. 1–19, 2016.

[7] Y. Sun, T. Fei, F. Schliep und N. Pohl, „Gesture Classification with Handcrafted Micro-Doppler Features using a FMCW Radar," in *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, 2018, S. 1–4.

[8] N. Kern, M. Steiner, R. Lorenzin und C. Waldschmidt, „Robust Doppler-Based Gesture Recognition With Incoherent Automotive Radar Sensor Networks," *IEEE Sensors Letters*, Band 4, Nr. 11, S. 1–4, 2020.

[9] R. P. Trommel, R. I. A. Harmanny, L. Cifola und J. N. Driessen, „Multi-target human gait classification using deep convolutional neural networks on micro-doppler spectrograms," in *2016 European Radar Conference (EuRAD)*, 2016, S. 81–84.

[10] Y. Lin, J. Le Kernec, S. Yang, F. Fioranelli, O. Romain und Z. Zhao, „Human Activity Classification With Radar: Optimization and Noise Robustness With Iterative Convolutional Neural Networks Followed With Random Forests," *IEEE Sensors Journal*, Band 18, Nr. 23, S. 9669–9681, 2018.

[11] B. Erol und M. G. Amin, „Radar Data Cube Processing for Human Activity Recognition Using Multisubspace Learning," *IEEE Transactions on Aerospace and Electronic Systems*, Band 55, Nr. 6, S. 3617–3628, 2019.

[12] C. Ding, Y. Jia, G. Cui, C. Chen, X. Zhong und Y. Guo, „Continuous Human Activity Recognition through Parallelism LSTM with Multi-Frequency Spectrograms," *Remote Sensing*, Band 13, Nr. 21, S. 4264, 2021.

[13] T. Stadelmayer, A. Santra, R. Weigel und F. Lurz, „Data-Driven Radar Processing Using a Parametric Convolutional Neural Network for Human Activity Classification," *IEEE Sensors Journal*, Band 21, Nr. 17, S. 19 529–19 540, 2021.

[14] C. Y. Aydogdu, S. Hazra, A. Santra und R. Weigel, „Multi-Modal Cross Learning for Improved People Counting using Short-Range FMCW Radar," in *2020 IEEE International Radar Conference (RADAR)*, IEEE, 2020.

[15] S. Wagner, „Combination of Convolutional Feature Extraction and Support Vector Machines for Radar ATR," *FUSION 2014 - 17th International Conference on Information Fusion*, 2014.

[16] S. Chen, H. Wang, F. Xu und Y.-Q. Jin, „Target Classification Using the Deep Convolutional Networks for SAR Images," *IEEE Transactions on Geoscience and Remote Sensing*, Band 54, Nr. 8, S. 4806–4817, 2016.

[17] J. Laviada, A. Arboleya, F. López-Gayarre und F. Las-Heras, „Broadband Synthetic Aperture Scanning System for Three Dimensional Through the Wall Inspection," *IEEE Geoscience and Remote Sensing Letters*, Band 13, Nr. 1, S. 97–101, 2016.

[18] S. A. Wagner, „SAR ATR by a combination of convolutional neural network and support vector machines," *IEEE Transactions on Aerospace and Electronic Systems*, Band 52, Nr. 6, S. 2861–2872, 2016.

[19] J. Geng, J. Fan, H. Wang, X. Ma, B. Li und F. Chen, „High-Resolution SAR Image Classification via Deep Convolutional Autoencoders," *IEEE Geoscience and Remote Sensing Letters*, Band 12, Nr. 11, S. 2351–2355, 2015.

[20] B. K. Kim, H.-S. Kang und S.-O. Park, „Drone Classification Using Convolutional Neural Networks With Merged Doppler Images," *IEEE Geoscience and Remote Sensing Letters*, Band 14, Nr. 1, S. 38–42, 2017.

[21] S. Haykin, „Cognitive radar: a way of the future," *IEEE Signal Processing Magazine*, Band 23, Nr. 1, S. 30–40, 2006.

[22] M. S. Greco, F. Gini, P. Stinco und K. Bell, „Cognitive Radars: On the Road to Reality: Progress Thus Far and Possibilities for the Future," *IEEE Signal Processing Magazine*, Band 35, Nr. 4, S. 112–125, 2018.

[23] M. Kozy, J. Yu, R. M. Buehrer, A. Martone und K. Sherbondy, „Applying Deep-Q Networks to Target Tracking to Improve Cognitive Radar," in *2019 IEEE Radar Conference (RadarConf)*, IEEE, 2019.

[24] P. Liu, Y. Liu, T. Huang, Y. Lu und X. Wang, „Cognitive Radar Using Reinforcement Learning in Automotive Applications," 2019. arXiv: 1904.10739 [eess.SP].

[25] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel und C. Waldschmidt, „Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band," *IEEE Transactions on Microwave Theory and Techniques*, Band 60, Nr. 3, S. 845–860, 2012.

[26] S. M. Patole, M. Torlak, D. Wang und M. Ali, „Automotive radars: A review of signal processing techniques," *IEEE Signal Processing Magazine*, Band 34, Nr. 2, S. 22–35, 2017.

[27] M. Dudek, D. Kissinger, R. Weigel und G. Fischer, „Consideration of higher frequency bands in accurate millimeter-wave FMCW-radar system simulations for automotive applications," in *WAMICON 2011 Conference Proceedings*, 2011, S. 1–4.

[28] F. B. Khalid, D. T. Nugraha, A. Roger, R. Ygnace und M. Bichl, „Distributed Signal Processing of High-Resolution FMCW MIMO Radar for Automotive Applications," in *2018 15th European Radar Conference (EuRAD)*, 2018, S. 513–516.

[29] H. Meinel und J. Dickmann, „Automotive Radar: From Its Origin to Future Directions," *Microwave Journal*, Band vol.56, S. 24–40, September 2013.

[30] C. Waldschmidt, P. Hügler und M. Geiger, „Radar as an emerging and growing technology for industrial applications: A short overview," in *Proceedings Sensor 2017*, AMA Service GmbH, Von-Münchhausen-Str. 49, 31515 Wunstorf, Germany, 2017.

[31] D. Brumbi, „Low power FMCW radar system for level gaging," in *2000 IEEE MTT-S International Microwave Symposium Digest (Cat. No.00CH37017)*, Bd. 3, 2000, 1559–1562 vol.3.

[32] G. Vinci, S. Linz, S. Mann, S. Lindner, F. Barbon, R. Weigel und A. Koelpin, „A Six-Port Radar System for Precise Distance Measurements and Vibration Monitoring in Industrial Environments," in *Sensors and Measuring Systems 2014; 17. ITG/GMA Symposium*, 2014, S. 1–5.

[33] C. Fischer und W. Wiesbeck, „An evaluation of sensor configurations for ground penetration radar," in *IGARSS 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No.00CH37120)*, Bd. 3, 2000, 993–995 vol.3.

[34] P. Hügler, T. Grebner, C. Knill und C. Waldschmidt, „UAV-Borne 2-D and 3-D Radar-Based Grid Mapping," *IEEE Geoscience and Remote Sensing Letters*, Band 19, S. 1–5, 2020.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg und L. Fei-Fei, „ImageNet Large Scale Visual Recognition Challenge," September 2014. arXiv: 1409.0575 [cs.CV].

[36] A. Krizhevsky, I. Sutskever und G. E. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger, Hrsg., Bd. 25, Curran Associates, Inc., 2012, S. 1097–1105.

[37] M. Altmann, P. Ott und C. Waldschmidt, „Deep Learning for Range-Doppler Map Single Frame Classifications of Cooking Processes," in *2018 15th European Radar Conference (EuRAD)*, Madrid, Spanien, 2018.

[38] M. Altmann, P. Ott, N. C. Stache und C. Waldschmidt, „Learning Dynamic Processes from a Range-Doppler Map Time Series with LSTM Networks," in *2019 16th European Radar Conference (EuRAD)*, Paris, Frankreich, 2019.

[39] M. Altmann, P. Ott, N. C. Stache, D. Kozlov und C. Waldschmidt, „A Cognitive FMCW Radar to Minimize a Sequence of Range-Doppler Measurements," in *2020 17th European Radar Conference (EuRAD)*, Utrecht, Niederlande, 2021.

[40] M. Altmann, P. Ott, N. C. Stache und C. Waldschmidt, „Multi-Modal Cross Learning for an FMCW Radar Assisted by Thermal and RGB Cameras to Monitor Gestures and Cooking Processes," *IEEE Access*, Band 9, S. 22 295–22 303, 2021.

[41] A. Demirbas, „Progress and recent trends in biodiesel fuels," *Energy Conversion and Management*, Band 50, Nr. 1, S. 14–34, 2009.

[42] D. Silva, T. Brányik, G. Dragone, A. Vicente, J. Teixeira und J. Silva, „High gravity batch and continuous processes for beer production: Evaluation of fermentation performance and beer quality," *Chemical Papers*, Band 62, Nr. 1, 2008.

[43] C. Schroeder und H. Rohling, „X-band FMCW radar system with variable chirp duration," in *2010 IEEE Radar Conference*, Washington, DC, USA: IEEE, 2010.

[44] H. Rohling, „Radar CFAR Thresholding in Clutter and Multiple Target Situations," *IEEE Transactions on Aerospace and Electronic Systems*, Band AES-19, Nr. 4, S. 608–621, 1983.

[45] M. Kronauge und H. Rohling, „Fast Two-Dimensional CFAR Procedure," *IEEE Transactions on Aerospace and Electronic Systems*, Band 49, Nr. 3, S. 1817–1823, 2013.

[46] C. C. Tappert, „Who Is the Father of Deep Learning?" In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2019.

[47] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare und J. Pineau, „An Introduction to Deep Reinforcement Learning," *Foundations and Trends® in Machine Learning*, Band 11, Nr. 3-4, S. 219–354, 2018.

[48] S. Russell, *Artificial intelligence : a modern approach*. Upper Saddle River, New Jersey: Prentice Hall, 2010.

[49] M. Hutter, *Universal Artificial Intelligence*. Springer-Verlag GmbH, 2004.

[50] N. J. Nilsson, *The Quest for Artificial Intelligence*. Cambridge University Press, 2015, 580 S.

[51] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[52] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2016, 760 S.

[53] Y. Kim und H. Ling, „Human Activity Classification Based on Micro-Doppler Signatures Using a Support Vector Machine," *IEEE Transactions on Geoscience and Remote Sensing*, Band 47, S. 1328–1337, 2009.

[54] M. G. Amin, Z. Zeng und T. Shan, „Hand Gesture Recognition based on Radar Micro-Doppler Signature Envelopes," in *2019 IEEE Radar Conference (RadarConf)*, 2019, S. 1–6.

[55] H. Schulz und S. Behnke, „Deep Learning," *KI - Künstliche Intelligenz*, Band 26, Nr. 4, S. 357–363, Mai 2012.

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

[57] L. P. Kaelbling, M. L. Littman und A. W. Moore, „Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Band 4, S. 237–285, 1996.

[58] M. van Otterlo und M. Wiering, „Reinforcement Learning and Markov Decision Processes," in *Adaptation, Learning, and Optimization*, Springer Berlin Heidelberg, 2012, S. 3–42.

[59] A. Karpathy und J. Johnson. (2017). „CS231n: Convolutional Neural Networks for Visual Recognition: Lecture Notes," Stanford University, Adresse: https://cs231n.github.io/neural-networks-1/#bio.

[60] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs und H. Lipson, „Understanding Neural Networks Through Deep Visualization," *CoRR*, Band abs/1506.06579, 2015. arXiv: 1506.06579.

[61] M. D. Zeiler und R. Fergus, „Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele und T. Tuytelaars, Hrsg., Cham: Springer International Publishing, 2014, S. 818–833.

[62] J. Long, E. Shelhamer und T. Darrell, „Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015.

[63] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink und J. Schmidhuber, „LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, Band 28, Nr. 10, S. 2222–2232, 2017.

[64] S. Hochreiter und J. Schmidhuber, „Long Short-Term Memory," *Neural Computation*, Band 9, Nr. 8, S. 1735–1780, 1997.

[65] F. A. Gers, J. Schmidhuber und F. Cummins, „Learning to forget: continual prediction with LSTM," in *9th International Conference on Artificial Neural Networks: ICANN '99*, IEEE, 1999.

[66] S. Sharma und R. Mehra, „Implications of Pooling Strategies in Convolutional Neural Networks: A Deep Insight," *Foundations of Computing and Decision Sciences*, Band 44, Nr. 3, S. 303–330, 2019.

[67] R. Caruana, S. Lawrence und L. Giles, „Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, Ser. NIPS'00, Denver, CO: MIT Press, 2000, S. 381–387.

[68] D. M. Hawkins, „The Problem of Overfitting," *Journal of Chemical Information and Computer Sciences*, Band 44, Nr. 1, S. 1–12, 2003.

[69] X. Ying, „An Overview of Overfitting and its Solutions," *Journal of Physics: Conference Series*, Band 1168, S. 022 022, 2019.

[70] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, Band 15, Nr. 56, S. 1929–1958, 2014.

[71] S. Ioffe und C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, 2015. arXiv: 1502.03167.

[72] V. Nair und G. E. Hinton, „Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Ser. ICML'10, Haifa, Israel: Omnipress, 2010, S. 807–814.

[73] R. Y. Rubinstein und D. P. Kroese, *The Cross-Entropy Method*. Springer New York, 2004.

[74] M. Sokolova und G. Lapalme, „A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, Band 45, Nr. 4, S. 427–437, 2009.

[75] A. Martins und R. Astudillo, „From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan und K. Q. Weinberger, Hrsg., Ser. Proceedings of Machine Learning Research, Bd. 48, New York, USA: PMLR, 2016, S. 1614–1623.

[76] J. Read, B. Pfahringer, G. Holmes und E. Frank, „Classifier chains for multi-label classification," *Machine Learning*, Band 85, Nr. 3, S. 333–359, 2011.

[77] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala und C. O. Aigbavboa, „A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, IEEE, 2018.

[78] S. Ruder, „An overview of gradient descent optimization algorithms," *CoRR*, Band abs/1609.04747, 2016. arXiv: 1609.04747.

[79] D. Kingma und J. Ba, „Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 2014. eprint: https://arxiv.org/abs/1412.6980.

[80] P. J. Werbos, „Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, Band 78, Nr. 10, S. 1550–1560, 1990.

[81] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu und X. Zheng, „TensorFlow: A System for Large-Scale Machine Learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, Ser. OSDI'16, Savannah, GA, USA: USENIX Association, 2016, S. 265–283.

[82] Q. H. Nguyen, H.-B. Ly, L. S. Ho, N. Al-Ansari, H. V. Le, V. Q. Tran, I. Prakash und B. T. Pham, „Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil," *Mathematical Problems in Engineering*, Band 2021, Y.-S. Shen, Hrsg., S. 1–15, 2021.

[83] J. Tan, J. Yang, S. Wu, G. Chen und J. Zhao, „A critical look at the current train/test split in machine learning," *CoRR*, Band abs/2106.04525, 2021. arXiv: 2106.04525.

[84] T. Chai und R. R. Draxler, „Root mean square error (RMSE) or mean absolute error (MAE)?" *Geosci. Model Dev.*, Band 7, 2014.

[85] K. Simonyan und A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. arXiv: 1409.1556 [cs.CV].

[86] X. Glorot und Y. Bengio, „Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh und M. Titterington, Hrsg., Ser. Proceedings of Machine Learning Research, Bd. 9, Chia Laguna Resort, Sardinia, Italy: JMLR Workshop und Conference Proceedings, 13–15 May 2010, S. 249–256.

[87] L. Lu, Y. Shin, Y. Su und G. E. Karniadakis, „Dying ReLU and Initialization: Theory and Numerical Examples," *Communications in Computational Physics*, Band 28, Nr. 5, S. 1671–1706, 2020.

[88] J. Wang, J. Zhang und X. Wang, „Bilateral LSTM: A Two-Dimensional Long Short-Term Memory Model With Multiply Memory Units for Short-Term Cycle Time Forecasting in Re-entrant Manufacturing Systems," *IEEE Transactions on Industrial Informatics*, Band 14, Nr. 2, S. 748–758, 2018.

[89] W. Zaremba, I. Sutskever und O. Vinyals, „Recurrent Neural Network Regularization," 2014. arXiv: 1409.2329 [cs.NE].

[90] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke und A. Rabinovich, „Going Deeper with Convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[91] K. He, X. Zhang, S. Ren und J. Sun, „Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 770–778.

[92] Y. Kim und B. Toomajian, „Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network," *IEEE Access*, Band 4, S. 7125–7130, 2016.

[93] P. Hügler, M. Geiger und C. Waldschmidt, „RCS measurements of a human hand for radar-based gesture recognition at E-band," 2016.

[94] P. Vincent, H. Larochelle, Y. Bengio und P.-A. Manzagol, „Extracting and Composing Robust Features with Denoising Autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland: Association for Computing Machinery, 2008, S. 1096–1103.

[95] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent und S. Bengio, „Why Does Unsupervised Pre-training Help Deep Learning?" *Journal of Machine Learning Research*, Band 11, Nr. 19, S. 625–660, 2010.

[96] D. Bank, N. Koenigstein und R. Giryes, „Autoencoders," 2020. arXiv: 2003.05991 [cs.LG].

[97] M. Abavisani, H. R. V. Joze und V. M. Patel, „Improving the Performance of Unimodal Dynamic Hand-Gesture Recognition With Multimodal Training," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019.

[98] N. Neverova, C. Wolf, G. Taylor und F. Nebout, „ModDrop: Adaptive Multi-Modal Gesture Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Band 38, Nr. 8, S. 1692–1706, 2016.

[99] N. Garcia, P. Morerio und V. Murino, „Modality Distillation with Multiple Stream Networks for Action Recognition," *The European Conference on Computer Vision (ECCV)*, 2018. arXiv: 1806.07110 [cs.CV].

[100] H. Zheng und X.-M. Zhang, „A Cross-Modal Learning Approach for Recognizing Human Actions," *IEEE Systems Journal*, S. 1–9, 2020.

[101] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama und T. Darrell, „Cross-modal adaptation for RGB-D detection," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016.

[102] Z. Guo, W. Liao, Y. Xiao, P. Veelaert und W. Philips, „Deep Learning Fusion of RGB and Depth Images for Pedestrian Detection," in *British Machine Vision Conference*, 2019.

[103] L. Zhen, P. Hu, X. Peng, R. S. M. Goh und J. T. Zhou, „Deep Multimodal Transfer Learning for Cross-Modal Retrieval," *IEEE Transactions on Neural Networks and Learning Systems*, S. 1–13, 2020.

[104] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber und A. Torralba, „Learning Cross-Modal Embeddings for Cooking Recipes and Food Images," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017.

[105] F. Pahde, M. Nabi, T. Klein und P. Jahnichen, „Discriminative Hallucination for Multi-Modal Few-Shot Learning," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018.

[106] J. Lezama, Q. Qiu und G. Sapiro, „Not Afraid of the Dark: NIR-VIS Face Recognition via Cross-Spectral Hallucination and Low-Rank Embedding," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017.

[107] K. Gunasekar, Q. Qiu und Y. Yang, „Low to High Dimensional Modality Hallucination Using Aggregated Fields of View," *IEEE Robotics and Automation Letters*, Band 5, Nr. 2, S. 1983–1990, 2020.

[108] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee und A. Ng, „Multimodal Deep Learning," in *International Conference on Machine Learning (ICML)*, 2011.

[109] O. Fried und R. Fiebrink, „Cross-modal Sound Mapping Using Deep Learning," in *New Interfaces for Musical Expression (NIME)*, 2013.

[110] O. Ronneberger, P. Fischer und T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells und A. F. Frangi, Hrsg., Cham: Springer International Publishing, 2015, S. 234–241.

[111] M. Siam, S. Elkerdawy, M. Jagersand und S. Yogamani, „Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017.

[112] M. D. Zeiler, D. Krishnan, G. W. Taylor und R. Fergus, „Deconvolutional networks," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010.

[113] V. Dumoulin und F. Visin, „A guide to convolution arithmetic for deep learning," 2016. arXiv: 1603.07285 [stat.ML].

[114] D. Zhang, J. Wang und X. Zhao, „Estimating the Uncertainty of Average F1 Scores," in *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ACM, 2015.

[115] A. H. Mirza und S. Cosan, „Computer network intrusion detection using sequential LSTM Neural Networks autoencoders," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2018.

[116] A. Essien und C. Giannetti, „A Deep Learning Framework for Univariate Time Series Prediction Using Convolutional LSTM Stacked Autoencoders," in *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, 2019.

[117] J. Schulman, F. Wolski, P. Dhariwal, A. Radford und O. Klimov, „Proximal Policy Optimization Algorithms," *CoRR*, Band abs/1707.06347, 2017. arXiv: 1707.06347.

[118] J. Thomas, C. F. Moss und M. Vater, *Echolocation in bats and dolphins*. Chicago: University of Chicago Press, 2004.

[119] M. Vespe, G. Jones und C. J. Baker, „Lessons for radar: waveform diversity in echolocating mammals," *IEEE Signal Processing Magazine*, Band 26, Nr. 1, S. 65–75, 2009.

[120] M. Greco und F. Gini, „Analysis and Modeling of Echolocation Signals Emitted by Mediterranean Bottlenose Dolphins," *EURASIP Journal on Advances in Signal Processing*, Band 2006, Nr. 1, 2006.

[121] S. Haykin, Y. Xue und P. Setoodeh, „Cognitive Radar: Step Toward Bridging the Gap Between Neuroscience and Engineering," *Proceedings of the IEEE*, Band 100, Nr. 11, S. 3102–3130, 2012.

[122] L. Kang, J. Bo, L. Hongwei und L. Siyuan, „Reinforcement Learning based Anti-jamming Frequency Hopping Strategies Design for Cognitive Radar," in *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, IEEE, 2018.

[123] M. Inggs, „Passive Coherent Location as Cognitive Radar," in *2009 International Waveform Diversity and Design Conference*, IEEE, 2009, S. 229–233.

[124] Q. Wang, P. Du, T. Dou, L. Gao und C. Li, „Cognitive passive radar system: software defined radio and deep learning approach," *The Journal of Engineering*, Band 2019, Nr. 21, S. 7326–7330, 2019.

[125] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan und D. Hassabis, „A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, Band 362, Nr. 6419, S. 1140–1144, 2018.

[126] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra und M. Riedmiller, „Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop*, 2013. arXiv: 1312.5602 [cs.LG].

[127] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver und D. Wierstra, „Continuous control with deep reinforcement learning," *Advances in Neural Information Processing Systems*, S. 2926–2934, 2015. arXiv: 1509.02971 [cs.LG].

[128] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver und K. Kavukcuoglu, „Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, Ser. ICML'16, New York, NY, USA: JMLR.org, 2016, S. 1928–1937.

[129] V. Konda und J. Tsitsiklis, „Actor-Critic Algorithms,“ in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen und K. Müller, Hrsg., Bd. 12, MIT Press, 2000.

[130] C. J. C. H. Watkins und P. Dayan, „Q-learning,“ *Machine Learning*, Band 8, Nr. 3-4, S. 279–292, 1992.

[131] H. van Hasselt, A. Guez und D. Silver, „Deep Reinforcement Learning with Double Q-learning,“ *CoRR*, Band abs/1509.06461, 2015. arXiv: 1509.06461.

[132] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg und D. Hassabis, „Human-level control through deep reinforcement learning,“ *Nature*, Band 518, Nr. 7540, S. 529–533, 2015.

[133] J. Schulman, S. Levine, P. Moritz, M. I. Jordan und P. Abbeel, „Trust Region Policy Optimization,“ *CoRR*, Band abs/1502.05477, 2015. arXiv: 1502.05477 [cs.LG].

[134] J. Schulman, P. Moritz, S. Levine, M. I. Jordan und P. Abbeel, „High-Dimensional Continuous Control Using Generalized Advantage Estimation,“ *CoRR*, Band abs/1506.02438, 2016.

Eigene Veröffentlichungen

Teile dieser Dissertation sind bereits in folgenden begutachteten Fachbeiträgen veröffentlicht.

Veröffentlichungen in Zeitschriften

- M. Altmann, P. Ott, N. C. Stache und C. Waldschmidt, „Multi-Modal Cross Learning for an FMCW Radar Assisted by Thermal and RGB Cameras to Monitor Gestures and Cooking Processes," *IEEE Access*, Band 9, S. 22 295–22 303, 2021.

Veröffentlichungen in Konferenzen

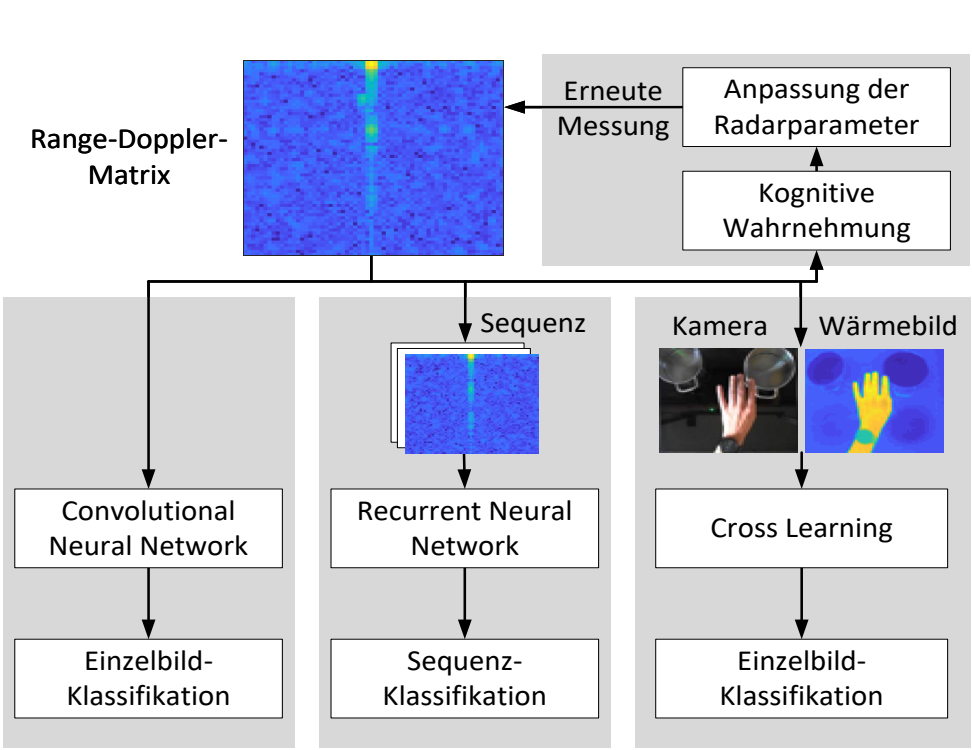
- M. Altmann, P. Ott, N. C. Stache, D. Kozlov und C. Waldschmidt, „A Cognitive FMCW Radar to Minimize a Sequence of Range-Doppler Measurements," in *2020 17th European Radar Conference (EuRAD)*, Utrecht, Niederlande, 2021.
- M. Altmann, P. Ott, N. C. Stache und C. Waldschmidt, „Learning Dynamic Processes from a Range-Doppler Map Time Series with LSTM Networks," in *2019 16th European Radar Conference (EuRAD)*, Paris, Frankreich, 2019.
- M. Altmann, P. Ott und C. Waldschmidt, „Deep Learning for Range-Doppler Map Single Frame Classifications of Cooking Processes," in *2018 15th European Radar Conference (EuRAD)*, Madrid, Spanien, 2018.
- M. Altmann und P. Ott, „Confocal radar system for improved FMCW range resolution," in *2017 International Conference on Research and Education in Mechatronics (REM)*, Wolfenbüttel, Deutschland, 2017.

- M. Altmann, P. Ott und N. C. Stache, „Deep Learning zur Erkennung von Kochprozessen," in *MATLAB Expo München*, München, Deutschland, 2018.
- M. Altmann und P. Ott, „Deep Learning for Radar and Camera Based Cooking Process Monitoring," in *NVIDIA GTC München (Poster-Session)*, München, Deutschland, 2018.
- D. Kozlov, P. Ott, O. Loffeld und M. Altmann, „Soft Iterative Method with Adaptive Thresholding for Reconstruction of Radar Scenes," in *2020 IEEE International Radar Conference (RADAR)*, Washington DC, USA, 2020.

Patente

- P. Ott und M. Altmann, „Vorrichtung zur Überwachung eines Kochvorgangs," Aktenzeichen DE 10 2017 106 392.7, September 2018.

Herausgeber:
Prof. Dr.-Ing. Christian Waldschmidt



In dieser Dissertation werden lernende Verfahren zur Gesten- und Kochklassifikation mittels Radarsensorik vorgestellt. Die Verfahren umfassen zum einen die Klassifikation der Radardaten mithilfe verschiedener tiefer neuronaler Netzwerke, teilweise unter Zuhilfenahme zusätzlicher Modalitäten. Zum anderen wird ein kognitives Radar vorgestellt, bei dem das Radar adaptiv auf die Szene reagieren und die Messparameter entsprechend anpassen kann.

Die Verfahren werden anhand der Anwendung Kochsensor evaluiert. In dieser dynamischen Szene wird ein Überkochschutz und eine Gestensteuerung realisiert.

ISBN: 978-3-948303-32-7 ISBN: 978-3-948303-33-4 (E-Book)