Freie Universität Berlin

Bachelorarbeit am Institut für Informatik
Dahlem Center for Machine Learning and Robotics
Biorobotics Lab

Dekodierung von Bienentänzen mithilfe eines künstlichen neuronalen Netzes

Adrian Defèr

Betreuer: Prof. Dr. Tim Landgraf
Erstgutachter: Prof. Dr. Tim Landgraf
Zweitgutachter: Prof. Dr. Raúl Rojas

18. August 2017

Zusammenfassung

Im Rahmen des Beesbook Projekts am Biorobotics Lab der FU Berlin wird eine Computer Vision Pipeline genutzt, um Bienentänze auf der Wabe zu detektieren, zu dekodieren und zu kartieren.
Zur Detektion der Tänze wird die Periodizität des Helligkeitsverlaufs an bestimmten Pixeln genutzt. Dabei wird der Helligkeitsverlauf der Pixel auf bestimmte Frequenzen hin untersucht, mit denen Bienen typischerweise schwänzeln. Mehrere Detektionen einer Frequenz werden wiederum räumlich, wie zeitlich geclustert und einzelnen Tänzen zugeordnet.
Da sich unter diesen Detektionen auch viele falsch positive Tänze befinden werden diese anschließend zu einem großen Teil von einem konvolutionalen neuronalen Netzwerk herausgefiltert. D.h. es existiert ein dediziertes Netz in der Pipeline, welches Ergebnisse des Waggle Dance Detectors binär, Tanz oder Nicht-Tanz, klassifiziert und somit unerwünschte Ergebnisse vor der Weiterverarbeitung herausfiltert.
Auf Basis der Bilddaten eines Tanzes wird im Anschluss der Winkel der Tanzrichtung mithilfe von Bilddifferenzen ermittelt.
Anhand der Länge des Tanzes und des Winkels der Tanzrichtung kann nun die von der Biene beworbene Position berechnet und auf einer Karte dargestellt werden.
Um die Qualität und Effizienz der Winkeldekodierung zu verbessern und außerdem die Anzahl der Module in der Pipeline zu verringern, war das Ziel dieser Arbeit ein künstliches neuronales Netz zu entwickeln, welches die Funktion des aktuellen konvolutionalen Netzwerks zur Filterung und zusätzlich die Funktion der Tanzwinkeldekodierung in sich vereint.

Inhaltsverzeichnis

1. Einleitung 5

2. Stand der Forschung / verwandte Arbeiten 6
   - 2.1. Der Schwänzeltanz . . . 6
   - 2.2. Aufbau zur durchgehenden Beobachtung . . . 9
   - 2.3. Bestehende Software . . . 11
     - 2.3.1. WDD . . . 12
     - 2.3.2. Filternetzwerk . . . 12
     - 2.3.3. Winkelextraktion . . . 13
     - 2.3.4. Mapping Modul . . . 14
     - 2.3.5. Zusammenfassung . . . 15
   - 2.4. Künstliche neuronale Netzwerke (KNN) . . . 16
     - 2.4.1. Multilayer Perceptron . . . 16
     - 2.4.2. Convolutional Neural Networks (CNN) . . . 18
     - 2.4.3. Recurrent Neural Network (RNN) . . . 18
   - 2.5. Verwandte Arbeiten . . . 18
     - 2.5.1. Arbeiten zu CNNs . . . 18
     - 2.5.2. Arbeiten zu RNNs . . . 20

3. Implementierung 21
   - 3.1. Datensatz . . . 21
   - 3.2. Anforderungen . . . 22
     - 3.2.1. Qualität . . . 22
     - 3.2.2. Performance und Technologie . . . 22
   - 3.3. Designentwurf . . . 23
   - 3.4. Architektur . . . 23
     - 3.4.1. Grundarchitektur . . . 23
     - 3.4.2. Verwendete Architektur . . . 24
   - 3.5. Training . . . 27
     - 3.5.1. Klassifizierungsnetz . . . 27
     - 3.5.2. Dekodierungsnetz . . . 28

4. Evaluierung 28
   - 4.1. Klassifizierungsnetz . . . 29
   - 4.2. Dekodierungsnetz . . . 30

5. Diskussion 32

6. Ausblick 33
7. Anhang 38

1. Einleitung

Die Tanzsprache der Honigbiene ist eine einzigartige Kommunikationsform im Tierreich, mit deren Hilfe beispielsweise neu entdeckte Futterquellen oder geeignete Nistplätze an ihre Artgenossen kommuniziert werden.
Eine von vielen Tanzarten der Bienen ist der Schwänzeltanz. Dieser dient, im Bienestock aufgeführt, dazu Sammlerbienen für eine Nahrungsquelle zu rekrutieren. Nach der Rückkehr in den Stock gibt eine Kundschafterin die Informationen, die dazu dienen die neue Futterquelle aufzufinden, an ihre Artgenossen, mit Hilfe dieses Tanzes, weiter.
Zoologe und Verhaltensforscher Karl von Frisch war der erste, der den Schwänzeltanz der Biene entschlüsselte[1]. Und bis heute ist der Schwänzeltanz, als bekannteste soziale Interaktionsform der Biene, oft Gegenstand multidisziplinärer Forschung gewesen. Denn er kann, durch seine Rolle in der Nahrungsbeschaffung, als einer der Schlüssel zum Überleben des Superorganismus des Bienenvolkes angesehen werden. Wiederum vom Überleben der Bienen abhängig ist die Bestäubung von ca. 80% aller Nutz- und Wildpflanzen hierzulande[2]. Was biologische Vielfalt und vielen Tieren und dadurch auch dem Menschen, direkt wie indirekt, die Ernährung sichert.
In den Anfangstagen mussten Verhaltensforscher, Biologen, Ökologen und Forscher vieler anderer Disziplinen das Verhalten der Bienen in allen Belangen manuell untersuchen. Beispielsweise galt es Schwänzeltänze mithilfe von Stoppuhr und Geodreieck in Echtzeit zu entschlüsseln. Später kam die Möglichkeit hinzu Videoaufnahmen anzufertigen, was die Anzahl der simultan möglichen Beobachtungen sowie die Genauigkeit dieser drastisch erhöhte. Jedoch keineswegs die aufzuwendende Zeit pro Beobachtung verringerte. Durch Digitalvideo wurde eine noch genauere Auswertung mithilfe von Softwarewerkzeugen möglich. Doch erst mit weiter fortgeschrittener Computertechnologie wurden Systeme zur automatischen Auswertung von Videodaten im Bezug auf das Verfolgen von Bienen-Individuen und Erkennen von Schwänzeltänzen entwickelt[3, 4, 5]. Doch keines dieser Systeme beherrscht die automatische Dekodierung von Schwänzeltänzen. Das einzige dem Autor bekannte System[11], welches dies in Echtzeit beherrscht, wurde am Institut für Informatik der Freien Universität Berlin entwickelt. Hier, am Biorobotics Lab[8], wird, eng zusammen mit Biologen und Neurobiologen, an der Erforschung der Schwarmintelligenz am Beispiel von Honigbienen und Fischen (speziell Guppys) gearbeitet. Erkenntnisse, die wir aus der Natur gewinnen, können mit großer Wahrscheinlichkeit auch in unserer technisierten Welt Anwendung finden. Beispielsweise könnten uns Erkenntnisse über Schwarmintelligenz bei der KI für selbstfahrende Autos[9] voranbringen.
Im Rahmen des Projekts RoboBee haben Tim Landgraf und Kollegen den Schwänzeltanz mithilfe von Video- und moderner Analysetechnik genauestens untersucht[6] und einen Bienenroboter gebaut, der von einem Bienenvolk akzeptiert wurde[7].

Das Ziel der Arbeit am Projekt Beesbook ist es nämlich das soziale Netzwerk des Bienenvolkes zu verstehen. Im Zuge dieses Projekts wurde dazu ein Tracking- und Dekodierungssystem entwickelt[10], welches ein Bienenvolk innerhalb eines am Institut installierten Stockes über dessen gesamte Lebensspanne hin trackt und Schwänzelläufe in Echtzeit automatisiert erkennt, dekodiert und in Form von Metadaten und den korrespondierenden Bilddaten abspeichert[11]. Im Rahmen von Bachelorarbeiten wurden Teile des Tanzerkennungs- und -dekodierungssystems verbessert[12, 13], jedoch nicht in die Echtzeitumgebung des ursprünglichen Systems integriert. Was dazu führte, dass nun 4 Module einzeln nacheinander ausgeführt werden müssen.
In dieser Arbeit soll nun ein künstliches neuronales Netzwerk entwickelt werden, welches die Aufgaben von 2 bestehenden[12, 13] übernimmt und dabei die Güte und Effizienz der Dekodierung der Schwänzelläufe nochmals verbessert. Dies soll ein erster Schritt in Richtung eines großen, automatischen, echtzeitfähigen Gesamtsystems sein, das alle Aufgaben in einem neuronalen Netz vereint.

2. Stand der Forschung / verwandte Arbeiten

2.1 Der Schwänzeltanz

Möchte eine Biene nach der Rückkehr in den Stock eine neu entdeckte Nahrungsquelle bewerben, so bewegt sie sich in die Mitte der Wabe, macht mit hecktischen Flügelbewegungen auf sich aufmerksam und beginnt dann den Schwänzeltanz (Abb.1). Dieser wird zwischendurch immer wieder für Kostproben an die Nachtänzerinnen, mittels Trophallaxis[1], unterbrochen.
Der Schwänzeltanz lässt sich in Schwänzel- und Rückkehrphase einteilen. In der Schwänzelphase (auch als Schwänzellauf bezeichnet) läuft die Biene eine gerade Strecke, während sie mit ihrem Hinterleib schnell, mit einer Frequenz von 13–15Hz, hin und her wackelt ("schwänzelt"). In der darauffolgenden Rückkehrphase läuft sie in einem Halbkreis zurück zu ihrem Ausgangspunkt. Diese beiden Phasen wiederholen sich beliebig oft, wobei die Richtung, in der die Biene zum Halbkreis einschlägt jedes Mal wechselt. In Abhängigkeit davon, welche Qualität das Futter und welche Bedeutung dieses für den Stock hat (herrscht gerade Futtermangel oder gibt es sowieso gerade ein Überangebot), tanzt die Biene energischer und/oder länger. Dabei ist mit energischer gemeint, dass die Amplitude des Schwänzelns höher ist. Mit länger ist hier gemeint, dass die Gesamtanzahl der Schwänzelläufe pro Tanz höher ist.

[1] Weitergabe von Futter an Artgenossen durch hochwürgen aus der Honigblase

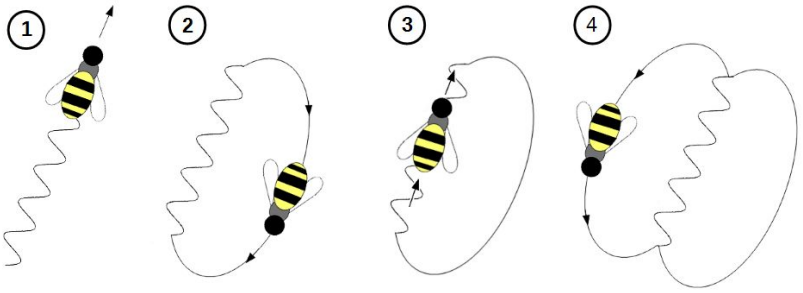

Abb.1: Der Schwänzeltanz. 1) Schwänzellauf – Biene läuft eine gerade Strecke, während sie mit ihrem Hinterleib hin und her wackelt 2) läuft in einem Halbkreis zurück zum Ausgangspunkt 3) wiederholter Schwänzellauf 4) läuft zur anderen Richtung in einem Halbkreis zurück zum Ausgangspunkt. Dieser Ablauf kann sich mehrmals wiederholen.
Quelle des Bildes: http://hmcbee.blogspot.de/2014/07/the-waggle-dance.html

Die Informationen, die dieser Tanz für die anderen Bienen enthält sind zum einen die Richtung und die Entfernung, in der die Futterquelle gesucht werden muss und zum anderen die Qualität der dort zu findenden Nahrung.
Dabei wird die Richtung in Form von Polarkoordinaten kodiert. Der Winkel, in dem die Biene in der Schwänzelphase relativ zum Lot des Stocks läuft, ist der Winkel, in deren Richtung sich die Nahrung relativ zum Azimut der Sonne[2] befindet (vgl. Abb.2).
Die Dauer der Schwänzelphase kodiert dabei die Entfernung zur Nahrungsquelle, also die Länge des Vektors.[1]
Tanzt eine Biene also in eine Richtung, sodass die Gerade der Strecke, die sie schwänzelt zu dem vertikalen Lot in einem Winkel alpha steht, so muss der Futterplatz entlang der Geraden gesucht werden, die mit dem Azimut der Sonne den Winkel alpha einnimmt.
Beide Werte werden von der Biene jedoch nur ungefähr angegeben. Der Tanz stellt nur den ersten Teil in einer Kette von wegweisenden Verhalten seitens der Bienen dar, um andere Bienen zu einer neuen Futterquelle zu lotsen.

[2] Vertikale Projektion des Sonnenstandes auf den Horizont

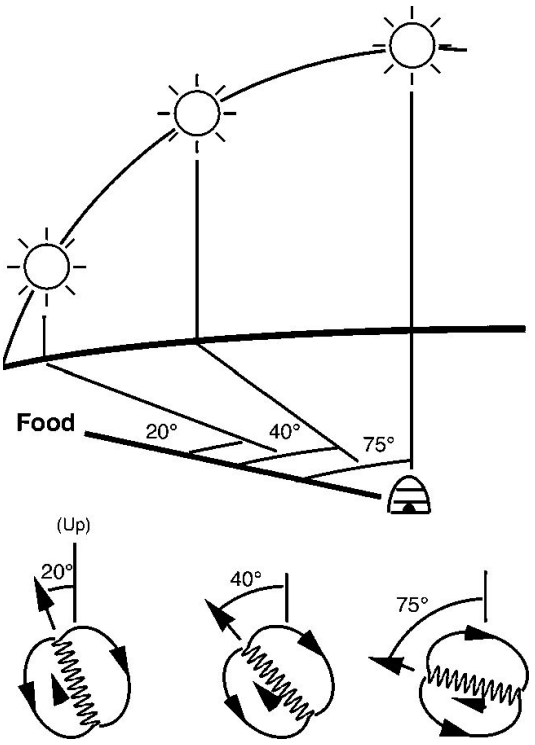

Abb.2: Der Tanzwinkel der Bienen. Oben: Die Richtung der Futterquelle relativ zum Azimut der Sonne. Unten: Korrespondierende Tänze im Bienenstock. In Dunkelheit tragen die Bienen den Winkel, in dem das Futter relativ zum Azimut der Sonne zu finden ist, vom Lot ab. Befindet sich die Futterstelle im Bezug auf den Azimut der Sonne (im oberen Bild Verbindungslinie zwischen Sonne und Bienenstock) in einer 20°-Richutng gegen den Uhrzeigersinn, so tanzt die Biene im Stock 20° relativ zum Lot des Bienenstockes entgegengesetzt dem Uhrzeigersinn. Die innere Uhr der Bienen ist so genau, dass sie auch nach langer Zeit in Dunkelheit immer genau wissen, wie die Sonne gerade steht. Und so passen sie auch ihren Tanzwinkel im Laufe des Tages beim Bewerben von ein und derselben Futterquelle dem aktuellen Sonnenstand an. Quelle des Bildes: Fred C. Dyer, 2002, The Biology Of The Dance Language

Möchte eine Folgebiene die Futterquelle aufsuchen, weiß sie nun ungefähr wo sie suchen muss. In der Nähe der Futterquelle dienen Brauseflüge anderer Bienen dazu, die Folgebienen zum genauen Ort zu locken. Bei Brauseflügen umfliegen die Bienen das Ziel erst ein paar Mal, bevor sie sich niederlassen und beduften den Ort dabei mit einem bieneneigenen Duftstoff, dem Geraniol. Der kreisende Flug und der Duftstoff soll andere Bienen optisch und olfaktorisch anlocken. Der Geruch der Nahrung selbst, den die Bienen beim Kosten im Stock wahrgenommen haben, animiert sie beim Erreichen eines Zielortes zum Landen.
Wenn bereits mehrere Bienen rekrutiert wurden und an diesem Ort sammeln, so hilft den Neulingen zusätzlich noch eine Art Flugkorridor, um die Nahrung schneller zu finden. Zusätzlich haben die Bienen, die diese Futterquelle schon länger anfliegen den Weg dorthin schon besser gelernt und machen deswegen immer weniger Fehler beim Tanzen (speziell bei der Entfernungsangabe[1]).
Doch der Schwänzeltanz ist immernoch der erste und wichtigste Teil bei der Rekrutierung von anderen Bienen zu einer neuen Futterquelle.

2.2 Aufbau zur durchgehenden Beobachtung

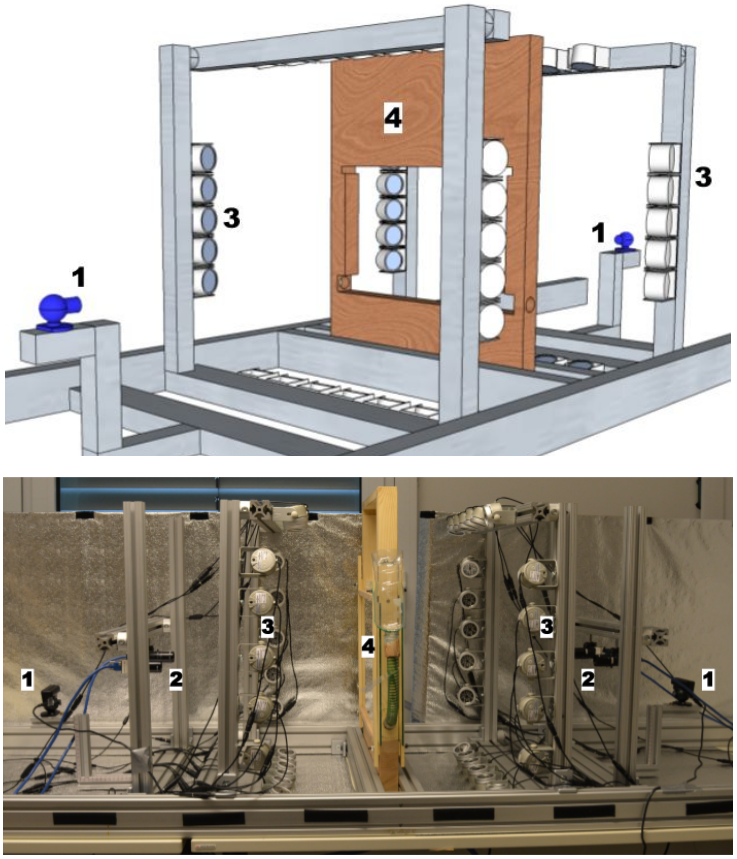

Abb.3: Aufnahme Setup. Installation der Wabe und der Kameras. Oben: Modell. Unten: Foto.
Zu sehen sind hier 1) PlayStation-Eye Kamera für jede Seite der Wabe. 2) je zwei hochauflösende Kameras pro Wabenseite. 3) IR-LED-Array auf jeder Wabenseite. Diese strahlen nicht frontal, sondern diffus, um homogenere Ausleuchtung und weniger Reflexionen auf der Scheibe zu erreichen. 4) Wabenrähmchen mit Deckscheiben – hier wird die Bienenwabe eingehangen.
Quelle beider Bilder: Wario et al., 2017, Automatic detection and decoding of honey bee waggle dances, submitted to PLUS ONE

Um Bienen untersuchen zu können, braucht es ein Bienenvolk. Dieses wurde direkt am Institut einquartiert. Es existiert ein dedizierter, abgedunkelter Raum, in dem eine Wabe auf eine Weise installiert wurde, sodass die Bienen mit Kameras über ihre gesamte Lebensspanne beobachtet werden können (vgl. Abb.3). Zu jeder Aufnahmesaison wird ein bestehendes Bienenvolk von ihrem Zuchtort in den "Bienenraum" umgesetzt. Dabei werden möglichst alle Bienen mit einem Tag auf ihrem Rücken versehen, der es erlaubt jede einzelne Biene eindeutig zu identifizieren. Dies ist für das Tracking von Wichtigkeit, das es erlaubt den zurückgelegten Weg einer jeden Biene auf der Wabe über ihr gesamtes Leben nachzuvollziehen. So lässt sich die soziale Struktur – welche Biene kommuniziert wo, wann und warum mit welcher anderen Biene – des Bienenvolkes analysieren. Diese sogenannte Tracking-Pipeline arbeitet mit

Bildmaterial von 4 hochauflösenden Kameras (2 Kameras pro Wabenseite, mit je 24MP), welche jedoch nur 3 Bilder pro Sekunde liefern. Ausreichend um die Tags auslesen zu können und Wege nachzuvollziehen. Eine Tanzerkennung ist auf diesem Material ebenfalls möglich. Für eine Tanzdekodierung ist das Material jedoch völlig unzureichend, da sich Schwänzelläufe meist im Bereich von Bruchteilen einer Sekunde abspielen. Die Dauer der Schwänzelphase könnte man nicht mit der nötigen Präzision bestimmen, da der genaue Start- und Endzeitpunkt des Schwänzellaufes schlicht nicht bekannt wäre. Und den Tanzwinkel müsste man im Ernstfall anhand von ein bis zwei geblurten Bildern der Biene bestimmen. Doch laufen die Bienen eher selten eine ideale gerade Strecke, weswegen der Mittelwert der Orientierungen mehrerer Momentaufnahmen des Schwänzellaufs für die letztendliche Orientierung genauer wäre.

Aus diesem Grund arbeitet das Tanzerkennungs- und -dekodierungssystem mit Bilddaten von zwei niedrigauflösenden Kameras (eine für jede Wabenseite), welche dafür jedoch 100 fps liefern. D.h. eine Periode des Schwänzelns kann auf ca. 6 bis 8 Bilder abgebildet werden, was eine mehr als ausreichende zeitliche Auflösung darstellt. Die Kameras sind dabei jedoch auf halbe VGA-Auflösung beschränkt. Was einer Auflösung von 320 x 240 Pixeln entspricht. In Abbildung 4 wird der Auflösungsunterschied der beiden Kameratypen verdeutlicht. Um uns Videoaufnahmen zu ermöglichen, die Bienen dabei jedoch nicht zu stören, wird die Wabe während des Aufnahmezeitraums lichtundurchlässig abgedeckt und die beiden Wabenseiten mit Infrarotlicht ausgeleuchtet, welches von Bienen optisch nicht wahrgenommen werden kann[3] – unseren Kameras, mit entfernten Infrarotfiltern, jedoch ermöglicht Licht einzufangen und somit Bilder aufzunehmen.

[3] Das Spektrum des für Bienen sichtbaren Lichts ist im Vergleich zum Menschen um etwa 150 Nanometer zum kurzwelligen UV-Licht hin verschoben. Bienen können also UV-Licht warnehmen - jedoch kein Rot. Diese Wellenlängen nehmen sie als schwarz war.[14]

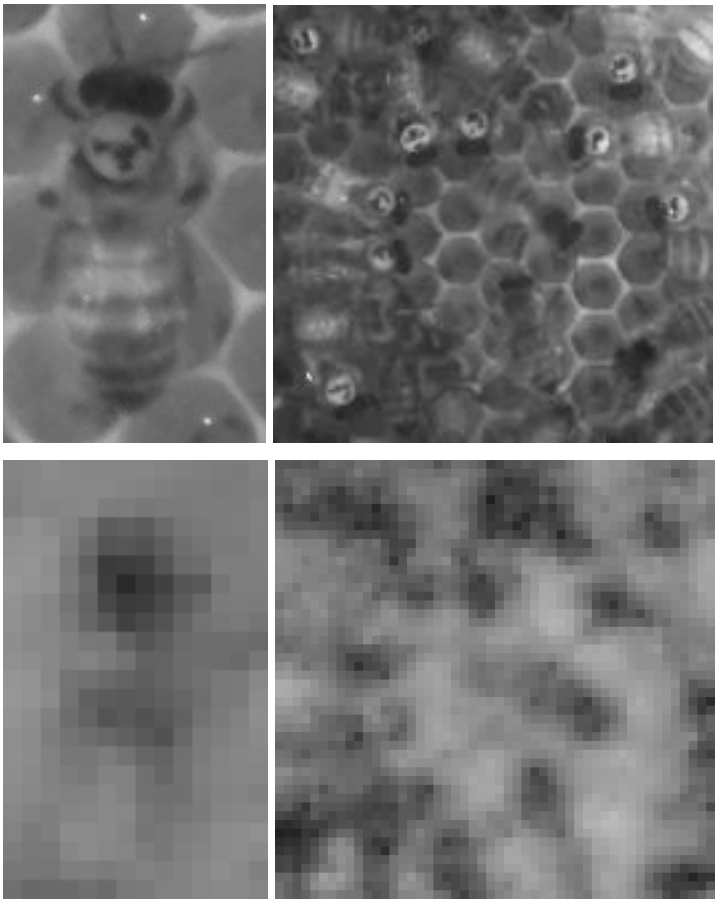

Abb.4: Visualisierung des Auflösungsunterschiedes zwischen den beiden Kameratypen für die Tracking-Pipeline (24MP) und die WDD-Pipeline (~0.077MP).

Links: Biene mit zirkulärem Binärcode auf dem Rücken

oben: Bildausschnitt einer der hochauflösenden Kameras (Tracking-Pipeline), auf dem eine Biene zu sehen ist. Man kann den Tag auf ihrem Rücken lesen.

unten: äquivalenter Bildausschnitt einer der niedrig auflösenden Kameras (WDD-Pipeline), auf dem eine Biene abgebildet ist. Sie lässt sich nur grob erkennen.

Rechts: Schwänzellauf – eine tanzende Biene umgeben von Folgebienen

oben: Bildausschnitt einer der hochauflösenden Kameras, der einen Schwänzellauf zeigt. Die schwänzelnde ist die obere der beiden umzingelten Bienen.

unten: ein 50x50px Bildausschnitt einer der niedrigauflösenden Kameras, so wie sie vom WDD abgespeichert werden. Der Ausschnitt zeigt den gleichen Tanz, wie im oberen Bild.

2.3 Bestehende Software

Im folgenden wird die Software Pipeline beschrieben, welche im Rahmen des Beesbook Projekts für die automatische Detektion und Dekodierung der Schwänzelläufe zuständig ist. Im folgenden nur WDD-Pipeline genannt.

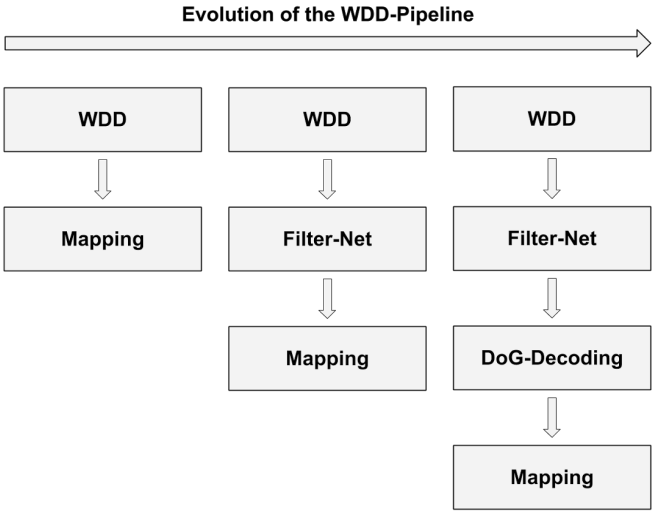

Abb.5: Entwicklung der WDD-Pipeline über die Zeit. Zunächst wurden die Tänze vom WDD[11] detektiert und auch dekodiert. Diese Daten wurden dann vom Mapping Modul genutzt, um die Tanzkoordinaten auf einer Karte darzustellen. Ein Filternetz[12] verbesserte später die Falsch-positiv-Rate. Eine neue Dekodiermethode auf Grundlage von Bilddaten[13] verbesserte die Dekodierqualität.

2.3.1 WDD

Diese Aufgabe übernahm zunächst der Waggle Dance Detector[11], im folgenden WDD genannt. Einige Komponenten dieses Systems wurden in der Vergangenheit bereits verbessert, bzw. ersetzt[12, 13]. Jedoch sei hier einmal seine Funktion erklärt.
Zur Detektion der Tänze wird die Periodizität des Helligkeitsverlaufs an bestimmten Pixeln – den Dot Detektoren – genutzt. Dabei wird der Helligkeitsverlauf der Pixel auf bestimmte Frequenzen hin untersucht, mit denen Bienen typischerweise schwänzeln (ca. 13 Hz [6]). Mehrere Detektionen einer Frequenz werden wiederum räumlich, wie zeitlich geclustert und einzelnen Schwänzelläufen zugeordnet. Abgespeichert werden dann die einzelnen, zugeschnittenen Frames der Läufe (statischer 50 x 50 Pixel großer Ausschnitt) als PNG-Dateien, sowie die Metadaten zum Tanz in einer CSV-Datei. Diese beinhalten die Startposition der Biene, also den Mittelpunkt der initial aktivierten Dot Detektoren, die Anzahl der Frames, woraus sich die Dauer des Laufes einfach berechnen lässt, einen Zeitstempel, den berechneten Tanzwinkel und eine Reihe von Koordinatenpaaren, welche allesamt die Trajektorie der Biene während des Schwänzellaufs darstellen. Der Tanzwinkel wird hierbei ganz simpel aus Start- und Endpunkt des Tanzes errechnet, entspricht also dem Winkel, den die Verbindungsgerade dieser beiden Punkte mit der Lotgeraden (y-Achse des Bildes) einnimmt.

2.3.2 Filternetzwerk

Da sich, aufgrund seiner Sensibilität, unter den Detektionen des WDD auch viele falsch positive Tänze befinden wurde die Präzision der Erkennung im Rahmen einer Bachlorarbeit[12] durch ein dahinter geschaltetes künstliches neuronales Netzwerk erhöht. Dieses klassifiziert Ausgaben des Waggle Dance Detectors binär, Tanz oder

Nicht-Tanz, womit unerwünschte Ergebnisse vor der Weiterverarbeitung herausgefiltert werden können.
Kurz vor dieser Arbeit wurde dieses Netz durch ein neues, vielversprechenderes Netz ersetzt, dessen Architektur in Abbildung 6 dargestellt wird. Mehr zu künstlichen neuronalen Netzen in Abschnitt 2.4.

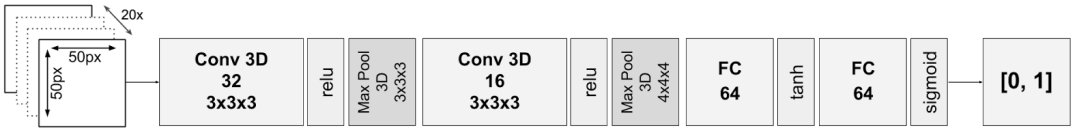

Abb.6: Architektur des Filternetzes. Das Netz nimmt als Eingabe 20 Frames mit einer Auflösung von je 50 x 50 px entgegen. Aufbau: Eine 3D-Konvolutionsschicht mit 32 3x3x3 Filtern und relu Aktivierung, gefolgt von 3D-Pooling mit 3x3x3 Kernel. Dann eine 3D-Konvolutionsschicht mit 16 3x3x3 Filtern und relu Aktivierung, gefolgt von 3D-Pooling mit 4x4x4 Kernel. Darauf folgt eine voll verbundene Schicht mit 64 Neuronen, tanh aktiviert, und eine voll verbundene Schicht mit 64 Neuronen und einer Sigmoidfunktion als Aktivierung. Die letzte voll verbundene Schicht stellt die Ausgabeschicht dar und besteht aus einem Neuron. Dieses liefert durch die Sigmoid-Funktion als Ausgabe eine Fließkommazahl zwischen 0 und 1, welche als Wahrscheinlichkeit für einen Tanz interpretiert wird.

2.3.3 Winkelextraktion

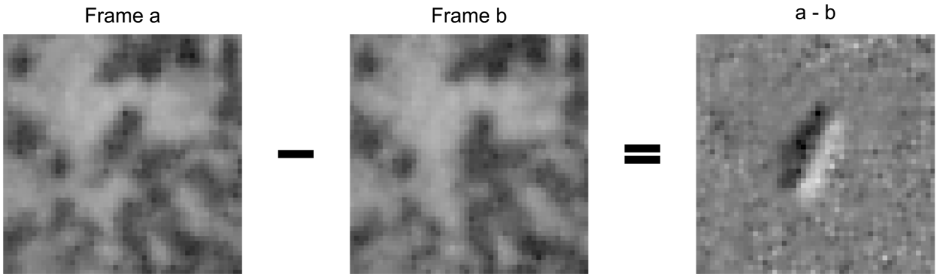

Abb.7: Differenzbild. rechts: Differenzbild aus zwei aufeinanderfolgenden Frames (links und mittig). Von den Pixelwerten von Frame a wurden die Pixelwerte von Frame b subtrahiert. Weiß und schwarz zu sehen sind die Orte im Bild, an denen die größte Helligkeitsveränderung von Frame a zu Frame b stattgefunden hat. Grau zu sehen sind die Orte, an denen ungefähr die selben Werte von einander abgezogen wurden.

Im Rahmen einer Bachelorarbeit[13] wurde hier die Winkelextrahierung stark verbessert, bzw. erstmals explizit implementiert. Auf Basis der abgespeicherten Bilddaten eines Tanzes wird im Anschluss der Winkel der Tanzrichtung mithilfe von Bilddifferenzen(Abb.7) ermittelt. An den verschiedenen Positionen des Abdomens der Biene (in aufeinander folgenden Bildern) entstehen lokale Minima und Maxima, siehe Abbildung 7 rechts. Diese Extremwerte entstehen durch den lateralen Ausschlag des Abdomens orthogonal zur Körperachse der Biene. Dort findet die höchste und großflächigste Änderung der Helligkeitswerte der Pixel im Bild statt. Mithilfe der FFT (fast Fourier transform) kann man diese Extremwerte der Bildreihe nun in einen Fourierraum transformieren (Abb.8), in dem sie nun unabhängig vom Bildort repräsentiert werden, jedoch die Informationen über die Orientierung der Biene nicht verlieren.

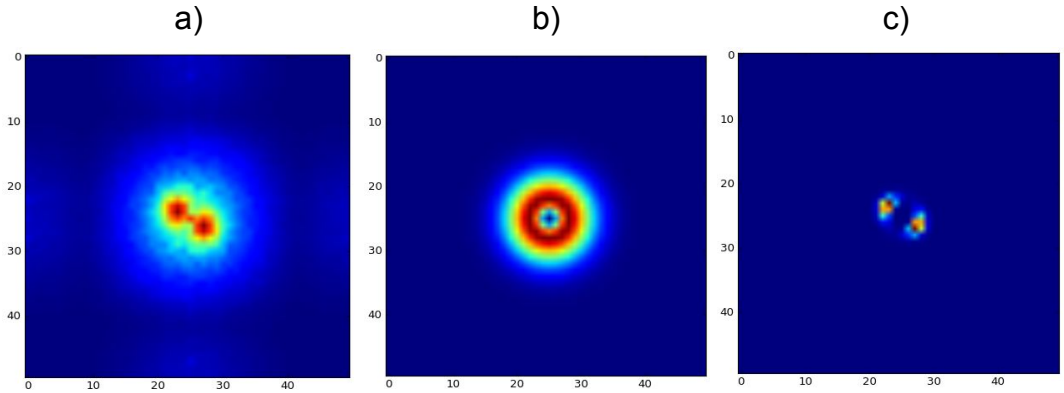

Abb.8: Im Fourierraum. (a) Summe der Fouriertransformationen aller Differenzbilder konsekutiver Frames eines Schwänzellaufs (b) Mexican-Hat-Kernel (c) Ergebnis der Konvolution mit b über a
Quelle: Wario et al., 2017, Automatic detection and decoding of honey bee waggle dances, submitted to PLUS ONE

Bildlich gesprochen: Wenn man durch diese 2 Maxima (Abb.8 c) eine gerade Linie zieht, dann ist die orthogonale auf dieser Geraden (die zweite Hauptkomponente) die Haupttanzachse. Ob die tatsächliche Tanzrichtung jedoch in die eine oder andere Richtung ausgehend von der Verbindungsgeraden zeigt ist jedoch nicht klar. Aus diesem Grunde wird die entgültige Tanzrichtung mithilfe der bereits erwähnten Trajektorie bestimmt, was jedoch nicht fehlerfrei möglich ist wenn die Vorwärtsbewegung der Biene zu gering ist. Die Verteilung der Fehler, die diese Methode birgt, hat einen Mittelwert von -3.29° und eine Standardabweichung von 7.17°.

2.3.4 Mapping Modul

Durch dieses Script werden die Tänze, bzw. die dadurch beworbenen Orte kartiert. Dazu werden die gatanzten Polarkoordinaten auf die echte Welt übertragen – so, wie es eine Biene, nach aktuellem Stand der Forschung, auch tun würde – und der Zielort als Punkt auf einer Karte dargestellt.
Hierbei handelt es sich tatsächlich um Tänze, nicht um Läufe. Die detektierten Läufe werden anhand ihrer Metadaten zeitlich und örtlich zu Tänzen geclustert. D.h. ein Tanz besteht nun aus Schwänzelläufen, die innerhalb eines bestimmten Zeitfensters an ungefähr dem gleichen Ort auf der Wabe stattgefunden haben. Diese Cluster werden danach mit dem RANSAC (RANdom SAmple Consensus)-Algorithmus von Ausreißern im Bezug auf die Tanzorientierung bereinigt. Die Dauer und Orientierung der übriggebliebenen Läufe werden dann pro Cluster gemittelt. Diese Mittelwerte, also die Polarkoordinaten mit dem Bienenstock als Pol, für jeden Tanz werden dann als Punkt auf einer Karte dargestelllt. Siehe Abbildung 9.

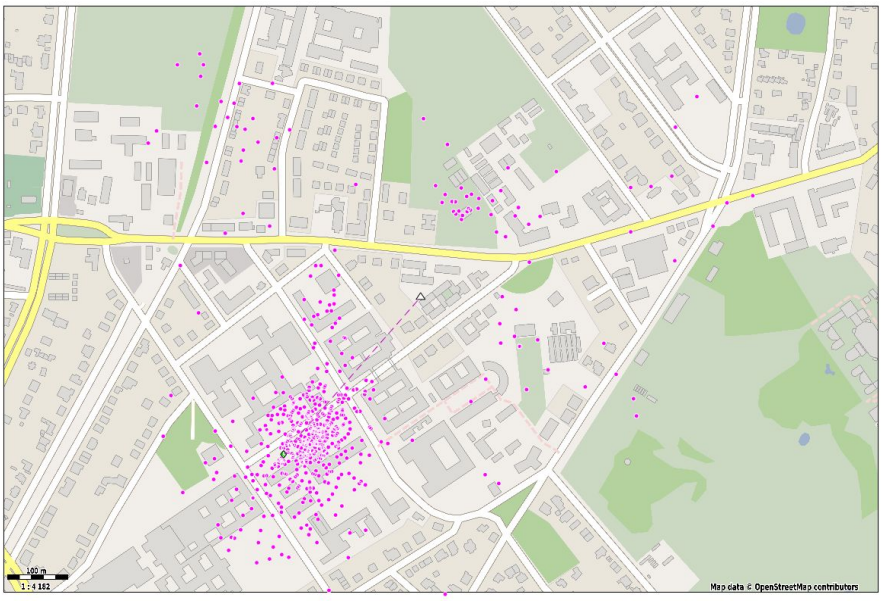

Abb.9: Geclusterte Tanzdetektionen. Jeder Punkt auf der Karte entspricht dem errechneten Ziel eines Schwänzeltanzes, der aus mindestens 4 Schwänzelläufen besteht. Weißes Dreieck: Bienenstock; Grüne Raute: Futterstelle; gestrichelte Linie: durchschnittliche Tanzrichtung
Quelle: Wario et al., 2017, Automatic detection and decoding of honey bee waggle dances, submitted to PLUS ONE

2.3.5 Zusammenfassung

Am Biorobotics Lab der FU Berlin existiert bereits eine Sammlung an Softwarekomponenten, welche in ihrer Gesamtheit die Aufgabe erfüllen Tänze zu erkennen, zu dekodieren und zu kartieren. Jedoch steckt in der Qualität und Effizienz bei der Erfüllung dieser Aufgaben noch Verbesserungspotenzial.
Die aktuelle Pipeline besteht aus vier Modulen, welche verschiedene Technologien miteinander verbinden. Die Echtzeitfähigkeit einzelner Module ist beschränkt oder wurde noch nicht umgesetzt. Der WDD läuft in Echtzeit, liefert jedoch eine große Zahl an falsch positiven Ausgaben. Was mit dem Filternetz ein zusätzliches Modul nötig macht. Jedoch bringt die Echtzeitanwendung den großen Vorteil, dass die aufzuzeichnenden Datenmengen stark reduziert werden, da nur kleine Ausschnitte potentieller Tänze abgespeichert werden müssen. Die Ergebnisse der aktuellen Winkelextraktion sind sehr solide, haben jedoch einen potentiellen 180° Fehler. Aktuell kann die Dekodierung nur im Rahmen des Postprocessings angewendet werden. Das Mapping Modul muss die einzelnen Schwänzelläufe erst noch zu Tänzen clustern, bevor sinnvolle Ausgaben entstehen und Läufe mit 180° Fehler aussortiert werden können. Momentan müssen also 4 verschiedene Komponenten manuell nacheinander ausgeführt werden, was einen großen zeitlichen, wie personellen Aufwand bedeutet und nicht wirklich der eigentlichen Idee der automatisierten Detektion und Dekodierung von Schwänzeltänzen entspricht.

2.4 Künstliche neuronale Netzwerke (KNN)

Die Realisierung einer genaueren und effizienteren Dekodierung mithilfe eines KNN wurde nun schon mehrfach postuliert. Im folgenden wird erklärt was künstliche neuronale Netze sind. In 2.4.2 und 2.4.3 werden nochmal zwei spezielle Ausprägungen, die CNNs (Convolutional Neural Networks) und RNNs (Recurrent Neural Networks) vorgestellt. Dies sind zwei Arten von KNNs, die sich gut für unser Problem eignen würden.

2.4.1 Multilayer Perceptron (MLP)

Ein künstliches neuronales Netz ist ein Modell für Informationsverarbeitung, welches grob vom natürlichen, biologischen neuronalen Netz, also einer Vielzahl miteinander verbundenen Neuronen in einem Nervensystem, inspiriert ist. Ein künstliches Neuron (Abb.10 B) repräsentiert hier in folgender Weise sein biologisches Pendant (Abb.10 A). Es hat mehrere Signaleingänge, so wie eine Nervenzelle über ihre Dendriten mit mehreren Neuriten anderer Nervenzellen verbunden ist und über diese Signale empfängt. Alle Signaleingänge sind gewichtet und werden im künstlichen Neuron aufsummiert, was grob der Integration von Potentialen in der Nervenzelle entspricht.
Danach überführt eine Aktivierungsfunktion (vgl. Abb.A1 im Anhang) die gewichtete Summe in den Ausgabewert.
Dieser Wert kann der Eingabewert für die nächsten verbundenen Neurone oder auch der Ausgabewert des gesamten Netzes sein. Bei einer Nervenzelle entscheidet sich am Axonhügel, ob ein Aktionspotential bei Erreichen eines bestimmten Schwellwertes ausgelöst wird oder nicht. Ob ein Signal an andere angebundene Nervenzellen abgegeben wird oder nicht. Die Signalabgabe eines künstlichen Neurons gestaltet sich hingegen nicht unbedingt binär. Es gibt in der Regel reelle Werte weiter und kann somit verschieden starke Signale weitergeben, was die Feuerrate (Frequenz der Aktionspotentiale) eines Neurons modellieren soll.

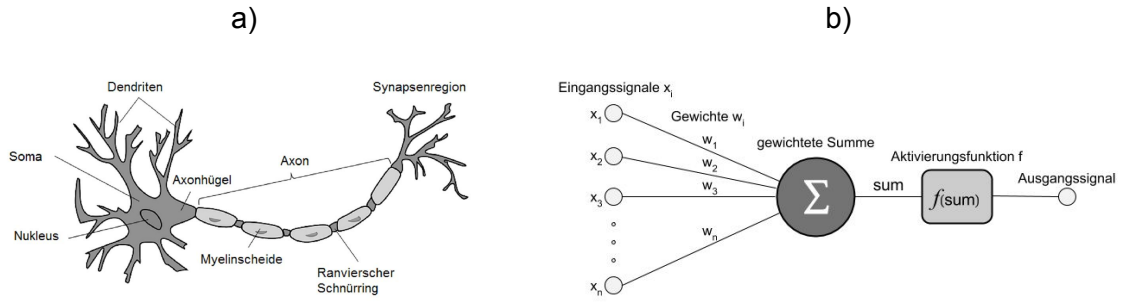

Abb.10: natürliches und künstliches Neuron. a) Nervenzelle **b)** künstliches Neuron
Das künstliche Neuron ist von der biologischen Nervenzelle inspiriert. Es modelliert grob die Signalaufnahme in den Dendriten und die Integration dieser Signale im Axonhügel mithilfe einer gewichteten Summe der Eingangssignale. Die Aktivierungsfunktion, der das Ausgangssignal entspringt, soll die Feuerrate des Neurons modellieren.
Quelle Bild A: http://wiki.sportsandscience.de/wiki/aufbau-der-nervenzelle/

So ein künstliches Neuron wird gemeinhin als einfaches Perzeptron, vom englischen perception, zu deutsch "Wahrnehmung", bezeichnet.
Ein einlagiges Perzeptron (Abb.11 A) besteht wiederum aus mehreren einfachen Perzeptronen, welche in einer Schicht angeordnet sind, also in einem Zeit- oder Verarbeitungsschritt gleichzeitig aktiv sind bzw. parallel abgearbeitet werden.
Ein mehrlagiges Perzeptron (MLP, multi-layer perceptron, Abb.11 B) besteht folglich aus mehreren Schichten von künstlichen Neuronen, wobei alle Neurone einer Schicht mit allen Neuronen der darauf folgenden Schicht (gewichtet) verbunden sind.

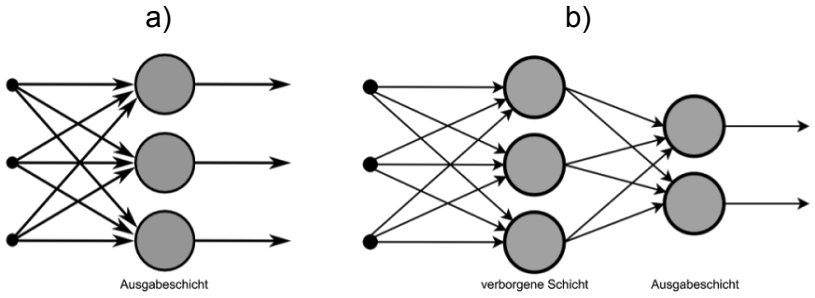

Abb.11: einfache künstliche neuronale Netze
a) einlagiges Perzeptron **b)** mehrlagiges, in diesem Falle 2-lagiges, Perzeptron
Quelle A: von Chrislb - Erstellt von Chrislb, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=224492; Quelle B: von Offnfopt - Eigenes Werk created using File:MultiLayerNeuralNetwork english.png as a reference., Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=39500777

Dieses mehrlagige Perzeptron ist das, was wir heute unter einem klassischen, einfachen künstlichen neuronalen Netz verstehen. Im Grunde ist so ein Netz als eine große Funktionskomposition zu verstehen, die im Stande ist jede nur erdenkliche Funktion sehr genau zu approximieren, wenn nur die Netzkapazität, d.h. die Anzahl der Neurone und Gewichte, hoch genug ist. Die grundlegende Funktion eines solchen Netzes besteht nun darin einen Eingabevektor (vgl. Abb.10 B, Eingangssignale x_i) in einen Ausgabevektor zu verwandeln ('forward pass'). Und ob dieser Ausgabevektor dem gewünschten entspricht entscheiden hauptsächlich die Gewichte. Diese müssen jedoch nicht von Hand festgelegt werden, sondern werden erlernt. KNNs müssen somit nicht explizit programmiert werden. Wenn die Topologie des Netzes und bestimmte Parameter festgelegt wurden erfolgt ein Training mit Eingabevektor-Ausgabevektor-Paaren. Dabei werden die Gewichte so angepasst, dass die tatsächliche Ausgabe des Netzes bei einer bestimmten Eingabe so genau wie möglich der gewünschten entspricht. Dafür ist eine geeignete Fehlerfunktion nötig, die den Abstand der tatsächlichen Ausgabe zur gewünschten quantifiziert. Dieser Fehler wird nun mithilfe des Backpropagation Algorithmus (Paul Werbos, 1974) in jedem Trainingsschritt von der Ausgabe- hin zur Eingabeschicht zurückgeführt und alle Gewichte auf dem Weg dorthin, abhängig von ihrem Einfluss auf den Fehler, geändert. Sodass bei erneuter identischer Eingabe der Fehler kleiner ausfällt.
Beim Training eins Netzwerks auf ein bestimmtes Problem hin, präsentiert man ihm nun eine große Anzahl von Repräsentanten dieses Problems, inkl. der gewünschten Ausgabe/Lösung, und hofft, dass beim Trainingsprozess eine Gewichtsbelegung

gefunden wird, sodass das Netz auch bei neuen, dem Netz unbekannten, Eingaben eine richtige Ausgabe liefert, also generalisieren kann.
Für tiefergehende Informationen wird an dieser Stelle auf [32] verwiesen.

2.4.2 Convolutional Neural Network (CNN)

Die zwei größten Unterschiede von CNNs[15] zu einfachen MLPs sind die lokale Konnektivität und die geteilten Gewichte. Das heißt, dass nicht alle Neurone einer Schicht mit allen Neuronen der nächsten Schicht verbunden sind, sondern die Neurone jeweils nur mit ausgewählten Neuronen der vorherigen Schicht verbunden sind. Außerdem teilen sich alle Neurone einer Schicht die selben Gewichte. Dies spart ungemein viele Parameter und folglich Verarbeitungsschritte, was größere und tiefere (im Sinne von mehr Schichten) Netze und auch viel größere Eingaben ermöglicht. Es ist nun möglich ganze Bilder, also 2D- und 3D-Matrizen als Eingabe zu verwenden und effizient zu verarbeiten. Genauso sind dann auch die Neurone in den einzelnen Schichten 2- oder 3-dimensional angeordnet. Der typische Aufbau von CNNs, mit sich abwechselnden Konvolutions- und Subsampling-Schichten, ist in Abbildung 12 dargestellt. Für tiefergehende, jedoch leicht verständliche, Erklärungen wird auf [16] verwiesen.

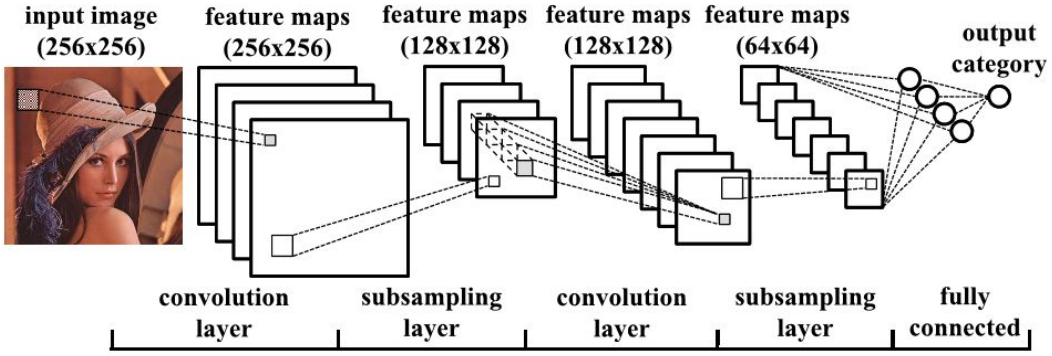

Abb.12: Beispiel eines Convolutional Neural Network
Eine 'feature map' ist das Ergebnis einer Konvolution der Eingabe mit ein und demselben Kernel. Ein Kernel, im Kontext der Computer Vision auch Filter genannt, entspricht hier einer Gewichtsmatrix. Hinter den Konvolutionsschichten, welche die relevanten Bildmerkmale herausfiltern, ist ein MLP geschaltet, welches ausschließliich für die Klassifikation zuständig ist.
Quelle: Cong J., Xiao B. (2014) Minimizing Computation in Convolutional Neural Networks. In: Wermter S. et al. (eds) Artificial Neural Networks and Machine Learning – ICANN 2014. ICANN 2014. Lecture Notes in Computer Science, vol 8681. Springer, Cham

2.4.3 Recurrent Neural Network (RNN)

RNNs sind neuronale Netze, welche nicht nur vorwärts gerichtete Verbindungen haben, sondern bspw. auch rückwärts gerichtete Verbindungen zu sich selbst im nächsten Verarbeitungsschritt. D.h. die Ausgabe eines Neurons ist im nächsten Zeitschritt Teil dessen Eingabe. Dies verschafft RNNs ein Gedächtnis, was sie für Anwendungen prädestiniert, bei denen sequenzielle Daten verarbeitet und/oder generiert werden sollen.

In der Praxis werden meist LSTMs[17] oder GRUs[18] genutzt.

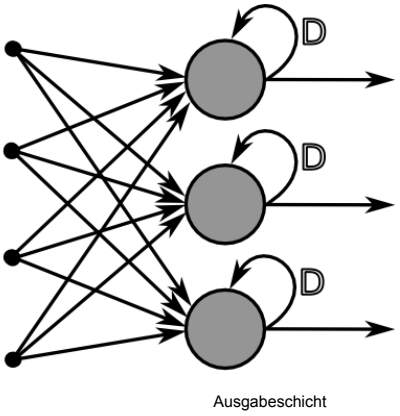

Abb.13: sehr einfache Form eines RNN mit direkter Rückkopplung. Die Neurone bekommen im nächsten Verarbeitungsschritt zusätzlich zu den normalen Eingaben ihre eigene Ausgabe aus aktuellem Schritt. 'D' steht hier für direkt – direkte Rückkopplung.
Außerdem möglich: indirekte Rückkopplung (Ausgang eines Neurons ist mit Eingang eines Neurons in vorgeschalteten Schichten verbunden), seitliche Rückkopplung (Ausgang eines Neurons ist mit Eingang eines Neurons in gleicher Schicht verbunden) und vollständige Verbindung (jedes Neuron ist mit jedem anderen Neuron verbunden).
Quelle: Von Chrislb – Erstellt von Chrislb, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=224500

2.5 Verwandte Arbeiten

Die lokale Konnektivität bei CNNs spiegelt das Prinzip des rezeptiven Feldes in unseren Sinnesorganen wieder. Die geteilten Gewichte folgen der Annahme, dass bestimmte Muster, im CNN durch die Filter repräsentiert, an allen Bildorten von Interesse sein könnten. Das macht CNNs (im folgenden auch ConvNets genannt) generell gut geeignet für Bilddaten (spatial data), während RNNs, wie bereits in 2.4.3 erwähnt, gut geeignet für Zeitreihen (temporal data) sind. Da unser Problem beides abdeckt – Zeitreihen, in denen jedes Datum der Reihe ein Bild darstellt – musste zunächst herausgefunden werden welche Grundarchitektur sich höchstwahrscheinlich am besten für unsere Aufgabe eigenen würde.
Dazu wurden Arbeiten ausgewertet, in denen CNNs[19, 20, 21, 22] und/oder RNNs[24, 25, 26] dazu genutzt wurden Verhaltenserkennung auf Videodaten zu realisieren. Deren Architektur sich also evtl. auf unser Problem anwenden lässt. Im Folgenden wird auf die jeweils relevanten Arbeiten eingegangen und Vor- und Nachteile im Bezug auf unsere Anwendung erläutert.

2.5.1 Arbeiten zu CNNs

In [19] wurde eine Architektur vorgestellt, welche Tätigkeiten von Menschen erkennt und klassifiziert. Input für dieses Netzwerk sind eine Reihe von Frames aus einem Video, welche dann von einer hardgecodeten Schicht, drei 3D-Konvolutionsschichten und einem voll verbundenen (fully connected (FC-) Layer) verarbeitet werden. Die erste hardgecodete Schicht extrahiert mit festgelegten Filtern

aus den Eingaben 5 Informationskanäle, welche die Eingabe für die nächsten Schichten sind. Zusätzlich wird ein 'human detector' genutzt, um das Netzwerk mit schon fertigen Ausschnitten von Menschen zu füttern. Die Vorverarbeitungsschritte sind also schon umfangreicher. Jedoch ist der Aufbau des Netzes sehr einfach gehalten. Es könnte für unser Problem ausreichend sein und könnte auch sehr einfach trainiert werden. Das Ausschneiden von ROIs (Region Of Interest) passt zudem gut zu unserem Konzept, der Ausgabe des WDD.

[20] stellt ein 2D-ConvNet mit Two-Stream Ansatz vor, welches die räumlichen Informationen (Frames) getrennt von den zeitlichen (optical flow konsekutiver Frames) in je einem separatem Netz verarbeitet und am Ende die Klassengewichte fusioniert.

Später wurde diese Architektur in [21] um 3D-Kovolution und 2 neue Fusionsmethoden ergänzt, welche die beiden Streams entweder schon vor der letzten Konvolutionsschicht fusioniert oder nach der letzten Konvolutionsschicht fusionieren, jedoch neben diesem fusionierten Hybridstream den räumlichen Stream unverändert parallel weiterverarbeiten und beide vor der Klassifikation nochmals entgültig fusionieren. Das trainierte Netz verarbeitete Vollbilder und keine zugeschnittenen Frames. Jedoch ist eine Vorverarbeitung dieser vonnöten.

In [22] wurde ein 3D-ConvNet für Videoklassifizierung entwickelt und dabei genauer auf die optimalen Hyperparameter, wie Strides (Schrittweiten) und Filterdimensionen eingegangen. Es wurde festgestellt, dass 3D-ConvNets generell besser auf Videodaten funktionieren als 2D-ConvNets und dass kleine, homogene Filterdimensionen am besten funktionieren. Auf dem UCF101[4] test split-1 lieferten 3x3x3-Konvolutionsfilter und 2x2x2-Pooling-Filter die besten Ergebnisse bei einer Eingabe von Vollbildern. Die vorgeschlagene Architektur ist also einfach und konventionell gehalten und liefert dabei auf allen getesteten Datensätzen, bis auf einen, durchweg bessere Ergebnisse gegenüber den besten bis dato publizierten.

2.5.2 Arbeiten zu RNNs

In [24] wird ein 3D-ConvNet durch ein RNN, anstatt wie sonst üblich durch ein MLP, als Klassifikator ergänzt. Das Netz wurde jedoch ausschließlich auf den KTH Datensatz trainiert. Dies ist ein älterer Datensatz, der lediglich aus 6 verschiedenen Klassen bzw. Aktivitäten (Laufen, Joggen, Rennen, Boxen, Winken und Klatschen) besteht, wobei diese von nur 25 verschiedenen Personen ausgeführt werden. Die Performance dieser Architektur lässt sich also nicht gut mit den anderen hier vorgestellten vergleichen. Auf dem KTH Datensatz jedoch lieferte sie gute Ergebnisse.

[25] stellen eine Architektur vor, welche für Tätigkeitserkennung von Menschen ein 2D-ConvNet als feature extractor (Merkmalsfilter, -extraktor) auf Frameebene und ein RNN als Klassifikator nutzt. Die zeitlichen Informationen werden also in diesem Falle nicht durch 3D-Konvolution extrahiert, sondern durch das RNN. Die Konvolutionsschichten filtern relevante Merkmale für jedes Frame heraus. Schritt für Schritt werden dann diese Ergebnisse in das RNN gegeben. In jedem Zeitschritt (für

[4] Datensatz mit 13.320 Videos von 101 verschiedenen Tätigkeiten aus Sport, Musik und Alltag [23]

jedes Frame) erfolgt durch das RNN eine Klassifikation. Die einzelnen Ergebnisse werden dann für die entgültige Klassifizierung gemittelt. Ein Vorteil dieser Architektur ist, dass das Netz Ende-zu-Ende trainierbar ist. Ein weiterer großer Vorteil in Hinblick auf die Ausgabe des WDD ist, dass die Eingabe nicht auf eine bestimmte Anzahl von Frames beschränkt ist. Als Eingabe wären also ganze Sequenzen unterschiedlicher Länge geeignet.

In [26] wird eine Lösung für die Tätigkeitserkennung in laufenden Videos im Rahmen der ActivityNet Challenge 2016[27] vorgestellt. Der Aufbau ist ähnlich zu dem in [25]. Jedoch werden nicht die einzelnen Frames als Eingabe für ein 2D-ConvNet genutzt. Stattdessen werden die Videos in 16 Frames große Abschnitte geteilt und jeweils in ein 3D-ConvNet wie in [22] gegeben. Die extrahierten räumlichen und zeitlichen Merkmale pro Abschnitt werden dann von einem RNN klassifiziert. Die ActivityNet Challenge besteht aus Detektion und Klassifikation von Tätigkeiten. Daher werden die resultierenden Klassifikationssequenzen anschließend genutzt, um die Tätigkeit zeitlich du lokalisieren. Dieser Ansatz könnte hilfreich sein, um die Bestimmung der Länge eines Schwänzellaufes zu verbessern. Da aktuell die Länge eines Schwänzellaufes durch die Anzahl der Frames einer Sequenz bestimmt wird, die Sequenzen jedoch oft nicht nur die Schwänzelphase beinhalten.

3. Implementierung

3.1 Datensatz

Der Gesamtdatensatz, auf den für diese Arbeit zurückgegriffen werden konnte sind WDD-Ausgaben der Saison 2016. Es handelt sich hierbei um Tanzaufzeichnungen aus 50 Tagen von Ende Juli bis Ende September. Die Anzahl der Detektionen pro Tag unterscheidet sich dabei sehr. Die Aufzeichnung eines Schwänzellaufes liegt, wie in 2.3.1 beschrieben, als Folge von Bilddateien der Größe 50 x 50px (vgl. Abb.4 unten rechts) vor. Die tanzende Biene ist darauf meist zuerst im Zentrum des ersten Bildes zu sehen und wandert dann, je nach Tanzwinkel, auf den folgenden Bildern in Richtung einer der Bildränder. Zu jeder Detektion gibt es zusätzlich zu den Bilddaten noch eine CSV-Datei, deren Inhalt ebenfalls in 2.3.1 beschrieben wurde.

Zum Evaluieren der Güte der Winkeldekodierung existiert ein Testdatensatz von 200 Schwänzelläufen die am 14.08.16 zwischen 10:00 und 10:30 aufgezeichnet und von 8 Menschen manuell, mithilfe eines Python-Tools, mit einem Tanzwinkel versehen wurden. Die Ground Truth Daten für diese 200 Läufe sind nun die jeweils 8 gemittelten Winkel. Diese 8 Menschen dekodierten die Winkel mit einer Standardabweichung von 6,66°. Gegen einen Teil dieses Datensatzes wurde auch die aktuelle Methode aus [13] getestet.

Außerdem existieren Ground Truth Daten zu 4239 Detektionen des WDD (vom 14.08.16 zwischen 09:00 und 16:00) im Bezug darauf, ob es sich bei der Detektion tatsächlich um einen Tanz handelt oder nicht. Neben 'ja' (Tanz) und 'nein' (kein Tanz) existiert noch das Label 'vielleicht' für Fälle in denen die zuständige Person nicht mit Sicherheit sagen konnte, ob es sich um einen Tanz handelt oder nicht. Mit äquivalenten

Ground Truth Daten, nur mit Bildmaterial aus 2014 und 2015, wurde das Filternetz[12] trainiert und getestet.

3.2 Anforderungen

Für diese Arbeit wurde das Ziel festgelegt, die Dekodierung der Schwänzelläufe mithilfe eines künstlichen neuronalen Netzes zu realisieren, in der Hoffnung damit die Qualität, jedoch vor allem auch die Effizienz der Dekodierung verbessern zu können. Dabei soll außerdem die Anzahl der Module veringert werden, indem dieses neue Netz zusätzlich die Funktion des Filternetzes mit übernimmt.

3.2.1 Qualität

Explizit sollen die Fehler der Dekodierung mindestens so gering, wie bei der aktuellen Methode[13] (μ = -3,29°, std = 7,17°) sein. Besser noch wäre es die Genauigkeit des Menschen zu erreichen. Sprich ein Mittelwert von 0° und eine Standardabweichung von 6,66° (vgl. 3.1 Abs. II).
Das aktuelle Filternetz erreicht eine Korrektklassifikation (accuracy) von 90,07% bei 89,8% Sensitivität (recall) und 95% Genauigkeit (precision).
Der WDD an sich liefert, je nach Feinjustierung seiner Parameter und Interpretation der Ausgaben[5], ca. 50% falsch positive Ausgaben[6]. Die Sensitivität kann nicht genau spezifiziert werden, da der WDD auf einem Videostream arbeitet und nur potentielle Tänze abgespeichert werden. Angesichts der hohen Falsch-positiv-Rate kann jedoch von einer sehr hohen Sensitivität nahe der 100% ausgegangen werden.
Folglich sollte die erarbeitete Lösung die Klassifikationsgüte des Filternetzes erreichen. Wobei eine höhere Sensitivität wünschenswert wäre.

3.2.2 Performance und Technologie

Die Lösung soll in Echtzeit anwendbar sein und sich in die bestehende Softwareumgebung integrieren lassen, wobei innerhalb der WDD-Pipeline hauptsächlich Python als Programmiersprache genutzt wird. So wurde für die Entwicklung eines Prototypen Keras[28] mit Theano als Backend genutzt. Keras ist ein high-level Deeplearning Framework für Python, mit dem man sehr einfach und schnell neuronale Netze implementieren kann. Sollte die Komplexität der Architektur die Möglichkeiten von Keras übersteigen oder eine andere Art der Implementierung für das Endprodukt gewünscht oder vonnöten sein, so sollte auf ein adäquates Framework umgestiegen werden.

[5] Bei manchen Detektionen kann auch ein Mensch nicht mit Sicherheit sagen, ob es sich um einen Tanz handelt oder nicht.
[6] evaluiert anhand von 8239 Detektionen

3.3 Designentwurf

Der Kern dieser Arbeit ist es das Filter-Netzwerk und den Dekoder in einem Netzwerk zu realisieren. D.h. es soll ein Dekoder-Netzwerk sein, welches falsch positive Tänze erkennt. Das Netzwerk bekommt als Eingabe die Ausgabesequenzen des WDD. Es entscheidet, ob es sich bei der Sequenz um einen Schwänzellauf handelt oder nicht und gibt bei positivem Ergebnis den Tanzwinkel aus. Dass ein Sample kein Tanz war soll geeignet kodiert werden. Denn um ohne eine Änderung des Mapping-Moduls auszukommen, müsste das Echtzeit-Framework dafür sorgen, dass das MM keine Daten bekommt, wenn es sich nicht um einen Tanz gehandelt hat.

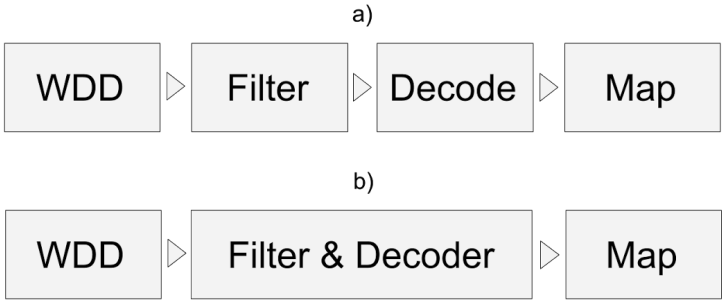

Abb.14: Aufbau der Dance-Detection-and-Decoding-Pipeline.
a) aktueller Aufbau
Der WDD (Attention-Module, kurz AM) liefert Detektionen an Filternetzwerk (FN).
FN liefert von falsch positiven bereinigte Tänze an Dekoder (Orientation Module, kurz OM).
OM liefert Tanzwinkel und -dauer an Mapping Module (MM).
MM clustert Läufe und stellt Tänze auf Karte dar (siehe Abb.9).
b) geplanter Aufbau
Das neue Dekodiernetzwerk (Mitte) bekommt als Eingabe Ausschnitte vom WDD. Es entscheidet, ob es sich um einen Tanz handelt oder nicht und dekodiert im Falle eines Tanzes. Das Framework müsste dann dafür sorgen, dass keine Daten an das Mapping Module gereicht werden, wenn es sich nicht um einen Tanz gehandelt hat.

3.4 Architektur

3.4.1 Grundarchitektur

Da unser Problem, die Tanzdetektion und -dekodierung, viel kleiner ist als beispielsweise die Klassifikation von 101 unterschiedlichen menschlichen Aktivitäten wurde entschieden, dass 3D-ConvNets völlig ausreichend für dessen Lösung sein müsste. Je nach Betrachtungsweise haben wir es nur mit 2 Klassen zu tun – Tanz oder Nicht-Tanz. Die Klasse Tanz gibt es dann nur noch in verschiedenen Ausprägungen (Länge, Winkel, Schwänzelfrequenz...), also mit einer relativ hohen Intravarianz. Somit wurde die Entwicklung eines 3D-ConvNets beschlossen, dessen Architektur sich grob an [22] orientieren soll.
Denn trotzdem beispielsweise die Berechnung des optical flows in Echtzeit möglich ist, soll das Ziel dieser Arbeit ein Netz sein, welches so wenig wie möglich

Vorverarbeitung des Inputs benötigt. Da der Input zwar nicht hoch aufgelöst ist, jedoch 100fps liefert und die Echtzeitfähigkeit eine wichtige Anforderung ist.

3.4.2 Verwendete Architektur

Im Laufe der Arbeit wurden viele Architekturen getestet. Zwei Ansätze zur Modellierung des Problems wurden dabei ausprobiert.

Einerseits wurde versucht es als Klassifizierungsproblem zu modellieren. Die Idee ist die Ausgabe eines Vektors der Länge 361, welcher die Wahrscheinlichkeitsverteilung für die 361 verschiedenen Klassen repräsentiert. Die ersten 360 Komponenten repräsentieren dabei die Oberklasse 'Tanz', Komponente 361 die Oberklasse 'kein Tanz'. Die Komponenten 0 bis 359 entsprechen dabei den Tanzwinkeln von 0° bis 359°. Bei einer einzigen Eingabe wäre die Dekodierung des Tanzwinkels also nur auf einen Grad genau möglich. Da das Dekodierungsergebnis später meist auf der Eingabe von mehreren versetzten Ausschnitten eines Schwänzellaufes basieren wird, wird die Ausgabe des Winkels, durch Mittelung mehrerer Werte, letztendlich trotzdem differenzierter ausfallen. Spätestens beim Clustering zu Tänzen (Kapitel 2.3.4) werden mehrere Läufe gemittelt und so die Genauigkeit des Tanzwinkels erhöht.

Der zweite Ansatz war die Modellierung als Regressionsproblem. Dabei soll die Ausgabe aus zwei Skalaren im Intervall [0, 1] bestehen, welche die beiden Komponenten (x, y) eines Vektors repräsentieren. Die Idee dabei ist, dass dieser Vektor entweder in die Tanzrichtung zeigt, die Ausgabe also der Tanzwinkel in Radien – berechnet durch $arctan2(y,\ x)$ – ist, oder beide Komponenten Null sind, bzw. die Länge des Vektors unterhalb eines bestimmten Thresholds liegt, was der Klasse 'kein Tanz' entspricht.

Da die ersten Prototypen mit Klassifizierungsansatz nicht zum Lernen gebracht werden konnten, war eine der Maßnahmen, um herauszufinden woran es lag, die Aufgaben des Netzes vorerst wieder zu trennen. Es sollten Architekturen getrennt (1) nur auf die Klassifizierung in Tanz oder nicht-Tanz, (2) auf die Dekodierung mit Klassifizierungsansatz oder (3) auf die Dekodierung mit Regressionsansatz trainiert werden. So konnte besser untersucht werden, ob die jeweilige Aufgabe von einer Netzarchitektur erlernt werden kann. Durch die isolierte Betrachtung konnten schnell Defizite in den genutzten Datensätzen festgestellt werden.
Im Folgenden werden nun die zwei relevanten Architekturen aus Kategorie (1) und (2) vorgestellt, mit denen sich die besten Ergebnisse erzielen ließen. Mit dem Regressionsansatz konnten im Rahmen dieser Arbeit leider keine guten Ergebnisse erzielt werden.

Wie in 2.3.2 bereits erwähnt, wurde das Filternetz aus [12] gerade durch ein neues ersetzt, da dies vermeintlich bessere Ergebnisse lieferte. Es wurde jedoch festgestellt, dass dieses Netz, welches mithilfe der in 3.1 erwähnten Ground Truth Daten vom 14.08.16 trainiert wurde, auf Tanzdaten vom 16.08.16 nicht so gut performt, wie es die Ergebnisse der Evaluation mit einem Validation-Split des Datensatzes vom 14.08.16

vermuten ließen. Aus diesem Grund musste die Architektur des Filternetzes, der Datensatz sowie der Trainingsprozess überarbeitet werden.
Die beiden hier vorgestellten Architekturen orientieren sich an der Architektur des nun ganz neuen Filternetzes. Die ersten Prototypen des Dekodierungsnetzwerkes sowie das vorherige Filternetz arbeiteten auf Ausschnitten von 20 Frames aus einer Tanzsequenz. Dies ist die Mindestlänge eines vom WDD abgespeicherten Ausschnittes. Auf den ersten und besonders den letzten Frames der Tanzsequenzen ist jedoch relativ oft keine tanzende Biene zu sehen. Auf den ersten Frames schwänzelt sie manchmal noch nicht und auf den letzten ist oft noch der Anfang der Rückkehrphase mit eingefangen. Und da die einzelnen Ausschnitte der Sequenzen die gleichen Labels bekommen, wie die Sequenz selbst, war eine der Vermutungen für den ausbleibenden Lernfortschritt der Prototypen sowie die schlechte Performance des Filternetzes, dass die Netze beim Training zu oft Ausschnitte zu sehen bekamen, die nicht ihr Label abbildeten. Also praktisch falsch gelabelte Trainingsbeispiele bekamen. Die Konsequenz dieser Überlegung war eine 128 Frame lange Eingabe. Damit sollen dem Netz einfach mehr Informationen für die Entscheidung zur Verfügung stehen – andererseits soll garantiert werden, dass selbst auf Ausschnitten von langen Sequenzen (Maximum vom 14. und 16.08.16: 422 Frames) dem Label entsprechender Inhalt zu sehen ist.

1) Klassifizierungsnetz

Beim Klassifizierungsnetz, wie in Abb.15 beschrieben, handelt es sich um ein Netz mit der gleichen Funktion wie der des Filternetzes. Es soll entscheiden, ob es sich bei dem präsentierten Ausschnitt um einen Schwänzellauf handelt oder nicht.

Abb.15: Modell des Klassifizierungsnetzes. Das Netz nimmt als Eingabe 128 Frames mit einer Auflösung von je 50 x 50 px entgegen. Aufbau: Drei 3D-Konvolutionsschichten, zwei vollverbundene Schichten. Die erste Konvolutionsschicht mit 32 5x5x5 Filtern, Stride 2 – die zweite und dritte mit 64 und 128 3x3x3 Filtern und Stride 1 – kein Zero-Padding. Nach jeder Konvolutionsschicht folgt eine Aktivierung durch SELU[29] und Pooling. Nach der ersten Konvolutionsschicht 3D-Max-Pooling mit 1x2x2 Kernel, nach der zweiten mit Kernel 2x2x2. Die dritte ist eine Global Average Pooling-Schicht, in diesem Fall mit 28x2x2 Kernel. Gefolgt von einer vollverbundenen Schicht mit 128 Neuronen, aktiviert durch die Sigmoid-Funktion. Die zweite vollverbundene ist die Ausgabeschicht und besteht aus einem Neuron. Dieses liefert durch die Sigmoid-Funktion als Ausgabe eine Fließkommazahl zwischen 0 und 1, welche als Wahrscheinlichkeit für einen Tanz interpretiert wird.

Die Eingabe für das Netz ist ein 128 Frame langer Ausschnitt einer detektierten Sequenz mit Framegröße 50 x 50px. Sprich ein Tensor mit Dimension 128 x 50 x 50. Um schnell auf kleinere Dimensionen zu kommen wird kein Zero-Padding verwendet und die erste Konvolutionsschicht wendet 5x5x5 große Filter, mit Stride 2 in allen Dimensionen, an. Die zweite und dritte Konvolutionsschicht sowie die erste und zweite Max-Pooling (vgl. Abb.A3 im Anhang) Schicht orientieren sich wieder an [22]. Die erste Max-Pooling Schicht arbeitet mit 1x2x2 Kernel, um anfangs so viel temporale Informationen wie möglich zu erhalten. Bei der dritten Pooling-Schicht kommt

Global-Average-Pooling[30] zum Einsatz. Jeder der 128 Filter, die aus der dritten Konvolutionsschicht resultieren, hat Dimension 28x2x2. Mit dem Global-Average-Pooling reduziert man die Dimension des gesamten Tensors auf 128x1x1x1. Es wird also nur ein Durchschnittswert für jeden 28x2x2 Filter gebildet.
Als state-of-the-art Aktivierungsfunktion wurde SELU[29] (vgl. Abb.A2 im Anhang) gewählt.

2) Dekodierungsnetz

Die Architektur des Dekodierungsnetzwerkes unterscheidet sich von 1) in der Eingabedimension und in der Größe der ersten vollverbundenen Schicht. Die Ausgabeschicht unterscheidet sich aufgrund der unterschiedlichen Aufgaben der Netze ebenfalls. Das Dekodierungsnetz wird (optimalerweise) ausschließlich mit Sequenzen gefüttert, auf denen Schwänzelläufe zu sehen sind und soll deren Orientierung ausgeben.

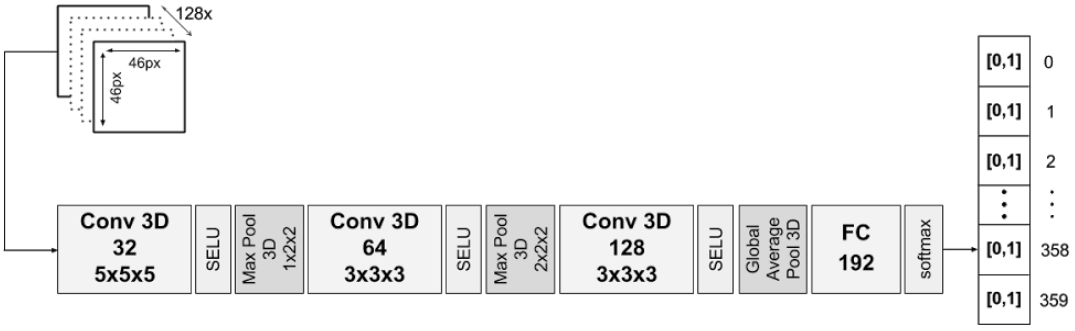

Abb.16: Modell des Dekodierungsnetzes. Das Netz nimmt als Eingabe 128 Frames mit einer Auflösung von je 46 x 46 px entgegen. Aufbau: Drei 3D-Konvolutionsschichten, zwei vollverbundene Schichten. Die erste Konvolutionsschicht mit 32 5x5x5 Filtern, Stride 2 – die zweite und dritte mit 64 und 128 3x3x3 Filtern und Stride 1 – kein Zero-Padding. Nach jeder Konvolutionsschicht folgt eine Aktivierung durch SELU[29] und Pooling. Nach der ersten Konvolutionsschicht 3D-Max-Pooling mit 1x2x2 Kernel, nach der zweiten mit Kernel 2x2x2[22]. Die dritte ist eine Global Average Pooling-Schicht, in diesem Fall mit 28x2x2 Kernel. Gefolgt von einer vollverbundenen Schicht mit 192 Neuronen, aktiviert durch die Softmax-Funktion. Die zweite vollverbundene ist die Ausgabeschicht und besteht aus 360 Neuronen. Die Neuronen 0 bis 359 geben je die Wahrscheinlichkeit für einen Tanz mit der Orientierung 0° bis 359° aus.

Die Dimension für die Eingabe ist im Gegensatz zum Klassifizierungsnetzwerk ein 128 Frame langer Ausschnitt mit einer Framegröße von 46 x 46px. Dies bringt mehr Flexibilität beim Training. Die erste vollverbundene Schicht hat hier 192 Neuronen. Die Ausgabeschicht besteht aus 360 Neuronen, welche jeweils die Wahrscheinlichkeit für die Orientierung des Schwänzellaufes entsprechend ihres Indexes, beginnend bei 0 (vgl. Abb.16), ausgeben. Mithilfe der Softmax-Funktion wird also die Ausgabe eines Vektors der Länge 360 erreicht, welcher die Wahrscheinlichkeitsverteilung für die 360 verschiedenen Klassen/Winkel repräsentiert.

3.5 Training

Das Training erfolgte bei allen Netzen mit einer Batchgröße[7] von 32 und dem Optimierer[8] Adam mit einer initialen Lernrate von 0.0005. Wie in 3.4.2 beschrieben, nehmen die Netze eine Folge von Frames der Länge 128 als Eingabe entgegen. Falls eine Sequenz aus weniger als 128 Frames bestand (Minimum pro Detektion am 16.08.16 sind 54 Frames), so wurde der Rest des Eingabetensors mit Nullen aufgefüllt. Die fehlenden also mit schwarzen Frames substituiert. Um die Rate an nicht für die Klassifikation relevanten Frames (vgl. 3.4.2, die ersten und letzten Frames) zu reduzieren, wurde ein Parameter *'trim'* eingeführt. Dieser bestimmt, wie viele Frames von einer Sequenz vorne wie hinten abgeschnitten werden, bevor sie weiter verarbeitet wird. Was die Vorverarbeitung der Daten betrifft wurden alle Bilder des Datensatzes Null zentriert.

3.5.1 Klassifizierungsnetz

Der Datensatz für das Training dieses Netzes waren die WDD-Detektionen vom 16.08.16 – insgesamt 16.802 (6103 Tänze / 10.699 Nicht-Tänze). Die Labels für diese Sequenzen wurden dabei nicht gänzlich manuell erstellt. Für diese Arbeit wurden 4000 Detektionen vom 16.08.16 manuell gelabelt (Tanz oder Nicht-Tanz). Dieser Satz an Ground Truth Daten diente unter anderem für das Training und die Validierung des neuen Filternetzes. Die restlichen 12.802 Detektionen wurden automatisch durch selbiges gelabelt.
War eine Sequenz länger als 128 Frames, so wurden die mittleren 128 Frames für die Eingabe genommen. *trim* war auf 15 gesetzt. Um Überanpassung entgegenzuwirken wurde Dropout[9] mit einer Wahrscheinlichkeit von 25%, also mit 25% Wahrscheinlichkeit inaktive Neurone (in Keras gegensätzlich zu [31]), vor der ersten vollverbundenen Schicht angewandt. Zusätzlich wurden alle Trainingssequenzen zufällig vertikal, horizontal oder vertikal und horizontal gespiegelt. Die Loss-Funktion war entsprechend des einzelnen Ausgabeneurons und der binären Klassifizierung binary crossentropy.

---

[7] Anzahl der Samples pro Trainingsschritt, nach deren Auswertung die Gewichte angepasst werden
[8] Algorithmus, der für die Gewichtsaktualisierungen im Trainingsprozess zuständig ist. Er entscheidet, nachdem die Fehlergradienten berechnet wurden, wie genau diese Gradienten für die Gewichtsanpassungen verrechnet werden.
[9] eine Methode, welche Überanpassung entgegenwirken soll. Dabei werden im Training alle Neurone einer Schicht mit der angegebenen Wahrscheinlichkeit aktiv gehalten, ansonsten deaktiviert. D.h. nur ein entsprechender Prozentsatz an Neuronen kann für den forward pass genutzt werden, was das Netz dazu zwingt mit den verbleibenden zurechtzukommen. Im Prinzip werden viele kleinere Subnetze trainiert, welche bei der Inferenz quasi als Ensemble genutzt werden. [31]

3.5.2 Dekodierungsnetz

Die Trainingsdaten für das Dekodierungsnetzwerk waren ebenfalls die Detektionen des WDD vom 16.08.16. Die Labels zu den Sequenzen, also der jeweilige Tanzwinkel, wurden dabei komplett aus einer automatischen Dekodierung aller Tänze durch die noch aktuelle Methode aus [13] erstellt. Für das Training wurden dann ausschließlich durch das Filternetz als Tanz klassifizierte Sequenzen genutzt. Ein Test-Split von 20% wurde zurückgelegt. Dementsprechend blieben für das Training der Winkeldekodierung noch 4883 von 6103 Tanz-Samples übrig. Für die entsprechenden Targets (Zielausgaben) für die Sequenzen wurde der jeweilige Tanzwinkel dazu erst auf ganze Grad gerundet und one-hot-kodiert und dann mit einem Gauß-Filter ($\sigma = 1,6$) weichgezeichnet, sodass sich die Wahrscheinlichkeitsverteilung über insgesamt 13 Winkel erstreckt. Dabei lag die Wahrscheinlichkeit für den korrekten Winkel bei rund 0,25 und für die nächsten vier angrenzenden Winkel in beide Richtungen bei jeweils rund 0,21, 0,11, 0,04 und 0,01. Die zirkuläre Eigenschaft des Winkels wird hierbei berücksichtigt, sodass sich die Verteilung auch bspw. über die Winkel 355° bis 7° erstrecken kann. Diese Weichzeichnung soll dem Netz das Lernen erleichtern. Das Target korrespondiert so besser zur Ausgabe der Softmax-Funktion und kleinere Abweichungen vom Zielwinkel werden in Verbindung mit dem categorical crossentropy loss weniger hart bestraft. Um Überanpassung zu vermeiden wurde im Training, neben gewöhnlichem Dropout vor der ersten vollverbundenen Schicht mit Wahrscheinlichkeit 0.3, auch auf 3D erweitertes Dropout (Spatial Dropout 3D) vor allen Konvolutionsschichten angewendet. Mit 10% Wahrscheinlichkeit werden hier Neurone konsistent in der dritten Dimension deaktiviert. Bildlich gesprochen bleiben hier also genau die gleichen Pixel im Bild über die gesamte Zeitachse, in jedem Frame, deaktiviert. Außerdem wurden von allen Sequenzen zufällige 46 x 46 px Bildausschnitte gewählt. Die zugeschnittenen Sequenzen wurden zusätzlich auf Frameebene noch jeweils zufällig vertikal gespiegelt, horizontal gespiegelt und um den Bildmittelpunkt rotiert. Bei letzterem wurde der Modus 'reflect' (vgl. Abb.A4 im Anhang) gewählt, bei dem die entstehenden Kanten bis zum Bildrand gespiegelt werden, um so wenig harte Kanten wie möglich entstehen zu lassen. Dies könnte den Fokus der Filter ablenken. Das Rotieren der Sequenzen wurde jedoch nicht ausschließlich als Gegenmaßnahme zur Überanpassung gewählt. Es sollte außerdem die unausgeglichene Verteilung der Tanzwinkel im Trainingsdatensatz (vgl. Abb.A5 im Anhang) ein wenig ausgleichen. Weniger oft repräsentierte Winkel könnten viel schwerer gelernt werden. Der Parameter *trim* war auf 10 gesetzt.

4. Evaluierung

In diesem Kapitel werden die Ergebnisse vorgestellt, welche die beiden trainierten Netzwerke auf den Testdaten erreichen und mit den Anforderungen abgeglichen.

4.1 Klassifizierungsnetz

Das Training eines reinen Klassifizierungsnetzes diente im Verlauf dieser Arbeit nur als Sanity Check, um ausschließen zu können, dass mit der generellen Architekur – das Dekodierungsnetz konnte nicht richtig lernen – etwas nicht stimmt. Die gleiche Architektur des zu der Zeit aktuellen Versuchsnetzes wurde dazu leicht angepasst und lediglich 50 Epochen auf die Filteraufgabe trainiert. Das trainierte Netzwerk wurde nicht abgespeichert, womit nun dessen Performance nicht genauer untersucht werden konnte und hier lediglich die unmittelbar ausgegebenen Ergebnisse zur Verfügung stehen. Diese sind in folgender Tabelle zusammengefasst.

Tab.1: Performance des trainierten Klassifizierungsnetzes auf dem Testdatensatz

	Anzahl der vorne und hinten abgeschnittenen Frames pro Sample				
	1	5	10	15	20
Accuracy	93.19%	94.50%	95.27%	95.86%	94.38%
Precision	84.18%	86.93%	88.62%	90.20%	93.00%
Recall	99.58%	99.50%	99.50%	99.16%	91.06%

20% des verwendeten Datensatzes, 3361 Samples (1220 davon Tänze), wurden für Testzwecke zurückgehalten. Die Samples dieses Testdatensatzes hat das Netzwerk beim Training nicht zu sehen bekommen. Die Ausgabe des Netzes ist, wie in 3.4.2 beschrieben ein Wert zwischen 0 und 1. Als Schwellwert für die Klassifizierung wurde 0,5 genutzt. Die Ausgabe des Netzwerkes wurde also einfach gerundet. War dieses Ergebnis eine 1, so handelte es sich lt. Netzwerk um einen Tanz – bei einer 0 um keinen Tanz. Bei Sequenzen mit mehr als 128 Frames, nach Abschneiden der *trim* Frames, wurden mehrere Subsequenzen – alle, die mit einer vorher festgelegten Schrittweite (stride) möglich waren – in das Netz gegeben und alle Ausgaben gemittelt. War eine Sequenz kürzer als 128 Frames, so wurde sie, wie im Training, mit Nullen aufgefüllt. Neben verschiedenen Werten für *trim* (1, 5, 10, 15, 20) wurden verschiedene strides ausprobiert – 1, 5, 10 und 20. Doch letztere hatten keinen Einfluss auf die Ergebnisse.
Maßgeblich für die in Tabelle 1 zusammengefassten unterschiedlichen Ergebnisse ist der *trim*-Wert. Hervorzuheben ist hier, dass die höchste Rate an Korrektklassifikationen, die accuracy, mit einem *trim*-Wert von 15 erreicht wird – der selbe wie im Training. Allgemein sind die Ergebniswerte sehr nahe an den geforderten dran.
Trotzdem sind sie nicht ernsthaft miteinander vergleichbar, da dieses Netzwerk hier mit den Ergebnissen des Netzwerkes trainiert wurde, mit dem es verglichen werden soll. Man kann nicht sagen, ob oder wie gut das hier trainierte Netz auf Videodaten von anderen Aufnahmetagen performt. Evtl. könnte es nicht gut generalisieren, so wie das Filternetz anfänglich. Das Klassifizierungsnetz scheint aber das Filternetz auch nicht

perfekt simulieren zu können. Dafür wären 100% bei allen drei Metriken nötig. Die dazu noch fehlende Performance könnte mit den relativ geringen Trainingsepochen zusammenhängen. Oder aber damit, dass rund ein Viertel der verwendeten Labels manuell erstellt worden sind, bei denen das Filternetz zu einem Teil anders entscheiden würde. Über den oben genannten Schwellenwert – quasi über eine Verschiebung der Auf- und Abrundungsgrenze – könnte man ebenfalls noch weiter optimieren.

4.2 Dekodierungsnetz

Der Datensatz zur Evaluierung des Dekodierungsnetzes wurde bereits in 3.1 beschrieben. Die Vorverarbeitung der Sequenzen für die Inferenz gleicht der beim Training, mit dem einzigen Unterschied, dass der 46 x 46 px Ausschnitt nicht zufällig gewählt, sondern stets mittig genommen wird. Die Vorgehensweise bei Sequenzen kürzer und länger als 128 Frames ist die gleiche wie in 4.1. Bei längeren Sequenzen wurden ebenfalls mehrere ausgegebene Winkel zu eingegebenen Subsequenzen (gewählt mit strides 2, 5 und 10) gemittelt, wobei selbstverständlich die zirkuläre Eigenschaft des Winkels berücksichtigt wurde. 359° und 1° werden also beispielsweise zu 0° gemittelt.
Die Gütemaße für eine Mehrklassen Klassifikation sind andere als die für eine binäre Klassifikation. Die hier angewandte Mehrklassen Klassifikation unterscheidet sich jedoch auch von den typischen (z.B. Klassifikation in Katze oder Hund oder Maus oder Schaf...), in dem Sinne, dass die Klassen hier nicht komplett disjunkt sind, sondern in einem engen Bezug zueinander stehen. Das Endergebnis entspricht eher dem einer Regression. Für das Dekodierungsnetzwerk schauen wir uns die Verteilung der Abstände, in die eine oder andere Richtung, der vorhergesagten zu den tatsächlichen Winkeln an. Wenn der vorhergesagte also nicht exakt dem tatsächlichen entspricht, so liegt sein Abstand entweder in (0, 180] oder (–180, 0). Ansonsten ist er Null. Als Maß für die Genauigkeit der Dekodierung im Durchschnitt nutzen wir μ – den Mittelwert der Abweichungen. Als Maß für den ungefähr zu erwartenden Fehler nutzen wir σ, die Standardabweichung der Winkel vom Mittelwert μ. Bei der Suche nach dem besten Ergebnis wurde mehr Wert auf eine geringere Standardabweichung als auf einen besseren Mittelwert gelegt, da der Mittelwert nicht unbedingt viel darüber aussagt, wie groß der Anteil an gut dekodierten Läufen ist. Erst zusammen mit einer geringeren Standardabweichung wird der Mittelwert interessanter.
Das beste Ergebnis, mit $\mu = 0,33°$ und $\sigma = 35,71°$, wurde mit stride 2 und *trim* 10 erzielt.

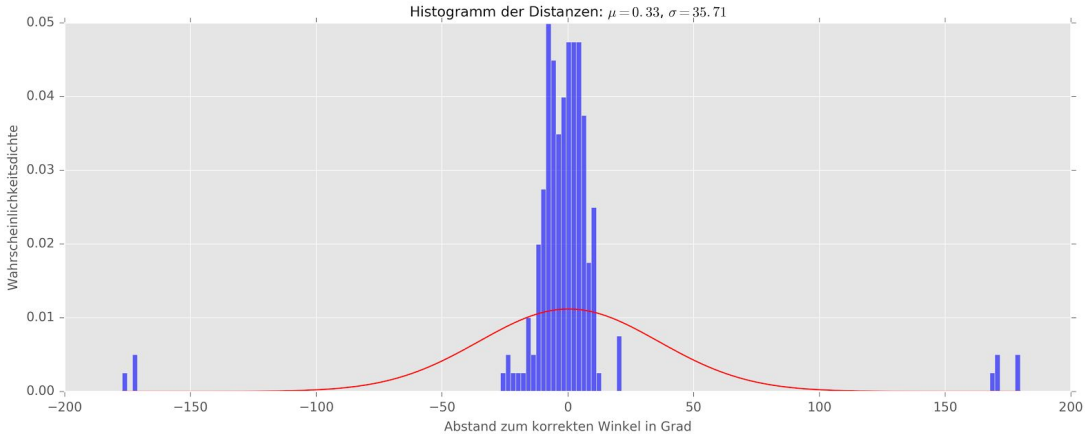

Abb.17: Histogramm der Winkelabstände zum tatsächlichen Winkel nach Inferenz der Evaluierungsdaten durch Dekodierungsnetz. In rot darüber gelegt: Gaußverteilung mit entsprechendem μ und σ. Größe der Bins: 2°

Auf Abbildung 17 ist jedoch zu sehen, dass für die vergleichsweise schlechte Standardabweichung von rund 35° nur ein paar einzelne Extremfälle verantwortlich sind. Es gab 8 Schwänzelläufe, bei denen das Netz mit der Haupttanzachse richtig, jedoch bei der korrekten Richtung falsch lag. Ähnlich der Fehler bei der aktuellen Dekodierungsmethode. Das entspricht bei 200 Testsamples einem Anteil von 4%. Wenn man diese 180°-Fehler außer Betracht lässt liegt die maximale Abweichung bei 26,5° in die negative Richtung. Wenn man diese 8 Ausreißer korrigieren – ihren Wert also um 180° drehen – würde, ergäbe sich eine bessere Verteilung (vgl. Abb.18) mit einer wesentlich besseren Standardabweichung von 7,85° und einem immernoch sehr guten Mittelwert von -1,47°.

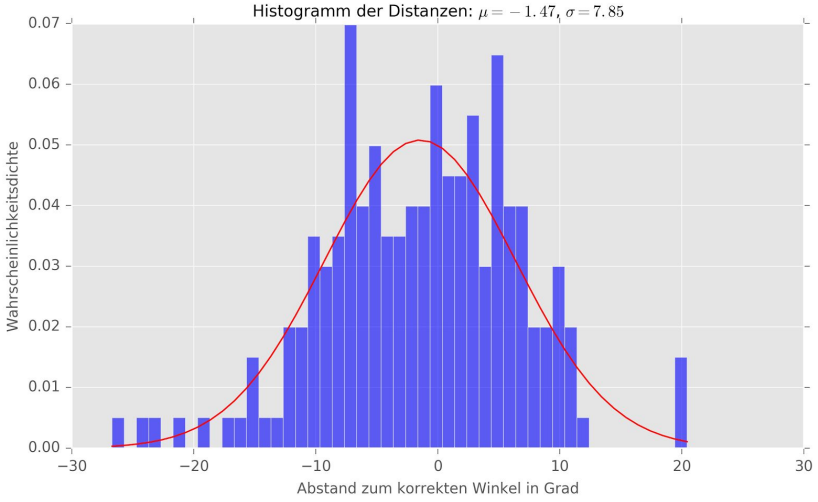

Abb.18: Histogramm der Winkelabstände zum tatsächlichen Winkel nach Inferenz der Evaluierungsdaten durch Dekodierungsnetz und Korrektur der 180°-Fehler. In rot darüber gelegt: Gaußverteilung mit entsprechendem μ und σ. Größe der Bins: 1°

Der Betrag dieser (korrigierten) Standardabweichung ist sehr nahe an dem der aktuellen Methode dran. Doch erreicht ihn nicht ganz. Was daran liegen könnte, dass das Netz maximal so gut werden kann, wie die Methode, von dessen Ergebnissen es lernt. Man könnte das Ergebnis so interpretieren, dass das Dekodierungsnetzwerk die aktuelle Methode schon fast perfekt, bis auf eine paar eklatante Fehler, simuliert.
Dass jedoch bessere Ergebnisse drin sind zeigt ein Blick auf die Ergebnisse mit dem Test-Split des Datensatzes, siehe Abbildung A8 im Anhang. Auf diesem Teildatensatz erreicht das identische Netzwerk nach Korrektur der 180°-Fehler (3,8%) eine geringere Standardabweichung von 4,77°, bei einem Mittelwert von -0,12°. Ohne Korrektur liegt die Standardabweichung jedoch ebenfalls bei über 30° (vgl. Abb.A7 im Anhang). Dies könnte darauf hindeuten, dass das Netzwerk nicht so gut auf Videosequenzen von anderen Aufnahmetagen generalisieren kann. Ähnlich dem anfänglichen Problem mit dem Filternetz. Eine andere Erklärung wäre wieder, wie bereits im letzten Absatz ausgeführt, dass das Netz die aktuelle Methode fast perfekt simuliert. Also einen gewissen Fehler zu machen mitgelernt hat, da es von latent fehlerhaften Daten (μ = -3,29°, σ = 7,17°) gelernt hat. Was jedoch nur auffällt, wenn man es gegen Ground Truth Daten testet, die diese Fehler nicht enthalten.

5. Diskussion

Im Rahmen dieser Arbeit wurde ein künstliches neuronales Netzwerk trainiert, welches die Winkeldekodierung von Schwänzeltänzen auf Bildsequenzen bewerkstelligt. Jedoch tut es dies nicht mit der nötigen Qualität, um die aktuelle Methode[13] ablösen zu können. Insbesondere sind die Ausgaben des trainierten Netzwerkes ebenfalls 180°-Fehler-behaftet. Auf dem kleinen Testdatensatz, sowie auf dem Test-Split ungefähr der gleiche Anteil.
Die Güte eines zweiten Netzwerkes, welches auf die Klassifizierung von Bildsequenzen in 'Tanz' oder 'Nicht-Tanz' trainiert wurde, lässt sich nicht genau beurteilen, da es nicht genauer evaluiert wurde. Insbesondere nicht mit den gleichen Testdaten, wie das aktuelle Filternetz. Die Performance auf WDD-Sequenzen zumindest eines anderen Tages müsste noch getestet werden, um das Generalisierungsvermögen des Netzes einschätzen zu können.
Nebenbei wurden 8239 neue Ground Truth Daten generiert, was für die Überprüfung der Qualität schon bestehender Ground Truth Daten nötig war und das Training des neuen Filternetzes mit ermöglichte – dem Lab jedoch auch bei zukünftigen Arbeiten auf dem Gebiet der WDD-Pipeline hilfreich sein könnte.

Das gesteckte Ziel ein neuronales Netzwerk für beide Aufgaben – Klassifikation und Winkeldekodierung – entwickelt zu haben wurde im Rahmen dieser Arbeit nicht erreicht. Zwei verschiedene Konzepte (vgl. 3.4.2) für die Umsetzung eines solchen Netzes existieren. Und einer davon – der Klassifikationsansatz – scheint auch zu funktionieren. Jedoch entstanden die in 4.2 präsentierten Ergebnisse erst kurz vor Ende dieser Arbeit, sodass keine Zeit mehr blieb die Ergebnisse zu optimieren und die beiden entwickelten Netze zu vereinen. Aus diesem Grund wurden auch noch keine

Anstrengungen dafür aufgewendet das Netz in die aktuelle Software Pipeline zu integrieren.
Das zeitaufwändigste war das manuelle Überprüfen der bestehenden Ground Truth Daten, sowie der automatisch generierten Trainingsdaten. Dazu kam noch die Abhängigkeit vom Filternetz, welches einen Teil der Trainingsdaten lieferte, jedoch erst neu trainiert werden musste.

6. Ausblick

Das Ziel dieser Arbeit konnte in ihrem engen zeitlichen Rahmen nicht erreicht werden. In zukünftigen Arbeiten könnte man nun an einigen Stellen ansetzen, um die Performance des Dekodierungsnetzes zu verbessern.
Grundsätzlich gilt es herauszufinden, warum das Netz ab einem gewissen Punkt nicht mehr besser wird und diese Ursache zu beseitigen.

Mögliche Ursachen für die Diskrepanz in der Performance zur aktuellen Dekodierungsmethode könnten sein:

a) Das Netzwerk generalisiert nicht gut auf Videodaten eines anderen Aufnahmetages.
   Dafür spricht, dass das Netz auf Daten, die dem selben Aufnahmetag entsprechen, wie die Trainingsdaten, eine wesentlich geringere Standardabweichung aufweist.
b) Das Netzwerk hat den Fehler, den die aktuelle Methode macht (Trainingsdaten stammen von dieser Methode) mitgelernt – inklusive der 180°-Fehler.
   Dies würde auf dem Test-Split des Trainingsdatensatzes nicht auffallen, jedoch wenn man es gegen Ground Truth Daten testet, die diese Fehler nicht enthalten.
   Eine manuelle Sichtung von 355 im Bezug auf die Tageszeit gleichverteilten Samples des Trainingsdatensatzes vom 16.08.16 ergab, dass etwa 15% der automatischen Dekodierungen den 180°-Fehler aufweisen.

Folgende Maßnahmen könnte man nun ergreifen:

1) <u>Gemischter Datensatz</u>
   Da die Überprüfung von a) am wenigsten Aufwand bedeutet, sollte man diese zuerst angehen.
   Ähnlich zum Training des neuen Filternetzes sollte das Dekodierungsnetz mit einem gemischten Trainingsdatensatz neu trainiert werden. D.h. mit Daten vom 14. (Quelle des Evaluierungsdatensatzes) und vom 16. August 2016. Wobei man die Samples, die auch im Evaluierungsdatensatzes enthalten sind vom Training ausschließen muss. Dies müsste die Generalisierungsfähigkeiten des Netzes verbessern.

2) <u>Neue Trainingsdaten</u>

Falls 1) keine (nur eine leichte) Verbesserung bringt, muss (zusätzlich auch) b) zutreffen. Denn es müsste jetzt auch die Eigenheiten des 14.er Datensatzes gelernt haben und dürfte keine Schwierigkeit mehr mit diesen im Evaluierungsdatensatz haben.
Es gibt nun 2 weitere Maßnahmen, die zum Erfolg führen könnten. Innerhalb des Mapping Moduls werden die geclusterten Läufe mithilfe des RANSAC-Algorithmus von Ausreißern bereinigt – speziell von 180°-Fehlern. Nun sollte man zuerst:

I. Alle Trainingssamples durch die RANSAC-Filterung laufen lassen und nur die übrig gebliebenen zum Training verwenden.

Nun sollte das Netz ohne Trainingsbeispiele mit 180°-Fehler trainiert werden. Da die 'RANSAC-Methode' ebenfalls erst zum Ende dieser Arbeit fertig wurde, konnte dies noch nicht ausprobiert werden.
Nun könnte, in Verbindung mit einer eventuellen Verbesserung durch einen gemischten Datensatz, die Performance des Netzes schon ausreichend gut sein. Denn falls nun keine 180°-Fehler mehr auftreten müssten wir theoretisch bei rund 7,85° Standardabweichung landen. Falls zusätzlich eine Verbesserung durch den gemischten Datensatz auftritt, könnte dies entscheidend sein um unter die 7,17° Standardabweichung von der aktuellen Dekodierungsmethode oder gar unter die 6,66° beim Menschen zu kommen.
Falls der Nachteil der nun viel geringeren Anzahl an Trainingssamples diese Verbesserungen nicht auffrisst.
Wenn dies der Fall sein sollte gibt es noch eine, wenn auch zeitaufwendige, Methode. Durch die RANSAC-Filterung könnte vermieden werden, dass das Netz den 180°-Fehler lernt. Jedoch nicht, dass es die allgemeine Ungenauigkeit lernt. Um dieses Problem zu lösen führt kein Weg daran vorbei neue Ground Truth Daten manuell zu erstellen. Man müsste nun also:

II. Einen Datensatz von geschätzt 1000 Samples des 14. und 16. August 2016 manuell in Bezug auf den Tanzwinkel labeln.

Diese Größe könnte ausreichend sein um das Netz nach einem Vortraining mit dem RANSAC-gefilterten Datensatz feinzutunen. Optimalerweise würden die 1000 Samples von jeweils mindestens 3 Menschen gelabelt werden, um auf eine ähnliche Genauigkeit, wie beim Evaluierungsdatensatz zu kommen. 1000 Samples zu labeln dauert ungefähr 4 Stunden. Zusätzlich müsste man auch Winkelaugmentierung, also das Rotieren der Trainingssamples, betreiben um möglichst alle Winkel abzudecken. Da bei 1000 Samples die Wahrscheinlichkeit gering ist, dass alle 360 Winkel mindestens einmal vorkommen.

Falls alle vorgeschlagenen Maßnahmen nicht helfen, müsste doch über eine gänzlich neue Architektur nachgedacht werden.

Literaturverzeichnis

[1] Karl von Frisch, 1965, *Tanzsprache und Orientierung der Bienen*, Springer-Verlag Berlin Heidelberg

[2] Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit, http://www.umwelt-im-unterricht.de/hintergrund/die-biene-eines-der-wichtigsten-nutztiere/

[3] Feldman, Balch, 2016, *Representing Honey Bee Behavior for Recognition Using Human Trainable Models*, Adaptive Behavior Vol 12, Issue 3-4, pp. 241 - 250 http://dx.doi.org/10.1177%2F105971230401200309

[4] A. Veeraraghavan, R. Chellappa and M. Srinivasan, *"Shape-and-Behavior-Encoded Tracking of Bee Dances,"* IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 30, no. 3, pp. 463-476, Mar. 2008.

[5] Toshifumi Kimura, Mizue Ohashi, Ryuichi Okada, und Hidetoshi Ikeno, 2011, *A new approach for the simultaneous tracking of multiple honeybees for analysis of hive behavior*, Apidologie, 42(5):607{617, 2011. ISSN 0044-8435, doi:10.1007/s13592-011-0060-6, http://dx.doi.org/10.1007/s13592-011-0060-6

[6] Landgraf T, Rojas R, Nguyen H, Kriegel F, Stettin K (2011) *Analysis of the Waggle Dance Motion of Honeybees for the Design of a Biomimetic Honeybee Robot*. PLoS ONE 6(8): e21354. https://doi.org/10.1371/journal.pone.0021354

[7] T. Landgraf, M. Oertel, D. Rhiel and R. Rojas, *"A biomimetic honeybee robot for the analysis of the honeybee dance communication system,"* 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, 2010, pp. 3097-3102. Doi: 10.1109/IROS.2010.5650930, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5650930&isnumber=5648787

[8] Biorobotics Lab am Dahlem Center for Machine Learning and Robotics, http://berlinbiorobotics.blog/

[9] AutoNOMOS Labs am Dahlem Center for Machine Learning and Robotics, autonomos-labs.com

[10] Wario F., Wild B., Couvillon MJ, Rojas R. and Landgraf T. (2015) *Automatic methods for long-term tracking and the detection and decoding of communication dances in honeybees*. *Front. Ecol. Evol*. 3:103. doi: 10.3389/fevo.2015.00103, https://www.mi.fu-berlin.de/inf/groups/ag-ki/publications/long-term-tracking/fevo-03-00103.pdf

[11] A. Rau, 2014, *Realtime Honey Bee Waggle Dance Decoding System*, https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/Betreute_Arbeiten/Master-Rau.pdf

[12] M. Halilovic, 2015, *Erkennung von Schwänzeltänzen mit tiefen neuronalen Netzen*, https://www.mi.fu-berlin.de/inf/groups/ag-ki/Theses/Completed-theses/Bachelor-theses/2016/Halilovic1/Bachelor-Halilovic.pdf

[13] S. Witte, 2017, *Automatische Dekodierung von Bienentänzen aus Bilddaten*

[14] W. Backhaus, *Color vision and color choice behavior of the honey bee*, Apidologie 24 (3) 309-331 (1993), DOI: https://doi.org/10.1051/apido:19930310

[15] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio: *Object Recognition with Gradient-Based Learning*, in Forsyth, D. (Eds), Feature Grouping, Springer, 1999, http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf

[16] http://cs231n.github.io/neural-networks-1/

[17] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation. MIT Press, 1997, http://www.bioinf.jku.at/publications/older/2604.pdf

[18] K. Cho et al., 2014, *On the properties of neural machine translation: Encoder-decoder approaches*. arXiv preprint arXiv:1409.1259, https://arxiv.org/abs/1409.1259

[19] S. Ji et al., 2013, *3D Convolutional Neural Networks for Human Action Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 35, Issue: 1, Jan. 2013, P.221 - 231)

[20] K. Simonyan and A. Zisserman, 2014, *Two-Stream Convolutional Networks for Action Recognition in Vi*deos, CoRR (abs/1406.2199), http://arxiv.org/abs/1406.2199

[21] C. Feichtenhofer, A. Pinz, A. Zisserman, 2016, *Convolutional Two-Stream Network Fusion for Video Action Recognition*, CoRR (abs/1604.06573), http://arxiv.org/abs/1604.06573

[22] Du Tran et al., 2014, *Learning Spatiotemporal Features with 3D Convolutional Networks*, CoRR (abs/1412.0767), http://arxiv.org/abs/1412.0767

[23] UCF101 - Action Recognition Data Set, http://crcv.ucf.edu/data/UCF101.php

[24] Moez Baccouche et al., 2011, *Sequential Deep Learning for Human Action Recognition*, Human Behavior Understanding: Second International Workshop, HBU 2011, Amsterdam, The Netherlands, November 16, 2011. Proceedings, P.29-39, Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-25446-8_4

[25] Jeff Donahue et al., *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*, CoRR (abs/1411.4389), http://arxiv.org/abs/1411.4389

[26] Alberto Montes et al.,2016, *Temporal Activity Detection in Untrimmed Videos with Recurrent Neural Networks*

[27] Heilbron, Escorcia and Niebles, 2015, *Activitynet: A largescale video benchmark for human activity understanding*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 961–970, https://github.com/imatge-upc/activitynet-2016-cvprw

[28] Keras: The Python Deep Learning library, https://keras.io/

[29] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, 2017, *Self-Normalizing Neural Networks*, CoRR (abs/1706.02515), http://arxiv.org/abs/1706.02515

[30] Min Lin, Qiang Chen, Shuicheng Yan, 2013, *Network In Network*, CoRR (abs/1312.4400), http://arxiv.org/abs/1312.4400

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, 2014, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, JMLR Vol.15(Jun): P.1929−1958, http://jmlr.org/papers/v15/srivastava14a.html

[32] R. Rojas, 1996, Neural Networks – A Systematic Introduction, Springer-Verlag Berlin Heidelberg

Anhang

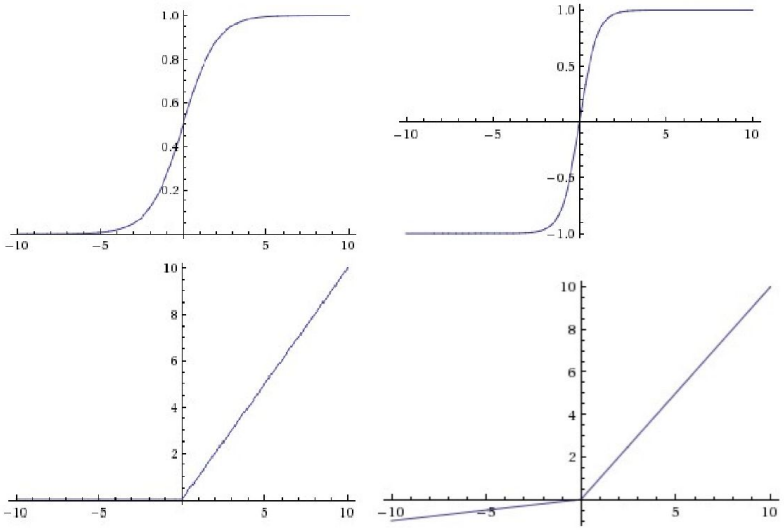

Abb.A1: Aktivierungsfunktionen.

Oben links: Sigmoid-Funktion, $f(x) = 1 / (1 + e^{-x})$, bildet auf reelle Zahl im Intervall [0,1] ab

Oben rechts: Tangens hyperbolicus, $f(x) = 1 - \frac{2}{e^{2x}+1}$, bildet auf reelle Zahl im Intervall [-1,1] ab

Unten links: ReLu (rectified linear unit), $f(x) = max(0, x)$, linear falls $x \geq 0$, sonst 0

Unten rechts: Leaky ReLu, $f(x) = max(\alpha x, x)$ für $\alpha \leq 1$, linear falls $x \geq 0$, sonst kleiner Anteil von x

Quellen oben und unten links: http://cs231n.github.io/neural-networks-1/;

Quelle unten rechts: https://x.ai/aspects-of-deep-learning-activation-functions/

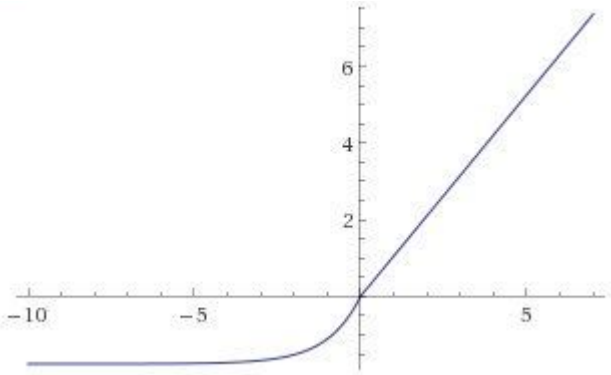

Abb.A2: SELU (scaled exponential linear unit)

Quelle: https://pbs.twimg.com/media/DB9F3hOW0AAw48o.jpg

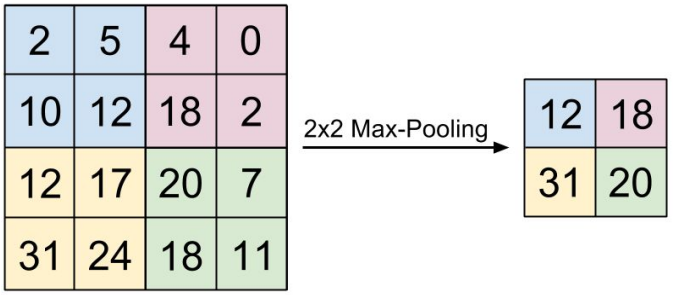

Abb.A3: Beispiel für Max-Pooling. Eine Subsampling-Methode, die nur die herausstechendsten Merkmale im Bild bewahrt. Hier mit Filtergröße 2x2. Die zu verarbeitenden Dimensionen der Eingabe werden reduziert und gleichzeitig wichtige Informationen erhalten.

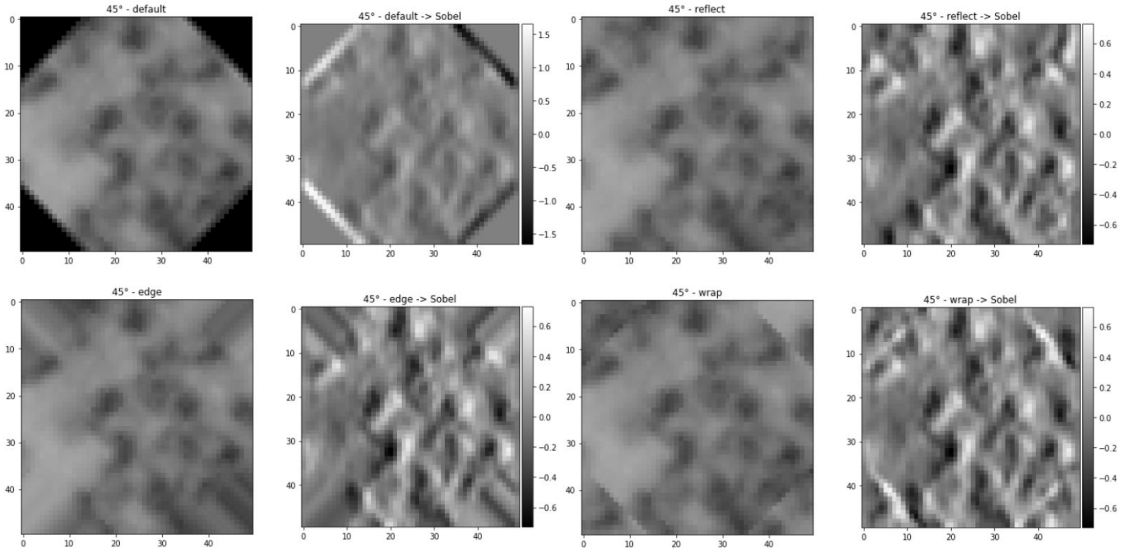

Abb.A4: 4 verschiedene Kantenbehandlungsmodi bei der Bildrotation und ihre Sobel-Ableitung. Im Modus 'reflect' entstehen die wenigsten harten Kanten. Das Bild sieht harmonischer aus.

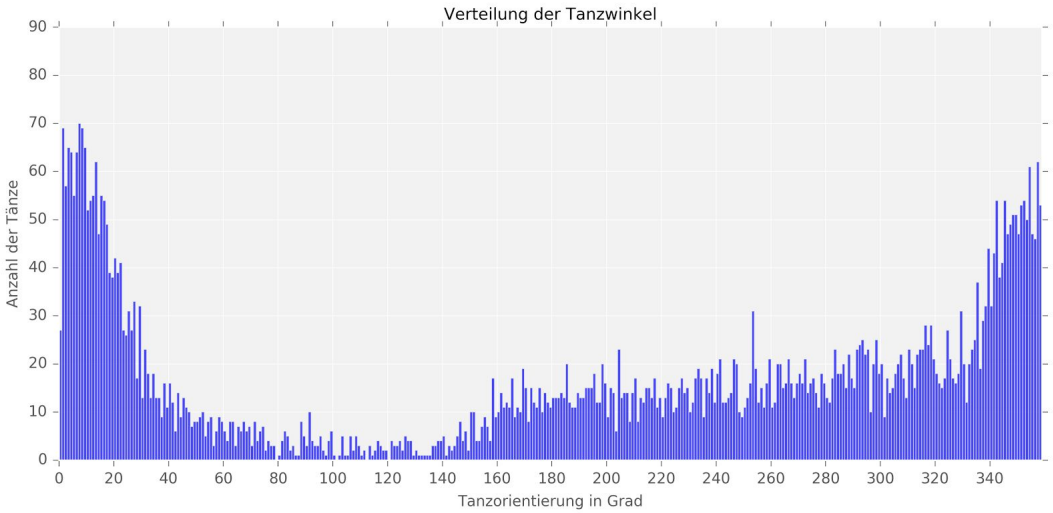

Abb.A5: Verteilung der Tanzwinkel im Trainingsdatensatz vom 16.08.16. Starker Peak bei etwa 5° ± 10°. Viele Winkel zwischen 75° und 145° sind stark unterpräsentiert.

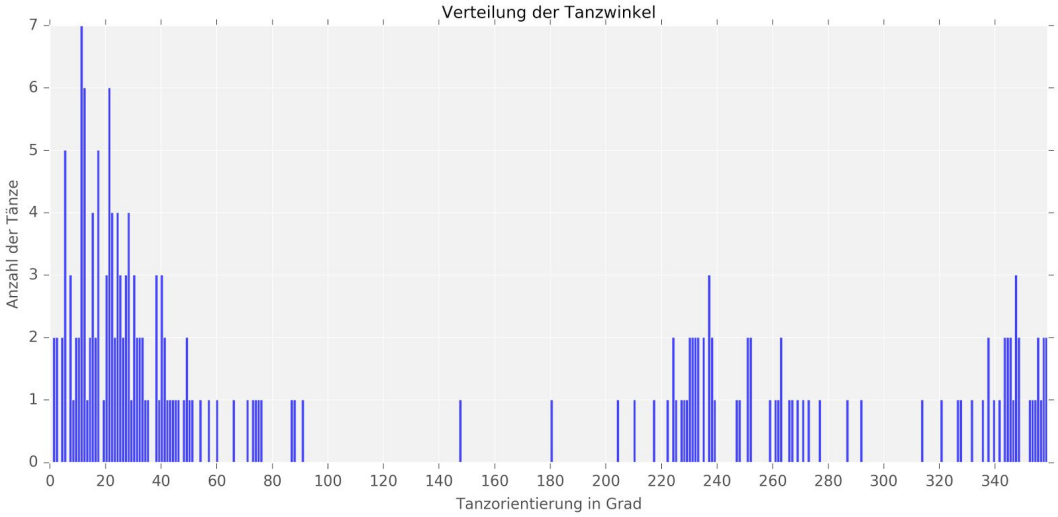

Abb.A6: Verteilung der Tanzwinkel im Testdatensatz vom 14.08.16. Starker Peak bei etwa 10°. Viele Winkel zwischen 90° und 200° und um die 300° herum sind überhaupt nicht vertreten.

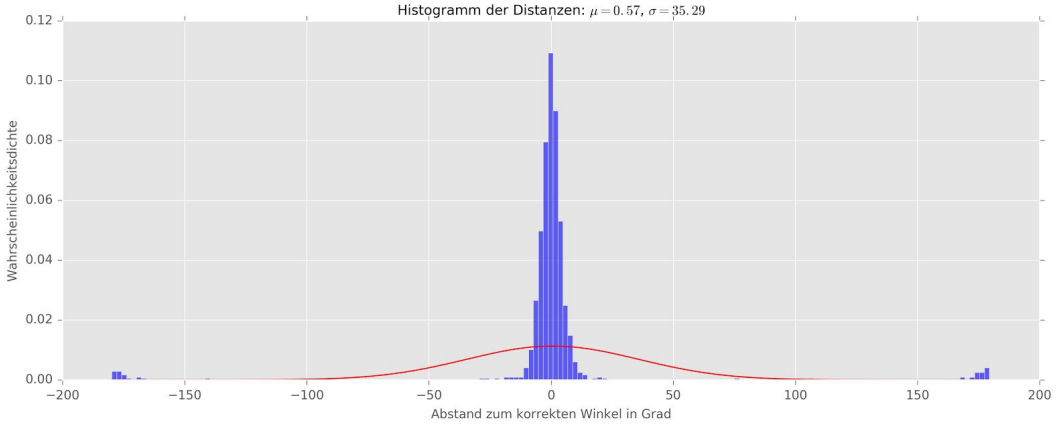

Abb.A7: Histogramm der Winkelabstände zum tatsächlichen Winkel nach Inferenz der Testdaten durch Dekodierungsnetz. In rot darüber gelegt: Gaußverteilung mit entsprechendem μ und σ. Größe der Bins: 2°

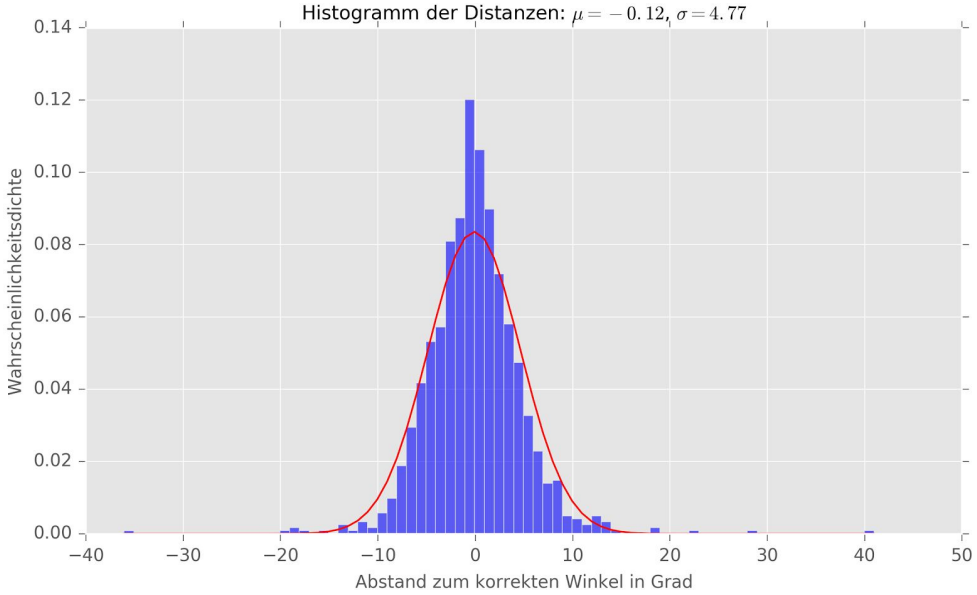

Abb.A8: Histogramm der Winkelabstände zum tatsächlichen Winkel nach Inferenz der Testdaten durch Dekodierungsnetz und Korrektur der 180°-Fehler. In rot darüber gelegt: Gaußverteilung mit entsprechendem μ und σ. Größe der Bins: 1°

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Ich erkläre weiterhin, dass die vorliegende Arbeit, in gleicher oder ähnlicher Form, noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Berlin, 18.08.17	Adrian Defèr
Ort, Datum	Unterschrift