Automatische Klassifikation von Vogelstimmen durch instanzbasiertes Lernen

Diplomarbeit

Ausgeführt zum Zweck der Erlangung des akademischen Grades
Dipl.-Ing. für technisch-wissenschaftliche Berufe

am Masterstudiengang Digitale Medientechnologien an der
Fachhochschule St. Pölten, Spezialisierung Audio Design

von:
Roman Schmid, BSc
DM121546

Betreuer/in und Erstbegutachter/in: Dipl.-Ing. Mag. Dr. Matthias Zeppelzauer
Zweitbegutachter/in: FH-Prof. Dipl.-Ing. Markus Seidl

Wien, 18.05.2015

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

- ich dieses Thema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter bzw. der Begutachterin beurteilten Arbeit überein.

....

Ort, Datum Unterschrift

Kurzfassung

Diese Arbeit ist im Gebiet der Audio-Informationsgewinnung angesiedelt und beschäftigt sich mit Mustererkennung und Klassifikation von Vogelstimmen.

Automatische Klassifikation von Tierlauten ist ein wichtiges Thema. So können, um einige wenige Anwendungsmöglichkeiten zu nennen, Gefahren entdeckt und verhindert, neue Tierarten entdeckt oder Überwachung von geschützten Arten gewährleistet werden. Dies hilft etwa Biologen, Forschern oder Menschen, die in der Nähe gefährlicher Tiere leben.

In dieser Arbeit wurden durch einen selbstentwickelten Klassifikator Vogelstimmen von südamerikanischen Singvögeln durch instanzbasiertes Lernen klassifiziert und ihrer Spezies zugeordnet.

Diverse Methoden und deren Ergebnisse werden erklärt und verglichen, um einen Überblick über die verschiedenen Herangehensweisen, ihre Grenzen, Möglichkeiten und ihre Vor- und Nachteile zu geben. Dazu wurden die Ergebnisse des Bird task und die Methode dieser Arbeit analysiert.

Die Resultate der Klassifikation waren mit bis zu 52,6% Genauigkeit sehr gut und mit den Ergebnissen der Teilnehmer des Bird task vergleichbar. Jedoch litt aufgrund der pragmatischen und einfachen Herangehensweise und der hohen Anzahl an generierten Daten die Performance des Klassifikators stark, weshalb das Datenset auf etwa 1/50 reduziert werden musste, um Experimente in einem realistischen Rahmen durchführen zu können. In der selbstentwickelten Methode wurde mit unterschiedlichen Audiomerkmalen gearbeitet, die sich rein aus den Spektrogrammen der Aufnahmen errechnen ließen.

Abstract

This work is located in the field of audio information retrieval. It covers pattern recognition and classification of birdsong.

Automatic classification of animal sounds is an important topic. Applications are the detection and prevention of threats, the discovery of new species and the monitoring of endangered. This could help biologists, scientists or people living in vicinity of dangerous animals.

In this thesis, a classification system has been developed that detects and analyses birdsong of south-American songbirds and classifies them with an instance-based classification method.

Several methods of classification and their results will be explained and compared, to give an overview of the different approaches, to show their limits, their possibilities and their advantages and disadvantages. The results of the self-implemented method and the results of the Bird task will be analysed.

The classification achieved up to 52.6% accuracy which can be compared to the results of the bird task participating teams. However the bad performance of the classifier due to the simple and pragmatic implementation and the high amount of generated data caused a reduction of the dataset to 1/50 to ensure the completion of the experiments in a realistic timeframe. The self-implemented method used different audio-features, computed only from the spectrograms of the given audio-data.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Automatische Klassifikation von Vogelstimmen ist eine wichtige Aufgabe im Bereich der bioakustischen Überwachung, sei es zur Erkennung neuer Arten, zur Beobachtung von Verhaltensweisen, Paarungsverhalten oder Migrationsmustern. Optisch sind Vögel oft schwierig oder unmöglich zu beobachten, da durch geografische oder botanische Faktoren die Erkennung ansässiger Tiere erschwert ist. Da sich Vögel aber mit akustischen Signalen verständigen, sind sie über Überwachungssysteme oder gezielte Aufnahmen gut zu überwachen.

Um Forschung im Bereich der Vogelstimmenerkennung zu fördern und um den Vergleich unterschiedlicher Methoden zu ermöglichen, wurde der Bird CLEF Task ins Leben gerufen (“Bird task | ImageCLEF - Image Retrieval in CLEF,” n.d.). Bird task war ein internationaler Wettbewerb, bei dem Methoden zur Vogelstimmenerkennung auf einem öffentlich verfügbaren Datensatz an Vogelstimmen entwickelt und objektiv evaluiert werden sollten. Die Thematik der Informationsgewinnung im Audiobereich ist sehr spannend und ermöglicht zahlreiche neue Entwicklungsansätze.

Das Fachgebiet der Arbeit ist Mustererkennung und maschinelles Lernen. Ziel der Arbeit ist es, einen Algorithmus für die Vogelstimmenerkennung basierend auf dem Bird CLEF Datensatz zu entwickeln. Die grundlegende Frage, die in dieser Diplomarbeit untersucht wird, ist, inwiefern instanzbasierte Methoden für die Vogelstimmenerkennung geeignet sind und wie diese im Vergleich zu alternativen Ansätzen, die auf komplexeren Merkmalen und Lernverfahren basieren, abschneiden.

1.2 Ziele und Forschungsfrage

Das Ziel, Vogelstimmen zu erkennen und zu klassifizieren ist keine neue Aufgabe im Bereich der Mustererkennung. Die Klassifikation von Vogellauten kann über unterschiedliche Arten erfolgen. Viele Ansätze versuchen, Daten weitgehend zu

abstrahieren und durch wenige repräsentative Werte kompakt darzustellen. Diese Methoden erfordern viel Vorwissen zur Bestimmung relevanter Merkmale und benötigen üblicherweise die Bestimmung einer großen Anzahl an Parametern. Das Ziel dieser Arbeit war es, eine möglichst einfache Methode zu entwickeln, die leicht verständlich und nachvollziehbar ist. Im Gegensatz zu komplexeren Methoden werden bei instanzbasiertem Lernen Vogellaute durch die Rohdaten selbst, im Zeit oder Frequenzbereich, repräsentiert. Diese werden dann mit einfachen Vergleichsmethoden mit unbekannten Daten verglichen, um diese klassifizieren zu können. Die größte Herausforderung dieser Vorgehensweise ist die Performance des Klassifikators, welche aufgrund der hohen Anzahl von Daten tendenziell schlechter ist als die von spezialisierteren Methoden. Der Fakt, dass die Rohdaten nicht abstrahiert werden, führt zu wesentlich größeren Datenmengen. Deshalb musste bei der Auswahl der Repräsentationsdaten stark darauf geachtet werden, dass die Dimensionalität der Daten möglichst gering, die Genauigkeit der Klassifikation jedoch möglichst hoch blieb.

1.3 Anwendungen

Mustererkennung im Audiobereich findet viele Anwendungen. Dabei sind die Aufgaben generell in drei Bereiche eingeteilt: Spracherkennung, Musikerkennung und die Erkennung von Umgebungsgeräuschen. Spracherkennung wird heutzutage in jedem Smartphone verwendet, um Suchanfragen, Nachrichten oder Anweisungen an das Telefons schneller eingeben zu können. Huggins-Daines et al. (2006) entwickelten einen Open Source Algorithmus zur Spracherkennung. Um ein anderes Beispiel zu nennen, kann wie von Stinson, Elliot, Kelly, & Liu (2009) für Gehörlose Sprache erkannt und in Schrift umgewandelt werden. Im musikalischen Bereich kann wie bei der populären Smartphone-App Shazam (Wang, 2006) ein Musikstück schnell und zuverlässig erkannt, oder wie in (Silla, Kaestner, & Koerich, 2007) Musik automatisch einem Genre zugeordnet werden.

Umgebungsgeräusche umfassen alle Geräusche, die weder Sprache noch Musik sind. Eine wichtige Anwendung im Kontext von Umgebungsgeräuschen ist die bioakustische Überwachung. Mit Hilfe von bioakustischer Überwachung können beispielsweise großräumige Überwachung von Populationen, Erkennung von Anwesenheit an speziellen Orten oder Lokalisierung von Tieren und die Erstellung von Bewegungsmustern gewährleistet werden.

Die Erkennung von Tierfamilien oder ihren Lauten ist gerade für Biologen ein wichtiges Thema. So können häufig aufgrund widriger Umstände, wie beispielsweise im dichten Dschungel, die Tiere schwer oder gar nicht gesehen

werden. Eine weitere Anwendung ist die akustische Früherkennung von Tieren. Beispielsweise werden im Projekt von Zeppelzauer, Stöger, & Breiteneder (2013), Elefantenherden anhand ihrer Laute aus weiter Entfernung erkannt. Die Art der Laute, die diese von sich geben, gibt Aufschluss über ihr voraussichtliches Verhalten. Sind die Elefanten aggressiv, können Dörfer präventiv evakuiert, oder die Herden umgelenkt werden. Auch die Erkennung neuer oder bereits als ausgestorben vermuteten Arten kann über akustische Überwachungssysteme bewerkstelligt werden. Weitere Beispiele wären die Überwachung des Verhaltens von Vogelarten im Bereich der Paarung (Baker, 2006), der Völkerwanderung (Bardeli et al., 2010) oder des Speziesreichtums einer Region (Gasc et al., 2013).

1.4 Organisation

Die vorliegende Diplomarbeit ist wie folgt strukturiert: Zunächst werden Grundbegriffe der Audioanalyse erklärt und diese dadurch für den Leser dieser und verwandter Arbeit verständlich gemacht. Kapitel 2 gibt eine Einführung in die automatische Geräuscherkennung und präsentiert verwandte Methoden für die Vogelstimmenerkennung. Kapitel 3 beschreibt die neu entwickelte Methode für die Erkennung von Vogelstimmen basierend auf instanzbasiertem Lernen. In Kapitel 4 und 5 sind durchgeführte Experimente und deren Ergebnisse zu finden, die Tests und Testreihen erklären und evaluieren sollen. Abschließend wird die Arbeit in Kapitel 6 zusammengefasst und ein Ausblick auf offene Themen gegeben.

1.5 Definition der wesentlichen Basisbegriffe

1.5.1 Merkmal

Ein Merkmal (engl. feature) ist eine kompakte Repräsentation, die eine Eigenschaft eines Signals darstellt. Beispielsweise kann ein Merkmal die Lautstärke oder Tonhöhe eines Signals sein.

1.5.2 Merkmalsvektor

Mehrere Merkmale können zu einem Merkmalsvektor (engl. feature vector) zusammengefasst werden. Dieser wird für das Training des Klassifikators und für die Klassifikation verwendet. Diese normierte Darstellung ist für die meisten Klassifikatoren notwendig.

1.5.3 Klassenzugehörigkeit

Jede Aufnahme muss einer Klasse (engl. class), also im Fall dieser Arbeit, einer Vogelspezies zugehörig sein. Diese kann im Vorhinein bekannt sein, oder durch die Klassifikation ermittelt werden. Die Klasse wird durch ein numerisches „Klassenlabel“ angegeben.

1.5.4 Abtastrate

Audiodateien werden bei der Digitalisierung abgetastet. Dies geschieht mit einer Abtastrate, die beispielsweise bei 44.100 Hz liegt. 44.100 Mal pro Sekunde werden digitale Daten von einem analogen Ton oder Geräusch abgetastet. Einer dieser Abtastzeitpunkte (engl. samples) repräsentiert somit eine 44.100-el Sekunde Audio.

1.5.5 Entscheidungsgrenze

Jeder Klassifikator muss Grenzen zwischen verschiedenen Klassen ziehen, um diese zu trennen. Diese Entscheidungsgrenze (engl. decision boundary) wird üblicherweise aus Trainingsdaten gelernt, um eine möglichst hohe Klassifikationsrate zu erzielen.

1.5.6 Genauigkeit

Die Qualität einer Klassifikation kann mit verschiedenen Maßen gemessen werden. Eines davon ist die Genauigkeit (engl. accuracy). Diese wird durch den

Prozentsatz der Aufnahmen berechnet, die von einem Klassifikator der richtigen Klasse zugeordnet wurden.

$$Genauigkeit = \frac{Anzahl\ der\ richtig\ gelabelten\ Dateien}{Anzahl\ aller\ gelabelten\ Dateien}$$

1.5.7 Fehlerrate

Die Fehlerrate (engl. error-rate) ist der Prozentsatz der Aufnahmen, die falschen Klassen zugeordnet wurden.

$$Fehlerrate = 1 - Genauigkeit$$

1.5.8 Frequenzband

Viele akustische Signalverarbeitungsmethoden arbeiten mit Frequenzbändern. Diese fassen je nach Einstellung einen kleineren oder größeren Bereich von Frequenzen zusammen, da die Berechnung für jede einzelne ganzzahlige Frequenz nicht praktikabel ist.

1.5.9 Unterscheidungsfähigkeit

Die Unterscheidungsfähigkeit (engl. discrimination ability) eines Merkmals gibt an, wie gut es zwei verschiedene Klassen unterscheiden kann. Ist es möglich, eine genaue Entscheidungsgrenze zwischen diesen zu ziehen, spricht man von guter Unterscheidungsfähigkeit. Ein Merkmal soll im Allgemeinen wenig Varianz innerhalb einer Klasse und viel Varianz über Klassen hinweg aufweisen.

1.5.10 Generalisierbarkeit

Die Generalisierbarkeit (engl. generalization) bestimmt, wie gut ein Klassifikator Klassen unterscheiden kann. Wenn er es schafft, zwischen der Grenze zweier Klassen einen großen Abstand (engl. margin) zu den Vertretern der Klasse herzustellen, spricht man von guter Generalisierbarkeit des Klassifikators. Der Mensch schafft es meist durch Sinneswahrnehmung, ein Objekt einer Klasse zuzuordnen. Als Beispiel kann er durch das Bellen und das Aussehen eines Hundes, diesen in die Kategorie Hund einordnen. Da ein Klassifikator diese Fähigkeit nicht besitzt, muss er durch Merkmalsvergleiche eine Grenze ziehen.

2 Grundlagen und verwandte Arbeiten

2.1 Gebiete der Audio-Klassifikation

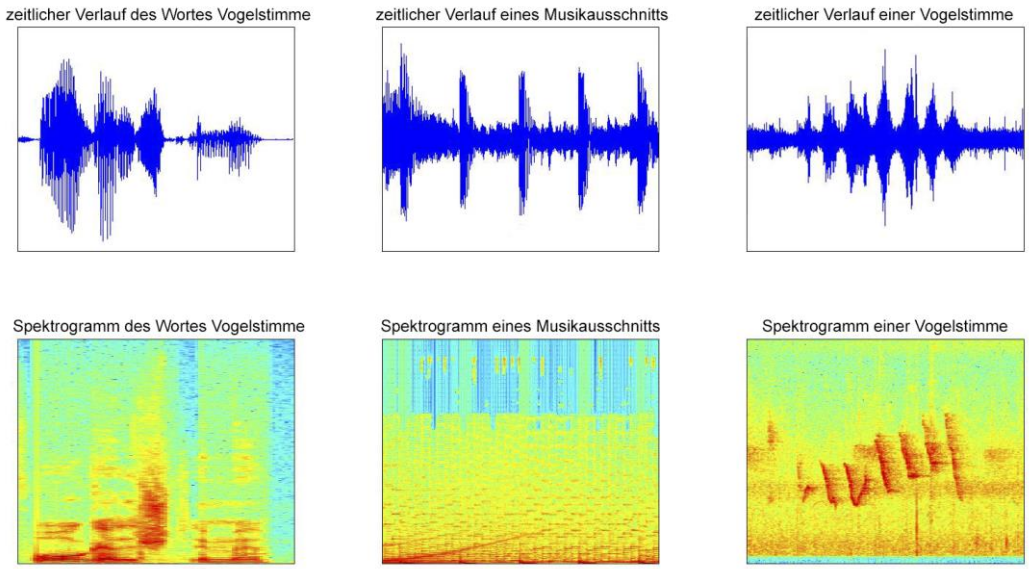


Abbildung 1: zeitliche und spektrale Darstellung von Sprache (li.), Musik und einer Vogelstimme (re.)

Im Bereich der Audioklassifikation gibt es drei große Gebiete: Sprache, Musik und Umgebungsgeräusche. Wie in (Abbildung 1) ersichtlich, unterscheiden sich diese in der zeitlichen Darstellung kaum. Lediglich repetitive Elemente sind in Musik und Vogelstimme erkennbar. In der spektralen Darstellung jedoch offenbaren sich große Unterschiede. Sprache hat für jedes Phonem, also jeden Laut, einen Bereich im Spektrum, der die Verteilung der Obertöne darstellt. Bei Musik sind rhythmische Elemente gut zu erkennen, wie eine Hi-Hat an den abgegrenzten Ausschlägen im Hochtonbereich. Sogar Melodiefolgen sind im tieffrequenten Bereich erkennbar. Bei der Darstellung der Vogelstimme sind wiederum klar definierte und sehr kurze Melodieverläufe sichtbar, die die Vogelstimme gut von der Umgebung abgrenzen. Dies ist ein großer Vorteil in der Vogelstimmenüberwachung und -klassifikation, den man sich zu Nutze machen kann.

Im Bereich der Audioklassifikation gibt es eine Vielzahl von Anwendungsmöglichkeiten wie Segmentierung von Stille, automatische Spracherkennung, Generierung von Musikinformationen oder Umgebungsgeräuscherkennung (engl. environmental sound recognition). Bei der Segmentierung wird Stille von auftretenden Signalen unterschieden. Dies ist wichtig, um gleichartige Teile in Aufnahmen zu finden. Mit Teilen, die nicht still sind, kann nun weitergearbeitet werden.

Spracherkennung arbeitet normalerweise auf einer syntaktischen Ebene, wie in (Lawrence Rabiner & Juang, 1993) beschrieben wird. Außerdem können neben der Erkennung von Worten auch die gesprochene Sprache oder sogar Emotionen erfasst werden (Ververidis & Kotropoulos, 2006).

Bei der Informationsgewinnung von Musik werden ähnliche oder gleiche Stellen, Instrumente, Künstler, Musikgenres oder generell musikalische Strukturen erkannt (Downie, 2003). Außerdem können wie in (Klapuri & Davy, 2006) erwähnt, interessante statistische Daten wie Tonhöhen, Hüllkurven, sowie Dauer und Quelle von Impulsen jedes vorkommenden Tones erkannt werden.

Das Thema dieser Arbeit, das sich mit Vogelstimmenerkennung befasst, ist im Bereich der Umgebungsgeräuscherkennung angesiedelt. Informationsgewinnung von Umgebungsgeräuschen betrachtet alle Signale, die weder Sprache noch Musik sind. Cowling & Sitte (2003) beschreiben in einem Artikel verschiedene Methoden zur Merkmalsextraktion und Klassifizierung von Umgebungsgeräuschen. Die Aufgabe, die normalerweise das menschliche Gehirn in einer Vielzahl von Fällen übernimmt, soll auch von einem Computer vorgenommen werden. Diese Fähigkeiten beinhalten, Gehörtes zu unterscheiden und einer Bedeutungskategorie oder bereits gehörten Lauten zuzuweisen. Computer repräsentieren Audio-Daten nur als numerische Folgen, was die Zuweisung einer Bedeutung weit schwieriger macht.

Das Vorwissen, das Menschen haben, hilft semantische Beziehungen herzustellen. Ein Hörer eines musikalischen Werkes kann gegebenenfalls Motive, Themen und Bewegungen in der Musik und damit verbundene Emotionen nachvollziehen und einordnen. Die Repräsentation dieser Informationen ist einem Computer aber nur möglich, wenn ein Mensch diesen sogenannten semantischen Spalt schließt. So werden aus numerischen Daten von Audio-samples zusammenhängende Noten mit bestimmten Tondauern. Die Bezeichnung von Daten durch den Menschen hilft, den Computer an die menschliche Wahrnehmung anzunähern und durch unterschiedliche Arten der Klassifikation die menschliche Kognition nachzubilden.

2.2 Aufbau eines akustischen Klassifikationssystems

Ein Klassifikationssystem folgt einer festgelegten Architektur, die im folgenden Abschnitt genauer erklärt wird. Unterschiedliche Optimierungen können zu Abweichungen dieser allgemeinen Architektur führen. Die grundlegenden Verarbeitungsschritte sind jedoch meist dieselben.

2.2.1 Aufnahme

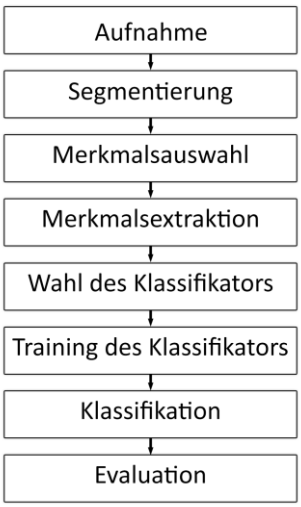


Abbildung 2: Aufbau eines akustischen Klassifikationssystems

Als erster Schritt werden Daten für die weitere Verarbeitung eingelesen. Diese Daten sind im Falle akustischer Überwachungs- und Klassifikationssysteme Audioaufnahmen. Bei der Audioaufnahme müssen Störquellen, wie geringe Datenraten, Rauschen oder andere zufällige Störsignale möglichst vermieden werden. Die Anforderungen der Aufgabe und die technischen Beschränkungen des jeweiligen Systems sind genau abzuklären, um diesen Problemen entgegenzuwirken. Nebengeräusche können mit Richtmikrofonen minimiert werden, die gezielt definierte Positionen abnehmen. Im Gegensatz dazu stehen unidirektionale Mikrofone, die jedoch einen größeren Bereich abdecken. Daher ist es sinnvoll, die Aufnahmetechnik der Aufgabe des Systems anzupassen. Sogenannte „Windshields“ minimieren Windgeräusche an den Mikrofonen, um eine rauschärmere Aufnahme zu ermöglichen. Bei der Aufnahme lassen sich zwei Arten von Systemen unterscheiden: Echtzeitsysteme, die die aufgenommenen Daten sofort und in Echtzeit analysieren und Systeme, die voraufgenommene Daten verarbeiten, sogenannte „Batchsysteme“. Systeme für den Bird CLEF fallen in die zweite Kategorie, da die aufgenommenen Daten im Nachhinein analysiert und klassifiziert werden.

2.2.2 Segmentierung und Vorverarbeitung der Aufnahmen

Ziel eines Audioklassifikationssystems ist es, akustische Ereignisse wie Laute, Lautanfänge oder Melodien zu erkennen, Ziel der Segmentierung, Relevantes von Irrelevantem zu trennen. Ein klassisches Beispiel für die Vorverarbeitung ist die Erfassung von Stille durch Schwellenwertbildung.

Um mit den relevanten Lauten der verschiedenen Spezies arbeiten zu können, müssen diese erst gefunden und extrahiert werden. Dies ist als Laie eine sehr schwierige Aufgabe, da die Arbeit in einem fremden Anwendungsgebiet viele Fehler erzeugen kann. Wie diese Segmentierung erfolgen kann, wird später in unterschiedlichen Herangehensweisen demonstriert.

2.2.3 Merkmalsauswahl

Welche Merkmale extrahiert werden, kann durch unterschiedliche Bewertungen erfolgen: Die Eingangsdaten zu analysieren hilft, Unterschiede in Spektrogrammen oder Aufnahmen herauszusehen/-hören und motiviert diese Merkmale gezielt herauszufinden. Vorkenntnisse zu den analysierten Daten wären hierfür von Vorteil. So kann der Entwickler etwa aus Domänenwissen damit rechnen, dass sich Vogelstimmen vor allem in der Frequenz unterscheiden, oder Spracherkennung durch Onset-Analyse sehr gut funktioniert. Diese Faktoren bestimmen stark die Auswahl eines Merkmals, da nicht erst während des Trainierens des Klassifikators Entscheidungen getroffen werden. Gute Merkmale sollten außerdem leicht zu extrahieren, rauscharm und nützlich zur Unterscheidung von Mustern sein. Es sollen Merkmale gefunden werden, die eine bestimmte Klasse von Aufnahmen sehr genau repräsentiert, sich aber von anderen Klassen möglichst stark unterscheidet.

2.2.4 Merkmalsextraktion

Bei der Merkmalsextraktion werden vorher definierte Daten aus Aufnahmen extrahiert, die dann als Merkmale die Originaldaten repräsentieren. So kann es sich im einfachsten Fall um die Länge eines Signals, den Tonumfang oder die gemittelte Lautstärke handeln. Merkmale werden in weiterer Folge zu einem Merkmalsvektor zusammengefasst. Dieser Vektor repräsentiert jeweils alle gewählten extrahierten Merkmale einer Aufnahme.

2.2.5 Wahl des Klassifikators

Der gewählte Klassifikator entscheidet, welche Daten in welcher Form verwendet werden, um die Klassifikation durchzuführen. Die Entscheidung, welcher Klassifikator für die gewählte Aufgabe gut geeignet ist, kann deshalb sehr schwierig sein, weil die Qualität eines Klassifikators oft von Zufallswerten oder falscher Verwendung der Daten herabgesetzt wird. Hier ist es als Benutzer wichtig unterscheiden zu können, welche Fehler der Klassifikator und welche der Benutzer und die Eingangsdaten verursachen.

Die Wahl des Klassifikators muss an die Anzahl der Trainingsdaten und an die Art der vorliegenden Merkmalsdaten angepasst werden. Bei einer hohen Dimensionalität der Merkmalsvektoren bieten sich Klassifikatoren wie Support Vector Machines an (Fagerlund, 2007). Sind die vorliegenden Daten extrahierte Segmente von Spektrogrammen, ist Clustering sinnvoll (Jain, Murty, & Flynn, 1999). Liegen die Daten als vollständige Audio-Aufnahmen vor, sind Hidden Markov Models (Trifa, Kirschel, Taylor, & Vallejo, 2008) oder ein Gauss Klassifikator (Zhou, Liu, & Duan, 2005) geeignet.

2.2.6 Training und Klassifikation

Für das Training eines Klassifikators werden grundsätzlich die Eingangsdaten so eingesetzt, dass ein Trainings- und ein Testset an Daten dazu verwendet wird, den Klassifikator zu trainieren und anschließend zu testen. Bei der sogenannten Kreuzvalidierung werden diese Daten mit einer definierten Wiederholungsanzahl zufällig gemischt und jeweils als Training- und Testset verwendet.

Das Training eines Klassifikators kann auf drei unterschiedliche Arten erfolgen: überwachtes Lernen (engl. supervised learning), teil-überwachtes Lernen (engl. semi-supervised learning) und unüberwachtes Lernen (engl. unsupervised learning). Beim überwachten Lernen muss im Vorfeld von einer wissenden Person jeder Eintrag der Trainingsdaten einer Klasse zugeordnet werden. Es muss außerdem sehr darauf geachtet werden, dass das Modell zur Klassifikation richtig erstellt wird, um Over- oder Underfitting zu vermeiden.

Overfitting bedeutet, dass das Modell zu empfindlich auf die Testdaten reagiert. Der Klassifikator versucht eine zu komplexe Entscheidungsgrenze zwischen verschiedenen Klassen zu ziehen. Die gewählten Merkmale und die Klassifikation weisen also eine schlechte Generalisierbarkeit auf. In solchen Fällen ist eine einfachere Form der Klassifikation und damit verbunden eine bessere Performance erstrebenswert, auch wenn schlechtere Klassifikationsergebnisse zu erwarten sind. Das Gegenteil des Overfittings ist das Underfitting; das heißt, dass das Modell zur Klassifikation zu ungenau definiert ist und es dadurch zu Fehlklassifikationen kommt. Daher ist es wichtig, das Trainings- und Testset durch Kreuzvalidierung zu mischen, um möglichst unabhängige Ergebnisse vom Trainingsset zu erhalten.

Im Gegensatz dazu sind beim unüberwachten Lernen keine Klassenzugehörigkeiten bekannt. Hier muss der Algorithmus Gemeinsamkeiten zwischen den verschiedenen Aufnahmen finden und diese der jeweils wahrscheinlichsten Klasse zuordnen. Diese Methode nennt sich Clustering. Es

gibt hier Unterschiede bei der Entscheidung zur Zuordnung einer Datei zu einem Cluster und bei der Art der Bestimmung der Clusteranzahl. Bei teil-überwachtem Lernen sind einige, aber nicht alle Testdaten einer Klasse zugeordnet.

Typischerweise wird der Klassifikator durch bekannte Klassenzugehörigkeiten der Test- und Trainingsdaten trainiert, bis die Genauigkeit der Klassifikation zufriedenstellend ist. Leider ist auf diese Art und Weise nur eine richtige oder falsche Zuordnung zu einer Klasse ersichtlich; aber nicht den Grund für die Entscheidung des Klassifikators.

Bei der Klassifikation werden die Merkmalsvektoren dazu verwendet, eine unbekannte Aufnahme einer Klasse zuzuordnen. Die Ähnlichkeit der Merkmalsvektoren einer Klasse sollte möglichst hoch sein. Dies ist aber durch Zufallswerte wie Rauschen oft erschwert, da zufällige Daten auch in jeder Art der Aufnahme oder Verarbeitung auftreten können.

Ein weiteres Problem ist, dass eventuell nicht aus jeder Aufnahme sinnvolle Daten extrahiert werden können. Wird zum Beispiel eine Aufnahme durch Störgeräusche unkenntlich gemacht, sind eventuell wichtige Parameter verfälscht. Auf solche Fehler muss mittels Detailbetrachtung der Ergebnisse eingegangen werden, um diese zu verhindern.

2.2.7 Evaluation

Die Qualität der Ergebnisse kann zum Beispiel mit der Fehlerrate gemessen werden, welche durch den Prozentsatz der falsch zugeordneten Aufnahmen repräsentiert wird. Diese sollte möglichst gering gehalten werden. Hierbei muss aber zwischen Kosten und Risiko abgewogen werden. Sind die Kosten für die Verbesserung des Klassifikators höher? Oder die der eventuell durch Fehlentscheidungen verursachten Schäden?

Es besteht außerdem auch die Möglichkeit mehrere Klassifikatoren kombiniert einzusetzen und unterschiedliche Aufgaben übernehmen zu lassen. Hierbei muss aber auf die Entscheidungshierarchie geachtet werden, um richtige aber eventuell wenig gewichtete Ergebnisse nicht durch falsche höher gewichtete zu überstimmen.

Durch fortwährendes Testen mit der Auswahl der Merkmale und des Klassifikators, kann der Entwickler nun evaluieren, welche die beste Klassifikationsmethode ist. Er muss nun Entscheidungen treffen, welche Parameter er an welchen Punkten so verändert, dass beide Qualitätsmerkmale (Genauigkeit und Laufzeit/Performance) möglichst gut werden.

Die Laufzeit ist ein wichtiger Faktor in der Erstellung einer Klassifikationsmethode. Die Komplexität eines Algorithmus skaliert mit der Anzahl der Merkmals-Dimensionen oder der Anzahl an Klassen. Der Entwickler muss anhand des Bereichs, in dem die verwendete Methode eingesetzt wird und damit verbundenen Rechenkapazitäten entscheiden, welches Klassifikationsmodell sinnvoll und praktikabel ist.

2.3 Charakteristika von Vogelstimmen

Vogelstimmen können in die beiden Kategorien Gesang (engl. song) und Ruf (engl. call) unterteilt werden. Diese unterscheiden sich sowohl syntaktisch, als auch in der Länge und im Aufgabenbereich des Lautes. Für die Erkennung einer Spezies ist Vorwissen über Charakteristika hilfreich, die diese Spezies auszeichnen können.

2.3.1 Einheitliche Charakteristika

Charakteristika, die Vogelstimmen in jedem Fall beschreiben und unterscheiden, sind laut Fagerlund Dauer, Frequenz und Komplexität (2007, p. 2). Die Dauer von Lauten variiert je nach Spezies, Einsatzzweck und Lautart. So sind etwa Vogelrufe kürzer als Gesänge. Der Frequenzbereich von Vogelstimmen wird hauptsächlich durch die Spezies festgelegt. Während eines Vogelgesangs variiert die Frequenz der gesungenen Töne, was den typischen Klang einer Spezies ausmacht. Die Komplexität der Melodien und Lautarten ist bei Gesängen deutlich höher als bei Rufen, da diese nicht zur Warnung, sondern zur Paarung oder Abschreckung von Rivalen bei der Gebietsmarkierung eingesetzt wird.

2.3.2 Vogelgesang

Nach Chou, Lee, & Ni (2007, p. 1) ist Vogelgesang, komplex, vielfältig, ansprechend und angenehm zu hören. Diese Laute werden üblicherweise von männlichen Vögeln produziert und dienen dem Anlocken eines Paarungspartners und der Gebietsmarkierung. Außerdem sind nach Chang-Hsing, Yeuan-Kuen, & Ren-Zhuang (2006, p. 1) Gesänge in Untergruppen unterteilt: Elemente/Noten (engl. elements/notes), Silben (engl. syllables), Motive/Phrasen (engl. motifs/phrases), Typen (engl. types) und Reihen (engl. bouts). Elemente und Noten sind die einfachsten individuellen Geräusche, die ein Vogel von sich gibt. Mehrere Elemente/Noten, die nacheinander in einem regulären Muster vorkommen, werden als Gesangssilbe (engl. song syllable) bezeichnet. Eine Sequenz dieser Silben, die wiederholt auftritt, wird als Gesangsmotiv oder -phrase

bezeichnet. Eine bestimmte Kombination von Motiven, die wiederholt vorkommt, ergibt einen Gesangstyp. Eine Gesangsreihe ist eine Sequenz von einem oder mehreren Motiven mit variabler Länge und mit stillen Pausen als Trennung zwischen dazwischen.

Catchpole & Slater (2008, p. 9) teilen Gesänge in Noten, Silben, Phrasen und Gesang (engl. song). Die gemeinsame Übereinstimmung mit Chou et al. (2007) ist also, Vogelgesang in Noten, Silben und Phrasen zu unterteilen, wobei die nächste übergeordnete Gruppe die Reihe oder der Gesang ist. In (Abbildung 3) ist der Aufbau von Vogelgesang grafisch dargestellt.

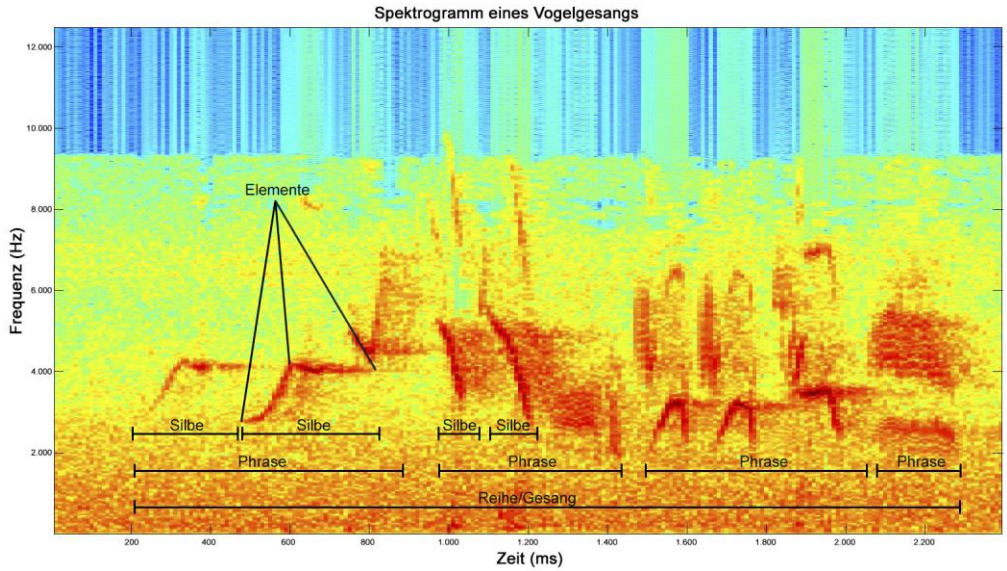
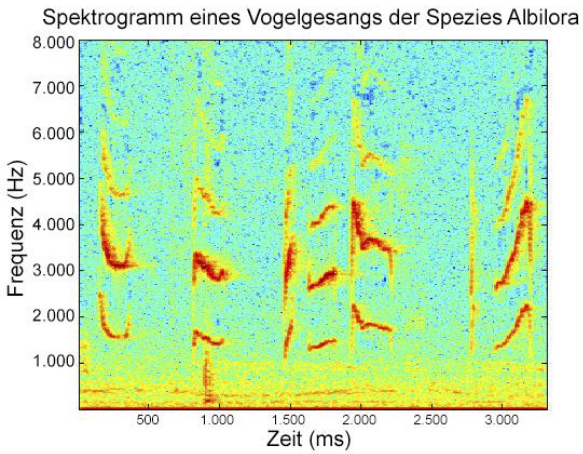
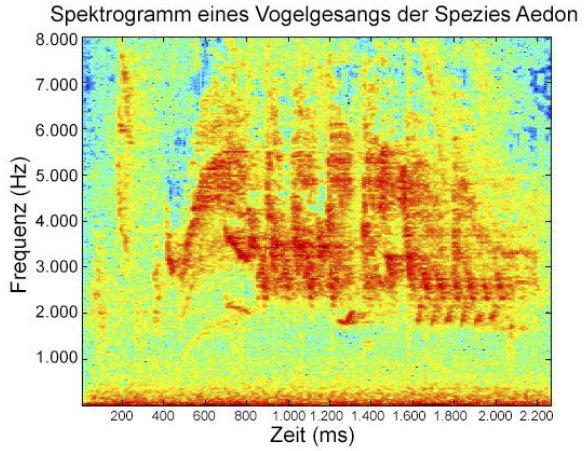


Abbildung 3: Spektrogramm und Aufbau eines typischen Vogelgesangs

Um ersichtlich zu machen, wie heterogen Vogelstimmen einzelner Spezies sein können, findet sich hier eine Gegenüberstellung einzelner Spektrogramme ähnlicher Länge mit Lauten der Spezies:

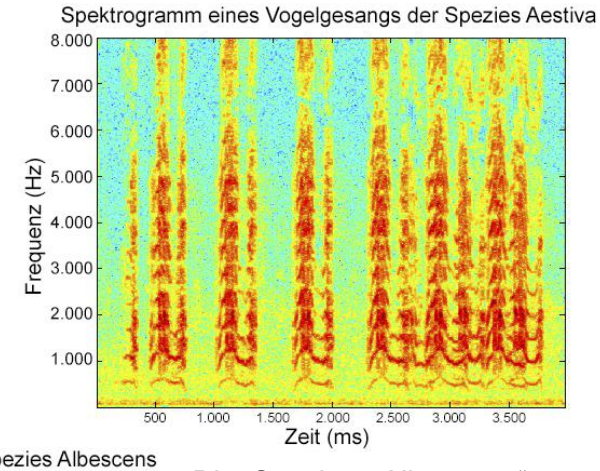
Die erste Spezies „Albilora“ produziert sehr markante und melodiöse Laute. Die Laute klingen ähnlich quietschend, wie das Auslassen von Luft aus einem Luftballon. Außerdem sind die einzelnen Silben verhältnismäßig lange, wobei die Phrasen zueinander größere Pausen aufweisen.

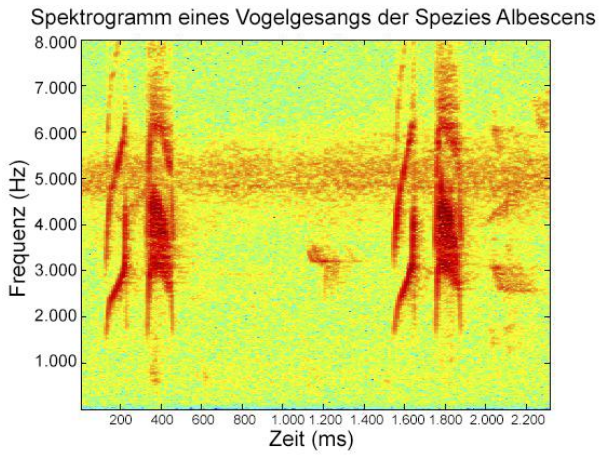




Die Spezies „Aedon“ erzeugt schrille, obertonreiche und vor allem sehr kurze Laute. Sie sind mit den uns bekannten Lauten von Meisen vergleichbar. Die Elemente sind also sehr kurz, die Phrasen aber durch eine hohe Anzahl an Wiederholungen sehr lange.

Die Laute der Spezies „Aestiva“ sind Kinderschreien sehr ähnlich. Sie sind kehlig und erwecken den Eindruck, Worte formen zu wollen. Im Fall dieser Aufnahme wäre das Wort „Mama“. Die Länge der Phrasen und Pausen liegen im Mittelfeld.





Die Spezies „Albescens“ macht verhältnismäßig lange Pausen zwischen den Phrasen. Eine Phrase besteht immer aus zwei Silben, einer mit aufsteigenden und eine mit gleichbleibender Melodie. Die Laute sind schrill und erinnern an den Gesang eines Finken.

Diese Gegenüberstellung zeigt nun, dass die Gesänge von Vögeln unterschiedlicher Spezies sehr charakteristisch sind und sich sehr gut dazu eignen, die Spezies zu klassifizieren. Dass diese Aufgabe sehr komplex werden kann, wird klar, wenn alle Parameter einbezogen werden, die das menschliche Ohr zur Klassifikation einer Spezies verwenden kann. So sollten Tonhöhe, Klangfarbe, Bandbreite, Tonlänge, Melodieverlauf, Pausenlänge, Lautstärke und

viele weitere Eigenschaften in die Klassifikation mit einfließen. Da ein Computer weder die Erfahrungswerte eines Ornithologen noch die Möglichkeit der menschlichen Kategorisierung besitzt, ist die Aufgabe der automatischen Vogelstimmenerkennung sehr komplex und schwierig.

2.3.3 Vogelrufe

In Gegenüberstellung zu Vogelgesang, der im vorigen Bereich behandelt wurde, beschreiben Chou et al. (2007, p. 1) Vogelrufe als monotone, kurze, wiederholte, festgelegte und geschlechtslose Laute, um Gefährten zu kontaktieren oder zu warnen. Sie unterscheiden sich von Gesang durch Dauer, Komplexität und Melodie. Da diese Rufe so kurz und monoton sind, ist es schwieriger, anhand dieser eine Spezies von einer anderen zu unterscheiden.

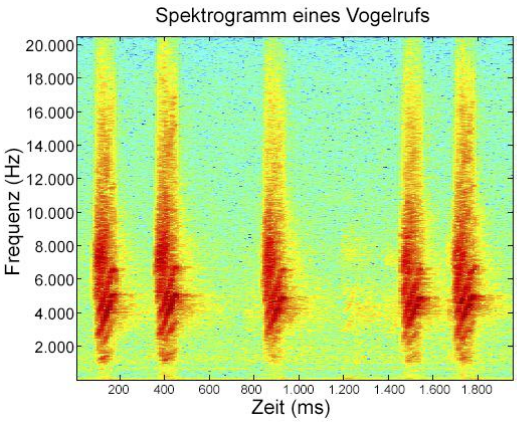


Abbildung 4: Spektrogramm eines Vogelrufs

2.4 Verwandte Arbeiten

Im folgenden Abschnitt werden verschiedene Methoden zur Vogelstimmenklassifikation erklärt. Diese wurden nach dem Klassifikator geordnet und sollen einen Überblick über Arbeiten und Methoden im Fachgebiet dieser Arbeit zeigen. Die unterschiedlichen Methoden und ihre Herangehensweisen beschreiben jeweils eine Möglichkeit, Vogelstimmen zu klassifizieren und die Ergebnisse und/oder Probleme der jeweiligen Methode.

2.4.1 Hidden Markov Models

Ein HMM ist ein Klassifikator, der mit Zeitdaten arbeitet. Er wird mit Abfolgen von Aufnahmen gespeist, um das Modell zu trainieren. Ein HMM funktioniert in der Regel folgendermaßen: ein 5-Tupel, also eine Menge von fünf Elementen (Ω_X, Ω_O, A, B, π), wird definiert. Ω_X = [S_1...S_N] ist eine definierte Menge mit N unterschiedlichen Zuständen. Ω_O = [v_1..v_k] ist eine Menge aus möglichen Beobachtungen. λ = (A, B, π) sind Parameter der HMM-Kette mit A_{NxN} als jene Matrix, die die Übergangswahrscheinlichkeiten enthält. B_{Nxk} enthält die Wahrscheinlichkeiten für jede Beobachtung in jedem einzelnen Zustand. π_{1xN} definiert die Verteilung der Anfangszustände.

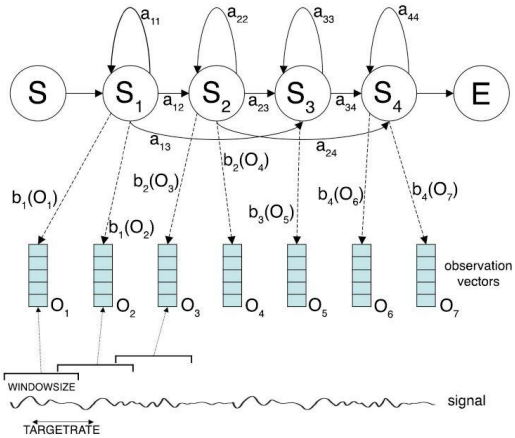


Abbildung 5: grafische Darstellung des Ablaufs eines Hidden Markov Models Quelle: (Young et al., 2002)

Praktisch funktioniert ein HMM folgendermaßen: ein Eingangssignal wird an den Klassifikator angelegt. Das erste vorkommende Element (in diesem Fall gefensterte Audio Daten mit einer definierten Fenstergröße und Überlappung [engl. windowsize und targetrate]) wird nun mit den möglichen Beobachtungen verglichen und die wahrscheinlichste Beobachtung wird für den ersten Zustand gewählt. Nach und nach werden die Eingangsdaten weiter verarbeitet, wobei Zustand zwei weiß, welche Beobachtung in Zustand eins gewählt wurde, und so weiter. Die Abfolge der Zustände ist aber nicht nach außen hin sichtbar, weshalb das Verfahren Hidden Markov Model heißt.

Ein HMM ist ein statistisches Werkzeug, das ein diskret-zeitliches dynamisches System modellieren kann. Dies geschieht mittels des Markov Prozesses mit

unbekannten Parametern. Das dynamische System ist in diesem Fall ein Vogel, der eine Sequenz von Beobachtungen produziert. Die Überlegung, jedes registrierbare Ereignis in einem Status darzustellen, wäre nicht komplex genug, da der Bezug zwischen sehr ähnlichen Ereignissen schwer herzustellen ist. Deswegen wurde das Modell so erweitert, dass jedes Ereignis mit einer gewissen Wahrscheinlichkeit einem Status zugeordnet wird. Für jede Vogelspezies wird ein eigenes HMM verwendet, um den zeitlichen Verlauf der akustischen Merkmale zu repräsentieren. Dadurch wird die Erkennung durch die höchste Ähnlichkeit der Trainingsdaten zur Sequenz der jeweiligen Testdatei ermöglicht. Die Schwierigkeit liegt hier bei der Einschätzung der Parameter der HMMs und deren Verwendung zur Annahme einer gleichen Abfolge von Ereignissen von Trainings- auf Testdaten.

Im Gegensatz zu Verfahren mit Mustervergleichen müssen im Fall der HMMs die Signale analysiert werden, um Merkmale an regelmäßigen diskreten zeitlichen Schritten zu finden. Danach können mit diesen Daten Laute verglichen werden. So werden an diskreten Schritten Merkmalsvektoren extrahiert und als eine Beobachtung gespeichert. Diese Serie an Beobachtungen wird folgend von einem HMM verwendet, um eine zeitliche Abfolge der Merkmale der jeweiligen Klasse zu erkennen. Diese Ergebnisse werden anschließend mit dem HMM einer unbekannten Klasse verglichen um herauszufinden, welche am wahrscheinlichsten die gleiche Sequenz an Beobachtungen produzieren würde.

Eine weitere Möglichkeit die Beobachtungen zu generieren, ist, das Eingangssignal mittels Linear Predictive Coding (LPC) zu analysieren und daraus die Beobachtungsvektoren zu erstellen. Dieses Verfahren versucht, aus Beziehungen vergangener samples die folgenden samples vorherzusagen. Dafür stehen Koeffizienten zur Verfügung, die so gewählt werden müssen, dass die Fehlerquote zwischen den Vorhersagen und den tatsächlichen Werten möglichst gering ist.

Trifa et al. (2008) stellten ein System zur Klassifikation von Vogelstimmen mit dem Einsatz von HMMs vor. Ziel war es, eine einfache aber effiziente Methode der Vogelstimmenklassifikation zu entwickeln, die leicht zu implementieren und verändern ist und in Echtzeit auf durchschnittlicher Hardware laufen soll. Die einzelnen Parameter der Hidden Markov Models wurden genau betrachtet und für den Einsatz als Trainings- und Testdaten analysiert. Dies führte zu Ergebnissen, wie der minimalen Anzahl an Stichproben für eine ausreichend genaue Erkennung.

Die aufgenommenen Vogelstimmen von Ameisenvögeln aus Mexiko aus 2005 und 2006 mit 44,1 KHz Samplerate und 16 Bit Samplerate wurden aus verschiedenen

Entfernungen gemacht, was zu sehr unterschiedlichen Signal-Rausch-Abständen führte. Die Aufnahmen wurden mit einem Hochpass-Filter bei 400 Hz gefiltert, um tieffrequente Störgeräusche zu entfernen. Im nächsten Schritt wurden 2-3,6 Sekunden lange Rufe herausgeschnitten, um sie für das Training des Modells verwenden zu können. Dies wurde mit dem Raven 1.3 program bewerkstelligt (Charif, Clark, & Fristrup, 2006). Die Qualität der einzelnen Wörter wurde subjektiv je nach SNR von A-E bewertet, wobei A die höchste Qualität repräsentierte. A-C wurden in Folge als hochqualitativ und D und E als zu verrauscht bezeichnet. Diese Bewertung wollten die Entwickler so einsetzen, dass sie die Ergebnisse mit menschlich gut hörbaren (A-C) und verrauschten (D-E) Wörtern vergleichen, um zu sehen, wie ähnlich HMM gegenüber dem menschlichen Hörvermögen arbeiten.

Das eingesetzte Hidden Markov Model Toolkit (HTK) ist eine Entwicklung des Cambridge University Engineering Departments und vereinfacht durch ein Toolkit (beinhaltet Bibliotheken und Werkzeuge für das Entwickeln, Trainieren, Testen und Analysieren von HMMs und deren Ergebnisse) die Anwendung und die Verständlichkeit der Ergebnisse eines HMM (Young et al., 2002). Der Code ist frei verfügbar und beinhaltet eine der häufigsten und flexibelsten Implementierungen der HMM.

Experimente haben gezeigt, dass das menschliche Gehör Frequenzen nicht linear abbildet und diese Art der Abstufung auch in der Mustererkennung bessere Ergebnisse erzielt. So werden bei Mel-Frequency Cepstral Coefficients (MFCC), im Gegensatz zur Fourier Transformation mit dem menschlichen Gehör ähnlichen logarithmischen Frequenzbändern (den Mel frequencies) und nicht mit einer linearen Skala gearbeitet. Dies könnte auch mit den Ergebnissen der Fourier Transformation und einer Filterbank mit gewichteten Frequenzbändern und einer Multiplikation dieser mit den transformierten Werten bewerkstelligt werden und würde dem menschlichen Gehör besser nachvollziehbare Ergebnisse liefern.

Die numerische Darstellung der Beobachtungen erfolgte bei Trifa et al. (2008, pp. 1-8) mit MFCC und LPC mit einer Fenstergröße von 25 ms und einer Überlappung von 15 ms, wobei die Werte erst durch Untersuchungen herausgefunden werden mussten. Unterschieden sich die Längen von Lauten zwischen verschiedenen Spezies stark, sollte der Überlappungsparameter erhöht werden. Sollte die Komplexität von Vogellauten in einem anderen Anwendungsfall höher sein sollte, kann die Anzahl an Markov-Zuständen angehoben werden. Im Fall dieses Experiments war eine Steigerung über 15 Zustände aber mit einem starken Abfall der Performance verbunden.

Die Ergebnisse der Studie mit fünf Spezies waren äußerst gut. Lediglich 10 samples wurden pro Spezies für das Training des Klassifikators verwendet und

eine Fehlerrate von 0,5% wurde festgestellt, wenn Dateien der Qualität A-C verwendet wurden. Die Performance wurde leicht schlechter, wenn A-E Dateien für die Berechnungen eingesetzt waren, die Fehlerrate stieg aber nicht merklich an. Die Anzahl an Trainings-Dateien korrespondierte merklich, aber nicht linear mit der Treffsicherheit. So war sie bei 10 Trainings-Dateien durchschnittlich bei 93,04% und bei 100 Trainings-Dateien bei 96,38%.

Da Vogellaute für HMMs als Zeitreihen betrachtet werden, kann die Länge der Laute unterschiedlich sein, ohne die Klassifikationserfolge signifikant zu beeinflussen, da jede einzelne Beobachtung statistisch unabhängig von vorhergehenden ist.

Die Qualität der Ergebnisse mittels LPC wurde allerdings bemängelt. Die in der HTK vorkommende Implementierung ist hauptsächlich für die menschliche Sprache und nicht für Vogellaute ausgelegt, was sich durch non-linearität in Vogellauten und schnelle Übergänge bemerkbar macht. Außerdem arbeitet LPC nicht mit der dominanten Frequenz, welche aber laut Nelson (1989) jedoch eine der effektivsten akustischen Hinweise von Tieren ist, um individuelle Informationen zu transportieren.

Chou et al. (2007, p. 1-4) verwendeten den Viterbi Algorithmus, um mittels HMM die wahrscheinlichste Spezies eines Vogels zu bestimmen. Auch bei diesem Projekt wurde mit Dateien mit 44,1 KHz und 16 Bit gearbeitet. Die Dateien umfassten Vogelgesang und Vogelrufe von 420 Spezies.

Die Merkmalsextraktion geschah mittels 512 samples großen Fenstern mit ¾ Überlappung, die angesetzt wurden, um in diesen Fenstern die FFT anzuwenden. Diese Berechnung in Fenstern bestimmter Größe, resultiert in einem Amplituden- und einem Phasenspektrum. Jedes berechnete Amplitudenspektrum, das durch ein begrenztes Zeit-Frequenz-Diagramm dargestellt wird, ergibt nebeneinandergereiht den Zeit-Frequenz-Verlauf eines Vogellautes, also ein Spektrogramm.

Für jedes Fenster wurde nun aus dem berechneten Spektrum die Hauptfrequenz ermittelt, indem das Frequenzband mit der höchsten Amplitude bestimmt wurde. Dieser Frequenzwert repräsentierte die dominante Wellenlänge des jeweiligen Fensters. Dieser Wert und der ihre zugehörige Amplitude wurden als Merkmalsvektor der einzelnen Fenster verwendet. Anschließend wurde der Mittelpunkt jeder Silbe über die maximalen Amplituden errechnet. Von diesen wurden jeweils vor- und rückwärts gehend die Enden der Silben errechnet. Diese Silben wurden im Anschluss daran segmentiert und gespeichert.

Die Silben wurden daraufhin mittels eines fuzzy c-mean (FCM) Clusteringalgorithmus so geclustert, dass ähnliche Silben für die weitere Berechnung nur einmal vorkamen. Die Ergebnisse des Clusterings wurden anschließend als Beobachtungen für die HMMs mit drei bis fünf Zuständen verwendet, wobei die Beobachtungen durch die dominante Wellenlänge von 0-255 repräsentiert wurden. Jede Silbengruppe hatte nun also eine Abfolge von dominanten Wellenlängen als Zustände. Zum Trainieren der HMM Parameter wurde der sogenannte Baum-Welch Algorithmus eingesetzt, der in (L. Rabiner, 1989, pp. 257–286) näher beschrieben wird. Der Viterbi Algorithmus konnte nun über die dominanten Wellenlängenabfolgen die wahrscheinlichste Spezies berechnen, die diese Abfolge generiert. Nachdem für jede Silbe eine Abfolge erstellt wurde, musste für die Gesamtzahl an Silben und ihren Wahrscheinlichkeiten eine Gewichtung festgelegt werden, um die wahrscheinlichste Spezies zu bestimmen. Dies geschah über die Häufigkeit der Zuordnung einer Silben-Zustandsabfolge zu einer Spezies.

Die Ergebnisse des Projekts ergaben, dass der Einsatz von drei Zuständen im HMM und ein Clustering auf drei Cluster die besten Ergebnisse erzielten. So wurden maximal 80,6% und durchschnittlich 78,3% aller Spezies richtig zugeordnet.

2.4.2 Support Vector Machines

Eine SVM arbeitet mit extrahierten Merkmalsvektoren aus Audiofenstern, um die Klassifikation durchzuführen. Sie kann also nicht mit zeitlichen Daten arbeiten, wie zum Beispiel ein HMM. Als binärer Klassifikator versucht eine SVM eine Hyperebene zu finden, die die Klassen möglichst gut unterscheiden kann. Dafür versucht sie einen größtmöglichen Abstand von der Hyperebene zu den Beispielen der Klassen mit n-dimensionalen Merkmalsvektoren zu erstellen. Da in den meisten Fällen keine lineare Hyperebene gezogen werden kann, versucht eine SVM im n-dimensionalen Merkmalsraum Grenzen zu ziehen, bis eine ausreichende Separierbarkeit gewährleistet ist. Um Overfitting zu verhindern, müssen gegebenenfalls Fehler akzeptiert werden, da die Generalisierbarkeit sonst stark leiden würde. Eine SVM versucht bei diesem Schritt einen Kompromiss zwischen Komplexität und Generalisierbarkeit zu treffen.

Fagerlund, ein Teilnehmer des finnischen AveSound projects mit dem Ziel, Vogellaute zu klassifizieren, hat in (Fagerlund, 2007, p. 6-7) SVMs im Detail getestet. Obwohl SVMs auch in der Lage sind, mehrere Klassen zu unterscheiden, wurde im Fall seiner Arbeit ein binärer Entscheidungsbaum mit mehreren SVMs erstellt, die jeweils nur zwei Klassen vergleichen sollten. Bei jedem Schritt im

Entscheidungsbaum wurde eine von zwei Klassen ausgeschlossen, wodurch am Ende die wahrscheinlichste Klasse einer unbekannten Aufnahme übrig bleibt.

Vorteile von SVMs sind ihre hohe Genauigkeit und ihre außergewöhnlich guten Eigenschaften für die Generalisierbarkeit. Sie basieren auf statistischen Lernmethoden und strukturellen Risiko-Minimierungen. In dieser Arbeit wurde die SVM mit dem sequential minimal optimization Algorithmus trainiert, der in (Platt, 1999) genauer erklärt wird.

Das Datenset wurde bereits für andere Klassifikationsmethoden verwendet und die Ergebnisse dieses Projektes mit den bekannten verglichen, um Schlüsse über die Qualität des Einsatzes einer SVM als Klassifikator zu treffen. Zwei Datensets mit sechs und acht Spezies waren bereits im AveSound project im Einsatz.

Die Einteilung in Elemente, Silben und Phrasen wurde auch hier verwendet, um Vogelgesang zu segmentieren. Die Segmentierung erfolgte mit einem iterativen Algorithmus in der Zeitdomäne, wie er in (Fagerlund, n.d., pp. 26-34) genauer beschrieben ist. Wichtig ist jedoch anzumerken, dass erkannte Silbenelemente, die weniger als 15 Millisekunden Abstand zueinander hatten, insgesamt als eine Silbe dargestellt wurden.

Von den segmentierten Silben wurden MFCC, und verschiedene Audiomerkmale berechnet. Die MFCC wurden mit 6ms großen Fenstern und 50% Überlappung errechnet. Die ersten 12 MFCC, sowie drei weitere Koeffizienten (der energy term, delta und delta-delta Koeffizienten) wurden für die weitere Verarbeitung verwendet. Die Audiomerkmale waren in diesem Fall 11, von denen sieben für eine Kurzzeitbeschreibung der Silben bestimmt waren. Für die Berechnungen wurden die Silben in 256 samples große Fenster mit 50% Überlappung geteilt. Nun wurden Merkmale für jedes Fenster berechnet und der Durchschnitt und die Varianz der Verläufe der Merkmale als weitere Merkmale verwendet.

In Tests mit dem ersten Datenset wurde darauf geachtet, dass nicht dieselben Daten für Training und Klassifikation verwendet wurden. Das zweite Set bestand aus manuell segmentierten Daten, die in Trainings- und Testset aufgeteilt wurden. Die Ergebnisse zeigen, dass in den meisten Fällen der Einsatz von gemischten Merkmalen zu den besten Ergebnissen führte. Diese Merkmale waren die MFCC mit den zusätzlichen delta und delta-delta Koeffizienten. Aus den Ergebnissen ließ sich ableiten, dass SVMs mindestens genauso gute Ergebnisse lieferten, wie in früheren Arbeiten des AveSound projects auf das Datenset angewandte Methoden wie neuronale Netze. Die Ergebnisse der Genauigkeit lagen durchschnittlich etwa bei 90%, maximal sogar bei 100%. Ein großer Vorteil des eingesetzten Entscheidungsbaumes war, dass ohne Vorwissen und mit zufälliger Anordnung

stets dieselben Ergebnisse bei der Klassifikation erreicht werden konnten. Ein mit Vorwissen erstellter Entscheidungsbaum, der die Verwandtschaft der Spezies miteinbezieht, könnte allerdings die Performance verbessern.

Als Teilnehmer des Bird task 2014 arbeiteten Martinez, Silvan, Villarreal, Fuentes, & Meza (2014) ebenfalls mit SVMs. Der erste Arbeitsschritt des Teams war das Heruntertakten der Aufnahmen von 44.100 auf 16.000 Hz. Ein Bandpassfilter von 500-4.500 Hz sollte das Signal auf die relevanten Informationen beschränken. Mit einem Kurzzeit-Energiefilter (engl. short time energy filter) wurden die Silben lokalisiert und herausgeschnitten. Die extrahierten Silben wurden auf 100x100 Werte skaliert, um eine konforme Länge aufzuweisen, siehe (Abbildung 6). Mit einer groben und einer feineren Einstellung wurden zwei Tests durchgeführt, um die Silben zu extrahieren.

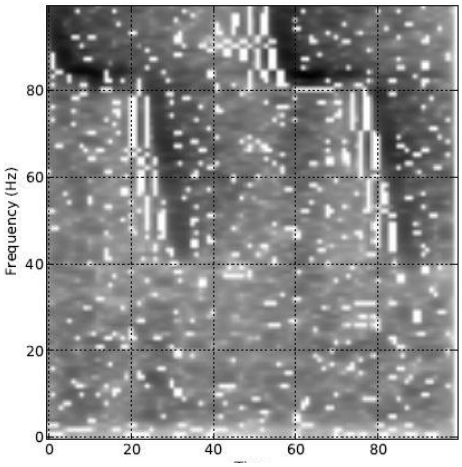


Abbildung 6: Beispiel einer Silbe mit einer normierten Größe von 100x100 Werten

Die SVM arbeitete nun mit den visuellen Merkmalen der Silben, um Vertreter für die Klassen zu finden. Ein Algorithmus für Gesichtserkennung wurde als Klassifikator umgeschrieben und verwendet.

Somit ist die verwendete Methode einer wörterbuchbasierten Klassifikation gleichzusetzen. Der Klassifikator durchsucht die Silben aller Spezies, um herauszufinden, welche Spezies am wahrscheinlichsten ein aktuell unbekanntes Muster generieren würde.

Als finale zwei Tests verwendete das Team 100 beziehungsweise 50 der wahrscheinlichsten Kandidaten für die Klassifikation, die als solche berechnet wurden. Außerdem wurden von Experten Silben ausgewählt, die einzelne Spezies besonders gut repräsentierten. Die Genauigkeit der Tests lag trotzdem bei nur maximal 12,9% beim Test mit 100 Kandidaten. Der andere Test schnitt schlechter ab. Nach Angaben der Teilnehmer war der Grund für die schlechte Klassifikationsrate vorrangig die hohe Anzahl an Spezies, sowohl die eventuell falsche Vermutung, dass die meisten Vogellaute zwischen 500 und 4.500 Hz vorkommen würden. Die Verwendung von Metadaten, die allen Teilnehmern des Bird task zur Verfügung standen, hätte möglicherweise eine positive Auswirkung auf die Genauigkeit haben können.

2.4.3 Wörterbuchbasierte Klassifikation

Bei dieser Art der Klassifikation wird mit sogenannten Wörterbüchern (engl. codebooks) gearbeitet, welche als Repräsentatoren einzelner Spezies fungieren. Aus den Aufnahmen einer Spezies werden die vorkommenden Vogellaute herausgeschnitten und gegebenenfalls geclustert, um dann als Wörterbuch diese Spezies zu repräsentieren.

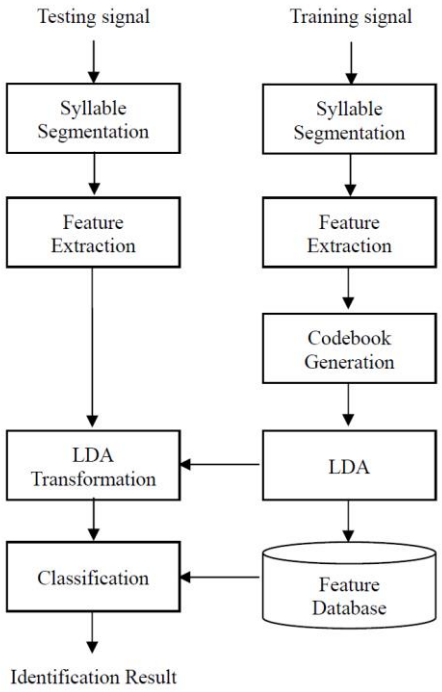


Abbildung 7: Ein Block-Diagramm des verwendeten Vogelgstimmen-Erkennungs-Systems

Chang-Hsing Lee et al. (2006, pp. 1–6) segmentierten die Vogellaute von 420 Spezies in Elemente/Noten, Silben, Motive/Phrasen, Typen und Reihen. Die Audio-Dateien hatten eine Abtastrate von 44,100 Hz und 16 Bit Tiefe. Für jede Silbe wurden die gemittelten Linear Predictive Coding Coefficients (ALPCC) und die gemittelten Mel Frequency Cepstral Coefficients (AMFCC) über alle Frames der Silben als stimmgebendes Merkmal berechnet. Ein Wörterbuch wurde für die extrahierten Ergebnisse jeweils eines Vogels erstellt, um die unterschiedlichen Charakteristiken der verschiedenen Silben zu speichern.

Die Trainingsphase beinhaltete folgende Schritte: die Segmentierung der Silben, die Merkmalsextraktion, die Erstellung des Wörterbuchs und Lineare Diskriminanzanalyse (LDA). Die Erkennungsphase bestand aus der Segmentierung der Silben, der Merkmalsextraktion, der LDA Transformation und der Klassifikation.

Die vorgestellte Art der Segmentierung in diesem Projekt stammt von Harma (2003). Der Vorteil dieser Methode ist die einfache Extraktion vieler Silben aus einer Aufnahme; auch dann, wenn mehrere Vögel gleichzeitig Laute von sich geben.

Für die Merkmalsextraktion wurden die segmentierten Silben in überlappende Fenster geteilt. Von diesen Fenstern wurden anschließend jeweils die LPCCs und MFCCs berechnet und als Merkmalsvektor jedes Fensters gespeichert. Bei diesem Projekt wurde mit jeweils 15 Koeffizienten gearbeitet. Die Ergebnisse aller

Fenster wurden gemittelt, um die ALPCC und die AMFCC für jeweils eine Silbe zu berechnen. Vergleiche mittels Euklidischer Distanz waren durch die Verwendung von cepstral coefficients möglich. In der Trainingsphase wurde außerdem pro Spezies der Durchschnitt von allen Silben als repräsentativer Merkmalsvektor einer Spezies errechnet. Dieser Vektor musste aber linear normalisiert werden, da der dynamische Umfang jedes Merkmalsvektors für die Durchschnittsberechnung variiert und daher nicht im selben Wertebereich lag.

Ein Vorteil dieser Vorgehensweise ist, dass unterschiedlich lange und unterschiedlich viele Silben immer durch einen durchschnittlichen Wert repräsentiert werden und daher die Dauer und Anzahl von Silben irrelevant ist.

Das Wörterbuch einer Spezies beinhaltete alle Merkmalsvektoren der einzelnen Silben, da diese teilweise große Unterschiede untereinander aufwiesen. Eine Clustering Methode, das Progressive Constructive Clustering (PCC), war danach dafür zuständig, die einzelnen Merkmalsvektoren der Silben einer Spezies zu clustern. Von jedem Cluster wurde ein gemittelter Merkmalsvektor errechnet, der die jeweilige Art von Silben repräsentieren sollte.

LDA war dafür zuständig, die Klassifikationsqualität zu erhöhen. LDA versucht verschiedene Klassen nicht zu vergleichen, sondern zu unterscheiden. Das Ziel war es, die Distanz innerhalb von Klassen zu minimieren, die zwischen Klassen aber zu maximieren. Dies geschah mittels einer Transformationsmatrix, die mit dem sogenannten Fisher criterion gebildet wird. Für diesen Schritt werden die Eigenvektoren der Verteilungsmatrix, innerhalb und zwischen den Klassen, herangezogen um eine optimale Transformationsmatrix zu erstellen. Diese Matrix soll das Verhältnis von Streuung zwischen Klassen und innerhalb von Klassen in einem niedrig-dimensionalen Raum maximieren.

In der Testphase wurden auch alle eingehenden Daten in unterschiedliche Silben segmentiert und die ALPCCs und MFCCs davon berechnet. Anschließend wurden die Ergebnisse ebenfalls linear normalisiert und mit der Transformationsmatrix in eine niedrigere Dimensionalität umgerechnet. Nun wurde die Euklidische Distanz zwischen dem Test-Merkmalsvektor und jedem repräsentierenden Merkmalsvektor berechnet.

In den Experimenten wurden jeweils 50% der Daten als Trainings- und Testset verwendet. Das Set von 420 Vogelgesängen wurde einerseits als solches verwendet, andererseits in insgesamt 561 Gesänge mit verschiedenen Aufgaben (Balzgesang o.ä.) beziehungsweise Charakteristika aufgeteilt. Die Ergebnisse von LPCC und MFCC wurden außerdem auf ihre Performance getestet. Auch die Auswirkung der Dimensionsreduktion durch die LDA wurde gemessen. Die

Ergebnisse des ersten Experiments zeigten, dass der Einsatz von AMFCC wesentlich bessere Resultate lieferte, als HMM oder ALPCC. So wurden mit AMFCC bis zu 58% der Spezies erkannt, mit ALPCC aber nur bis zu 36% und mit HMM nur bis zu 28%. Der Einsatz von LDA reduzierte die Anzahl an Merkmalen von 15 auf jeweils 11 und verbesserte die Ergebnisse durchgehend, maximal sogar um 13%.

In einem zweiten Experiment wurden die erstellten Wörterbücher für die Klassifikation der Testdaten herangezogen. Die Berechnungszeit stieg durch den Einsatz mehrerer Merkmalsvektoren an, jedoch waren die Ergebnisse im Vergleich zum ersten Experiment besser. So konnte mittels CMFCC (codebook-based MFCC) eine Genauigkeit von bis zu 84% und mit CLPCC (codebook-based LPCC) bis zu 51% erreicht werden. Nach der Reduktion durch LPA von 15 auf 11 beziehungsweise 12 Merkmale, konnten sogar bis zu 87% für CMFCC und bis zu 60% Genauigkeit für CLPCC erreicht werden.

Aus den Ergebnissen der Experimente identifizieren die Mitarbeiter des Projekts zwei Hauptgründe, die für die falsche Klassifikation verantwortlich waren. Probleme traten nach falscher Silbensegmentierung und starkem Hintergrundrauschen auf. Diese beiden Aspekte wurden bei der Erstellung dieser Arbeit in Abschnitt 5.2 untersucht und verbessert.

Als Teilnehmer des Bird task verwendeten Joly, Champ, & Buisson (2014) ebenfalls einen wörterbuchbasierten Klassifikator. Um Rauschen in den Aufnahmen zu vermindern wurde SoX (“SoX - Sound eXchange | HomePage,” n.d.) verwendet, ein plattformunabhängiges Open Source Kommandozeilen-programm zur Audiobearbeitung. Zuerst wurde das Signal gefiltert, um Rauschen zu entfernen, die dann entstandenen stillen Teile der Aufnahmen zusammengekürzt. Dadurch waren die rauschfreien Signalanteile nahe beieinander. Die Parameter zur Rauschminimierung wurden iterativ bestimmt und so gewählt, dass die Aufnahmen nach der Filterung und Bearbeitung nicht weniger als 20% ihrer Originalgröße hatten. Mit dem Open Source Framework marsyas (“MARSYAS,” n.d.) wurden MFCC mit den Parametern der Bird task Veranstalter berechnet. Die MFCC-Vektoren wurden mittels k-NN Clustering in 30 Cluster geordnet und die einzelnen MFCC Merkmale auf ihre Qualität getestet. Testreihen wurden mit allen Merkmalen mit einer berechneten Gewichtung durch ihre Qualität und nur mit den hochqualitativen Merkmalen durchgeführt (Merkmale mit einer hohen Unterscheidungsfähigkeit). Auch Tests mit reverse k-NN Clustering und einer Kombination beider Methoden wurden durchgeführt, wobei der Einsatz von k-NN Clustering die besten Ergebnisse erzielte. Weitere Tests haben gezeigt, dass der Einsatz der mitgelieferten Metadaten zu den Aufnahmen die

Klassifikationsqualität steigerte. Die Metadaten Ortsbestimmung (engl. geo-location), Höhe (engl. altitude) und Tageszeit (engl. time-of-day) wurden für die Klassifikation eingesetzt.

Für die Klassifikation wurden die Aufnahmen zu ¾ als Trainingsset und ¼ als Testset verwendet. Die Ergebnisse zeigen, dass die Klassifikationsrate bei ausschließlichem Einsatz der Audiomerkmale bei maximal 32,8% lag und bei Verwendung der Metadaten sogar bei 36,5%. Die Rauschverminderung und die Art des Clusterings fanden auch in dieser Arbeit Einfluss.

2.4.4 Neuronale Netze

Neuronale Netze als Klassifikator funktionieren ähnlich neuronaler Verkettungen im menschlichen Gehirn. Knotenpunkte stimulieren benachbarte Knotenpunkte, um einen Impuls weiterzugeben, der verarbeitet wird. Bei neuronalen Netzen gibt es eine Eingabeschicht, eine versteckte Schicht und eine Ausgabeschicht. Daten, die am Eingang ankommen, werden auf die einzelnen Knotenpunkte verteilt. In der Ausgabeschicht wird definiert, welche Ergebnisse durch das neuronale Netz berechnet werden sollen. Jeder Knoten aller drei Schichten hat anfangs eine zufällige Gewichtung zu angrenzenden Punkten. Diese Gewichtung bestimmt, welche Daten von welchem Knoten angenommen werden. So erhält zum Beispiel ein Knoten in der versteckten Schicht die gewichteten Werte von zwei Knoten aus der Eingabeschicht und errechnet seinen eigenen Wert. Dieser Vorgang wird für jeden Knoten in der versteckten Schicht ausgeführt, wobei in diesem eine definierte Anzahl an Knoten existiert (in vertikaler und horizontaler Richtung). Nach der Berechnung der versteckten Knoten werden die Knoten in der Ausgabeschicht berechnet. Folgend wird verglichen, wie stark die errechneten Ergebnisse von den angegebenen abweichen und die Verbindungsgewichtungen werden rücklaufend neu definiert. Dieser Prozess wird nun so oft wiederholt, bis die Gewichtungen der versteckten Knoten die gewünschten Ergebnisse erzielen. Dieser Prozess ist gut nachvollziehbar, leider aber äußerst langsam und rechenintensiv.

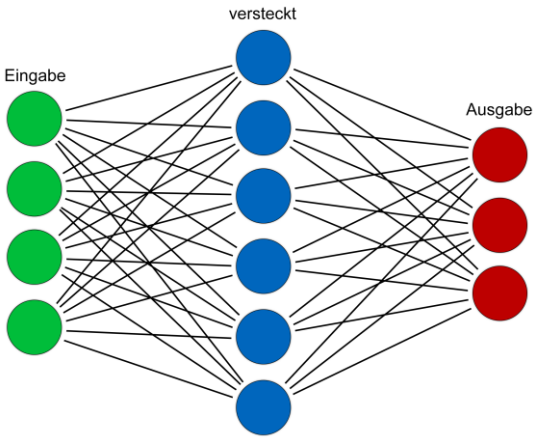


Abbildung 8: schematische Darstellung eines neuronalen Netzes

Als Teilnehmer des Bird task 2014 arbeitete ein Team aus den Niederlanden mit neuronalen Netzen (Koops, Balen, & Wiering, 2014). Die verwendeten Daten waren ebenfalls mit 44.100 Hertz abgetastet und 16 Bit tief. Die Segmentierung der Daten erfolgte durch die Annahme, dass die lautesten Anteile einer Aufnahme die Vogellaute beinhalteten. Die Aufnahmen wurden durch einen Faktor vier heruntergetaktet, wodurch sie auf 11.025 Hertz Abtastrate gebracht wurden, was das Spektrum bei etwa 5.500 Hertz abschnitt. Dieser Schritt sollte die irrelevanten Daten im Hochtonbereich des Spektrums eliminieren, da dort keine relevanten Daten für die Vogelstimmenerkennung vorkommen. Das abgeschnittene Spektrum wurde in weiterer Folge bei 1.000 Hertz hochpassgefiltert, um tieffrequente Störgeräusche zu entfernen. Vom nun stark gefilterten Signal wurde die dominante Frequenz des Signals durch die maximale Amplitude errechnet. Unterhalb dieser Frequenz wurden mit einem weiteren Filter tieffrequente Signalanteile weggefiltert. Nun wurden von den lautesten Signalanteilen ausgehend die zeitlichen Grenzen durch eine Schwellwertberechnung gesucht. Der Wert für den Schwellenwert wurde empirisch auf 17 Dezibel festgelegt. Diese Berechnung resultierte in N Segmenten mit definierten Start- und Endpunkten. Kurze gefundene Segmente wurden anschließend verbunden, um längere Abschnitte zu formen. Dies wurde mit einem Clusteringalgorithmus vorgenommen, der anhand der Abstände zwischen den Lauten clusterte. Durch Tests ergab sich, dass geclusterte Segmente von 800 ms Länge optimal waren.

Von diesen Segmenten wurden anschließend MFCC berechnet. Drei Methoden zur aggregierten Darstellung der Wörter mit unterschiedlicher Anzahl von MFCC waren folgende: Methode eins verwendete die Mittelwerte der MFCC der einzelnen Segmente, Methode zwei enthielt Mittelwert und Varianz der MFCC und in Methode drei verwendete Mittelwert, Varianz, sowie die 1. und 2. Ableitung, und die Mittelwerte von Varianz, 1. und 2. Ableitung. Durchschnittlich wurden 4,83 Segmente pro Aufnahme errechnet, was in 46.799 Segmenten pro Methode resultierte. Diese wurden zufällig in 75% Trainingsdaten und 25% Testdaten aufgeteilt und für die neuronalen Netze verwendet. Durch Tests wurde definiert, dass ein Training mit 250 zufällig gewählten Segmenten am besten funktionierte. Für jede der drei Methoden wurde ein neuronales Netz trainiert, wobei in zwei Tests überprüft wurde, ob versteckte Schichten, die kleiner waren als die Eingabeschicht, bessere Ergebnisse erzielten als größere. Die versteckten Schichten waren in zwei gleich große Bereiche aufgeteilt, außer bei der dritten Methode, in der drei Schichten vorhanden waren. Die Ausgabeschicht war 501 Knoten groß, da 501 Spezies im Datensatz vorhanden waren.

Die Klassenzuordnung einer Aufnahme wurde durch gewichtete Ergebnisse der einzelnen Segmente erzielt. Die besten Ergebnisse erzielten die

Zusammenstellungen 240-350-350-501 (Eingabeschicht - versteckte Schicht - versteckte Schicht - Ausgabeschicht) mit 73% Genauigkeit, 96-64-64-501 mit 60% Genauigkeit und 48-40-40-501 mit etwa 23% Genauigkeit. Die Klassifikationsrate nahm mit zunehmender Knotenanzahl stark ab. Die Teilnehmer gaben an, dass Overfitting bei der Klassifikation die Genauigkeit verminderte und eine zufälligere Aufteilung in Trainings- und Testset die Klassifikationsrate noch steigern könnte.

Die Behauptung der Teilnehmer, dass Vogellaute im Bereich von 1.000 bis 6.000 Hertz vorkommen, stützt die Art und Weise der Vorverarbeitung der selbstentwickelten Methode dieser Arbeit.

3 Methode

3.1 Lösungsansatz und Überblick

Das Ziel dieser Arbeit war, bekannte und gut funktionierende Methoden zur Klassifikation von Vogelstimmen mit einer viel pragmatischeren und einfacheren zu vergleichen. Die Vorteile von instanzbasiertem Klassifizieren sind unter anderem, dass weniger Vorwissen über das Datenset notwendig ist, und dass bei diversen Extraktions- und Verarbeitungsmethoden weniger Parameter definiert werden müssen, die Erfahrung und Wissen voraussetzen. Die Wahl der Parameter beeinflusst die Klassifikationsrate deutlich und erfordert immer noch eine Menge an manueller Arbeit. Ein weiterer Vorteil ist, dass andere Klassifikationsmethoden eine wesentlich höhere Anzahl an Trainingsdaten voraussetzen, instanzbasiertes Lernen aber schon mit wenigen Exemplaren und deren definierter Klassenzugehörigkeit funktioniert.

Aus den Ergebnissen sollen Schlüsse über die Berechtigung von Klassifikation durch instanzbasiertes Lernen gezogen werden oder andererseits die bekannten Methoden dadurch gestützt werden, dass diese höhere Klassifikationsergebnisse erzielen. Die Vorgangsweise, wirklich auf fast alle gängigen Algorithmen zu verzichten, war nicht dadurch bedingt, dass das Wissen zur Verwendung dieser fehlte. Die meisten dieser Methoden sind fertig implementierte Funktionen, die nur eingebunden werden müssen. Vielmehr war es der Test auf die Notwendigkeit komplexerer Methoden und die eigene Fähigkeit einen großen Datensatz zu klassifizieren.

Im Folgenden wird die entwickelte und eingesetzte Methode beschrieben und erklärt, um dem Leser einen Einblick in den Entwicklungsprozess zu gewähren. In Abschnitt 3.2 werden die Dateistruktur und die Erstellung des Trainingssets erklärt. Abschnitt 3.3 beschreibt die Erzeugung, Filterung und Bearbeitung der Spektrogramme jeder einzelnen Aufnahme. In 3.4 wird der Prozess der Lokalisierung von Maxima im Audiosignal beschrieben. Abschnitt 3.5 erklärt die Erstellung der Wörterbücher als Repräsentanten jeder Spezies. In 3.6 wird die Art und Verwendung des Clusterings beschrieben, das die Menge der repräsentativen Daten reduziert. Abschnitt 3.7 erklärt schließlich die Klassifikation und Testreihen, die zur Genauigkeits- und Performancemessung durchgeführt wurden.

3.2 Verarbeitung und Sortierung des Trainingssets

Im Testset des Wettbewerbs befanden sich 9.688 Aufnahmen mit der Benennung LIFECLEF2014_BIRDAMAZON_XC_WAV_RNXXXXX, wobei XXXXX für die fortlaufende Nummer der jeweiligen Aufnahme steht. Jeder Audio-Datei lag eine XML Datei mit den eingetragenen Metadaten bei.

Am Anfang der Entwicklung der Methode wurde eine Matlab Funktion erstellt, die nach dem Einlesen des Verzeichnisses, in dem alle Dateien des Trainingssets lagen, aus den beiliegenden XML Dateien die Spezies des Vogels herauslas und die Vögel dann in Ordner sortiert auf der Festplatte ablegte.

Nach der Sortierung aller Dateien des Trainingssets, begann die Entwicklung der Klassifikationsmethode. Der erste Schritt war es nun, aus den vorhandenen Audio-Dateien gute Merkmale zu finden und zu extrahieren. Für jede Spezies standen mehrere Dateien zur Verfügung, die nacheinander abgearbeitet wurden.

3.3 Erzeugung des Spektrogramms

Da die Wellenform einer Aufnahme keinen Aufschluss über Position und Frequenzverteilung der Vogellaute bietet, wurden Spektrogramme der Aufnahmen erstellt. Diese Darstellung ist für die weitere Verarbeitung der Signale von Vorteil, da die Audiodaten in dieser Form besser verständlich sind. Für die Erzeugung wurden die jeweiligen Aufnahmen eingelesen. Diese waren mit 44.100 Hertz abgetastet und hatten eine Tiefe von 16 Bit. Aus den Daten wurde mit der in der Signal Processing Toolbox enthaltenen Funktion spectrogram() ein Spektrogramm erstellt. Ein solches ist in (Abbildung 9) zu sehen. Das Spektrogramm wurde als nächstes logarithmiert, damit die Werte besser vergleichbar wurden.

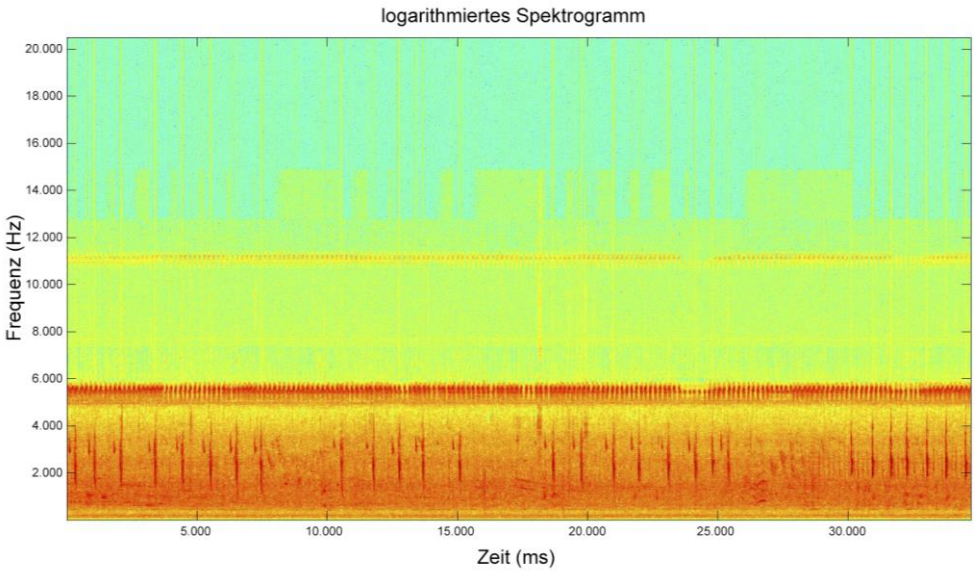


Abbildung 9: logarithmiertes Spektrogramm einer Klasse

Nachfolgend wurde das Spektrogramm bei 8.000 Hz abgeschnitten, da darüber keine relevanten Daten für die Klassifikation ersichtlich waren. Das Ergebnis ist in (Abbildung 10) zu sehen.

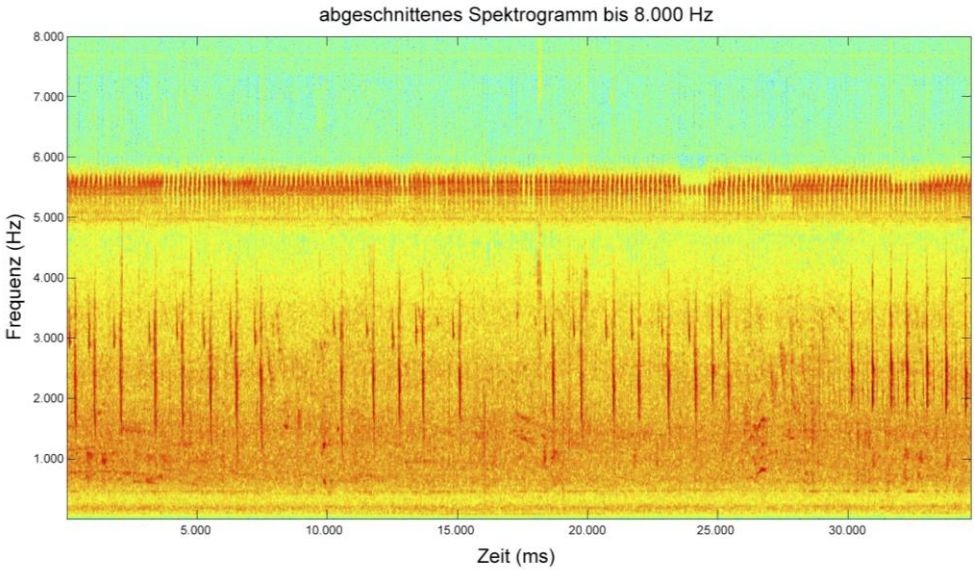


Abbildung 10: abgeschnittenes Spektrogramm bis 8.000 Hz

Danach wurde für jede Zeile des Spektrogramms der Durchschnitt berechnet und abgezogen, um eventuell auftretendes Rauschen zu reduzieren. Das Ergebnis ist in (Abbildung 11) zu sehen. Es ist sofort sichtbar, welche Qualitätsverbesserung

dieser simple Rechenschritt bringt, da die Vogellaute (rote Stellen) deutlich besser freigestellt sind und das Rauschen im Bereich um 5.700 Hertz entfernt wird.

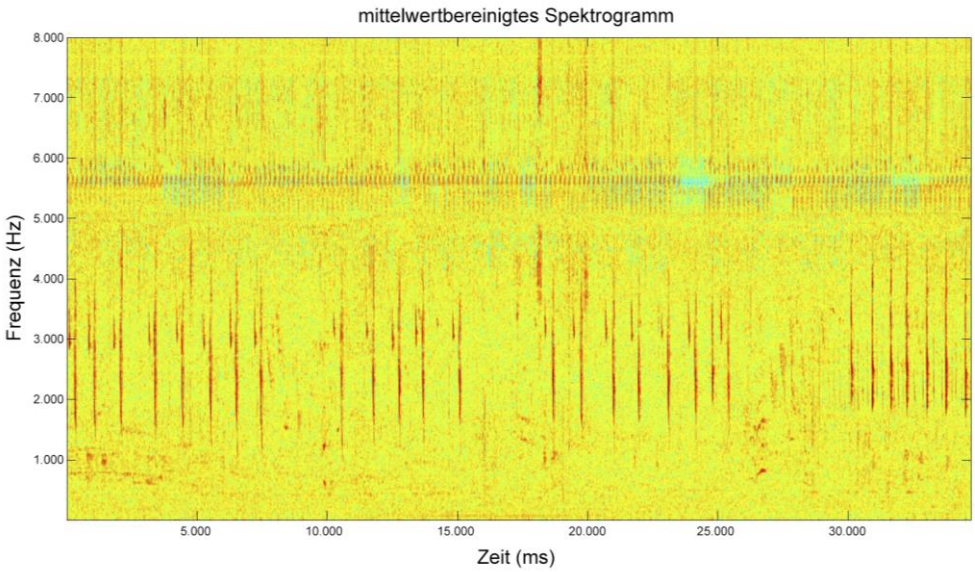


Abbildung 11: Mittelwertbereinigtes Spektrogramm

3.4 Erkennung lokaler Maxima

Als nächster Schritt sollte nun herausgefunden werden, an welchen Positionen sich lokale Maxima, also Amplitudenspitzen, im Spektrogramm befinden, da sich an diesen die lautesten Signalanteile und somit potentiell Vogellaute befinden. Dafür wurde die Funktion findpeaks() verwendet. Die auf Maxima zu analysierenden Daten entsprachen einem einfachen Funktionsgraphen (also der Summe aller Spaltenwerte des Spektrogramms), der in (Abbildung 12) mit den eingezeichneten erkannten lokalen Maxima ersichtlich ist.

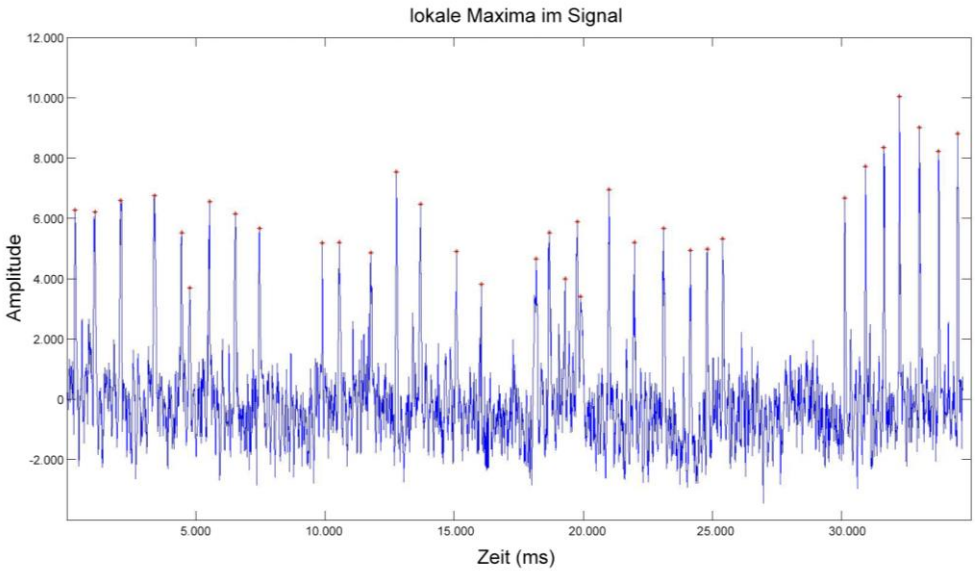


Abbildung 12: automatisch erkannte und markierte Positionen im Signal mit höchster Gesamtenergie pro Frame

Die Maxima sind in (Abbildung 12) als rote Sterne dargestellt. Die Höhe der Maxima beschreibt verhältnismäßig ihre Amplitude, wie in (Abbildung 11) die Rotintensität.

In (Abbildung 13) wurden Maxima und Spektrogramm in einer Grafik dargestellt. Um die Maxima auf einen darstellbaren Wertebereich zu bringen, wurden sie auf 1/100 ihrer Höhe skaliert.

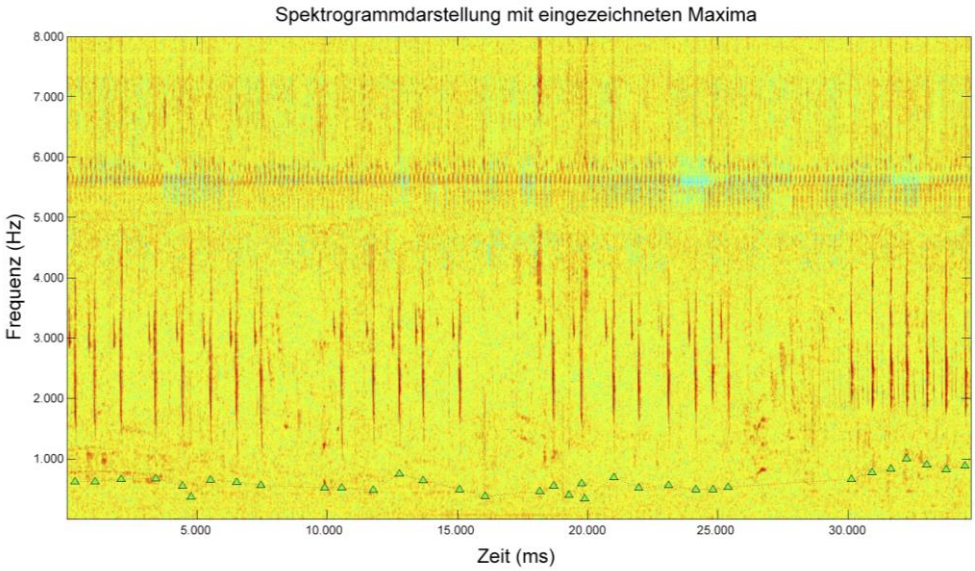


Abbildung 13: Spektrogrammdarstellung mit eingezeichneten Maxima

Die Maxima sind in (Abbildung 13) als grüne Dreiecke dargestellt. Es ist gut ersichtlich, dass die Erkennung der lokalen Maxima, also die einzelnen Laute des Vogels, gut funktionierte.

3.5 Generierung von Wörterbüchern

Um einen Vogel repräsentativ durch einige wenige Laute darstellen zu können, wurden zuerst alle gefundenen Laute jeder Aufnahmen einer Spezies gespeichert. Dafür wurden an den erkannten lokalen Maxima Fenster angesetzt, die jeweils einen Ausschnitt des Spektrogramms ausschnitten. Der Sinn dahinter ist es, eine Spezies durch einige wenige Laute darstellen zu können, um die generierten Daten gering zu halten und die Rechenzeit bei der Klassifikation zu verkürzen, da nicht alle gefunden Lauten verglichen werden müssen.

Die Extraktion der Wörter aus dem Spektrogramm brachte einige Probleme mit sich. Da verschiedene Positionen der Fenster nicht funktionieren würden, musste zwischen verschiedenen Fällen unterschieden werden: Fall eins und auch der häufigste, das Wort befand sich nicht an den Rändern des Spektrogramms. Fall zwei, das Wort war weiter hinten positioniert, als es die Fensterbreite zulassen würde, da das Ende des Spektrogramms erreicht war. In diesem Fall wurde das Wort an der letztmöglichen Position aus dem Spektrogramm ausgeschnitten. Die Möglichkeit, das Fenster zu verkleinern, um dieses Problem zu verhindern hatte den Nachteil, dass die Homogenität der Fenstergrößen damit nicht mehr übereinstimmen würde. Fall drei traf zu, wenn das Wort zu früh im Spektrogramm saß, als es die Fensterbreite zulassen würde. In diesem Fall wurde das Fenster an den Beginn des Spektrogramms gesetzt.

Das Wörterbuch wurde abschließend unter dem generierten Dateinamen für den weiteren Verlauf der Hauptfunktion gespeichert. In (Abbildung 14) ist eine grafische Repräsentation eines solchen Wörterbuchs zu sehen. Die Wörter sind hier durch Nullen voneinander getrennt und dargestellt.

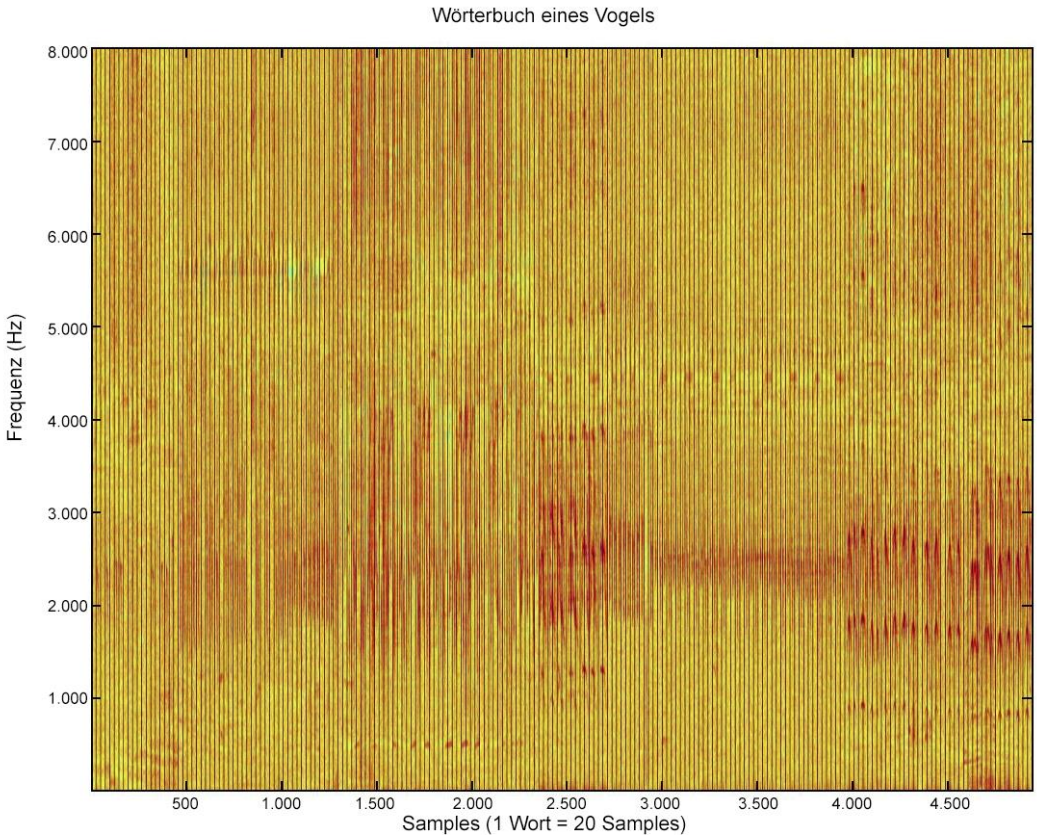


Abbildung 14: Wörterbuch eines Vogels

3.6 Clustering

Da die Anzahl an Wörtern für jede Aufnahme und jede Spezies variabel war, wurden die Wörter anschließend einerseits zur Datenreduktion und andererseits aus Redundanzgründen geclustert. Da viele erkannte Wörter sehr ähnlich waren, konnten die Wörter durch das ihnen zugeordnete Clusterzentrum dargestellt werden.

Die Wörter, die bis hier als Ausschnitte des Spektrogramms gespeichert waren, wurden hierfür als Vektoren gespeichert, indem jede Spalte hinter die vorige gehängt wurde, da der k- Means Clustering Algorithmus nur mit Vektoren arbeiten kann.

Nach der Vektorisierung wurde die entstandene Matrix aus Wortvektoren noch um 90° gedreht, um die Vektoren für den Einsatz des k-Means Clusterings vorzubereiten.

Das Clustering wurde mit 4, 6 und 10 Clustern durchgeführt, die für unterschiedliche Testreihen bei der Klassifikation verwendet wurden.

Die in die Statistics Toolbox integrierte Funktion kmeans() clusterte die Wörter der Clustering-Matrix in die jeweils angegebene Anzahl an Clustern, sodass ähnliche Worte gruppiert wurden. Für jedes entstandene Clusterzentrum wurden für spätere Merkmalsvergleiche die Varianz mittels der Funktion var() und der Mittelwert jeder Zeile mittels mean() berechnet und gespeichert.

In (Abbildung 15) werden die Ergebnisse des Clusterings gezeigt. Jedes Clusterzentrum, das dem Mittelwert aller zugeordneten Wörter entspricht, hatte dementsprechend eine Länge von 20 ms und stellte je nach Testreihe 4.000-12.000 Hertz dar. Die berechneten Zentren und Merkmale aller Vögel wurden im nächsten Schritt für die Klassifikation unbekannter Vögel verwendet.

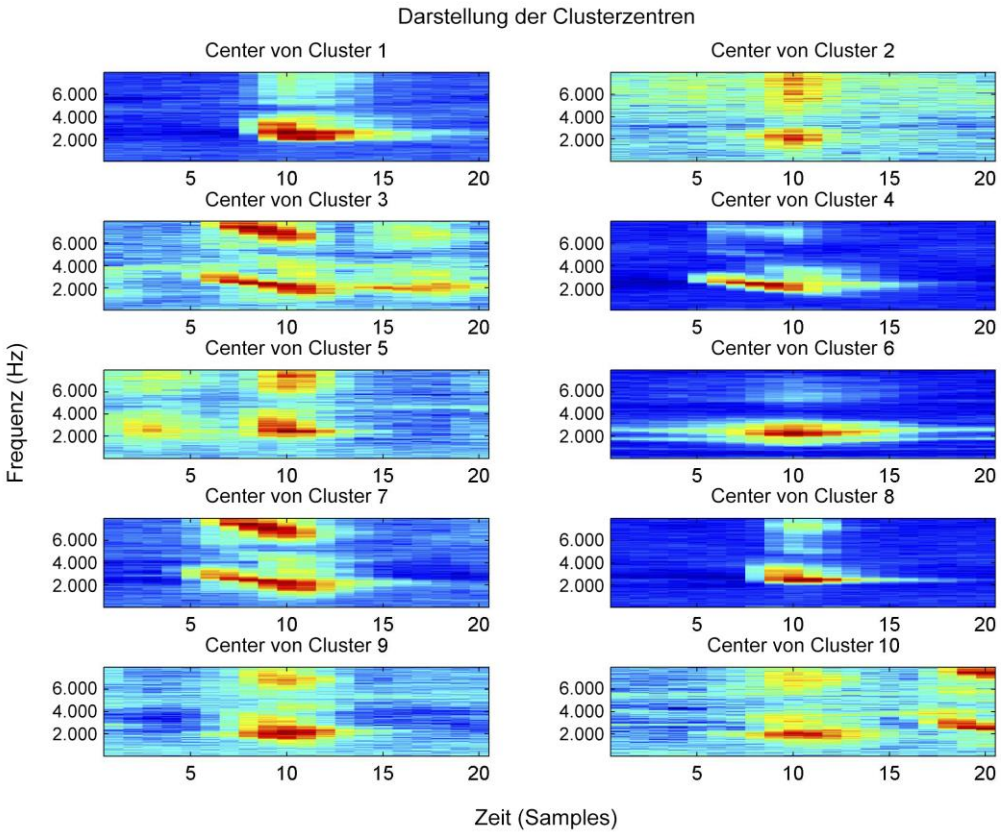


Abbildung 15: Darstellung der Clusterzentren und damit der Laute einer Spezies

3.7 Klassifikation

Um nun einen unbekannten Vogel klassifizieren zu können, mussten für alle Aufnahmen des Testsets die gleichen Berechnungen durchgeführt werden wie für die Trainingsdaten. Die einzige Ausnahme war das Clustering, das für die Testdaten nicht durchgeführt wurde.

Die Klassifikation funktionierte durch Ähnlichkeitsberechnungen. In verschiedenen Experimenten wurden die Test- und Trainingsdaten durch drei unterschiedliche Merkmale dargestellt und mittels Euklidischer Distanz verglichen. Erstens wurden die Spektrogramme selbst miteinander verglichen, welche als 2-dimensionale Bilder gesehen werden können. Versuch 2 verglich die Ähnlichkeit der Varianzvektoren. Versuch 3 verglich die Ähnlichkeit der Mittelwertvektoren und Versuch 4 die Ähnlichkeit eines kombinierten Vektors aus Varianz und Mittelwert.

Diese Experimente wurden durchgeführt, um die verschiedenen Merkmale auf ihre Klassifikationsrate hin zu vergleichen. Die Klassifikation funktionierte dadurch, dass jedes erkannte Wort der Testaufnahme mit jedem Clusterzentrum der Trainingsdaten über Spektrogramm, Varianz- und/oder Mittelwertvektoren miteinander verglichen und die geringsten Distanzen gespeichert wurden. Die geringste Distanz über alle Clusterzentren aller Spezies bestimmte dann die Klasse, die dem Wort am nächsten und somit am wahrscheinlichsten war. Ein weiterer Test analysierte die Berechtigung einer anderen Zuordnung einer unbekannten Spezies. Über Mehrheitsbestimmung aller Wörter der Testdatei wurde die häufigste nächste und damit wahrscheinlichste Klasse bestimmt.

4 Experimente

4.1 Der Bird Task Datensatz

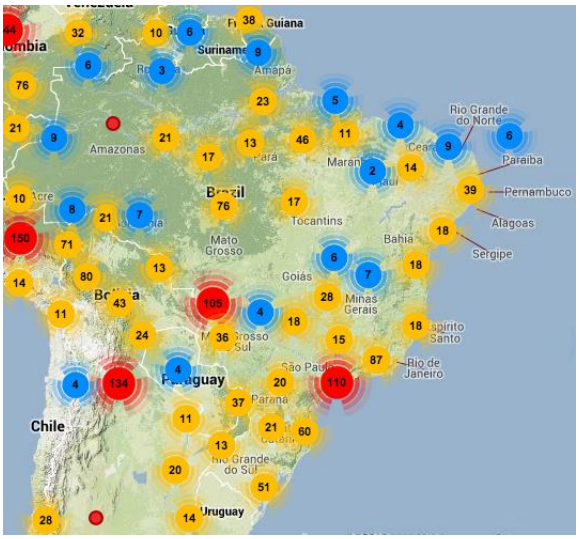


Abbildung 16: Verteilung der verwendeten Spezies des Datensatzes in Südamerika. Quelle: ("Bird task | ImageCLEF - Image Retrieval in CLEF," n.d.)

Der Datensatz des Bird task 2014 beinhaltet über 500 Spezies von südamerikanischen Vögeln rund um Brasilien. Die Aufnahmen wurden aus der Xeno-Canto Datenbank entnommen, die im Dezember 2014 197.468 Aufnahmen beinhaltete ("xeno-canto :: Vogelstimmen aus aller Welt teilen," n.d.). In den Datensets befanden sich Audio-Dateien von Aufnahmen jeder Spezies (mindestens 15, maximal 91) und XML Dateien, die zusätzliche Informationen über jede Aufnahme enthielten (Spezies des Vogels, Art des Lautes [Ruf, Gesang, Alarm, Flucht, etc.], Spezies von Vögeln die im Hintergrund der Aufnahme vorkamen, Uhrzeit und Ort der Aufnahme, Kommentare, u.a.). Über 14.000 Aufnahmen wurden für die Teilnehmer des Wettbewerbs bereitgestellt, um unterschiedliche Herangehensweisen zu testen und deren Ergebnisse zu präsentieren.

Die Audio-Dateien wurden alle normalisiert, um Pegelunterschiede nicht in die Klassifikation einfließen zu lassen und mit 44.100 Hz abgetastet. Die Bitrate waren 16 Bit. Für Vogellaute spezialisierte MFCC wurden von jeder Datei berechnet und den Teilnehmern des Wettbewerbs zur Verfügung gestellt. Dabei handelte es sich um 16 MFCC, die mit 11,6 ms großen Fenstern, alle 3,9 ms, also etwa 70% Überlappung, berechnet wurden.

4.2 Beschreibung der Daten

Die erhaltenen Daten wurden wie in 3.2 beschrieben vorsortiert und geordnet. Aus den gesamten Daten wurden aus Performancegründen für diese Arbeit die ersten 10 Spezies für die Klassifikation verwendet. (Tabelle 1) zeigt die Verteilung der Dateien in Test- und Trainingsset.

Tabelle 1: Verteilung der Aufnahmen in Test- und Trainingsset

Spezies	**Anzahl der Aufnahmen im Trainingsset**	**Anzahl der Aufnahmen im Testset**	**Anzahl an Lauten**	**p(Klasse)**	**p(Klasse)^2**
abeillei	7	7	~310	9,09%	0,83%
accipitrinus	7	6	~610	7,79%	0,61%
acer	8	7	~205	9,09%	0,83%
acutirostris	6	5	~380	6,49%	0,42%
aedon	10	10	~250	12,99%	1,69%
aestiva	9	8	~190	10,39%	1,08%
aethiops	10	10	~535	12,99%	1,69%
albescens	8	7	~360	9,09%	0,83%
albicollis	10	10	~510	12,99%	1,69%
albilora	8	7	~375	9,09%	0,83%
Zufallswahrscheinlichkeit					**10,47%**

Die Wahrscheinlichkeit, dass eine Aufnahme Mitglied einer Klasse ist, ist in Spalte vier sichtbar. Aus allen Wahrscheinlichkeiten folgend ist die gemittelte Zufallswahrscheinlichkeit jeder Klasse etwa 10,5%. Dieses Ergebnis würde ein zufälliger Klassifikator erzielen und ist der auf jeden Fall zu überbietende Wert einer durchgeführten selbstentwickelten Klassifikation.

4.3 Setup des Experiments

In verschiedenen Versuchen wurden die Distanzen zwischen einer unbekannten Testaufnahme und allen Wörterbüchern berechnet und gespeichert. die Klassifikation wurde auf 2 Arten vorgenommen: erstens durch die Vermutung, dass der ähnlichste Vektor des Trainingssets auch derselben Klasse der Testaufnahme angehört. Dieser Ansatz entspricht der Nearest Neighbor Regel, die die Distanzen zu vorkommenden Datenpunkten analysiert. Für die zweite Art wurde über Mehrheitsbestimmung die Klasse für die Testaufnahme gewählt, die am häufigsten die geringsten Distanzen hatte. Der zweite Ansatz ähnelt der K-NN Klassifikation, wobei hier K die Anzahlt der Testlaute darstellt.

4.4 Parameter der Verarbeitungsschritte

In diesem Abschnitt werden verschiedene Parameter der Methode erklärt und beschrieben, um ein besseres Verständnis über die Wahl der Funktionen und deren Parameter zu ermöglichen.

```
[S,F,T,P] = spectrogram(wav,1764,1323,2*2048,44100)
```

Die Funktion spectrogram() erzeugt wie bereits erwähnt ein Spektrogramm, also eine grafische Darstellung der Frequenzverteilung einer Aufnahme. Die Übergabeparameter dieser Funktion waren: an erster Stelle die Audiodaten, die in der Variable wav gespeichert waren. Der zweite Übergabeparameter war die Fenstergröße, die für das Hammingfenster verwendet wurde. Als dritter Parameter war die Größe des Überlappungsbereiches der Fenster festgelegt und an vierter Stelle war die Länge der FFT. Zu guter Letzt stand die Abtastrate der Audio-Daten. Die Parameter sind in samples angegeben und können über die Abtastrate in Millisekunden umgerechnet werden. Das Hammingfenster für die Spektrogrammberechnung war 1.764 Frames groß. Dies entspricht einer Länge von 40 ms (44.100 / 1.000 * 40 = 1.764). Der Überlappungsbereich wurde auf 30 ms und die Länge der FFT auf 4096 Werte festgelegt. Die Parameter für Fenstergröße, Überlappung und Länge der FFT wurden iterativ festgelegt, um einen guten Kompromiss zwischen Zeit- und Frequenzauflösung zu gewährleisten. Die Samplerate war 44.100 Hertz.

```
[pks,locs]=findpeaks(summeSpalten,'MINPEAKHEIGHT',2*abs(std(summeSpalt
en)),'MINPEAKDISTANCE',10);
```

Für das Finden der lokalen Maxima wurde die Funktion findpeaks() aus dem Signal Processing Toolkit verwendet. Die Übergabewerte hierbei waren die zu analysierenden Daten und optionale Einstellungen. In diesem Fall mussten alle vorkommenden Maxima über einem definierten Wert liegen, der zwei Mal dem Absolutwert der Standardabweichung des Spektrogramms entsprach. Auch dieser Wert wurde iterativ gefunden und lieferte auf allen Daten gute Ergebnisse. Der zweite optionale Parameter bewirkt, dass zwei gefundene Maxima mindestens 10 frames auseinanderliegen mussten, wobei die gefundenen Maxima in absteigender Reihenfolge ausgeschlossen wurden. Wenn also zwei Maxima innerhalb von 10 Frames gefunden wurden, wurde der höhere gewählt und der niedrigere verworfen. Dies war notwendig, da einzelne Laute durch mehrere Maxima repräsentiert wurden, was zu einer zu hohen Zahl an gefundenen Maxima geführt hätte.

```
woerterbuchErzeugen(spectrogrammBreite,20,durchschnAbgezogenesSpec,loc
s);
```

Die Erzeugung der Wörterbücher funktionierte durch eine selbstentwickelte Methode. Die Übergabewerte des Funktionsaufrufs waren zuerst der Name des Vogels zur Speicherung, dann die Breite des Spektrogramms, die anfangs in einer Variable gespeichert wurden, die Breite der Wortfenster in samples, das Spektrogramm selbst und die Positionen der gefundenen Maxima. Die Funktion erstellte damit selbstständig Wörterbücher zur Repräsentation der Vogellaute.

```
kMeansClustering(clusteringMatrix,k(l),woerterbuch20,vogelname);
```

Das Clustering geschah auch mittels einer selbstentwickelten Funktion, welche die in der Statistics Toolbox enthaltene Funktion kmeans() verwendete. Die Übergabeparameter waren die generierte Clustering-Matrix, die Anzahl an Clustern, das Wörterbuch für die Wortlänge, welche für die Darstellung der Cluster innerhalb der Funktion verwendet wurde, und der Vogelnamen für die Speicherung der Daten.

4.5 Testreihen

Die Testreihen, die durchgeführt wurden, variierten in verschiedenen Aspekten. Die drei Änderungen waren die Anzahl der Cluster, mit denen verglichen wurde, die Art der Klassenzuordnung einer Aufnahme des Testsets und der Frequenzbereich, der für die Verarbeitung verwendet wurde. Die Anzahl der Clusterzentren war in drei Versuchsreihen 10, 6 und 4 und die Klassenzuordnung funktionierte beim ersten Teil der Testreihen durch den jeweils am nächsten gefundenen Vektor aus dem Trainingsset (Nearest Neighbor). Beim zweiten Teil wurde der Vogel mit der Spezies klassifiziert, die die meisten nahen Vektoren hatte, also deren Klassennummer bei der Distanzbestimmung der einzelnen Wörter der Aufnahme des Testsets am häufigsten aufschien (K-Nearest Neighbor). Die Frequenzbereiche variierten zwischen 0 und 12.000 Hz in unterschiedlichen Ausschnitten, wie in (Tabelle 3) beschrieben wird.

5 Ergebnisse

5.1 Qualitative Ergebnisse

In diesem Abschnitt werden qualitative Ergebnisse präsentiert, die für die Verwendung und Parameterwahl der eingesetzten Funktionen sprechen soll.

Die spektrale Darstellung der Energieverteilung im Spektrum eines Signals als Spektrogramm muss immer an die Anforderungen der Verwendung angepasst werden. Der sogenannte Zeit-Frequenz-Kompromiss (engl. time-frequency-tradeoff) besagt, dass bei einer guten Frequenzauflösung die zeitliche Auflösung leidet und umgekehrt. Dies ist in (Abbildung 17) und (Abbildung 18) veranschaulicht.

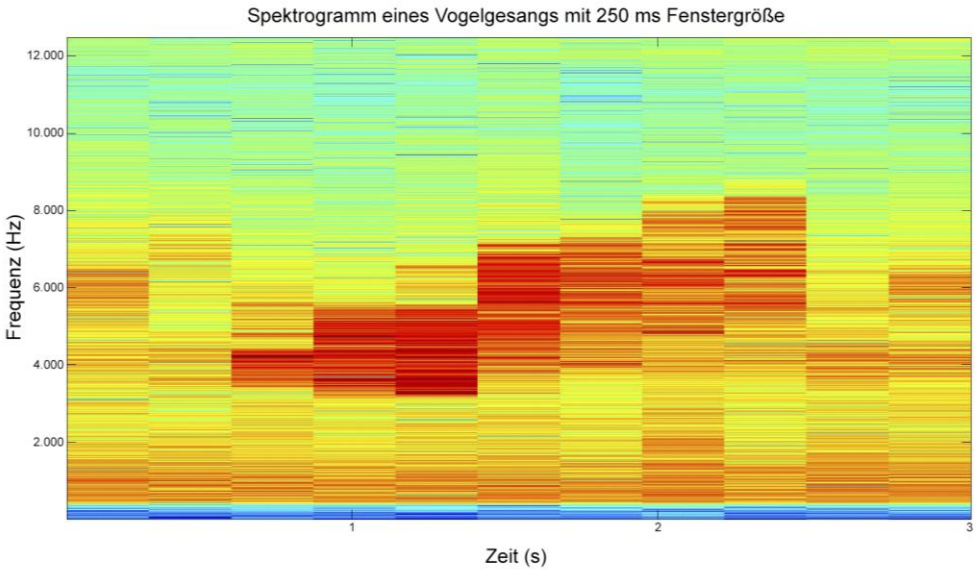


Abbildung 17: Spektrogrammdarstellung mit guter Frequenzauflösung

Die Auflösung im Frequenzbereich ist in (Abbildung 17) sehr gut. Auch die Fenstergröße der Berechnung kann sehr genau erkannt werden. Die Parameter sind hierbei eine Fenstergröße von 250 ms und eine Überlappung von etwa 2 ms. Im Vergleich dazu ist die Fenstergröße in (Abbildung 18) nur etwa 2,5 ms, die Überlappung wie im ersten Fall etwa 2 ms. Das Ergebnis ist eine sehr gute zeitliche Auflösung, es ist aber ebenfalls gut ersichtlich, dass dabei die Frequenzauflösung verschwimmt und sehr ungenau wird.

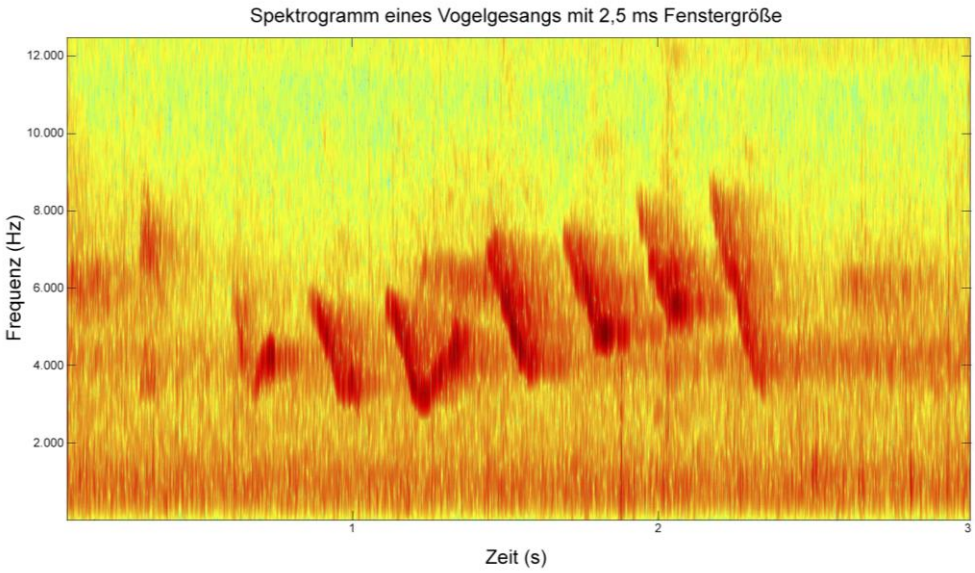


Abbildung 18: Spektrogrammdarstellung mit guter zeitlicher Auflösung

Durch iterative Verbesserungen wurde in dieser Arbeit mit 40 ms Fenstergröße und 30 ms Überlappung gearbeitet, was einen guten Kompromiss aus Rechendauer, zeitlicher und Frequenzauflösung gewährleistet. Das Ergebnis ist in (Abbildung 19) zu sehen.

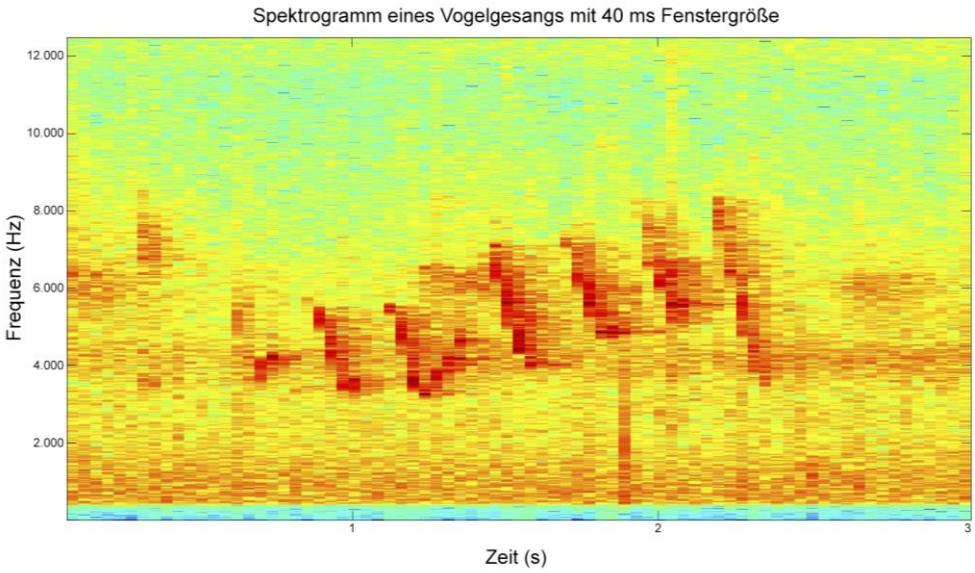


Abbildung 19: Spektrogrammdarstellung mit Kompromiss zwischen Zeit- und Frequenzauflösung

Ein weiterer sehr wichtiger Aspekt dieser Arbeit war die Erkennung der Vogellaute und die Abgrenzung zu Hintergrundgeräuschen. Die Funktion findpeaks() lieferte

fast immer sehr nachvollziehbare Ergebnisse, wie an den grünen Dreiecken in (Abbildung 20) zu erkennen ist. In einigen Fällen jedoch waren die Resultate durch Rauschen oder ähnliches sehr zufällig, wie in (Abbildung 21).

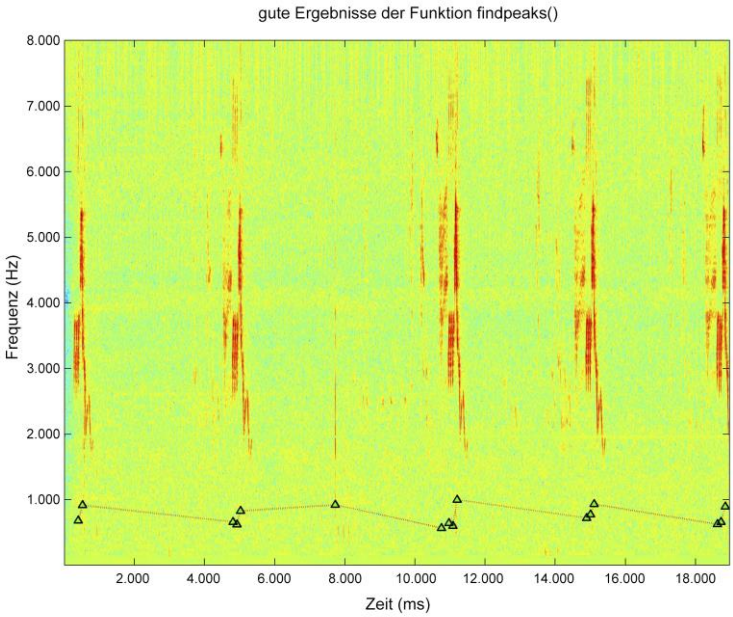


Abbildung 20: qualitativ gute Ergebnisse der Funktion findpeaks() mit gut sichtbaren lokalen Maxima

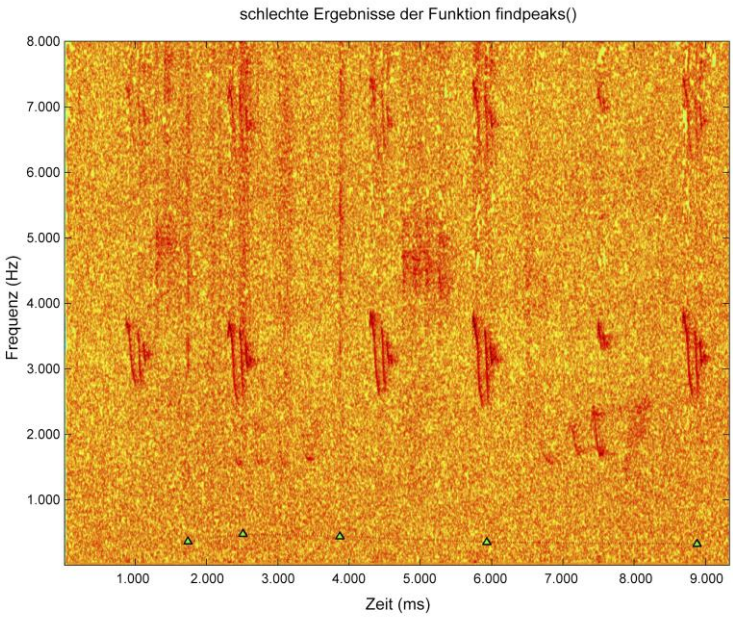


Abbildung 21: qualitativ schlechte Ergebnisse der Funktion findpeaks() durch Übersehen von Vogellauten durch Störgeräusche

Der Schritt, das Rauschen nach der Berechnung des Spektrogramms zu minimieren, funktionierte nur bis zu einem gewissen Grad, weshalb Ergebnisse wie in (Abbildung 21) nicht zu verbessern waren. Es wurde auch mit einer alternativen Funktion zur Erkennung der lokalen Maxima gearbeitet, die aber durch fehlende Flexibilität schlechtere Ergebnisse lieferte.

Bei der Ansicht der Wörterbücher ist deutlich sichtbar, dass die Extraktion der Vogellaute und die damit verbundene Erstellung der Wörterbücher gut funktionierten. In der grafischen Darstellung des Wörterbuchs eines Vogels sind die Laute gut in den einzelnen Wörtern festgehalten und sogar eine sichtbare Frequenzspanne der Vogellaute ist erkennbar, die im Falle von (Abbildung 22) von etwa 1.000 bis 6.000 Hz war.

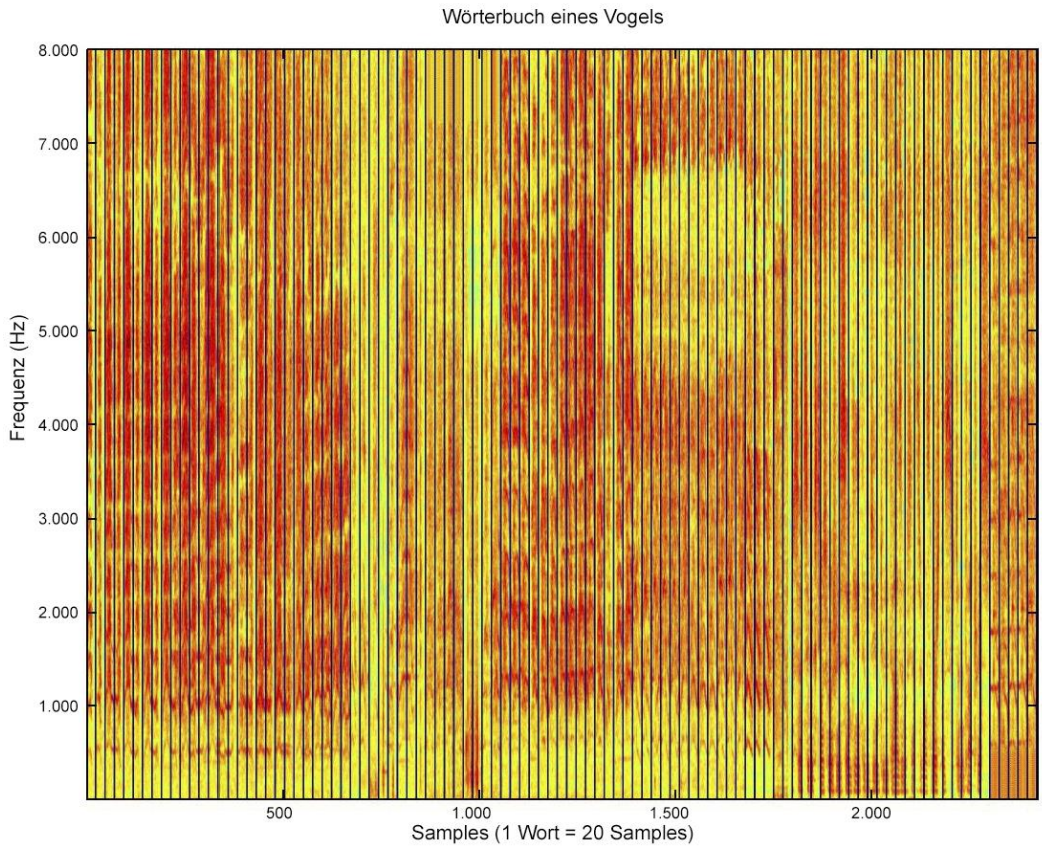


Abbildung 22: Wörterbuch eines Vogels über alle Wörter

Die Ergebnisse des Clusterings waren ebenfalls zufriedenstellend. In (Abbildung 23) ist auf der linken Seite der Linie das jeweilige Clusterzentrum zu sehen, das aus den Wörtern der rechten Seite durch Mittelwertberechnung ermittelt wird. Sowohl das Clustering, also die Auswahl der zu gruppierenden Wörter, als auch die Repräsentationen durch die Clusterzentren sind gut nachvollziehbar. Ziel des

Clusterings war, die Anzahl an Daten stark zu reduzieren, indem gleiche oder ähnliche Wörter zusammengefasst wurden. Redundante Daten konnten so eliminiert werden.

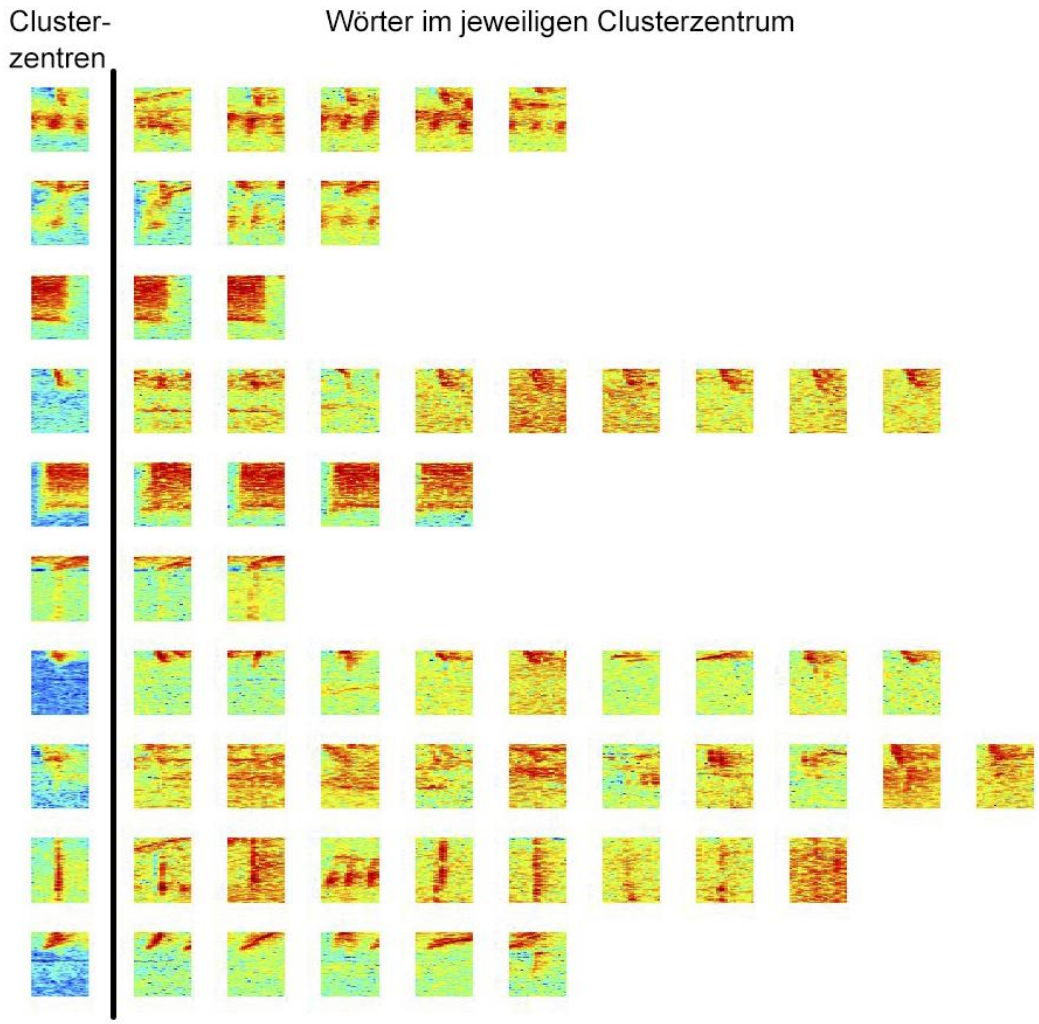


Abbildung 23: Darstellung der Clustererstellung (li. Clusterzentren | re. Wörter, die das Clusterzentrum repräsentieren)

5.2 Quantitative Ergebnisse

Da der Rechenaufwand des instanzbasierten Lernens durch die wesentlich größere Anzahl an Merkmalsvektoren viel höher ist als der von anderen Klassifikationsmethoden, wurden die zu klassifizierenden Spezies für die Experimente auf 10 reduziert. Durch Messung der Laufzeiten in mehreren Durchgängen und mit unterschiedlichen Anzahlen von Spezies wurde ermittelt, dass die Klassifikation mit durchschnittlich 15 Aufnahmen pro Spezies trotz

effizienter Programmierung im Mindestfall von vier Clusterzentren etwa 26,5 Tage gedauert hätte. Die Formel zur ungefähr resultierenden Rechendauer R skalierte also linear mit der Anzahl der Klassen und lautete:

$$R = 190 * A + 100$$

Dabei war A die Anzahl der zu analysierenden Klassen und das Ergebnis wurde in Sekunden berechnet.

Da die hochgerechnete Dauer keine praktikablen Experimente vorhersagte, wurde der Umfang reduziert, um in einem kleineren und überschaubareren Rahmen die Qualität des Klassifikators zu testen.

In (Tabelle 2) befinden sich die Ergebnisse der ersten Testreihe mit einem Frequenzbereich von 0-8.000 Hz. In Spalte eins ist die Anzahl an Clustern gegeben, die in den Experimenten 4, 6 und 10 Cluster waren. In Spalte zwei findet sich die Art der Klassifikation, entweder über den nächstliegenden Vektor der Testaufnahme (Nearest Neighbor) und einem Clusterzentrum einer Klasse, oder über die Mehrheit der minimalen Vektordistanzen aller Wörter der Testdatei (K-Nearest Neighbor). Die Berechnung erfolgte in allen Fällen über die Euklidische Distanz zweier Vergleichsvektoren. Spalte drei gibt die Rechendauer in Minuten an, die bei durchschnittlich acht Aufnahmen pro Klasse je Test- und Trainingsset notwendig war, um allen Dateien im Testset einer Klasse zuzuordnen. Die Spalten vier bis sieben geben die Ergebnisse der Genauigkeit der jeweiligen Distanzmessung zur Klassifikation an.

Tabelle 2: Ergebnisse der ersten Testreihe zur Klassifikation der selbstentwickelten Methode (Frequenzbereich 0-8.000 Hz)

k	**Art der Klassenzuordnung**	**Rechenzeit in Minuten**	**Test 1 Genauigkeit über Vektoren des Spektrogramms**	**Test 2 Genauigkeit über Varianz**	**Test 3 Genauigkeit über Mittelwert**	**Test 4 Genauigkeit über Varianz und Mittelwert**
10	Nearest Neighbor	77,0	**18,42%**	7,89%	14,47%	**10,53%**
6	Nearest Neighbor	47,2	13,16%	5,26%	18,42%	**10,53%**
4	Nearest Neighbor	32,8	15,79%	**10,53%**	**19,74%**	**10,53%**
10	K-Nearest Neighbor	77,9	39,47%	21,05%	32,89%	**13,16%**
6	K-Nearest Neighbor	47,9	40,79%	18,42%	**38,16%**	11,84%
4	K-Nearest Neighbor	32,2	**44,74%**	**23,68%**	**38,16%**	10,53%

Die jeweils besten Klassifikationsergebnisse wurden fett markiert. Es ist gut ersichtlich, dass die Art der Klassenzuordnung über die Mehrheit der nahen Vektoren besser funktionierte als die Zuordnung über den nächstliegenden Vektor. Meistens wurde die höchste Klassifikationsrate durch den Vergleich der

Spektrogramme erzielt, im besten Fall 44,74%. Danach folgte die Klassifikation über die Mittelwertvektoren mit maximal 38,16% und mit Abstand danach die Klassifikation über die Varianzvektoren mit maximal etwa 23,68%. Eine Kombination beider Vektoren erzielte durchgehend keine guten Ergebnisse mit einer maximalen Genauigkeit von etwa 13%. Außerdem ist gut ersichtlich, dass die Klassifikation mit 4 Clusterzentren in den meisten Fällen am besten funktionierte. Auf diesen Ergebnissen und dieser Erkenntnis wurde eine weitere Testreihe (Tabelle 3) durchgeführt. In dieser Reihe wurde evaluiert, welche Frequenzbereiche sich am besten für die Klassifikation eignen. Die Clusteranzahl k war in dieser Testreihe 4. Die Ergebnisse mit der höchsten Genauigkeit wurden wieder fett markiert.

Tabelle 3: Ergebnisse der zweiten Testreihe zur Klassifikation der selbstentwickelten Methode mit k von 4

Frequenzbereich	**Rechenzeit in Minuten**	**Test 1 Genauigkeit über Vektoren des Spektrogramms**	**Test 2 Genauigkeit über Varianz**	**Test 3 Genauigkeit über Mittelwert**	**Test 4 Genauigkeit über Varianz und Mittelwert**
0-4.000 Hz	5,35	27,63%	21,05%	26,32%	**10,53%**
0-6.000 Hz	16,55	34,21%	18,42%	26,32%	**10,53%**
0-8.000 Hz	35,83	44,74%	**23,68%**	38,16%	**10,53%**
0-10.000 Hz	60,18	44,74%	11,84%	**47,37%**	**10,53%**
0-12.000 Hz	99,97	39,47%	14,47%	35,53%	7,89%
1.000-8.000 Hz	24,43	**52,63%**	17,11%	44,74%	**10,53%**
2.000-8.000 Hz	17,65	42,11%	6,58%	30,26%	9,21%
1.000-10.000 Hz	48,48	35,53%	**23,68%**	38,16%	**10,53%**

Die besten Ergebnisse erzielte wieder der Vergleich der Spektrogrammvektoren mit maximal 52,63%. Die Genauigkeit über die Varianzvektoren erzielte maximal 23,68%, die Genauigkeit über den Mittelwert bis zu 47,37% und die Genauigkeit über die Kombination aus Varianz- und Mittelwertvektoren 10,53%, die allerdings auch ein zufälliger Klassifikator erreichen würde. Warum die Ergebnisse der Kombinationsvektoren so schlecht waren, lag an den unterschiedlichen Wertebereichen der extrahierten Daten. Selbst eine Normalisierung der Kombinationsvektoren brachte keine Verbesserung der Ergebnisse.

Die folgende (Abbildung 24) zeigt die Ergebnisse aus (Tabelle 3) als Liniendiagramm an. Hier sind die Qualitäten der unterschiedlichen Tests gut sichtbar.

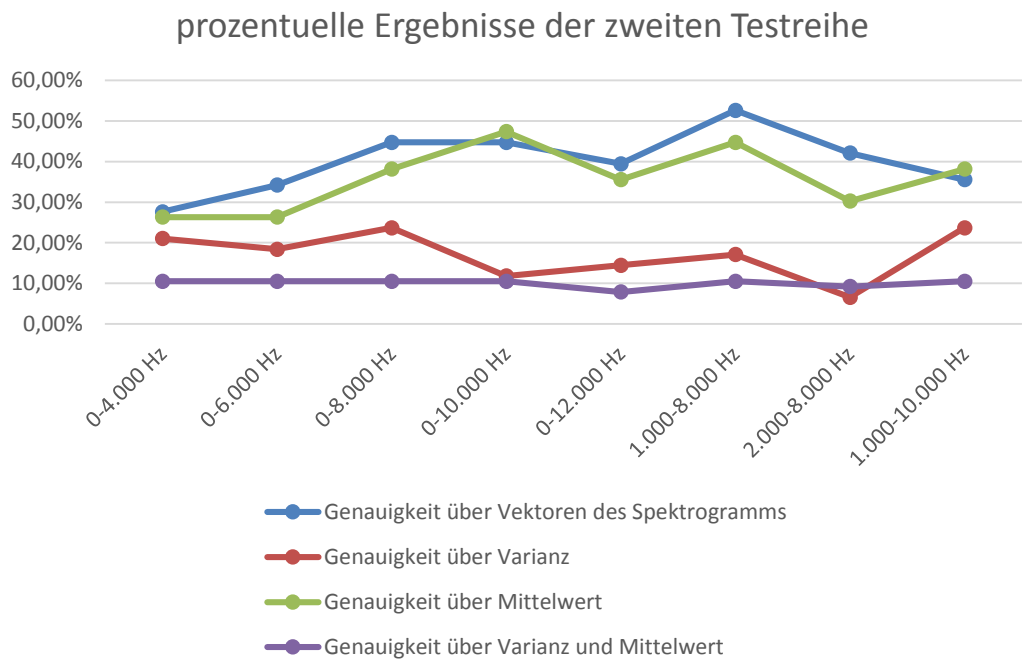


Abbildung 24: prozentuelle Ergebnisse der zweiten Testreihe

Ein dritter Test sollte zeigen, ob die Verwendung der MFCC-Matrizen, die die Wettbewerbsveranstalter zur Verfügung gestellt haben, zu einer Steigerung der Klassifikationsrate führen kann. Der Test verglich die normalen Spektrogrammvektoren, die auch als Varianz- und Mittelwertvektoren zur Verwendung kamen, mit den Werten der MFCC Berechnungen, welche ebenfalls als Varianz- und Mittelwertvektoren ausgewertet wurden. Die Ergebnisse sind in (Tabelle 4) ersichtlich und zeigten, dass lediglich bei der Kombination der Varianz- und Mittelwertvektoren (Test 4) eine Steigerung der Genauigkeit von etwa 4% erreicht wurde. Bei den anderen drei Vektorarten wurde eine Verschlechterung der Genauigkeit von bis zu 25% festgestellt. Allerdings war die Rechendauer der Klassifikation mit MFCC etwa 1/3 so lange wie die mit Spektrogrammen. In Summe war die Klassifikation mit Spektrogrammen um 47,36% besser.

Tabelle 4: Ergebnisse des dritten Tests mit MFCC-Matrizen

Art der Daten	**Rechenzeit in Minuten**	**Genauigkeit über Vektoren des Spektrogramms**	**Genauigkeit über Varianz**	**Genauigkeit über Mittelwert**	**Genauigkeit über Varianz und Mittelwert**
Spektrogramme	24,43	**52,63%**	**17,11%**	**44,74%**	10,53%
MFCC	**8,14**	27,63%	6,58%	28,95%	**14,47%**

Der abschließende vierte Test ermittelte die Notwendigkeit der Rauschverminderung aus Kapitel 3.3, indem die Klassifikation mit und ohne dieser durchgeführt wurde. Die Ergebnisse zeigten eindeutig, dass der vorgestellte Rechenschritt hervorragend funktionierte und einen großen Stellenwert bei der Qualität der Klassifikation aufwies. So waren die Ergebnisse mit Rauschverminderung um bis zu 31% besser als jene ohne diese. Aber auch von der Performance der Klassifikation war leicht zu erkennen, dass die durch Rauschen fehlerhaft erkannten Signalspitzen unnötig große Datenmengen verursachten. So wurde die Rechenzeit der Klassifikation auf mehr als einen Faktor 4 verlängert, wenn die Rauschverminderung nicht angewandt wurde. (Abbildung 25) zeigt, wie viele Maxima durch die falsche Erkennung gefunden wurden. Hier ist eine Gegenüberstellung zu sehen, bei der im linken Teil des Bildes keine Rauschverminderung durchgeführt wurde. In diesem Bereich sind sofort die fehlerhaft erkannten Maxima zwischen den in rot erkennbaren lautesten Signalanteilen, also den Vogellauten, sichtbar und mit einer roten Ellipse markiert. Im rechten Teil ist die Rauschverminderung aktiv, was schon im Erscheinungsbild des Spektrogramms zu erkennen ist. Hier werden die lokalen Maxima der Vogellaute ohne Fehler erkannt und angezeigt. Die Parameter der Funktion zum Erkennen der lokalen Maxima waren in beiden Anwendungen identisch.

Alle Testergebnisse unterstützen also die Aussage, dass sich die Rauschverminderung positiv auf die Klassifikationsrate auswirkt und die für die Klassifikation wichtigsten Signalanteile zwischen 1.000 und 8.000 Hz liegen.

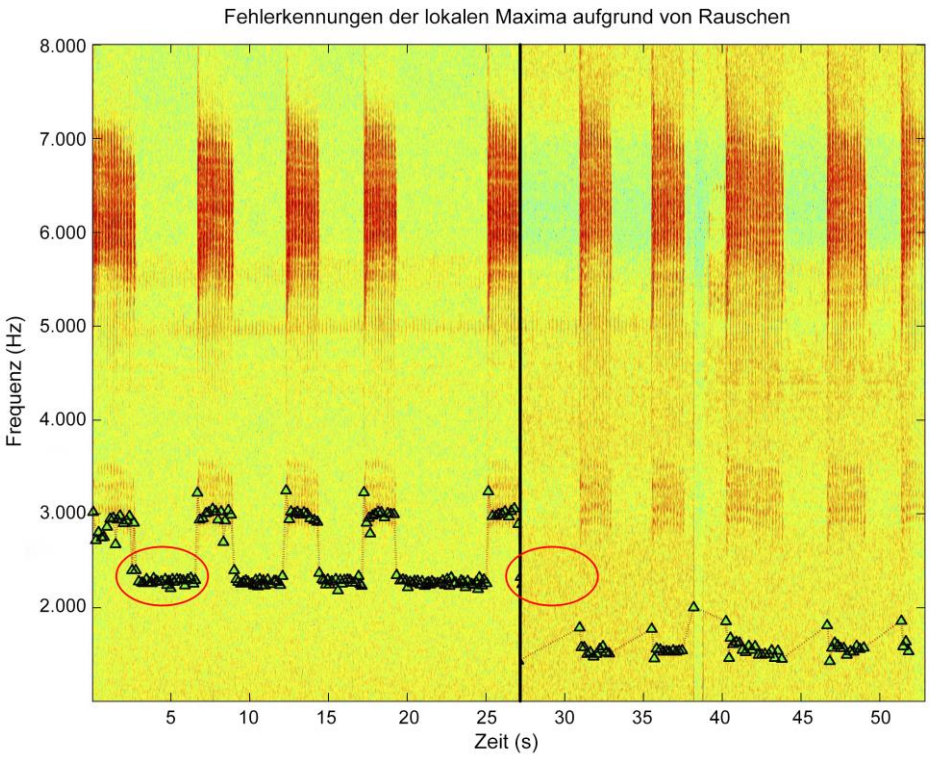


Abbildung 25: Fehlerkennungen der lokalen Maxima aufgrund von Rauschen (li. Rauschen und Fehler | re. rauschvermindert und bessere Erkennung)

6 Zusammenfassung und Ausblick

Die Klassifikation von Vogelstimmen kann mit unterschiedlichen Methoden erzielt werden. So können Klassifikatoren mit Merkmalen oder Audiodaten selbst arbeiten, um eine Klassifikation durchzuführen. Die Art der Klassifikation ist an den jeweiligen Anwendungsfall und die vorhandenen Daten anzupassen. Die Anwendung der instanzbasierten Klassifikation wie in dieser Arbeit ist eine pragmatische, leicht verständliche und simple Methode. Die Rechenkomplexität des Klassifikators ist aber durch die große Anzahl an Daten sehr hoch. Im Laufe der Arbeit wurde der Datenumfang daher deutlich reduziert, um die Rechendauer zu verringern. Die Genauigkeit der Klassifikation war mit maximal 52,63% sehr zufriedenstellend und zeigen, dass instanzbasiertes Lernen für Vogelstimmenklassifikation erfolgreich eingesetzt werden kann. Die Verwendung von spezialisierten Methoden erzielt, wie in den Ergebnissen des Bird task ersichtlich, größtenteils bessere Ergebnisse; das benötigte Hintergrundwissen ist jedoch ein entscheidender Faktor für den Erfolg zur Wahl der geeigneten Parameter (Goëau et al., 2014).

Um die Klassifikationsrate weiter zu steigern, könnte in weiteren Testreihen evaluiert werden, ob eine Änderung der Fenstergröße für die Wortrepräsentationen bessere Ergebnisse erzielen könnte. Weiters merkten Ren, William Dennis, & Huy Dat (2014, p. 5) an, dass ihre entwickelte Methode zur Klassifikation für den Bird task mittels Metadaten gleich gut funktionieren würde wie die Verwendung von Merkmalen der Audioaufnahmen. Auch dieser Ansatz eröffnet Verbesserungsmöglichkeiten. Des Weiteren ist in den Metadaten die Anzahl von Spezies ersichtlich, die in einer Aufnahme vorkommen. Die Auswahl von Dateien, in denen nur eine Spezies vorkommt, könnte die Klassifikationsrate möglicherweise auch steigern.

Literaturverzeichnis

Baker, M. C. (2006). Differentiation of Mating Vocalizations in Birds: Acoustic Features in Mainland and Island Populations and Evidence of Habitat-Dependent Selection on Songs. *Ethology*, *112*(8), 757–771. http://doi.org/10.1111/j.1439-0310.2006.01212.x

Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K.-H., & Frommolt, K.-H. (2010). Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, *31*(12), 1524–1534. http://doi.org/10.1016/j.patrec.2009.09.014

Bird task | ImageCLEF - Image Retrieval in CLEF. (n.d.). Retrieved May 11, 2015, from http://www.imageclef.org/2014/lifeclef/bird

Catchpole, C. K., & Slater, P. J. B. (2008). *Bird Song: Biological Themes and Variations* (Auflage: 2). Cambridge England ; New York: Cambridge University Press.

Chang-Hsing, L., Yeuan-Kuen, L., & Ren-Zhuang, H. (2006). Automatic Recognition of Birdsongs Using Mel-frequency Cepstral Coefficients and Vector Quantization., 331–335.

Charif, R. A., Clark, C. W., & Fristrup, K. M. (2006). Raven 1.3 user's manual. *Technical Report, Cornell Laboratory of Ornithology, Ithaca, NY*.

Chou, C.-H., Lee, C.-H., & Ni, H.-W. (2007). Bird Species Recognition by Comparing the HMMs of the Syllables. In *Second International Conference on Innovative Computing, Information and Control, 2007. ICICIC '07* (pp. 143–143). http://doi.org/10.1109/ICICIC.2007.199

Cowling, M., & Sitte, R. (2003). Comparison of Techniques for Environmental Sound Recognition. *Pattern Recogn. Lett.*, *24*(15), 2895–2907. http://doi.org/10.1016/S0167-8655(03)00147-8

Downie, J. S. (2003). Music information retrieval. *Annual Review of Information Science and Technology*, *37*(1), 295–340. http://doi.org/10.1002/aris.1440370108

Fagerlund, S. (2007). Bird Species Recognition Using Support Vector Machines. *EURASIP J. Appl. Signal Process.*, *2007*(1), 64–64. http://doi.org/10.1155/2007/38637

Fagerlund, S. (n.d.). Automatic Recognition of Bird Species by Their Sounds.

Gasc, A., Sueur, J., Jiguet, F., Devictor, V., Grandcolas, P., Burrow, C., … Pavoine, S. (2013). Assessing biodiversity with sound: Do acoustic diversity indices reflect phylogenetic and functional diversities of bird communities? *Ecological Indicators*, *25*, 279–287. http://doi.org/10.1016/j.ecolind.2012.10.009

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R., Rauber, A., & Joly, A. (2014). LifeCLEF bird identification task 2014. In *CLEF2014*.

Harma, A. (2003). Automatic identification of bird species based on sinusoidal modeling of syllables. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)* (Vol. 5, pp. V–545–8 vol.5). http://doi.org/10.1109/ICASSP.2003.1200027

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and Their Applications*, *13*(4), 18–28. http://doi.org/10.1109/5254.708428

Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M., & Rudnicky, A. I. (2006). Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In *2006 IEEE International Conference on Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings* (Vol. 1, pp. I–I). http://doi.org/10.1109/ICASSP.2006.1659988

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Comput. Surv.*, *31*(3), 264–323. http://doi.org/10.1145/331499.331504

Joly, A., Champ, J., & Buisson, O. (2014). Instance-based bird species identication with undiscriminant features pruning-LifeCLEF 2014. In *CLEF2014*.

Klapuri, A., & Davy, M. (2006). *Signal Processing Methods for Music Transcription* (Auflage: 2006). New York: Springer.

Koops, H. V., Balen, J. V., & Wiering, F. (2014). A Deep Neural Network Approach to the LifeCLEF 2014 Bird Task. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.* (pp. 634–642). Retrieved from http://ceur-ws.org/Vol-1180/CLEF2014wn-Life-KoopsEt2014.pdf

MARSYAS. (n.d.). Retrieved March 3, 2015, from http://marsyas.info/

Martinez, R., Silvan, L., Villarreal, E., Fuentes, G., & Meza, I. (2014). Svm candidates and sparse representation for bird identification. *Working Notes of CLEF 2014 Conference*.

Nelson, D. A. (1989). The Importance of Invariant and Distinctive Features in Species Recognition of Bird Song. *The Condor*, *91*(1), 120–130. http://doi.org/10.2307/1368155

Platt, J. C. (1999). Advances in Kernel Methods. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), (pp. 185–208). Cambridge, MA, USA: MIT Press. Retrieved from http://dl.acm.org/citation.cfm?id=299094.299105

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286. http://doi.org/10.1109/5.18626

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Ren, L. Y., William Dennis, J., & Huy Dat, T. (2014). Bird classification using ensemble classifiers. In *Working notes of CLEF 2014 Conference*.

Silla, C. N., Kaestner, C. A. A., & Koerich, A. L. (2007). Automatic music genre classification using ensemble of classifiers. In *IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC* (pp. 1687–1692). http://doi.org/10.1109/ICSMC.2007.4414136

SoX - Sound eXchange | HomePage. (n.d.). Retrieved March 3, 2015, from http://sox.sourceforge.net/

Stinson, M. S., Elliot, L. B., Kelly, R. R., & Liu, Y. (2009). Deaf and Hard-of-Hearing Students' Memory of Lectures with Speech-to-Text and Interpreting/Note Taking Services. *The Journal of Special Education*, *43*(1), 52–64. http://doi.org/10.1177/0022466907313453

Trifa, V. M., Kirschel, A. N. G., Taylor, C. E., & Vallejo, E. E. (2008). Automated species recognition of antbirds in a Mexican rainforest using hidden Markov models. *The Journal of the Acoustical Society of America*, *123*(4), 2424–2431. http://doi.org/10.1121/1.2839017

Ververidis, D., & Kotropoulos, C. (2006). Emotional speech recognition: Resources, features, and methods. *Speech Communication*, *48*(9), 1162–1181. http://doi.org/10.1016/j.specom.2006.04.003

Wang, A. (2006). The Shazam music recognition service. *Commun. ACM*, *49*, 44–48. http://doi.org/10.1145/1145287.1145312

xeno-canto :: Vogelstimmen aus aller Welt teilen. (n.d.). Retrieved December 3, 2014, from http://www.xeno-canto.org/

Young, S., Evermann, G., Hain, T., Kershaw, D., Moore, G., Odell, J., … Woodland, P. (2002). The HTK Book (for HTK Version 3.2. 1), 2002. *Cambridge University Engineering Department*.

Zeppelzauer, M., Stöger, A., & Breiteneder, C. (2013). Acoustic Detection of Elephant Presence in Noisy Environments. In *MAED '13 Proceedings of the 2nd*

ACM international workshop on Multimedia analysis for ecological data. New York, NY, USA: ACM. http://doi.org/10.1145/2509896.2509900

Zhou, Q., Liu, X., & Duan, H. (2005). ECG Beat Classification Using Mirrored Gauss Model. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the* (pp. 5587–5590). http://doi.org/10.1109/IEMBS.2005.1615752

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A. Code der Arbeit

A.1. Funktion für Auslesen der XML Daten und Sortierung der Dateien:

```
speziesArray={'abeillei', 'accipitrinus', 'acer', 'acutirostris',
'aedon', 'aestiva', 'aethiops', 'albescens', 'albicollis',
'albilora'};

files=dir('train/*.xml');
for i=1:1:length(files)

    xmlData=xml_load(files(i).name);
    spezies=xmlData.Species;

    if any(ismember(speziesArray,spezies))==0;
        continue;
    end

    originalName=files(i).name;
    originalName=originalName(1:end-4);
    audioName=strcat(originalName,'.wav');
    speziesOrdner = strcat(pwd,'/train/',spezies,'/');
    if exist(speziesOrdner,'dir') <= 0
        mkdir(speziesOrdner);
    end

    wavFileName=strcat(pwd,'/train/',audioName);
    xmlFileName=strcat(pwd,'/train/',files(i).name);

    wavMoveName=strcat(spezies,'/',audioName);
    copyfile(wavFileName,speziesOrdner);
    copyfile(xmlFileName,speziesOrdner);

end
```

A.2. Hauptfunktion:

```
clear all;
d = dir('./train/'); % verzeichnis
isub = [d(:).isdir]; % alle ordner hernehmen
nameFolds = {d(isub).name}';
nameFolds(ismember(nameFolds,{'.','..'})) = [];
% versteckte ordner wegschneiden

 for c=1:length(nameFolds)
    vogelname=char(nameFolds(c)); % name des vogels herausfinden

    path=strcat('./train/',vogelname,'/*.wav');
```

```
    files=dir(path);
    spekPfad=strcat('train/',vogelname,'/spektrogramme/');
    if ~exist(spekPfad, 'dir')
        mkdir(spekPfad);
    end
    clustPfad=strcat('train/',vogelname,'/clustering/');
    if ~exist(clustPfad, 'dir')
        mkdir(clustPfad);
    end

    woerterbuch20=[];

    for i=1:1:length(files);
        wav=wavread(['./train/' vogelname filesep files(i).name]);

        % --------------------SPEKTROGRAMM----------------

            [S,F,T,P] = spectrogram(wav,1764,1323,2*2048,44100);
%               [S,F,T,P] =
spectrogram(wav,512,341,2*2048,44100); % für MFCC
            spectrogramm = 10*log10(P);

            figure('Visible','off'); imagesc(spectrogramm);
            set(gca,'YDir','normal');
            head=strcat('Spektrogramm');
            title(head);
            xlabel('Zeit (ms)');
            ylabel('Frequenz (Hz/10)');

            abschneidenAb=8000;
            spectrogramm((abschneidenAb/10+1):end,:) = [];
            abschneidenTief=1000;
            if abschneidenTief~=0
                spectrogramm((1:(abschneidenTief/10)),:) = [];
            end

            spectrogrammHoehe=size(spectrogramm,1);
            spectrogrammBreite=size(spectrogramm,2);

            filename=files(i).name;
            filename=filename(1:end-4);


jpgPrintName=strcat('train/',vogelname,'/spektrogramme/',filename,'.jp
g');

            figure('Visible','off'); imagesc(spectrogramm);
            set(gca,'YDir','normal');
            head=strcat('Spektrogramm bis 8000 Hz');
            title(head);
```

```
            xlabel('Zeit (ms)');
            ylabel('Frequenz (Hz/10)');

        % ---------------DURCHSCHNITT ABZIEHEN-------------------

            spectroDurchschnitt = mean(spectrogramm,2);

            wiederholterDurchschnitt = repmat(spectroDurchschnitt,[1
spectrogrammBreite]);
            durchschnAbgezogenesSpec = spectrogramm-
wiederholterDurchschnitt;

            figure('Visible','off');
            imagesc(durchschnAbgezogenesSpec);
            set(gca,'YDir','normal');
            head=strcat('Spektrogramm mit abgezogenem Durchschnitt');
            title(head);
            xlabel('Zeit (ms)');
            ylabel('Frequenz (Hz/10)');

        % ---------------------FINDPEAKS------------------
             spectroDurchschnittsWert=std2(durchschnAbgezogenesSpec);
            pks={};
            locs={};

            summeSpalten = sum(durchschnAbgezogenesSpec,1);

[pks,locs]=findpeaks(summeSpalten,'MINPEAKHEIGHT',2*abs(std(summeSpalt
en)),'MINPEAKDISTANCE',10);

             if isempty(pks) % falls keine peaks gefunden werden
                 continue; % Datei überspringen
             end

             figure('Visible','off'); plot(summeSpalten); hold on;
             plot(locs,pks,'r*');
             head=strcat('Maxima im Signal');
             title(head);
             xlabel('Zeit (ms)');
             ylabel('Gewicht der Maxima');

        % ---------------------PEAKS PLOTTEN-----------------

        figure('Visible','off');
        imagesc(durchschnAbgezogenesSpec);
        set(gca,'YDir','normal');
        hold on;
        head=strcat('Spektrogramm mit Maxima');
        title(head);
        xlabel('Zeit (ms)');
```

```
        ylabel('Frequenz (Hz/10)');
        skaliertePks=pks.*0.01; % dividiert 1/100

plot(locs,skaliertePks,':r^','LineWidth',1,'MarkerEdgeColor','k','Mark
erFaceColor','g','MarkerSize',5);

        print(jpgPrintName,'-djpeg','-r300');



        % -------------------MFCC Matrix-------------------

%         trainingFilename=strcat('train/',vogelname,'/*.txt'); %
datei laden
%         trainingFiles=dir(trainingFilename);
%         vergleichsFile=trainingFiles(i).name;
%
vergleichsFilePath=strcat('train/',vogelname,'/',vergleichsFile);
%         inhalt = load(vergleichsFilePath);
%         MFCCMatrix=rot90(inhalt); % bis hier datei einlesen
%         spectrogrammBreite=size(MFCCMatrix,2);



        % -------------------WOERTERBÜCHER----------------

aktuellesWoerterbuch=woerterbuchErzeugen(spectrogrammBreite,20,durchsc
hnAbgezogenesSpec,locs);
        woerterbuch20=[woerterbuch20 aktuellesWoerterbuch];

    end %% ende for schleife files
    close all;



filename=strcat('train/',vogelname,'/',vogelname,'Woerterbuch',num2str
(20),'File.mat');
    save(filename,'woerterbuch20');

        % -----------------PDFs--------------------


    pdfErzeugen(woerterbuch20,20,spectrogrammHoehe,vogelname);


        % -----------CLUSTERING VEKTOREN ERSTELLEN---------------


    clusteringMatrix=[];
    for m=1:1:numel(woerterbuch20)
        aktuelleMatrix=woerterbuch20{m};
        vektor=aktuelleMatrix(:);
        clusteringMatrix=[clusteringMatrix vektor];
    end
    clusteringMatrix=clusteringMatrix';


        % --------CLUSTERING MIT KMEANS------------
```

```
    k=[4,6,10];
    for l=1:size(k,2)

kMeansClustering(clusteringMatrix,k(l),woerterbuch20,vogelname);
    end
end
```

A.3. Funktion zur Erstellung der Wörterbücher:

```
function [ woerterbuch ] = woerterbuchErzeugen(
spectrogrammBreite,fenstergroesse,durchschnAbgezogenesSpec,peakLocs )

    for m=1:numel(peakLocs)
        position=peakLocs(m);
        halbeFenstergroesse=round(fenstergroesse/2);
        %mittig
        if position>halbeFenstergroesse &
position<(spectrogrammBreite-halbeFenstergroesse)
            woerterbucheintrag=durchschnAbgezogenesSpec(:,position-
(halbeFenstergroesse-1):position+((halbeFenstergroesse)));
            woerterbuch{m}=woerterbucheintrag;
        %ende
        elseif position>(spectrogrammBreite-halbeFenstergroesse-1)
            ende=(spectrogrammBreite-fenstergroesse)+1;

woerterbucheintrag=durchschnAbgezogenesSpec(:,ende:spectrogrammBreite)
;
            woerterbuch{+m}=woerterbucheintrag;
        %anfang
        else

woerterbucheintrag=durchschnAbgezogenesSpec(:,1:fenstergroesse);
            woerterbuch{m}=woerterbucheintrag;
        end
    end
end
```

A.4. Funktion für das Erzeugen der PDF Dateien:

```
function [ ] = pdfErzeugen(
woerterbuchX,fenstergroesse,spectrogrammHoehe,vogelname )


    myMatrix=[];
    tempMatrix=[];
    for i=1:1:numel(woerterbuchX)
        currentWord=woerterbuchX{i};
        tempMatrix = [currentWord
(ones(spectrogrammHoehe,(fenstergroesse/5))*-100)];
        tempAll = myMatrix;
        myMatrix = [tempAll tempMatrix];
```

```
    end

    figure('Visible','off');
        set(gcf,'Units','inches');
        screenposition = get(gcf,'Position');
        set(gcf,...
        'PaperPosition',[0 0 screenposition(3:4)],...
        'PaperSize',[screenposition(3:4)]);
    imagesc(myMatrix);
    set(gca,'YDir','normal');

pdfPrintName=strcat('train/',vogelname,'/',vogelname,'Woerterbuch',num
2str(fenstergroesse),'.pdf');
    print(gcf, '-dpdf', '-r300', pdfPrintName);


end
```

A.5. Funktion für das Clustering:

```
function [ ] = kMeansClustering(
clusteringMatrix,k,woerterbuch20,vogelname )


    clusteranzahl=k;
    [IDX,C,sumd,D] = kmeans(clusteringMatrix,clusteranzahl);
    varianzMatrix=[];
    mittelwertMatrix=[];

    % cluster center darstellen
    figure('Visible','off');
    for i=1:1:clusteranzahl
        subplot(clusteranzahl/2,2,i);
        clusterCenter=C(i,:);
        clusterCenter=reshape(clusterCenter,size(woerterbuch20{1}));
        imagesc_flip(clusterCenter,false);
        varianz=var(clusterCenter,0,2);
        mittelwert=mean(clusterCenter,2);
        varianzMatrix = [varianzMatrix varianz];
        mittelwertMatrix = [mittelwertMatrix mittelwert];
        titel=strcat('center von cluster: ', num2str(i));
        title(titel);
    end


clusterPrintName=strcat('train/',vogelname,'/clustering/',vogelname,'_
k',num2str(clusteranzahl),'_clusterCenter.mat');
    save(clusterPrintName,'C')

clusterPrintName=strcat('train/',vogelname,'/clustering/',vogelname,'_
k',num2str(clusteranzahl),'_clusterCenter.jpg');
    print(clusterPrintName,'-djpeg','-r300');

clusterVarianz=strcat('train/',vogelname,'/',vogelname,'_k',num2str(cl
usteranzahl),'_varianzMatrix.mat');
    save(clusterVarianz,'varianzMatrix')
```

```
clusterMittelwert=strcat('train/',vogelname,'/',vogelname,'_k',num2str
(clusteranzahl),'_mittelwertMatrix.mat');
    save(clusterMittelwert,'mittelwertMatrix')


end
```

A.6. Funktion für das Austesten des Testsets:

```
clear all;
path=strcat('./test/*.wav');
files=dir(path);
spekPfad=strcat('test/spektrogramme/');
if ~exist(spekPfad, 'dir')
    mkdir(spekPfad);
end


accuracyDistanz=0;
accuracyVarianz=0;
accuracyMittelwert=0;
accuracyVarUndMit=0;
accuracyHaeufigsteDistanz=0;
accuracyHaeufigsteVarianz=0;
accuracyHaeufigsterMittelwert=0;
accuracyHaeufigsteVarUndMit=0;
uebersprungen=0;

for i=1:1:length(files);
    wav=wavread(['./test/' files(i).name]);

    % ------------------------SPECTROGRAMM-------------------------


        [S,F,T,P] = spectrogram(wav,1764,1323,2*2048,44100);
%               [S,F,T,P] =
spectrogram(wav,512,341,2*2048,44100); % für MFCC
        spectrogramm = 10*log10(P);


        abschneidenAb=8000;
        spectrogramm((abschneidenAb/10+1):end,:) = [];
        abschneidenTief=1:)000;
        if abschneidenTief~=0
            spectrogramm((1:(abschneidenTief/10)),:) = [];
        end


        spectrogrammHoehe=size(spectrogramm,1);
        spectrogrammBreite=size(spectrogramm,2);


        filename=files(i).name;
        filename=filename(1:end-4);


        jpgPrintName=strcat('test/spektrogramme/',filename,'.jpg');
```

```
    % ----------------DURCHSCHNITT ABZIEHEN--------------------

        spectroDurchschnitt = mean(spectrogramm,2);

        wiederholterDurchschnitt = repmat(spectroDurchschnitt,[1
spectrogrammBreite]);
        durchschnAbgezogenesSpec = spectrogramm-
wiederholterDurchschnitt;

    % ----------------------FINDPEAKS--------------------------

        spectroDurchschnittsWert=std2(durchschnAbgezogenesSpec);
        pks={};
        locs={};

        summeSpalten = sum(durchschnAbgezogenesSpec,1);

[pks,locs]=findpeaks(summeSpalten,'MINPEAKHEIGHT',100*max(max(durchsch
nAbgezogenesSpec)),'MINPEAKDISTANCE',10);
         if isempty(pks)
            uebersprungen=uebersprungen+1; % überspringen, wenn keine
peaks gefunden werden
            continue;
         end
         figure('Visible','off'); plot(summeSpalten); hold on;
         plot(locs,pks,'r*');
    % --------------------PEAKS PLOTTEN-----------------------

    figure('Visible','off');
    imagesc(durchschnAbgezogenesSpec);
    set(gca,'YDir','normal');
    hold on;
    head=strcat('Spectrogramm von: ',filename);
    title(head);
    xlabel('Zeit (ms)');
    ylabel('Frequenz (Hz/10)');
    skaliertePks=pks.*0.01; % dividiert 1/100

plot(locs,skaliertePks,':r^','LineWidth',1,'MarkerEdgeColor','k','Mark
erFaceColor','g','MarkerSize',5);

    print(jpgPrintName,'-djpeg','-r300');


        % --------------------MFCC Matrix--------------------
%
%       trainingFilename=strcat('test/','/*.txt'); % datei laden
%       trainingFiles=dir(trainingFilename);
%       vergleichsFile=trainingFiles(i).name;
%       vergleichsFilePath=strcat('test/','/',vergleichsFile);
```

```
%         inhalt = load(vergleichsFilePath);
%         MFCCMatrix=rot90(inhalt); % bis hier datei einlesen



    % --------------------WOERTERBÜCH-------------------------

woerterbuch20=woerterbuchErzeugen(spectrogrammBreite,20,durchschnAbgez
ogenesSpec,locs);

filename=strcat('test/',filename,'Woerterbuch',num2str(20),'File.mat')
;
    save(filename,'woerterbuch20');


    % -----------------VEKTOREN ERSTELLEN----------------------


    testWoerterMatrix=[];
    for m=1:1:numel(woerterbuch20)
        aktuelleMatrix=woerterbuch20{m};
        vektor=aktuelleMatrix(:);
        testWoerterMatrix=[testWoerterMatrix vektor];
    end
    testWoerterMatrix=testWoerterMatrix';


    % ---------------------DISTANZMESSUNG----------------------


    d = dir('./train/'); % verzeichnis
    isub = [d(:).isdir]; % alle ordner hernehmen
    nameFolds = {d(isub).name}';
    nameFolds(ismember(nameFolds,{'.','..'})) = []; % versteckte
ordner wegschneiden


    distanzCell={}; % initialisieren
    distanzWoerter={};
    distanzVarianz={};
    distanzMittelwert={};
    distanzMatrix=[];


    for x=1:size(testWoerterMatrix,1) % für jedes aktuelle wort


        testVektor=testWoerterMatrix(x,:); % nimm den jeweiligen wort-
vektor vom test vogel
        testVarianz=var(woerterbuch20{x},0,2);
        testMittelwert=mean(woerterbuch20{x},2);


        testVarUndMit=vertcat(testVarianz,testMittelwert);


        for z=1:10 % für jeden vogel im trainings set


            trainingsPfad=char(nameFolds(z)); % name des vogels
herausfinden
```

```
trainingFilename=strcat('train/',trainingsPfad,'/','clustering/*.mat')
; % cluster datei laden
            trainingFiles=dir(trainingFilename);
            vergleichsFile=trainingFiles(2).name; % 10,4,6

vergleichsFilePath=strcat('train/',trainingsPfad,'/','clustering/',ver
gleichsFile);
            inhalt = load(vergleichsFilePath);
            inhaltFile = fieldnames(inhalt);
            inhaltData = inhalt.(inhaltFile{1});
            trainingsClusteringMatrix=inhaltData; % bis hier datei
einlesen


            trainingFilename=strcat('train/',trainingsPfad,'/*.mat');
% varianz datei laden
            trainingFiles=dir(trainingFilename);
            vergleichsFile=trainingFiles(5).name; % wb, 10mitM,
10varM, 4MitM, 4varM, 6mitM, 6varM

vergleichsFilePath=strcat('train/',trainingsPfad,'/',vergleichsFile);
            inhalt = load(vergleichsFilePath);
            inhaltFile = fieldnames(inhalt);
            inhaltData = inhalt.(inhaltFile{1});
            trainingsVarianzMatrix=inhaltData; % bis hier datei
einlesen


            trainingFilename=strcat('train/',trainingsPfad,'/*.mat');
% mittelwert datei laden
            trainingFiles=dir(trainingFilename);
            vergleichsFile=trainingFiles(4).name; % wb, 10mitM,
10varM, 4MitM, 4varM, 6mitM, 6varM

vergleichsFilePath=strcat('train/',trainingsPfad,'/',vergleichsFile);
            inhalt = load(vergleichsFilePath);
            inhaltFile = fieldnames(inhalt);
            inhaltData = inhalt.(inhaltFile{1});
            trainingsMittelwertMatrix=inhaltData; % bis hier datei
einlesen



            for y=1:size(trainingsClusteringMatrix,1) % für jeden
cluster center des trainings vogels
                trainingsVektor=trainingsClusteringMatrix(y,:); %
cluster center vektor holen
                trainingsVarianz=trainingsVarianzMatrix(:,y);
                trainingsMittelwert=trainingsMittelwertMatrix(:,y);


                woerterDistanz = norm(testVektor - trainingsVektor); %
distanz errechnen
                distanz=log10(max(max(woerterDistanz))); % distanz
logarithmieren
                distanzWoerter{z,y}=distanz; % distanzen in XY achsen
eintragen.
```

```
                varianzDistanz = norm(testVarianz - trainingsVarianz);
% distanz errechnen
                distanz=log10(max(max(varianzDistanz))); % distanz
logarithmieren
                distanzVarianz{z,y}=distanz; % distanzen in XY achsen
eintragen.


                mittelwertDistanz = norm(testMittelwert -
trainingsMittelwert); % distanz errechnen
                distanz=log10(max(max(mittelwertDistanz))); % distanz
logarithmieren
                distanzMittelwert{z,y}=distanz; % distanzen in XY
achsen eintragen.
                % x = N cluster der jeweiligen vögel - y = vögel im
trainingsset



trainingsVarUndMit=vertcat(trainingsVarianz,trainingsMittelwert);

distanz=pdist2(testVarUndMit,trainingsVarUndMit,'euclidean'); %
euclidean distance % distanz errechnen
                distanz=log10(max(max(distanz))); % distanz
logarithmieren
                distanzVarUndMit{z,y}=distanz; % distanzen in XY
achsen eintragen.


            end
        end


        distanzWoerterCell{1,x}=(distanzWoerter); % erste zeile der
cell beinhaltet die distanzmatrix der testwörter
        distanzWoerterMatrix=cell2mat(distanzWoerter);
        [M,I] = min(abs(distanzWoerterMatrix));
        [Mminimum,Iminimum] = min(M);
        distanzWoerterCell{4,x}=M(Iminimum);
        distanzWoerterCell{5,x}=I(Iminimum);
        distanzWoerterCell{2,x}=M; % speichere minimum-wert
        distanzWoerterCell{3,x}=I; % speichere dazugehörigen index


        distanzVarianzCell{1,x}=distanzVarianz; % erste zeile der cell
beinhaltet die distanzmatrix der testwörter
        distanzVarianzMatrix=cell2mat(distanzVarianz);
        [M,I] = min(abs(distanzVarianzMatrix));
        [Mminimum,Iminimum] = min(M);
        distanzVarianzCell{4,x}=M(Iminimum);
        distanzVarianzCell{5,x}=I(Iminimum);
        distanzVarianzCell{2,x}=M; % speichere minimum-wert
        distanzVarianzCell{3,x}=I; % speichere dazugehörigen index


        distanzMittelwertCell{1,x}=distanzMittelwert; % erste zeile
der cell beinhaltet die distanzmatrix der testwörter
        distanzMittelwertMatrix=cell2mat(distanzMittelwert);
```

```
        [M,I] = min(abs(distanzMittelwertMatrix));
        [Mminimum,Iminimum] = min(M);
        distanzMittelwertCell{4,x}=M(Iminimum);
        distanzMittelwertCell{5,x}=I(Iminimum);
        distanzMittelwertCell{2,x}=M; % speichere minimum-wert
        distanzMittelwertCell{3,x}=I; % speichere dazugehörigen index

        distanzVarUndMitCell{1,x}=distanzVarUndMit; % erste zeile der
cell beinhaltet die distanzmatrix der testwörter
        distanzVarUndMitMatrix=cell2mat(distanzVarUndMit);
        [M,I] = min(abs(distanzVarUndMitMatrix));
        [Mminimum,Iminimum] = min(M);
        distanzVarUndMitCell{4,x}=M(Iminimum);
        distanzVarUndMitCell{5,x}=I(Iminimum);
        distanzVarUndMitCell{2,x}=M; % speichere minimum-wert
        distanzVarUndMitCell{3,x}=I; % speichere dazugehörigen index

    end

Ab hier Klassenzuordnung über Häufigkeit oder Nähe (2. Auskommentiert)
    % häufigste Distanz
    vogelCheck=[] % erstelle matrix mit allen mininum-ids
    for g=1:size(testWoerterMatrix,1)
        vogelCheck=[vogelCheck distanzWoerterCell{3,g}];
    end
    haeufigsteId=mode(vogelCheck,2); % schau, welcher vogel am
häufigsten am nähesten ist
    vogelName=char(nameFolds(haeufigsteId)); % gib den namen des
vogels aus


    filesXML=dir('test/*.xml'); % aus XML richtige spezies herauslesen
    xmldata = xml_load(filesXML(i).name);
    species = xmldata.Species;


    erkanntHaeufigsteDistanz=strcmp(vogelName,species); % auf
gleichheit überprüfen
    if erkanntHaeufigsteDistanz==1
        accuracyHaeufigsteDistanz=accuracyHaeufigsteDistanz+1;
    end
    checkString=strcat('erkannt ueber haeufigste distanz:
',vogelName,' - richtig: ',species,' - erkannt?:
',num2str(erkanntHaeufigsteDistanz));
    disp(checkString); % ergebnis ausgeben



%     % näheste Distanz
%     distanzWoerterErgebnisse=distanzWoerterCell(4,:);
%     distanzWoerterErgebnisse=[distanzWoerterErgebnisse;
distanzWoerterCell(5,:)];
%     distanzWoerterErgebnisse=cell2mat(distanzWoerterErgebnisse);
%     [M,I] = min(distanzWoerterErgebnisse(1,:));
```

```
%     vogelId=distanzWoerterErgebnisse(2,I);
%
%     vogelName=char(nameFolds(vogelId)); % gib den namen des vogels
aus
%
%     filesXML=dir('test/*.xml'); % aus XML richtige spezies
herauslesen
%     xmldata = xml_load(filesXML(i).name);
%     species = xmldata.Species;
%
%     erkanntDistanz=strcmp(vogelName,species); % auf gleichheit
überprüfen
%     if erkanntDistanz==1
%         accuracyDistanz=accuracyDistanz+1;
%     end
%     checkString=strcat('erkannt ueber distanz: ',vogelName,' -
richtig: ',species,' - erkannt?: ',num2str(erkanntDistanz));
%     disp(checkString); % ergebnis ausgeben


    % häufigste varianz
    vogelCheck=[] % erstelle matrix mit allen mininum-ids
    for g=1:size(testWoerterMatrix,1)
        vogelCheck=[vogelCheck distanzVarianzCell{3,g}];
    end
    haeufigsteId=mode(vogelCheck,2); % schau, welcher vogel am
häufigsten am nähesten ist
    vogelName=char(nameFolds(haeufigsteId)); % gib den namen des
vogels aus


    filesXML=dir('test/*.xml'); % aus XML richtige spezies herauslesen
    xmldata = xml_load(filesXML(i).name);
    species = xmldata.Species;


    erkanntHaeufigsteVarianz=strcmp(vogelName,species); % auf
gleichheit überprüfen
    if erkanntHaeufigsteVarianz==1
        accuracyHaeufigsteVarianz=accuracyHaeufigsteVarianz+1;
    end
    checkString=strcat('erkannt ueber haeufigste varianz:
',vogelName,' - richtig: ',species,' - erkannt?:
',num2str(erkanntHaeufigsteVarianz));
    disp(checkString); % ergebnis ausgeben


%     % näheste Varianz
%     distanzVarianzErgebnisse=distanzVarianzCell(4,:);
%     distanzVarianzErgebnisse=[distanzVarianzErgebnisse;
distanzVarianzCell(5,:)];
%     distanzVarianzErgebnisse=cell2mat(distanzVarianzErgebnisse);
%     [M,I] = min(distanzVarianzErgebnisse(1,:));
%     vogelId=distanzVarianzErgebnisse(2,I);
%
%     vogelName=char(nameFolds(vogelId)); % gib den namen des vogels
aus
```

```
%
%     filesXML=dir('test/*.xml'); % aus XML richtige spezies
herauslesen
%     xmldata = xml_load(filesXML(i).name);
%     species = xmldata.Species;
%
%     erkanntVarianz=strcmp(vogelName,species); % auf gleichheit
überprüfen
%     if erkanntVarianz==1
%         accuracyVarianz=accuracyVarianz+1;
%     end
%     checkString=strcat('erkannt ueber varianz: ',vogelName,' -
richtig: ',species,' - erkannt?: ',num2str(erkanntVarianz));
%     disp(checkString); % ergebnis ausgeben



    % häufigster Mittelwert
    vogelCheck=[] % erstelle matrix mit allen mininum-ids
    for g=1:size(testWoerterMatrix,1)
        vogelCheck=[vogelCheck distanzMittelwertCell{3,g}];
    end
    haeufigsteId=mode(vogelCheck,2); % schau, welcher vogel am
häufigsten am nähesten ist
    vogelName=char(nameFolds(haeufigsteId)); % gib den namen des
vogels aus


    filesXML=dir('test/*.xml'); % aus XML richtige spezies herauslesen
    xmldata = xml_load(filesXML(i).name);
    species = xmldata.Species;


    erkanntHaeufigsterMittelwert=strcmp(vogelName,species); % auf
gleichheit überprüfen
    if erkanntHaeufigsterMittelwert==1
        accuracyHaeufigsterMittelwert=accuracyHaeufigsterMittelwert+1;
    end
    checkString=strcat('erkannt ueber haeufigsten mittelwert:
',vogelName,' - richtig: ',species,' - erkannt?:
',num2str(erkanntHaeufigsterMittelwert));
    disp(checkString); % ergebnis ausgeben



%     % nähester Mittelwert
%     distanzMittelwertErgebnisse=distanzMittelwertCell(4,:);
%     distanzMittelwertErgebnisse=[distanzMittelwertErgebnisse;
distanzMittelwertCell(5,:)];
%
distanzMittelwertErgebnisse=cell2mat(distanzMittelwertErgebnisse);
%     [M,I] = min(distanzMittelwertErgebnisse(1,:));
%     vogelId=distanzMittelwertErgebnisse(2,I);
%
%     vogelName=char(nameFolds(vogelId)); % gib den namen des vogels
aus
%
```

```
%     filesXML=dir('test/*.xml'); % aus XML richtige spezies
herauslesen
%     xmldata = xml_load(filesXML(i).name);
%     species = xmldata.Species;
%
%     erkanntMittelwert=strcmp(vogelName,species); % auf gleichheit
überprüfen
%     if erkanntMittelwert==1
%         accuracyMittelwert=accuracyMittelwert+1;
%     end
%     checkString=strcat('erkannt ueber mittelwert: ',vogelName,' -
richtig: ',species,' - erkannt?: ',num2str(erkanntMittelwert));
%     disp(checkString); % ergebnis ausgeben



    % häufigste Varianz+Mittelwert
    vogelCheck=[] % erstelle matrix mit allen mininum-ids
    for g=1:size(testWoerterMatrix,1)
        vogelCheck=[vogelCheck distanzVarUndMitCell{3,g}];
    end
    haeufigsteId=mode(vogelCheck,2); % schau, welcher vogel am
häufigsten am nähesten ist
    vogelName=char(nameFolds(haeufigsteId)); % gib den namen des
vogels aus


    filesXML=dir('test/*.xml'); % aus XML richtige spezies herauslesen
    xmldata = xml_load(filesXML(i).name);
    species = xmldata.Species;


    erkanntHaeufigsteVarUndMit=strcmp(vogelName,species); % auf
gleichheit überprüfen
    if erkanntHaeufigsteVarUndMit==1
        accuracyHaeufigsteVarUndMit=accuracyHaeufigsteVarUndMit+1;
    end
    checkString=strcat('erkannt ueber haeufigste varianz und
mittelwert: ',vogelName,' - richtig: ',species,' - erkannt?:
',num2str(erkanntHaeufigsteVarUndMit));
    disp(checkString); % ergebnis ausgeben



%     % näheste Varianz+Mittelwert
%     distanzVarUndMitErgebnisse=distanzVarUndMitCell(4,:);
%     distanzVarUndMitErgebnisse=[distanzVarUndMitErgebnisse;
distanzVarUndMitCell(5,:)];
%     distanzVarUndMitErgebnisse=cell2mat(distanzVarUndMitErgebnisse);
%     [M,I] = min(distanzVarUndMitErgebnisse(1,:));
%     vogelId=distanzVarUndMitErgebnisse(2,I);
%
%     vogelName=char(nameFolds(vogelId)); % gib den namen des vogels
aus
%
%     filesXML=dir('test/*.xml'); % aus XML richtige spezies
herauslesen
```

```
%     xmldata = xml_load(filesXML(i).name);
%     species = xmldata.Species;
%
%     erkanntVarUndMit=strcmp(vogelName,species); % auf gleichheit
überprüfen
%     if erkanntVarUndMit==1
%         accuracyVarUndMit=accuracyVarUndMit+1;
%     end
%     checkString=strcat('erkannt ueber varianz und mittelwert:
',vogelName,' - richtig: ',species,' - erkannt?:
',num2str(erkanntVarUndMit));
%     disp(checkString); % ergebnis ausgeben


%%
end %% ende for schleife files

    % ----------------------EVALUIERUNG--------------------------


% häufig
accDist=accuracyHaeufigsteDistanz/i*100;
accVar=accuracyHaeufigsteVarianz/i*100;
accMit=accuracyHaeufigsterMittelwert/i*100;
accVarUndMit=accuracyHaeufigsteVarUndMit/i*100;


% nähe
% accDist=accuracyDistanz/i*100;
% accVar=accuracyVarianz/i*100;
% accMit=accuracyMittelwert/i*100;
% accVarUndMit=accuracyVarUndMit/i*100;


ergebnisse=strcat('accuracy Distanz: ',num2str(accDist),'% accuracy
Varianz: ',num2str(accVar),'% accuracy Mittelwert:
',num2str(accMit),'% accuracy varianz und mittelwert:
',num2str(accVarUndMit),'% uebersprungen: ',num2str(uebersprungen));
disp(ergebnisse);
```