Exzellenzcluster
Cognitive Interaction Technology
Ambient Intelligence
Dr. Thomas Hermann

Bachelorarbeit

Kognitive Informatik

Akustische Szenenerkennung auf Android-Mobilgeräten

Tobias Rodehutskors

(Matr.-Nr.: 1917608)

Betreuer:

Dipl. Inform. Christian Leichsenring, M. Sc. Sebastian Hammerl

Bielefeld
September 2011

Inhaltsverzeichnis

1 Einleitung

Das Ziel dieser Arbeit ist die Entwicklung eines Systems zur automatischen Erkennung von akustischen Szenen auf Mobilgeräten mit dem Android-Betriebssystem. Eine akustische Szene ist dabei die Gesamtheit der Geräusche, die das Mobilgerät zu einem Zeitpunkt umgeben. Akustische Szenenerkennung bezeichnet Verfahren, die nur aufgrund von akustischen Informationen, die das Mobilgerät umgebende Szene erkennen. Dies erlaubt die Anpassung der Software eines Mobilgeräts an den aktuellen Kontext des Benutzers, wodurch die Benutzung vereinfacht werden kann, oder Störungen des Benutzers vermieden werden können. So ist es zum Beispiel möglich, dass das Mobilgerät in einem Gespräch den Klingelton abschaltet oder Anrufe nicht annimmt, um dem Benutzer nicht zu unterbrechen.

Im Rahmen dieser Arbeit wird eine Evaluation verschiedener Klassifikatoren durchgeführt, die für die Erkennung akustischer Szenen eingesetzt werden können. Ebenso wird untersucht welchen Einfluss die Wahl verschiedener Parameter einer Merkmalsextraktion aus Audiodaten auf die Güte der Erkennung hat.

Die Anwendung Earacle wird als Implementierung eines solchen Systems vorgestellt. Earacle kann durch den Benutzer auf die Erkennung verschiedener akustischer Szenen trainiert werden und stellt eine Schnittstelle bereit mit der andere Anwendungen die Erkennungsfähigkeit von Earacle nutzen können.

2 Verwandte Arbeiten

Das in dieser Arbeit verwendete System versucht das Audiosignal ohne Separation in einzelne Geräusche zu interpretieren (Acoustic Scene Recognition). Dies ist abzugrenzen von der Analyse akustischer Szenen (Acoustic Scene Analysis). Diese versucht die gesamte Geräuschkulisse einer Szene in einzelne Geräusche zu zerlegen und diese zu interpretieren, um eine Aussage über die gesamte Szene treffen zu können (vgl. Bregman [5]). Hörtests, die von Peltonen, Eronen, Parviainen, und Klapuri [14] durchgeführt wurden, haben gezeigt, dass der Mensch seine akustische Umgebung analysiert indem er sie in einzelne Geräusche zerlegt. Eine zukünftige Ausweitung des Systems in diese Richtung scheint also sinnvoll zu sein.

Von Sawhney und Maes [16] wurden verschiedene Merkmalsextraktionen und Klassifikatoren zur Erkennung akustischer Umgebungen verglichen. Wobei die Testbeispiele einem der fünf vorgegebenen Klassenlabel *People*, *Subway*, *Traffic*, *Voice* und *Other* zugeordnet werden mussten. Die besten Resultate wurden in diesen Versuchen mit einer Merkmalsextraktion auf Basis des Frequenzbereichs des Audiosignals und einem einfachen Nearest-Neighbor-Klassifikator erzielt. Mit diesem System wurden 68% der Testbeispiele richtig klassifiziert. Die benutze Merkmalsextraktion ähnelt der Berechnung von Mel-Frequency-Cepstral-Coefficients, die in Abschnitt 3.2 beschrieben und in dieser Arbeit als Merkmalsextraktion verwendet wird. Der Nearest-Neighbor-Klassifikator ist ein Spezialfall des k-Nearest-Neighbor-Klassifikators, der in Abschnitt 3.1 erläutert wird.

Peltonen, Tuomi, Klapuri, Huopaniemi, und Sorsa [13] untersuchten ebenfalls die Erkennung von alltäglichen akustischen Szenen. Dabei wurden 17 mögliche Klassenlabel vorgegeben. Durch die Benutzung von MFCCs und einem Gaussian-Mixture-Model konnte eine Klassifikationsrate von 63,8% erreicht werden. Wenig schlechter schnitt das Modell bestehend aus einer Merkmalsextraktion basierend auf dem Verhältnis der Energie verschiedener Frequenzbereiche des Audiosignals in Verbindung mit einem Nearest-Neighbor-Klassifikator ab. Durch Generalisierung der 17 Klassenlabel auf 6 Klassenlabel konnte eine Steigerung der Klassifikationsgenauigkeit auf 68,6% erzielt werden. Um eine gute Klassifikationsgenauigkeit zu erreichen, wird daher in dieser Arbeit ebenfalls eine geringe Anzahl von Klassenlabel verwendet. Es wurde zudem gezeigt, dass durch eine Kombination verschiedener Merkmalsextraktionen die Klassifikationsgenauigkeit gesteigert werden kann. Da jedoch ein Mehraufwand bei

der Vorverarbeitung des Audiosignals entsteht, ist dieser Ansatz auf Mobilgeräten als kritisch anzusehen.

Die Erkennung akustischer Szenen wurde von Lukowicz et al. [10] eingesetzt, um Aktivitäten in einer Werkstatt zu erkennen. Dazu wurde der Benutzer sowohl mit Beschleunigungssensoren als auch mit zwei Mikrophonen ausgestattet. Ein Mikrophon wurde am Handgelenk angebracht, ein weiteres im Hüftbereich. Die Einordnung von Geräuschen wurde in eine von drei Klassen vorgenommen. Durch die verschiedenen Positionierungen der Mikrophone konnte zusätzlich deren Intensitätsunterschied benutzt werden, um die Quelle eines Geräusches zu lokalisieren. Durch die Fusion der akustischen Informationen mit den Daten der Beschleunigungssensoren konnte eine sehr gute Klassifikationsgenauigkeit bei der Identifikation verschiedener Aufgaben im Werkstattbereich erreicht werden. Ziel der Erkennung von Arbeitsschritten in diesem Umfeld ist, den Benutzer auf einen vergessenen Arbeitsschritt hinzuweisen oder ihm Zusatzinformationen zur aktuellen Arbeit zu präsentieren. Da die in dieser Arbeit verwendeten Mobilgeräte nur über ein Mikrofon verfügen, ist eine Lokalisierung der Geräusche nicht möglich. Durch eine Fusion von Informationen verschiedener Sensoren kann jedoch die Klassifikationsgenauigkeit, der in dieser Arbeit vorgestellte Szenenerkennung, verbessert werden.

Eine weitere Anwendung für die akustische Szenenerkennung findet sich im Bereich der Hörgeräte, die von Hamacher et al. [8] beschrieben wird. Moderne Hörgeräte verfügen über eine Vielzahl von unterschiedlichen Algorithmen, die für spezifische Situationen ausgelegt sind. Wichtig ist die richtige Auswahl der Algorithmen und deren Parametrisierung passend zur aktuellen akustischen Umgebung. Eine manuelle Auswahl der richtigen Einstellungen ist, ebenso wie ständige Änderungen der Konfiguration eines Mobilgeräts, unpraktikabel für den Benutzer, so dass ein System zur Erkennung von akustischen Szenen benutzt werden kann, um diese Auswahl zu treffen.

Zulkernain, Madiraju, Ahamed und Stamm [24] stellten ein System vor, dass die Unterbrechung durch Anrufe minimieren soll. Dazu entscheidet das System bei jedem eingehenden Anruf in welcher Situation sich der Benutzer zur Zeit befindet, und ob der Anruf in der gegebenen Situation störend ist und somit nicht angenommen werden soll. Zur Bestimmung des aktuellen Kontextes des Benutzers verwendet das System Informationen aus dem Kalender des Benutzers, die Tageszeit und Informationen über den aktuellen Aufenthaltsort und baut anhand dieser Informationen einen Entscheidungsbaum auf. Dieses System könnte durch die Einbeziehung des akustischen Kontextes noch bessere Entscheidungen für den Nutzer ermöglichen, um zum Beispiel in einem Gespräch, das nicht im Kalender des Benutzers eingetragen ist, einen Anruf nicht anzunehmen.

3 Erkennung akustischer Szenen durch Klassifikation

Zur Bestimmung der aktuellen akustischen Szene wird ein Klassifikator eingesetzt. Ein Klassifikator ist ein überwachtes Lernverfahren, dass Eingabedaten auf eine diskrete Menge von Ausgaben abbildet. Die Parameter des Modells, d.h. die Zuordnung zwischen Ein- und Ausgaben, werden dabei durch Beispiele gelernt. Ein Beispiel besteht aus einem Merkmalsvektor als Eingabe und einem oder mehreren zugeordneten Klassenlabeln als Sollausgabe. Ziel des Trainings eines Klassifikators ist es ein generalisiertes Modell zu erhalten, dass unbekannten Merkmalsvektoren ein Klassenlabel zuordnen kann. Dieser Vorgang wird Klassifikation genannt. Dazu bestimmt der Klassifikator üblicherweise für jedes mögliche Klassenlabel eine Konfidenz, d.h. einen Wert, der das Vertrauen des Klassifikators in die Richtigkeit der Zuordnung von Merkmalsvektor und Klassenlabel ausdrückt. Das Klassenlabel mit der höchsten Konfidenz ist das Ergebnis der Klassifikation.

Damit der Klassifikator nicht direkt auf den hochdimensionalen Audiodaten arbeiten muss, findet zuerst eine Vorverarbeitung statt. Eine solche Vorverarbeitung ist die in Abschnitt 3.2 beschriebene Erzeugung von MFCCs aus den Audiodaten. Das Audiomaterial wird in kurze Fenster zerlegt und für jedes Fenster werden MFCCs erzeugt. Die MFCCs eines Fensters bilden einen Merkmalsvektor. Um Paare aus Merkmalsvektoren und Klassenlabeln zu erhalten wird einer Menge von Merkmalsvektoren per Hand ein Klassenlabel zugewiesen. Die Menge der Paare wird dann in einen Trainingsdatensatz und einen Testdatensatz geteilt. Der Klassifikator wird mit dem Trainingsdatensatz trainiert. Die Güte der Klassifikation wird mit Hilfe des Testdatensatzes gemessen. In Abschnitt 4.1 wird der Aufbau eines Trainings- und Testdatensatzes beschrieben.

Es wird zwischen Singlelabel- und Multilabel-Klassifikation unterschieden. Bei der Singlelabel-Klassifikation ist jedem Merkmalsvektor aus der Menge der Beispiele genau ein Klassenlabel zugeordnet. Bei einer Multilabel-Klassifikation können jedem Merkmalsvektor mehrere Klassenlabel zugeordnet sein.

3.1 k-Nearest-Neighbor-Algorithmus

Ein Sonderfall der Klassifikation ist das instanzbasierte Lernen. Dabei werden in der Trainingsphase des Algorithmus keine Parameter für das Modell bestimmt, sondern es findet erst eine Auswertung des Modells statt, wenn ein unbekanntes Beispiel klassifiziert werden soll. Dazu werden alle Beispiele mit ihren bekannten zugehörigen Klassenlabeln gespeichert. Ein solcher instanzbasierter Lernalgorithmus ist der k-Nearest-Neighbor-Algorithmus (kNN).

Zur Klassifikation neuer unbekannter Beispiele sucht der kNN-Algorithmus im Eingaberaum der Daten nach k bekannten Beispielen, die dem unbekannten Beispiel bezüglich einer Metrik am nächsten sind. Das Klassifikationsergebnis für das zu klassifizierende Beispiel wird jetzt aus einer Mittelung der Klassenlabel der Nachbarn gewonnen.

Ein Beispiel für eine Klassifikation ist Abbildung 3.1 zu entnehmen. Es gibt zwei mögliche Klassenlabel, blau und rot. Bekannte Beispiele werden durch blaue und rote Kreise dargestellt. Im Beispiel wird $k = 5$ gewählt. Die fünf nächsten Nachbarn des zu klassifizierenden Beispiels v sind alle dem blauen Klassenlabel zugeordnet. Daher wird der Merkmalsvektor v ebenfalls als dem blauen Klassenlabel zugehörig klassifiziert. Im Gegensatz dazu wird der Merkmalsvektor w dem roten Klassenlabel zugeordnet. Hierbei ist jedoch zu beachten, dass nicht alle fünf Nachbarn des Merkmalsvektors w dem roten Klassenlabel angehören. Dadurch, dass ein Nachbar dem blauen Klassenlabel angehört, beträgt die Konfidenz für das blaue Klassenlabel 1/5, die Konfidenz für das rote Klassenlabel 4/5.

Eine Verbesserung der Klassifikation kann durch verschiedene Abwandlungen des kNN-Algorithmus erreicht werden. So ist es zum Beispiel möglich die Klassifikation der Nachbarn mit dem reziproken Abstand zwischen zu klassifizierendem und bekanntem Beispiel zu gewichten.

Der kNN-Algorithmus ist sowohl für Singlelabel-Klassifikationen, als auch durch Problemtransformationen für Multilabel-Klassifikationen anwendbar.

3.2 Mel-Frequency-Cepstral-Coefficients

Damit der Klassifikationsalgorithmus nicht direkt auf den aufgenommenen hochdimensionalen Audiodaten arbeiten muss, wird eine Vorverarbeitung durchgeführt. Aufgabe dieser Vorverarbeitung ist es, zum einen für das zu lösende Problem, in diesem Fall die Klassifikation von akustischen Szenen, wichtige Eigenschaften in den Eingabedaten zu finden, zum anderen eine Dimensionsreduktion durchzuführen.

Eine in der Spracherkennung weit verbreitete Vorverarbeitung sind die Mel-Frequency-Cepstral-Coefficients (MFCCs). Die MFCCs sind motiviert durch die Art der Wahrneh-

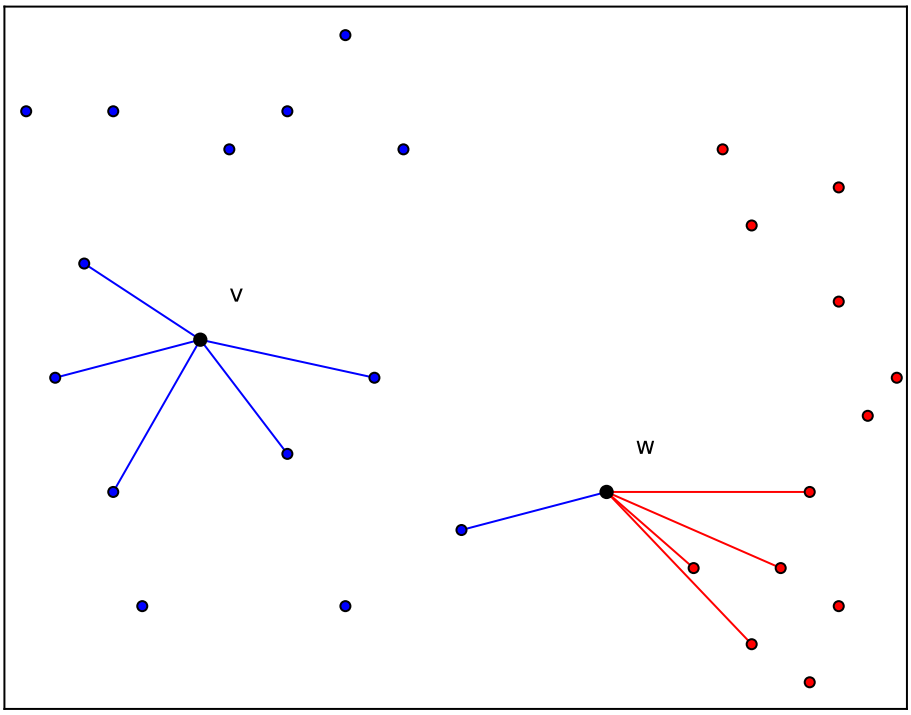

Abbildung 3.1: Klassifikation durch einen kNN-Klassifikator. Der zu klassifizierende Merkmalsvektor wird dem Klassenlabel zugeordnet, dass die Mehrheit der fünf nächsten Nachbarn beschreibt.

mung von Geräuschen durch den Menschen. Sie werden aus den Audiodaten erzeugt und repräsentieren in kompakter Form das Frequenzspektrum. Eine Beschreibung der Gewinnung der MFCCs wurde von Logan et al. [9] gegeben und wird an dieser Stelle kurz zusammengefasst.

Zur Gewinnung dieser Merkmale wird das aufgenommene Audiosignal in einzelne Fenster unterteilt. Für jedes Fenster werden MFCCs berechnet. Mit Hilfe einer diskreten Fouriertransformation wird für jedes Fenster das Amplitudenspektrum aus den Audiodaten berechnet. Im nächsten Schritt wird der Logarithmus des Amplitudenspektrums berechnet, da die wahrgenommene Lautstärke eines Signals als approximativ logarithmisch angenommen werden kann. Anschließend wird die Anzahl der Frequenzbänder reduziert. Zu diesem Zweck werden Dreiecksfilter benutzt. Die Dreiecksfilter sind jedoch nicht gleichmäßig über den abzudeckenden Frequenzraum verteilt, sondern folgen der Mel-Skala. Die Mel-Skala drückt den Zusammenhang zwischen tatsächlicher Frequenz und der vom Menschen wahrgenommenen Tonhöhe aus und wurde zuerst von Stevens,

Volkmann und Newman [18] beschrieben. Eine populäre Approximation der Mel-Skala wurde von O'shaughnessy [12] vorgeschlagen:

$$M(f) = 2595 * \lg(1 + \frac{f}{700}) \tag{3.1}$$

Abbildung 3.2 zeigt die Approximation der Mel-Skala. Der Zusammenhang zwischen Frequenz und Tonhöhe ist im unteren Frequenzbereich bis 1000 Hz annähernd linear, während er über 1000 Hz stark logarithmisch ist.

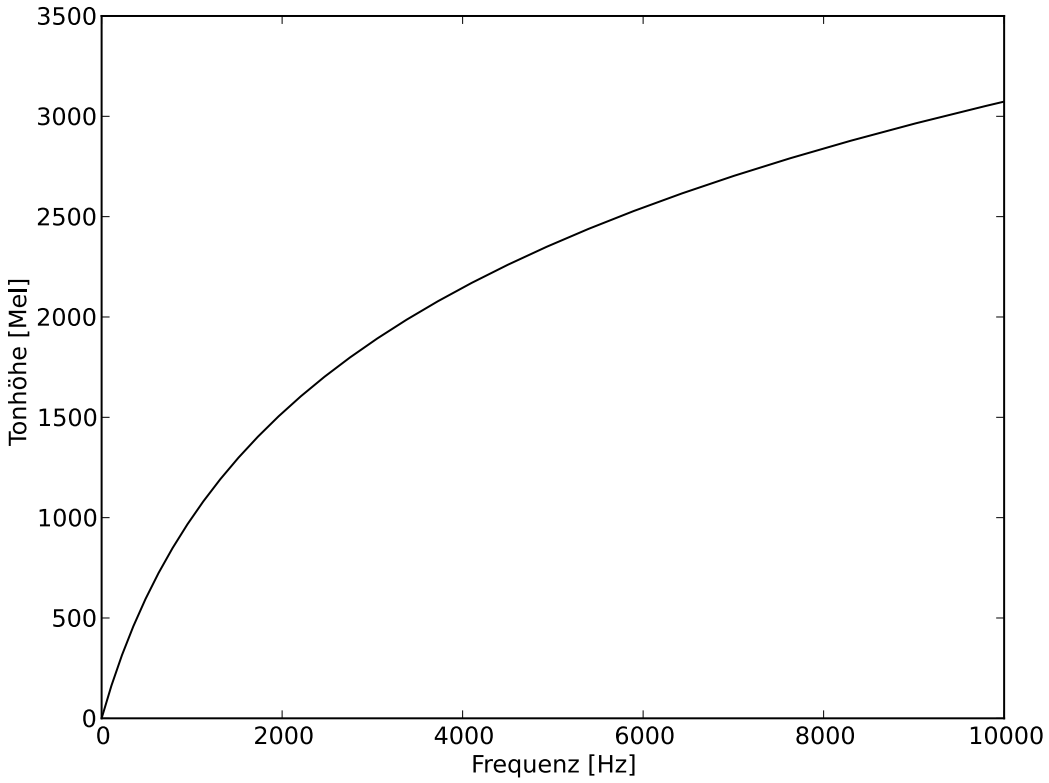

Abbildung 3.2: Die Mel-Skala drückt den Zusammenhang zwischen tatsächlicher Frequenz und der vom Menschen wahrgenommenen Tonhöhe aus.

Die Aufteilung der Dreiecksfilter mit Hilfe der Mel-Skala sorgt dafür, dass der untere Frequenzbereich stärker abgedeckt ist als der obere Frequenzbereich, da tiefere Frequenzen für die menschliche Wahrnehmung wichtiger sind als höhere Frequenzen [9].

Da die Ergebnisse der Dreiecksfilterung stark korreliert sind, wird abschließend eine diskrete Kosinus-Transformation durchgeführt, welche für eine Dekorrelation sorgt. Die Ergebnisse der Dreiecksfilterung werden dafür als Signal aufgefasst. Es werden üblicherweise 13 MFCCs erzeugt.

4 Evaluation verschiedener Klassifikatoren

In diesem Abschnitt wird untersucht, wie sich die Wahl von unterschiedlichen Klassifikatoren auf das Klassifikationsergebnis auswirkt. Des Weiteren wird die Auswirkung der Wahl unterschiedlicher Parameter, die die Extraktion der Merkmale aus den Audiodaten beeinflussen, evaluiert. Dazu werden zuerst die verschiedenen zur Evaluation benutzten Datensätze vorgestellt, die sich in einen Singlelabel- und einen Multilabel-Datensatz unterteilen. Es werden sowohl Singlelabel- als auch Multilabel-Klassifikatoren untersucht. Die verwendeten Multilabel-Klassifikatoren MLkNN [23], BRkNN [17] und IBLR-ML [6] basieren alle auf einer Abwandlung des k-Nearest-Neighbor-Klassifikators (kNN), der, wie in Kapitel 2 beschrieben, bereits erfolgreich zur Erkennung akustischer Szenen eingesetzt wurde. Es werden jeweils die zehn nächsten Nachbarn betrachtet. Zusätzlich wird der naiver Bayes-Klassifikator [15] als Multilabel-Klassifikator untersucht, wobei eine Problemtransformation über die Label-Powerset-Methode [19] stattfindet (naiver Bayes mit LP). Als Singlelabel-Klassifikatoren werden ein kNN-Klassifikator und ein naiver Bayes-Klassifikator untersucht.

Es werden die Metriken vorgestellt anhand derer die Güte des Klassifikationsergebnisses beurteilt wird. Als letztes wird eine empirische Auswertung vorgenommen.

4.1 Test- und Trainingsdatensätze

Um die Auswirkungen der unterschiedlichen Wahlen der Parameter, die die Extraktion der Merkmale beeinflussen, als auch den Einfluss von unterschiedlichen Klassifikatoren auf das Klassifikationsergebnis zu untersuchen, wurde ein eigener Datensatz von Audiomaterial aufgenommen. Dies ist erforderlich, da die Audiohardware eines Mobilgeräts im Allgemeinen nicht mit hochwertiger Audiohardware zu vergleichen ist. So leiden die Aufnahmen zum Beispiel unter einem starken Rauschen. Es liegt die Vermutung nahe, dass die Klassifikationsgenauigkeit unter der schlechteren Aufnahmequalität leidet.

Die Daten wurden mit dem eingebauten Mikrophon eines *HTC Desire Z* aufgenommen. Die Auflösung der Aufnahme betrug 16 Bit. Es wurde ein Kanal aufgenommen und eine Abtastrate von 44100 Hz gewählt. Diese Abtastrate ist auf allen Android-Mobilgeräten

garantiert verfügbar [3]. Andere Abtastraten werden nicht zwingend unterstützt. Um den Einfluss der Abtastrate auf das Klassifikationsergebnis zu untersuchen, wurden die aufgenommenen Daten auch auf eine Abtastrate von 8000 Hz überführt. Das Mobilgerät lag während der Aufnahmen entweder auf einer Oberfläche oder wurde in der Hand gehalten.

Es wurden insgesamt 125 Einzelaufnahmen mit einer Länge von jeweils mindestens 10 Sekunden erstellt. Die längste Aufnahme dauert 104 Sekunden. Die Gesamtdauer aller Aufnahmen beträgt ungefähr 1 Stunde und 23 Minuten.

Es wurden zwei unterschiedliche Datensätze aus den Beispieldaten erzeugt. Der Singlelabel-Datensatz erlaubt pro Beispiel nur die Zuweisung von einer Klasse, während der Multilabel-Datensatz die Zuweisung von mehreren Klassen zu einem Beispiel zulässt. Diese Unterscheidung wurde getroffen, um den Einfluss der Zuweisung von mehreren Klassenlabeln auf das Klassifikationsergebnis zu untersuchen. Da das Verbot von mehreren Zuweisungen zu einer akustischen Szene der Realität widerspricht, wurde den Beispielen des Singlelabel-Datensatzes jeweils das dominantere Klassenlabel zugeordnet. Also dasjenige Klassenlabel, das die Aufnahme besser beschreibt, da es auffallender ist.

Die Beispieldaten wurden den Klassenlabeln *Speech*, *Music*, *Traffic*, *Crowd* und *Silence* zugeordnet. Das Klassenlabel *Silence* umfasst hauptsächlich Aufnahmen von Gesprächen sowie teilweise Aufnahmen von Nachrichten- oder Radiosendungen. Unter dem Klassenlabel *Music* wurden Aufnahmen verschiedener Musikrichtungen zusammengefasst. Dabei wurden fast nur Aufnahmen aus dem Radio oder von digitalen Tonträgern berücksichtigt. Das Klassenlabel *Traffic* umfasst Fahrten mit verschiedenen Verkehrsmitteln, wie zum Beispiel Autos, Zügen, Straßenbahnen oder Bussen, sowie Szenen in denen die Aufnahme in der Nähe von Verkehrsmitteln angefertigt wurde, wie zum Beispiel auf dem Gehweg an einer Straße. Das Klassenlabel *Crowd* umfasst Situationen in denen eine Menge von Menschen zusammentrifft ohne das dabei einzelne Gespräche erkennbar sind. Es wurden Aufnahmen in Fußgängerzonen, Restaurants und Bahnhofshallen angefertigt. Das Klassenlabel *Silence* umfasst verschiedene ruhige Situationen, in denen keine dominante Tonquelle zu erkennen ist. Diese Unterteilung schließt natürlich eine große Menge an Situationen aus. Im Rahmen der Evaluation wurde jedoch diese Einschränkung getroffen, um eine übersichtliche Testmenge zu erhalten.

Die Anzahl der für jedes Klassenlabel aufgenommenen Daten und deren Gesamtlänge ist für den Multilabel-Datensatz der Tabelle 4.1 und für den Singlelabel-Datensatz der Tabelle 4.2 zu entnehmen. Jede Aufnahme des Multilabel-Datensatzes ist höchstens mit drei unterschiedlichen Klassenlabeln versehen. Die Kardinalität des Multilabel-Datensatzes beträgt 1.19. Es wurden ganze Aufnahmen entweder dem Test- oder dem Trainingsdatensatz zugeordnet. Der Gesamtdatensatz wurde so getrennt, dass ungefähr

	Trainingsdaten		Testdaten		Gesamt	
	Aufnahmen	Länge [s]	Aufnahmen	Länge [s]	Aufnahmen	Länge [s]
Traffic	34	1630	19	808	53	2438
Speech	19	736	13	376	32	1112
Music	14	533	6	290	20	823
Crowd	25	798	9	399	34	1197
Silence	8	262	4	119	12	381

Tabelle 4.1: Anzahl der Aufnahmen und deren Gesamtlänge aufgeschlüsselt über die verschiedenen Klassenlabel des Multilabel-Datensatzes

	Trainingsdaten		Testdaten		Gesamt	
	Aufnahmen	Länge [s]	Aufnahmen	Länge [s]	Aufnahmen	Länge [s]
Traffic	30	1399	16	737	46	2136
Speech	16	655	12	317	28	972
Music	11	437	5	252	16	689
Crowd	19	606	4	203	23	809
Silence	8	262	4	119	12	381

Tabelle 4.2: Anzahl der Aufnahmen und deren Gesamtlänge aufgeschlüsselt über die verschiedenen Klassenlabel des Singlelabel-Datensatzes

zwei Drittel der Daten als Trainingsdaten und ein Drittel als Testdaten zu Verfügung stehen.

Es wird davon ausgegangen, dass die Daten zu einem gewissen Grad falsch deklariert sind, da die Zuweisung der einzelnen Beispiele zu den Klassenlabeln nur von einer Person durchgeführt wurde und sowohl das Erkennen einer akustischen Szene als auch das Erkennen der dominanten Szene einer Beispielaufnahme teilweise subjektiv sind.

4.2 Metriken zur Evaluation von Singlelabel- und Multilabel-Klassifikation

Um die Güte einer Klassifikation zu messen sind verschiedene Metriken vorgeschlagen worden. In dieser Arbeit werden die Metriken Classification-Accuracy, One-Error und Hamming-Loss verwendet. Ein besonderes Problem an dieser Stelle ist die Vergleichbarkeit von Singlelabel-Klassifikation und Multilabel-Klassifikation, da beide unterschiedliche Eigenschaften aufweisen.

Zur Beschreibung der verschiedenen Metriken wird folgende Notation verwendet. Sei D eine Menge von Beispielen. Sei weiter L die Menge aller möglichen Klassenlabel.

Jedes Beispiel aus D besteht aus einem Merkmalsvektor x_i und einer Menge von Klassenlabeln $Y_i \subseteq L$. Falls $|Y_i| = 1$ für jedes Tupel aus D, dann ist jedem Beispiel genau ein Klassenlabel zugeordnet und es handelt sich um eine Singlelabel-Klassifikation. Ist jedoch $|Y_i| > 1$ für irgendein Tupel aus D, dann ist mindestens einem Tupel mehr als ein Klassenlabel zugeordnet und es handelt sich um eine Multilabel-Klassifikation.

Sei ferner $h(\cdot)$ ein Klassifikator und $Z_i \subseteq L$ das Ergebnis der Klassifikation durch $h(\cdot)$.

Dann lässt sich die Classification-Accuracy wie durch Tsoumakas und Vlahavas [22] angegeben, durch

$$Classification\text{-}Accuracy(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} I(Z_i = Y_i) \tag{4.1}$$

berechnen, wobei

$$I(true) = 1, I(false) = 0 \tag{4.2}$$

ist. Diese Metrik gibt den Prozentsatz der Beispiele an, die richtig klassifiziert wurden. Der Wert dieser Metrik liegt zwischen 0 und 1, 0 falls alle Beispiele falsch markiert wurden, 1 falls alle Beispiele richtig markiert wurden. Diese Metrik kann sowohl auf Singlelabel-Klassifikationen als auch auf Multilabel-Klassifikationen angewandt werden. Hierbei ist jedoch zu beachten, dass im Falle einer Multilabel-Klassifikation alle Klassenlabel korrekt erkannt werden müssen, damit die Klassifikation als Erfolg gewertet wird. Gibt die Klassifikation nicht alle tatsächlichen Klassenlabel eines Beispiels an wird die Klassifikation als Misserfolg gewertet. Dies verdeutlicht, dass diese Metrik im Falle einer Multilabel-Klassifikation viel strikter ist als bei einer Singlelabel-Klassifikation, da mehrere Fälle zu einem Misserfolg führen können. Ein Vergleich einer Multilabel-Klassifikation mit einer Singlelabel-Klassifikation anhand dieser Metrik ist also nur unter Berücksichtigung dieses Umstandes durchzuführen.

Sei $f(\cdot, \cdot)$ eine Funktion, die für ein Datenbeispiel x_i und ein Klassenlabel $l \in L$ einen Rang ermittelt, so dass Klassenlabel mit einer besseren Passung zu einem Beispiel einen höheren Rang erhalten als Klassenlabel mit einer schlechteren Passung.

Dann lässt sich wie von Zhang und Zhou [23] beschrieben, der One-Error berechnen als

$$One\text{-}Error(f, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \arg\max_{l \in L} f(x_i, l) \notin Y_i \tag{4.3}$$

Der One-Error gibt somit an, wie häufig sich das Klassenlabel, dass von $f(\cdot, \cdot)$ als Klassenlabel mit der besten Passung angegeben wurde, nicht in der tatsächlichen Menge von Klassenlabeln befindet, die zu einem Beispiel gehören.

Im Falle einer Singlelabel-Klassifikation entspricht der One-Error dem Prozentsatz der falsch klassifizierten Instanzen. One-Error liefert somit eine Möglichkeit Singleabel- und Multilabel-Klassifikationen rudimentär zu vergleichen.

Eine der bekanntesten Metriken für Multilabel-Klassifikationen ist Hamming-Loss, der von Tsoumakas, Katakis und Vlahavas [20] als

$$Hamming\text{-}Loss(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|} \tag{4.4}$$

angegeben wird. Dabei bezeichnet Δ die symmetrische Differenz zwischen zwei Mengen. $|Y_i \Delta Z_i|$ ist also genau dann 0, falls $|Y_i = Z_i|$. Falls die Klassifikation nicht richtig ist, gibt $|Y_i \Delta Z_i|$ an, wie viele tatsächliche Klassenlabel nicht erkannt wurden und wie viele nicht vorhandene Klassenlabel fälschlicherweise vorhergesagt wurden.

Über diese Metriken hinaus sind auch weitere Faktoren für die Beurteilung eines Modells zu berücksichtigen. An dieser Stelle sind vor allem Kriterien zu nennen, die im Bezug zur Schonung von Ressourcen auf einer mobilen Plattform stehen. Dies betrifft den Aufwand für die Extraktion der Merkmale, den Aufwand des Trainings eines Klassifikators, die Speichergröße des Klassifikators, wie auch den Aufwand zum Klassifizieren einzelner unbekannter Instanzen.

4.3 Statistische Auswertung der Evaluation

Um den Einfluss der verschiedenen Parameter zu untersuchen wurde eine statistische Auswertung durchgeführt. Dazu wurden 100 mal zufällige Parameter gewählt. Mit jedem Satz von Parametern wurden dann die Merkmale aus den Trainings- und Testdaten extrahiert. Die Klassifikatoren wurden dann jeweils mit dem Trainingsdatensatz trainiert und mit dem Testdatensatz getestet. Es kann somit eine Untersuchung für jeden Klassifikator über 100 Experimente durchgeführt werden.

Dabei unterlag die Auswahl der Parameter einigen Einschränkungen. Die Breite der einzelnen Fenster, in die das Audiomaterial unterteilt wurde, lag zwischen 0,02 und 10 Sekunden. Benachbarte Fenster konnten sich bis zu 50% überlappen. Bis zu zehn benachbarte Fenster konnten zu einem Merkmalsvektor zusammengefasst werden mit dem dann der Klassifikator trainiert wurde. Falls benachbarte Fenster zu einem Merkmalsvektor zusammengefasst wurden, konnten sich diese Merkmalsvektoren bis zu 50% überlappen. Die Abtastrate konnte entweder 8000 oder 44100 Hz betragen.

	One-Error	Classification-Accuracy	Hamming-Loss
BRkNN	20,58	65,85	12,57
MLkNN	**16,55**	68,16	10,84
IBLR-ML	17,28	68,18	11,05
naiver Bayes mit LP	20,37	**71,66**	**10,49**

Tabelle 4.3: Ergebnisse der Klassifikation des Multilabel-Datensatzes mit Multilabel-Klassifikatoren gemittelt über 100 Experimente.

Es werden nur solche Parametersätze zugelassen, bei denen die damit extrahierten Merkmalsvektoren maximal eine Länge von 10 Sekunden abdecken. Diese Einschränkung bedeutet, dass die Parameter nicht unkorreliert gezogen werden können. So verhindert eine Fensterbreite von 6 Sekunden, dass zwei benachbarte Fenster zu einem Vektor zusammengefasst werden, da sonst insgesamt 12 Sekunden von diesem Vektor abgedeckt werden.

Da frühere Experimente eine besonders gute Klassifikationsrate bei der Extraktion der Merkmale durch MFCCs zeigten, wurden nur diese untersucht.

Tabelle 4.3 zeigt den Wert für jede der drei verwendeten Metriken, gemittelt über alle 100 Experimente, aufgeschlüsselt nach den verschiedenen benutzten Klassifikatoren auf dem Multilabel-Datensatz. Auffällig ist, dass der native Bayes-Klassifikator mit Problemtransformation sowohl in der Classification-Accuracy als auch im Hamming-Loss alle kNN-basierten Klassifikatoren aussticht. Von den kNN-basierten Klassifikatoren liefert MLkNN die besten Ergebnisse.

Tabelle 4.4 schlüsselt die Ergebnisse der Klassifikation des Singlelabel-Datensatzes mit Multilabel-Klassifikatoren auf. Wie beim Multilabel-Datensatz dominiert auch hier wieder der naive Bayes-Klassifikator mit Problemtransformation. Dieser liefert jedoch bessere Ergebnisse als auf dem Multilabel-Datensatz. Dies lässt sich über die Arbeitsweise der Problemtransformation erklären. Sie erzeugt die Potenzmenge der Menge der Klassenlabel und führt für jedes Element der Potenzmenge eine eigene Klassifikation durch. So muss der Klassifikator im Falle des Multilabel-Datensatzes unter einer größeren Menge an Klassen unterscheiden als im Falle des Singlelabel-Datensatzes.

Insgesamt schneiden die Multilabel-Klassifikatoren bezüglich der One-Error-Metrik auf dem Multilabel-Datensatz besser ab, als auf dem Singlelabel-Datensatz. Dies lässt sich dadurch erklären, dass die One-Error-Metrik im Falle des Multilabel-Datensatzes das korrekte Vorhersagen eines von mehren Klassenlabeln belohnt.

Tabelle 4.5 stellt die Ergebnisse der Singlelabel-Klassifikatoren auf dem Singlelabel-Datensatz dar. Auch hier schneidet der naive Bayes-Klassifikator deutlich besser ab als der kNN-basierte Klassifikator.

	One-Error	Classification-Accuracy	Hamming-Loss
BRkNN	30,86	65,67	11,52
MLkNN	25,23	67,32	10,13
IBLR-ML	25,66	69,43	9,79
naiver Bayes mit LP	**21,71**	**78,53**	**8,59**

Tabelle 4.4: Ergebnisse der Klassifikation des Singlelabel-Datensatzes mit Multilabel-Klassifikatoren gemittelt über 100 Experimente.

	One-Error	Classification-Accuracy
kNN	30,59	69,41
naiver Bayes	**21,47**	**78,53**

Tabelle 4.5: Ergebnisse der Klassifikation des Singlelabel-Datensatzes mit Singlelabel-Klassifikatoren gemittelt über 100 Experimente.

Insgesamt ist zu bemerken, dass die kNN-basierten Klassifikatoren in allen Konstellationen zu ungefähr gleichen Ergebnissen führen. Bei Verwendung dieser Klassifikatoren stellt eine Zuordnung eines Beispiels zu mehreren Klassenlabeln also keinen Nachteil dar. Im Gegenteil dazu profitieren die naiven Bayes-Klassifikatoren von einem Singlelabel-Datensatz.

Da die folgenden Auswertungen bezüglich des Klassifikationsergebnisses unter Berücksichtigung der Fensterbreite bei allen Klassifikatoren und Datensätzen dem gleichen Trend folgen, sind diese in Grafik 4.1 gemittelt zusammengefasst. Bei der Mittelung über den Hamming-Loss- wurden die Singlelabel-Klassifikatoren nicht berücksichtigt.

Zu erkennen ist, dass die Classification-Accuracy bei Fensterbreiten zwischen 3 und 5 Sekunden ihr Maximum erreicht. Für die gleichen Fensterbreiten erreicht der Hamming-Loss sein Minimum. Der One-Error nimmt für Fensterbreiten zwischen 4 und 5 Sekunden günstige Werte an. Der Einfluss der Fensterbreite auf den Hamming-Loss ist gering, wohingegen die anderen beiden Metriken deutlicher von einer guten Wahl profitieren.

Der Einfluss der Anzahl der verbundenen Fenster auf das Klassifikationsergebnis wird in den Grafiken 4.2 und 4.3 dargestellt. Hier wurde zwischen den kNN-basierten Klassifikatoren und den Bayes-Klassifikatoren unterschieden, da diese einem gegenläufigen Trend folgen.

Während die kNN-basierten Klassifikatoren für zwei verbundene Fenster ihr Optimum erreichen, bevorzugen die Bayes-Klassifikatoren eine größere Anzahl von zusammengefassten Fenstern. Für Werte von 4 bis 8 nehmen sie ihr Optimum an. Hier ist jedoch zu bemerken, dass auf Grund der Korrelation zwischen den Parametern die Datengrundlage für viele zusammengefasste Fenster schlechter wird.

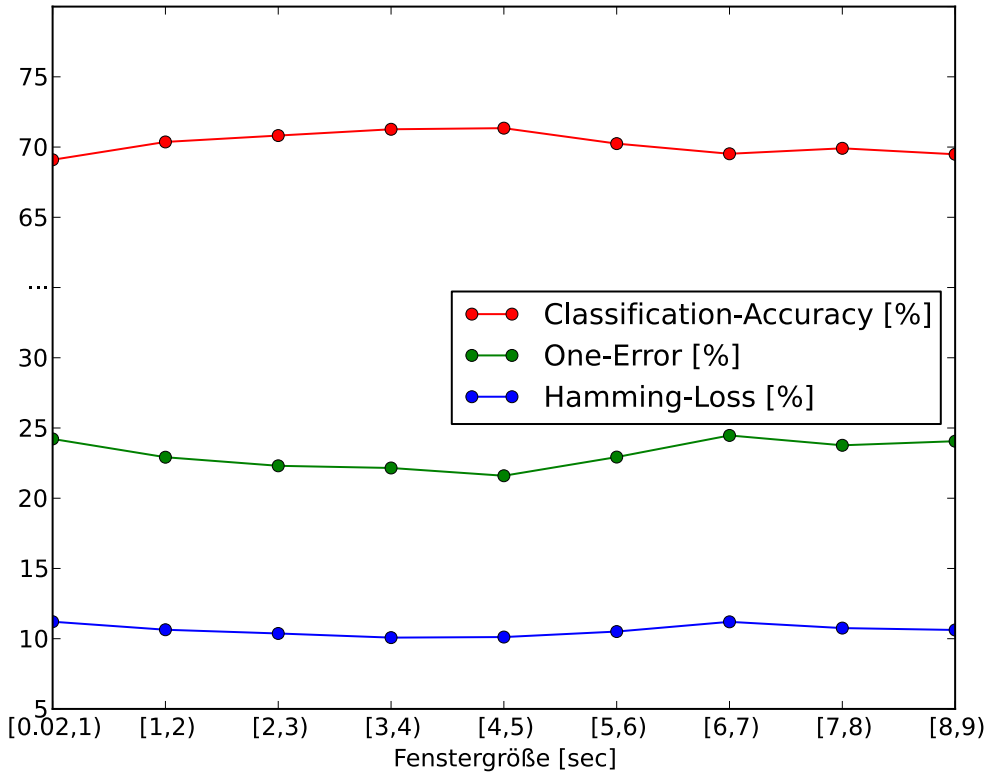

Abbildung 4.1: Mittelung der Classification-Accuracy, One-Error und Hamming-Loss unter Berücksichtigung der Fenstergröße.

In Tabelle 4.6 sind die Klassifikationsergebnisse, aufgeschlüsselt nach verwendeter Abtastrate, aufgelistet. Da sowohl bei den Klassifikatoren, die auf dem Singlelabel-Datensatz arbeiten, als auch denen, die auf dem Multilabel-Datensatz arbeiten, der gleiche Effekt zu erkennen ist, wurde auch hier grob nach den Klassifikatoren gruppiert. Hierbei ist zu bemerken, dass die kNN-basierten Klassifikatoren besser auf Daten arbeiten, die mit 8000 Hz abgetastet wurden, während die Bayes-Klassifikatoren besser auf den feiner abgetasteten Beispieldaten arbeiten. Da der Aufwand für die Berechnung der MFCCs linear mit der Anzahl der zu verarbeiteten Werte wächst, können die MFCCs bei einer Abtastrate von 8000 Hz ungefähr 5,5 mal schneller berechnet werden als bei einer Abtastrate von 44100 Hz. Die kNN-basierten Klassifikatoren haben an dieser Stelle also den Vorteil, dass sie besser auf den schneller extrahierbaren Daten arbeiten, während die naiven Bayes-Klassifikatoren auf den langsam extrahierbaren Daten bessere Ergebnisse erzielen.

Das Überlappen der einzelnen Fenster scheint keinen entscheidenden Einfluss auf das Klassifikationsergebnis auszuüben. Die Unterschiede zwischen der optimalen Wahl dieses

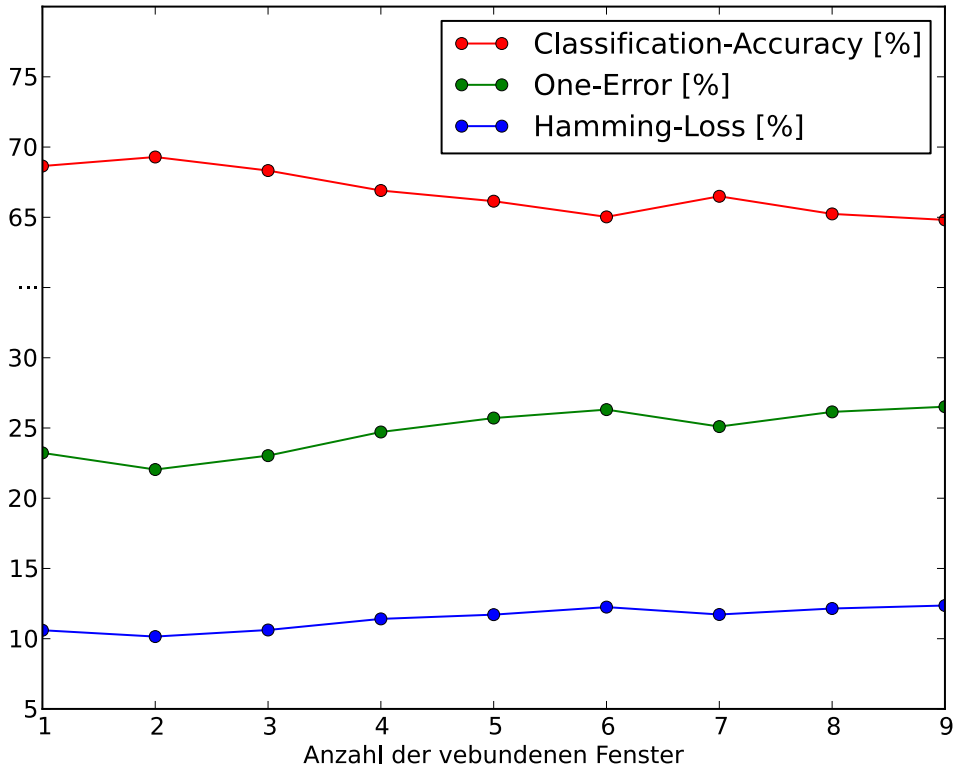

Abbildung 4.2: Classification-Accuracy, One-Error und Hamming-Loss gemittelt über alle kNN-basierten Klassifikatoren aufgetragen nach der Anzahl der verbundenen Fenster.

Parameters und der schlechtesten Wahl des Parameters für einen Klassifikator macht bei den meisten Metriken weniger als einen halben Prozentpunkt aus. Das Überlappen der einzelnen Fenster, dass in der Sprachverarbeitung sehr stark benutzt wird, oft mit einer Überlappung von 50%, scheint bei der Erkennung von akustischen Szenen keinen Einfluss zu haben. Dies mag daran liegen, dass im Gegensatz zur Spracherkennung, wo einzelne kurze Phoneme erkannt werden müssen, hier der Fokus auf die Hintergrundgeräusche fällt. Außerdem werden im Gegensatz zur Sprachverarbeitung relativ breite Fenster verwendet, was zu einer Mittelung führt. Das Überlappen der einzelnen Fenster ist also zu vernachlässigen. Insbesondere ist hier anzumerken, dass durch eine Überlappung der Fenster der Rechenaufwand erhöht wird. So verdoppelt sich der Aufwand für die Berechnung der MFCCs insgesamt bei einer Überlappung von 50%. Im Hinblick auf die Ressourcenschonung sollte also auf ein Überlappen der Fenster verzichtet werden.

Das Überlappen der Vektoren von benachbarten Fenstern scheint ebenfalls keinen entscheidenden Einfluss auf das Klassifikationsergebnis auszuüben. Ein eindeutiger

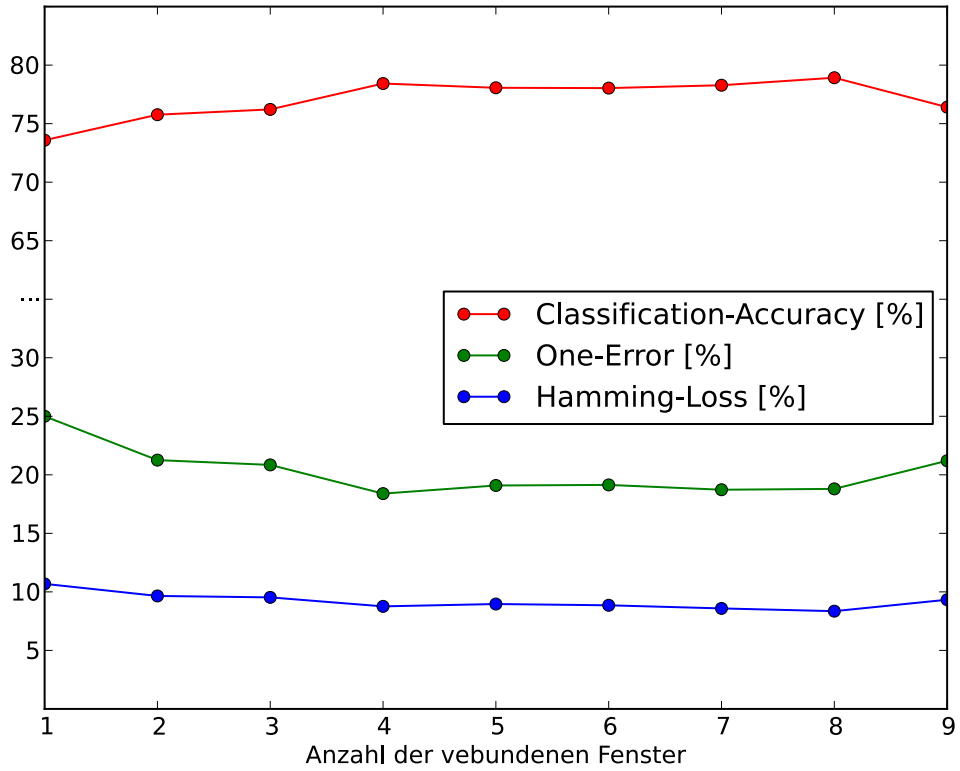

Abbildung 4.3: Classification-Accuracy, One-Error und Hamming-Loss gemittelt über alle naiven Bayes-Klassifikatoren aufgetragen nach der Anzahl der verbundenen Fenster.

Trend ist aus diesem Parameter nicht abzulesen. Zum Teil ist es so, dass ein Klassifikator in der einen Metrik besser auf eine höhere Überlappung anspricht, während in einer anderen Metrik genau das Gegenteil der Fall ist. Zudem macht die optimale Wahl gegenüber der schlechtesten Wahl des Parameters bei den meisten Klassifikatoren und Metriken weniger als einen halben Prozentpunkt Unterschied aus.

Die folgenden Messungen über die Dauer der einzelnen Schritte wurden nicht auf einem Mobilgerät sondern auf einem Desktopcomputer durchgeführt und sind nur untereinander vergleichbar.

Da die Klassifikatoren BRkNN und kNN ihr Modell erst auswerten, wenn eine Vorhersage getroffen werden muss, schneiden diese unter Betrachtung des Aufwands für das Training besonders gut ab. Auch die naiven Bayes-Klassifikatoren zeigen eine sehr gute Performance. Diese benötigen im Schnitt 31 ms für das Training des Klassifikators mit dem Trainingsdatensatz. Schlecht schneiden die Verfahren MLkNN und IBLR-ML ab.

	Classification-Accuracy		One-Error		Hamming-Loss	
	8000Hz	44100Hz	8000Hz	44100Hz	8000Hz	44100Hz
kNN	68,47	66,80	22,51	25,43	10,46	11,62
naiver Bayes	74,23	78,70	23,20	18,72	9,95	9,03

Tabelle 4.6: Ergebnisse der Klassifikation aufgeschlüsselt nach verwendeter Abtastrate und gruppiert nach kNN-basierten Klassifikatoren und naiven Bayes-Klassifikatoren.

Sie benötigen im Mittel 4026 ms für den Aufbau des Klassifikators. Dieser Wert wird vor allem von einigen Ausreißern stark gesteigert.

Da die kNN-basierten Klassifikatoren zur Vorhersage alle gespeicherten Instanzen durchsuchen müssen, schneiden sie mit einer durchschnittlichen Vorhersagezeit von 1863 ms für alle Testdaten deutlich schlechter ab als die naiven Bayes-Klassifikatoren. Sie benötigen im Schnitt nur 65 ms. Auch an dieser Stelle wird der Wert der kNN-basierten Klassifikatoren durch Ausreißer nach oben beeinflusst.

Da die kNN-basierten Verfahren alle Instanzen zur Laufzeit vorhalten müssen, liegt die Größe dieser Klassifikatoren im Speicher im Schnitt bei 624 kB, wohingegen die Bayes-Klassifikatoren im Schnitt nur 27 kB belegen.

Basierend auf den obigen Erkenntnissen wird ein neuer Parametersatz gewählt. Dabei wird die Fensterbreite auf 4,5 Sekunden gelegt. Es werden zwei Fenster verbunden und als Abtastrate werden 8000 Hz gewählt. Es wird keine Überlappung der Fenster und keine Überlappung der zusammengefassten Fenster genutzt. Als Klassifikator wird MLkNN gewählt, da dieser bezüglich der Metriken Classification-Accuracy, One-Error und Hamming-Loss auf dem Multilabel-Datensatz von den kNN-basierten Verfahren am besten abschneidet.

Der auf den so aus dem Multilabel-Datensatz extrahierten Daten arbeitende Klassifikator weißt mit einer Classification-Accuracy von 72,22%, einem One-Error von 8,33% und einem HammingLoss von 8,22% durchweg gute Werte auf. Mit 155 ms für den Aufbau des Klassifikators auf allen Trainingsdaten und 80 ms für die Vorhersagen aller Testdaten liegt er deutlich besser als der Durchschnitt der kNN-basierten Verfahren.

5 Implementierung von Earacle

Um das vorgestellte Modell zur Erkennung von akustischen Szenen zu testen, wurde Earacle entwickelt. Earacle ist eine Anwendung für das Mobilgeräte-Betriebssystem Android. Android ist ein quelloffenes Betriebssystem, das auf dem Linux-Kernel basiert und seit 2007 von der Open Handset Alliance, einem Konsortium aus Hard- und Software-Herstellern, Telekommunikations- und Marketingunternehmen, entwickelt wird [1, 4].

Android ermöglicht die Programmierung mit Java. Dazu verwendet Android eine eigene virtuelle Maschine, die den kompilierten Bytecode ausführt und für den Einsatz auf Mobilgeräten optimiert ist. Dem Entwickler steht eine umfangreiche API zur Verfügung, die den Zugriff auf viele Systemkomponenten und die einfache Entwicklung neuer Anwendungen erlaubt.

Eine Einführung in die Programmierung mit Android kann in dieser Arbeit nicht gegeben werden, da dies den Umfang der Arbeit übersteigen würde. Für weiterführende Informationen über die einzelnen Klassen und deren Benutzung sei an dieser Stelle auf die Dokumentation verwiesen [2]. Die Implementierung verwendet die Android-API in der Version 8; dies entspricht Android in der Version 2.2.

Zur Extraktion der MFCCs aus den Audiodaten wurden Teile der jAudio-Bibliothek in Version 1.0.4 verwendet [11]. Diese Bibliothek ermöglicht die einfache Extraktion von verschiedenen Audiomerkmalen aus Audiodaten.

Zur Klassifikation der Audiodaten wird die Bibliothek Mulan in der Version 1.3.0 verwendet [21]. Diese Bibliothek stellt verschiedene Multilabel-Klassifikatoren bereit. Mulan basiert auf Weka, einer Bibliothek für das maschinelle Lernen für Singlelabel-Klassifikation [7].

Im Folgenden wird ein Einblick in die Leistungsfähigkeit und die Programmstruktur von Earacle gegeben. Nähere Informationen über die Implementierung sind der Dokumentation zu entnehmen.

5.1 Leistungsumfang

Earacle klassifiziert aufgenommenes Audiomaterial anhand von bekannten Beispielen mit denen ein Klassifikator trainiert wird. Die Anwendung nimmt dazu durchgehend das Audiosignal vom Mikrophon des Mobilgeräts ab, teilt dieses in Fenster und klassifiziert sie. Der Benutzer kann den Klassifikator, die Art der Merkmalsextraktion, sowie die Parameter, die die Merkmalsextraktion beeinflussen, wählen. Es können ein Satz von Klassenlabeln definiert und Beispieldaten aufgenommen werden, die den gewählten Klassenlabeln zugeordnet werden können. Ein solches Klassifikationsmodell, bestehend aus dem Klassifikator, den Beispieldaten sowie dem Satz von Klassenlabeln und Parametern der Merkmalsextraktion, kann gespeichert und wieder geladen werden. Die Ergebnisse der Klassifikation können angezeigt, gespeichert und dem Android-System bekannt gemacht werden.

Die Anwendung besteht aus einer graphischen Oberfläche und einem unabhängig von der Oberfläche lauffähigen Hintergrunddienst. Dies ermöglicht es der Anwendung die Szenenerkennung durchzuführen, während die Oberfläche nicht sichtbar ist. Der Hintergrunddienst wird gemeinsam mit der Oberfläche beim Start der Anwendung gestartet. Allerdings kann die Oberfläche unabhängig vom Hintergrunddienst verlassen werden. Der Dienst klassifiziert dann weiterhin die aktuellen Audiodaten. Falls der Hintergrunddienst ausgeführt wird und die Benutzeroberfläche erneut aufgerufen wird, startet diese keinen neuen Dienst sondern die Oberfläche verbindet sich auf den bestehenden Hintergrunddienst, so dass die Manipulation dessen wieder möglich ist. Zur Laufzeit macht der Hintergrunddienst die aktuelle Klassifikation dem System über einen Broadcast bekannt (siehe Abschnitt 5.5).

Die Oberfläche der Anwendung besteht aus einer einzigen Activity, die die vier Tabs, die in den Abbildungen 5.1, 5.2, 5.3 und 5.4 zu sehen sind, darstellt. Die Activity stellt im Android-System die Hauptklasse einer Anwendung dar und ermöglicht die Interaktion mit dem Benutzer und dem System.

Im ersten Tab (Abbildung 5.1) werden die Ergebnisse der aktuellen Klassifikation, geordnet nach der Konfidenz, die der Klassifikator liefert, angezeigt. Dabei werden nur Klassenlabel angezeigt, deren Konfidenz einen gewissen Schwellwert übersteigt.

Im zweiten Tab (Abbildung 5.2) ist es für den Benutzer möglich den Klassifikator mit neuen Beispielen zu trainieren. Es können neue Klassenlabel angelegt oder bestehende gelöscht werden. Ein Start des Lernvorgangs sorgt dafür, dass dem Klassifikator neue Beispiele hinzugefügt werden. Dazu muss mindestens ein Klassenlabel ausgewählt werden, das die aktuelle akustische Situation charakterisiert.

Der dritte Tab (Abbildung 5.3) zeigt dem Benutzer verschiedene Angaben über das aktuelle Klassifikationsmodell. Dies sind zum einen der gewählte Klassifikator, die

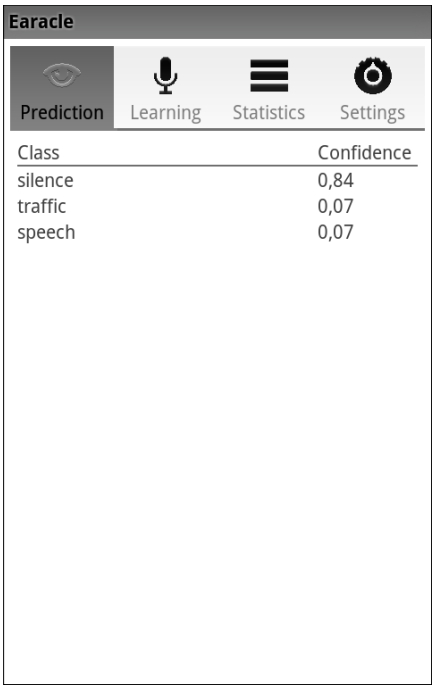

Abbildung 5.1: Anzeige der aktuellen Klassifikationen

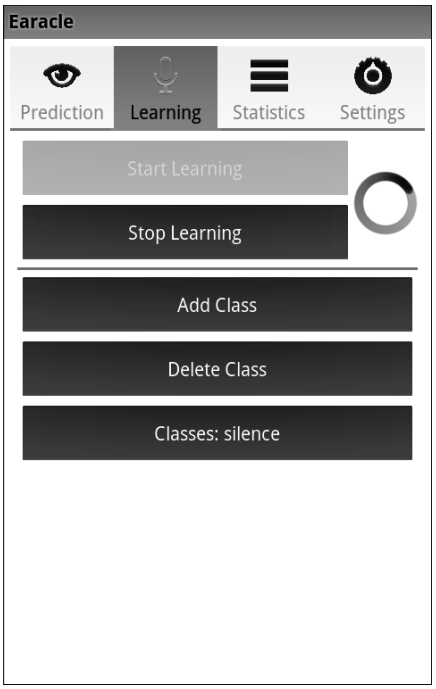

Abbildung 5.2: Tab zum Hinzufügen neuer Beispieldaten

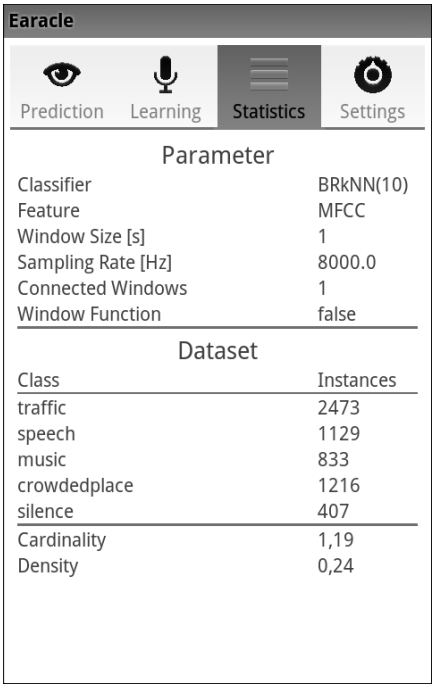

Abbildung 5.3: Angaben über das aktuelle Klassifikationsmodell

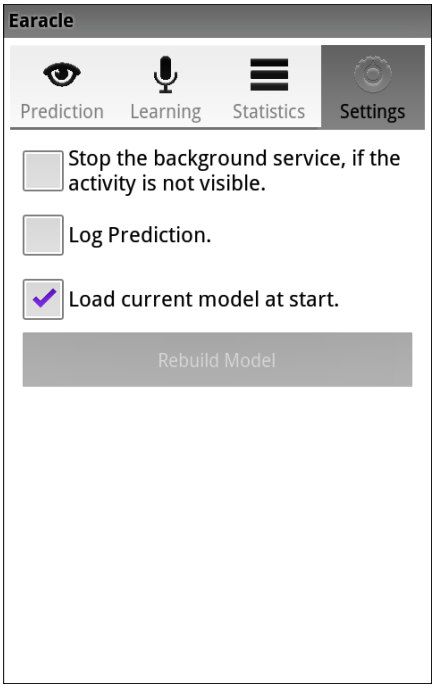

Abbildung 5.4: Einstellungsmöglichkeiten der Anwendung

Art der Merkmalsextraktion, sowie die Parameter der Merkmalsextraktion und zum anderen Angaben über den Satz von Beispielen.

Im vierten Tab (Abbildung 5.4) kann vom Benutzer entschieden werden, ob die Klassifikationen in eine Datei gespeichert werden sollen und ob das aktuelle Klassifikationsmodell bei jedem Start der Anwendung automatisch geladen werden soll. Außerdem ist es der Anwendung möglich, im Hintergrund ausgeführt zu werden, um auf Änderungen der akustischen Szene zu reagieren, während die Anwendung selbst nicht sichtbar ist. Dazu muss der Benutzer der Anwendung mitteilen, dass der Hintergrunddienst nicht mit dem Verlassen der Oberfläche beendet werden soll. Da das Trainieren mancher Klassifikatoren sehr aufwändig ist, geschieht dies nicht immer direkt bei der Aufnahme neuer Beispiele. Stattdessen hat der Benutzer dann die Möglichkeit, in diesem Tab das Training des Klassifikators zu einem ihm passenden Zeitpunkt durchzuführen.

5.2 Programmstruktur

Die Implementierung orientiert sich stark am Model-View-Controller-Entwurfsmuster. Die Klasse EaracleActivity, die von einer Android-Activity erbt, ist für die Anzeige der Benutzeroberfläche und die Reaktion auf Benutzereingaben zuständig. Dazu erzeugt sie die in Abschnitt 5.1 beschriebenen Views und stellt diese als Tabs dar. Ebenso erzeugt sie Controller, die auf Benutzereingaben reagieren. Die Klasse EaracleService, welche von einem Android-Service erbt, nimmt die Aufgaben des Modells im MVC-Entwurfsmuster war. Dazu erzeugt sie verschiedene Klassen, wie zum Beispiel die Klasse MultiLabelInstances, die für die Speicherung der Beispieldaten und die Speicherung der Klassifikationen zuständig ist, oder die Klasse ProcessingManager, die für die Koordination von Audioaufnahme und Verarbeitung beziehungsweise Klassifikation verantwortlich ist. Der Service wird als Hintergrunddienst ausgeführt und ist unabhängig von der Activity lauffähig.

Der EaracleService wird beim ersten Aufruf der EaracleActivity gestartet. Danach wird durch die Benutzung eines Android-Binders die Verbindung zwischen Service und Activity hergestellt. Dabei handelt es sich um ein Interface, das den Zugriff auf den EaracleService gestattet. Falls die Activity beendet und wieder gestartet wird, kann sie sich wieder mit Hilfe eines Binders mit dem Service verbinden, ohne diesen neu starten zu müssen.

Die Activity implementiert einen IServiceListener. Damit meldet sich die Activity als Listener beim EaracleService an, um auf Ereignisse im Hintergrunddienst reagieren zu können. Umgekehrt implementiert der Service das Interface IEaracleService über das es von der Activity angesprochen werden kann. Service und Activity werden im gleichen Prozess ausgeführt. Dies führt dazu, dass keine aufwändige Inter-Prozess-

Kommunikation benötigt wird, um Daten zwischen den beiden Bereichen auszutauschen. Stattdessen teilen sich beide einen gemeinsamen Speicherbereich. Dies ermöglicht, dass die Activity direkt Listener auf den Modellklassen des Services anmeldet, so dass diese auf Änderungen in den Daten reagieren können. Die Verbindung zwischen Service und Activity ist in Abbildung 5.5 vereinfacht dargestellt.

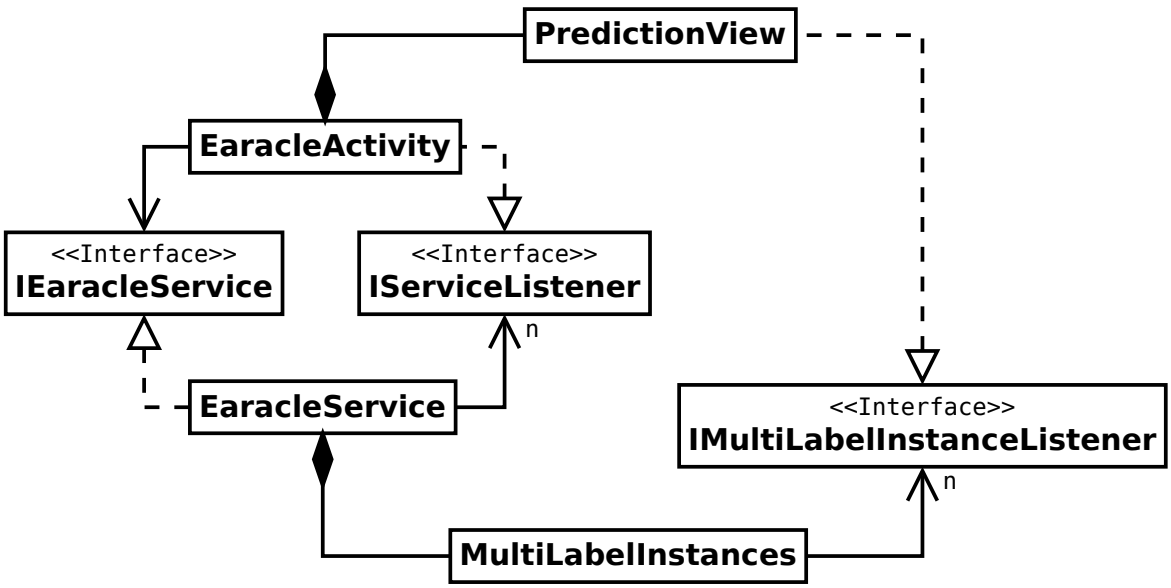

Abbildung 5.5: Ausschnitt der Programmstruktur der Earacle-Anwendung. Zu Sehen ist die über Interfaces entkoppelte Verbindung zwischen Service und Activity. Außerdem ist beispielhaft die Verbindung zwischen einem View und einer Modellklasse über einen Listener dargestellt.

Wie in Abbildung 5.5 zu sehen ist, implementiert der PredictionView, der für die Anzeige der aktuellen Klassifikation zuständig ist, einen IMultiLabelInstancesListener. Sobald sich die Activity mit dem Service verbindet, registriert sie diesen View als Listener bei der Klasse MultiLabelInstances, welche für die Speicherung neuer Klassifikationen zuständig ist. Dadurch wird der PredictionView über jede neue Klassifikation informiert und kann diese anzeigen.

Falls die Activity vom Benutzer beendet wird, kann sie auch den Hintergrunddienst beenden. Sollte der Benutzer dies nicht wünschen, so dass die Klassifikation im Hintergrund fortgesetzt werden kann, müssen alle Listener von den Klassen des Services abgemeldet werden. Dies ist nötig, da sonst durch eine Benachrichtigung über eine Änderung der Daten Code in den Views ausgeführt wird, obwohl diese nicht sichtbar sind.

5.3 Koordination von Audioaufnahme und Audioverarbeitung

Da die Aufnahme der Audiodaten sowie die Extraktion und Klassifikation der Merkmalsvektoren sehr aufwändig sind, werden diese nicht im Hauptthread der Anwendung durchgeführt sondern in separaten Threads. Zu diesem Zweck werden zwei Android-AsyncTasks verwendet. Die Klasse AsyncTask erlaubt durch das Überschreiben von Methoden die einfache Benutzung von Threads unter Android. Ebenso kann auch aus dem Thread heraus Code im Hauptthread ausgeführt werden, um die Benutzeroberfläche zu aktualisieren.

Ein AsyncTask wird für die Aufnahmen der Audiodaten verwendet, während der zweite AsyncTask die Verarbeitung der Daten durchführt, die die Extraktion und Klassifikation der Merkmalsvektoren umfasst.

Die Initialisierung und Koordination der Zusammenarbeit der AsyncTasks wird durch die Klasse ProcessingManager durchgeführt. Da das Anfordern von Speicher auf der Android-Plattform verhältnismäßig aufwändig ist, werden vom ProcessingManager einmalig zwei Buffer alloziert, welche die Rohdaten der Aufnahmen aufnehmen. Abwechselnd wird jeweils ein Buffer für die Aufnahme verwendet, während aus den Daten im anderen Buffer der Merkmalsvektor extrahiert wird.

Sobald der Buffer der Audioaufnahme gefüllt ist, benachrichtigt der AsyncTask, der die Aufnahme durchgeführt hat, den ProcessingManager. Ebenso wird der ProcessingManager vom Daten verarbeitenden AsyncTask über die fertige Extraktion und Klassifikation des Merkmalsvektors informiert. Dazu ist der ProcessingManager bei beiden AsyncTasks als Listener angemeldet. Falls beide AsyncTasks die Bearbeitung abgeschlossen haben, werden die Buffer getauscht, so dass aus dem Buffer in den soeben aufgenommen wurde der nächste Merkmalsvektor extrahiert wird und in den Buffer aus dem der letzte Merkmalsvektor extrahiert wurde die nächste Aufnahme gespeichert wird. Da dieser Tausch der Buffer nur durchgeführt werden kann, wenn beide AsyncTasks die Bearbeitung abgeschlossen haben, muss jeweils ein AsyncTask auf den anderen warten. Dabei ist variabel, welcher Thread seine Bearbeitung zuerst abschließt.

In Abbildung 5.6 ist ein möglicher Ablauf von Aufnahme und Verarbeitung vorgestellt. Der ProcessingManager erhält das Signal, das die Verarbeitung starten soll. Dies ist zum Beispiel der Fall, wenn die Anwendung gestartet und ein Modell geladen wurde. Der ProcessingManager erzeugt einen Thread zur Aufnahme von Audiomaterial und einen Thread zur Verarbeitung der Audiodaten. Die Verarbeitung umfasst sowohl die Extraktion der Merkmale aus den Audiodaten, als auch die Klassifikation

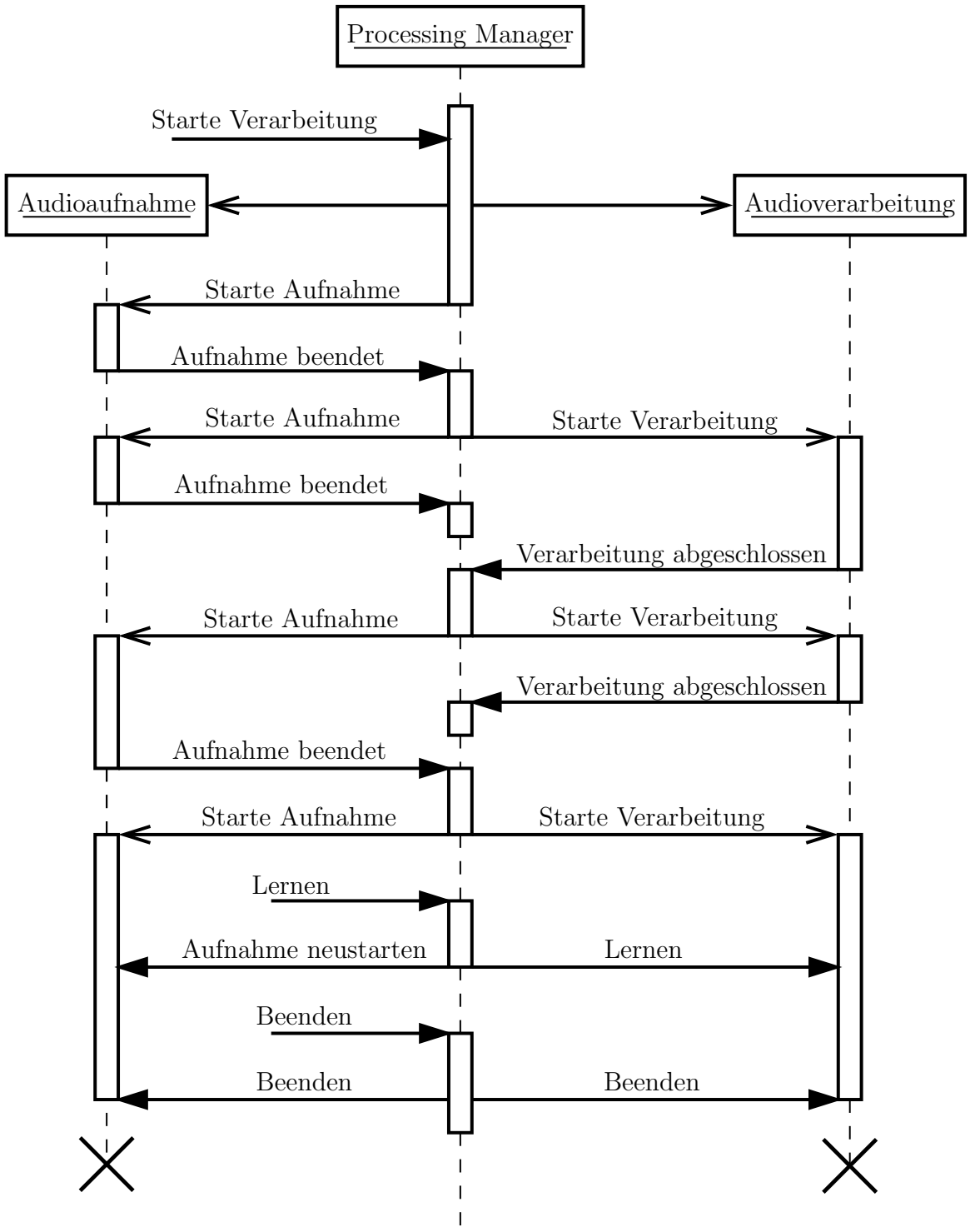

Abbildung 5.6: Möglicher Ablauf der Koordination zwischen Audioaufnahme und Verarbeitung. Die Verarbeitung umfasst dabei die Extraktion des Merkmalsvektors, als auch entweder die Klassifikation des Merkmalsvektors oder das Hinzufügen des Merkmalsvektors zu den Beispieldaten.

der Merkmalsvektoren, beziehungsweise das Hinzufügen von neuen Beispielen zum Klassifikationsmodell.

Sobald die beiden Threads gestartet sind, beginnt der Aufnahmethread mit der Aufnahme von Audiomaterial in den ersten Buffer. Währenddessen wartet der Verarbeitungsthread auf eine erste fertige Aufnahme. Der Aufnahmethread füllt den ersten Buffer und meldet die Fertigstellung der Aufnahme an den ProcessingManager. Jetzt wird der Verarbeitungsthread damit beauftragt den ersten Buffer zu bearbeiten, während gleichzeitig eine neue Aufnahme in den zweiten Buffer gestartet wird. Die Klassifikation hinkt also der Aufnahme hinterher. Nun kann entweder der Aufnahmethread mit der Aufnahme in den zweiten Buffer oder der Verarbeitungsthread zuerst seine Aufgabe erfüllen. In beiden Fällen muss der bereits fertige Thread auf den noch arbeitenden Thread warten, da die Buffer nur getauscht werden können, wenn beide Threads die Bearbeitung abgeschlossen haben.

Falls ein Merkmalsvektor nicht klassifiziert werden soll, sondern neue Beispieldaten aufzunehmen sind, wird dies dem ProcessingManager durch einen externen Aufruf, der zum Beispiel durch einen Klick in der Benutzeroberfläche ausgelöst wurde, mitgeteilt. Der ProcessingManager sendet an den Aufnahmethread das Signal die aktuelle Aufnahme abzubrechen und eine neue Aufnahme zu starten, da die aktuelle Aufnahme ungültig geworden ist. Gleichzeitig wird dem Verarbeitungsthread mitgeteilt, dass der nächste extrahierte Merkmalsvektor nicht klassifiziert werden soll sondern dem Klassifikationsmodell als neues Beispiel hinzugefügt werden soll. Ebenso wie bei der Klassifikation werden die Buffer zwischen Aufnahme und Verarbeitung wiederholt gewechselt.

Falls die Anwendung beendet wird oder aus einem anderen Grund die Verarbeitung der Audiodaten beendet werden soll, sendet der ProcessingManager den AsyncTasks das Signal zum Beenden. Diese brechen daraufhin ihre Verarbeitung so schnell wie möglich selbst ab.

5.4 Audioaufnahmen unter Android

Es gibt unter Android zwei Möglichkeiten eine Audioaufnahme anzufertigen. Die Klasse MediaRecorder stellt ein einfaches Aufnahmeinterface bereit. Dieses erlaubt jedoch nur bestimmte Kodierungen der Aufnahme, sowie ausschließlich die direkte Aufnahme in eine Datei. Dem Gegenüber steht die Klasse AudioRecord, welche es erlaubt einen Stream von PCM-Daten zu lesen. Aufgrund dieser größeren Freiheit wird in der Implementierung die AudioRecord-Klasse verwendet.

Es stellen sich jedoch verschiedene Problematiken, die nur unzureichend gelöst werden können. Bei der Instanziierung der AudioRecord-Klasse muss gewählt werden mit

welcher Abtastrate die Aufnahme angefertigt werden soll. Die verfügbaren Abtastraten variieren von Gerät zu Gerät, wobei nur eine Abtastrate von 44100 Hz auf allen Geräten garantiert verfügbar ist [3]. Es ist jedoch nicht möglich die unterstützten Abtastraten des jeweiligen Geräts zur Laufzeit abzufragen. Stattdessen muss ein AudioRecord-Objekt mit der gewünschten Abtastrate instanziiert werden. Sollte die Abtastrate nicht unterstützt werden, wird eine IllegalArgumentException geworfen, die von der Anwendung gefangen werden muss.

Ein weiteres Problem besteht darin, dass zu jeder Zeit nur eine einzige Anwendung auf einem Android-System auf die Aufnahmehardware zugreifen kann. Anders als bei dem Abspielen von Ton unter Android, gibt es bei der Aufnahme jedoch keine Möglichkeit abzufragen, ob die Aufnahmehardware von einer anderen Anwendung belegt ist, oder ob sie genutzt werden kann. Um die Verfügbarkeit der Audiohardware zu testen, muss ein Objekt der Klasse AudioRecord instanziiert werden. Anschließend kann geprüft werden, ob es gültig initialisiert wurde. Liegt keine gültige Initialisierung vor, muss davon ausgegangen werden, dass die Aufnahmehardware bereits belegt ist. Falls jedoch eine andere Anwendung die Aufnahmehardware bereits belegt hatte, wird dieser Anwendung dennoch die Audiohardware entzogen, so dass nun beide Anwendungen keinen Zugriff auf die Aufnahmehardware haben. Falls dieser Fall des plötzlichen Entzugs der Aufnahmehardware nicht von der Anwendung, der die Hardware entzogen wurde, berücksichtigt wurde und sie in diesem Fall die Hardware nicht vollständig freigegeben hat, so befindet sich die Aufnahmehardware nun in einem undefinierten Zustand, aus dem sie programmatisch nicht neu angesteuert werden kann. Einzig ein Neustart des Systems schafft an dieser Stelle Abhilfe.

5.5 Integration von Earacle in andere Anwendungen

Um die Integration der akustischen Szenenerkennung, die Earacle zur Verfügung stellt, in anderen Anwendungen nutzen zu können, sendet Earacle bei jeder neuen Klassifikation einen Broadcast an das System. Dieser Broadcast kann von einem Broadcast-Receiver empfangen werden. Zusammen mit dem Broadcast wird eine Hashmap gesendet, die alle Klassenlabel des Klassifikationsmodells sowie deren aktuelle Konfidenzen enthält.

Mit diesen Informationen ist es einer Anwendung möglich auf verschiedene akustische Szenen zu reagieren, ohne die Aufnahme und Klassifikation selbst durchführen zu müssen. So ist es zum Beispiel möglich, dass eine Anwendung automatisch die Profileinstellungen des Mobilgeräts abhängig von der aktuellen akustischen Szene anpasst.

6 Evaluation von Earacle

Um die Klassifikation durch Earacle zu testen, wurden im Folgenden verschiedene Versuche durchgeführt, bei denen die Klassifikation in Echtzeit auf den über das eingebaute Mikrofon aufgenommenen Audiodaten durchgeführt wurde.

Es wurden zwei Feldexperimente durchgeführt, die die Klassifikation während einer Stadtbahnfahrt und während des Aufenthalts in einer Bahnhofshalle aufzeichnen. In zwei kontrollierten Versuchen wurden die Unterschiede der Klassifikationsgenauigkeit bei unterschiedlicher Positionierung und Verdeckung des Mobilgeräts im Vergleich zum Halten des Mobilgeräts in der Hand untersucht.

Für die Versuche wurde ein BRkNN-Klassifikator mit den in Abschnitt 4.1 beschriebenen Test- und Trainingsdatensätzen trainiert. Es wurde eine Fensterbreite von einer Sekunde gewählt, da die zu betrachtenden Ereignisse von relativ kurzer Dauer sind. Die Aufnahmen wurde bei einer Abtastrate von 8000 Hz angefertigt. Zur Merkmalsextraktion wurden MFCCs verwendet. Die Konfidenzen der Klassenlabel wurden über 20 Sekunden gemittelt. Die Klassifikationen wurde zur Auswertung in eine Log-Datei geschrieben. Die Tests wurden mit einem HTC Desire Z durchgeführt.

6.1 Evaluation in unkontrollierter Umgebung

Im ersten Versuch wurden die Geräusche einer Stadtbahnfahrt untersucht. Die Stadtbahn fuhr während des Versuchs mehrere Haltestellen an. Ungefähr ein Dutzend Fahrgäste sorgten für durchgehende Gespräche während der Fahrt. Das Mobilgerät wurde während des Versuchs in der Hand gehalten. In Abbildung 6.1 sind die Konfidenzen für die einzelnen Klassenlabel über den Fahrverlauf aufgetragen. Vertikale Linien markieren die Zeitpunkte an denen die Stadtbahn an einer Haltestelle zum Stehen kam. An dem Verlauf der Konfidenz des Klassenlabels *Traffic* sind sehr eindeutig die verschiedenen Abschnitte der Fahrt zu erkennen. Zwischen den Haltestellen steigt die Konfidenz deutlich, teilweise über 90 Prozent, an. Die Konfidenz an den Haltestellen ist hingegen deutlich niedriger. Es ist zu bemerken, dass durch die Mittelung über mehrere zurückliegende Klassifikationen kein rapider An- und Abstieg der Konfidenzen stattfindet. Stattdessen entstehen durch die Mittelung ansteigende und abfallende Flanken.

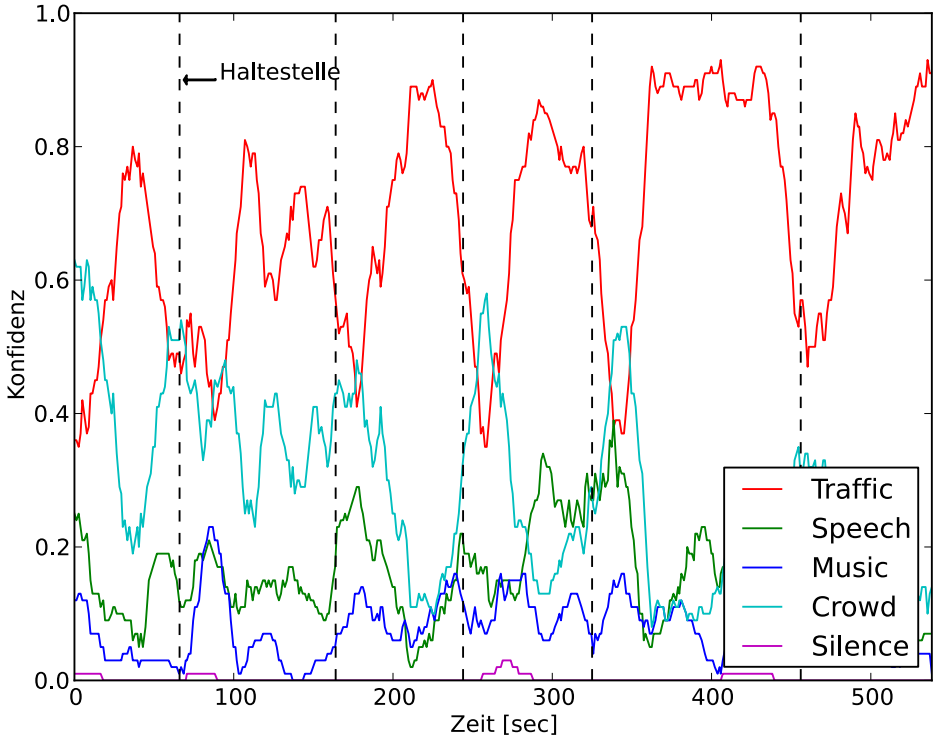

Abbildung 6.1: Konfidenzen der einzelnen Klassenlabel über den Fahrverlauf einer Stadtbahnfahrt aufgetragen. Zeitpunkte, an denen die Stadtbahn an einer Haltestelle zum Stehen kam, sind durch vertikale Linien markiert.

Außerdem hinken die Konfidenzen den Ereignissen zeitlich hinterher. Während die Gespräche der Fahrgäste während der Fahrt durch die Fahrgeräusche unterdrückt werden, sind diese an den Haltestellen deutlicher zu erkennen. Somit steigt die Konfidenz für das Klassenlabel *Crowd* deutlich an. Es kommt während des Versuchs fälschlicherweise zu leicht erhöhten Werten für das Klassenlabel *Music*. *Silence* wird während der Fahrt korrekterweise nicht erkannt.

In einem zweiten Versuch wurden die Klassifikationsergebnisse während eines Aufenthalts in der Bahnhofshalle des Bielefelder Hauptbahnhofes getestet. Die Bahnhofshalle war zum Zeitpunkt des Versuchs sehr belebt. Das Mobilgerät wurde während des Versuchs wieder in der Hand gehalten.

Die Ergebnisse des Versuchs sind Abbildung 6.2 zu entnehmen. Deutlich besitzt das Klassenlabel *Crowd* durchgehend die höchste Konfidenz. Fälschlicherweise hat jedoch auch das Klassenlabel *Traffic*, gerade am Beginn und Ende des Versuchs, eine hohe Konfidenz. Die übrigen Klassenlabel werden korrekterweise nicht erkannt und haben eine sehr geringe Konfidenz aufzuweisen.

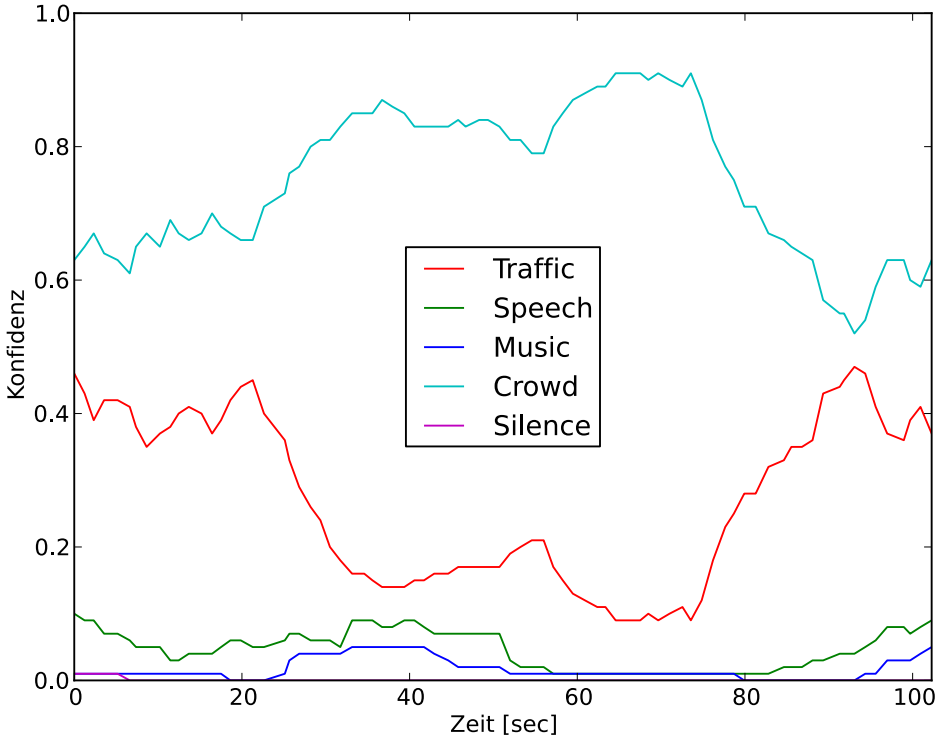

Abbildung 6.2: Konfidenzen der Klassenlabel während des Aufenthalts in einer belebten Bahnhofshalle.

6.2 Evaluation mit kontrolliertem Versuchsablauf

Um die Klassifikationsgenauigkeit von Earacle bei unterschiedlicher Positionierung und Verdeckung des Mobilgeräts zu testen, wurde eine akustische Szene konstruiert. Der Versuch wird zweimal durchgeführt. Beim ersten Versuch wird das Mobilgerät unbedeckt auf einem Tisch positioniert. Im zweiten Versuch wird das Mobilgerät durch eine Lederhülle geschützt in einer Jeanshosentasche platziert. Durch die doppelte Isolierung des Mobilgeräts durch Hülle und Hosentasche wird das Signal, das das Mikrofon erreicht, deutlich gedämpft.

Der Versuch besteht aus 5 Phasen. Zu Beginn wird für ungefähr 60 Sekunden kein Geräusch produziert. Anschließend werden 60 Sekunden aus einem Musikstück abgespielt. In der dritten Phase kehrt für 30 Sekunden wieder Ruhe ein. Danach wird für weitere 75 Sekunden Sprache präsentiert. Im letzten Abschnitt des Versuchs wird wiederum kein Geräusch präsentiert.

In Abbildung 6.3 sind die Konfidenzen der Klassenlabel während des Versuchs aufgetragen. Sehr gut zu erkennen sind die durch die Mittelung entstehenden abfallenden

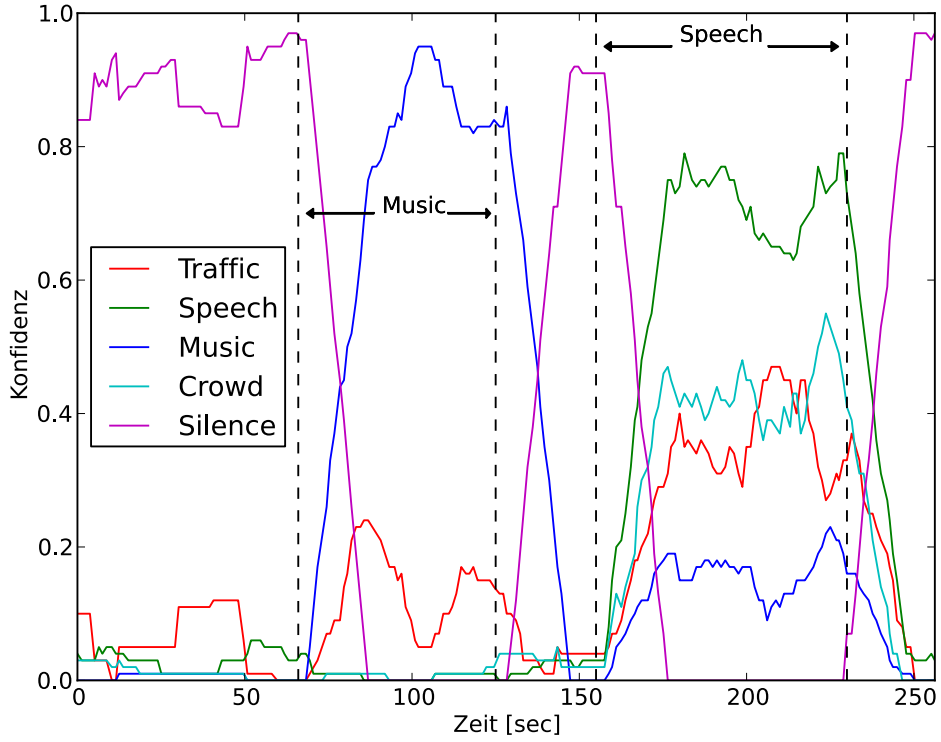

Abbildung 6.3: Konfidenzen der einzelnen Klassenlabel aufgetragen auf den Verlauf einer konstruierten akustischen Szene. Das Mobilgerät lag während des Versuchs auf einem Tisch. Das Auftreten von Musik und Sprache ist durch vertikale Linien markiert.

und ansteigenden Flanken bei dem Wechsel der akustischen Einflüsse. Die Klassenlabel *Silence* und *Music* werden mit Konfidenzen deutlich über 80 Prozent sehr gut erkannt. Während der Präsentation von Sprache dominiert die Konfidenz des Klassenlabels *Speech* korrekterweise. Deutlich sind jedoch auch die Konfidenzen für *Traffic* und *Crowd* erhört, was als Fehlerkennung beurteilt werden muss.

In Abbildung 6.4 sind die Konfidenzen der Klassenlabel während des Versuchs mit dem bedeckt positionierten Mobilgerät aufgetragen. Wie im vorhergehenden Versuch sind auch hier die abfallenden und ansteigenden Flanken während eines Wechsels des akustischen Signals zu erkennen. Deutlich wird das Klassenlabel *Silence* in der ersten, dritten und fünften Phase erkannt, wobei die Erkennung in der ersten Phase im Vergleich zum vorhergehenden Versuch starken Schwankungen unterliegt. Die Erkennung der Musik in der zweiten Phase des Versuchs schlägt fehl. Zum einen weist das Klassenlabel *Music* fast durchgehend eine Konfidenz von weniger als 50 Prozent auf, zum anderen haben die Klassenlabel *Traffic* und *Crowd* deutlich erhöhte Werte. Das

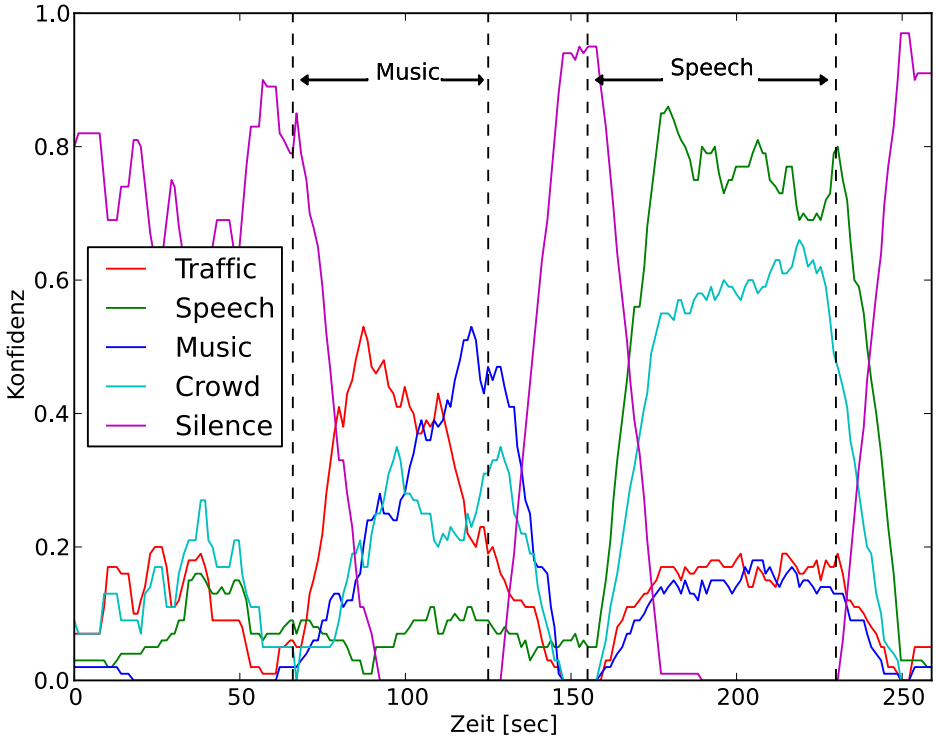

Abbildung 6.4: Konfidenzen der einzelnen Klassenlabel aufgetragen auf den Verlauf einer konstruierten akustischen Szene. Das Mobilgerät befand sich während des Versuchs in einer Lederhülle und in der Tasche einer Jeanshose. Das Auftreten von Musik und Sprache ist durch vertikale Linien markiert.

Klassenlabel *Traffic* dominiert fälschlicherweise einen großen Teil dieses Abschnittes. In der vierten Phase wird korrekt Sprache erkannt. Die Konfidenz für das Klassenlabel *Speech* dominiert den Abschnitt, wobei jedoch auch das Klassenlabel *Crowd* sehr hohe Werte aufweist.

Insgesamt ist zu bemerken, dass die Erkennungsgenauigkeit im zweiten Versuch zwar nicht so gut ist wie im ersten Versuch, jedoch dennoch brauchbar. Es ist allerdings anzumerken, dass beim zweiten Versuch keine Bewegungen stattgefunden haben, so dass der Jeansstoff keine Geräusche durch Reibung produziert hat.

7 Fazit und Ausblick

In dieser Arbeit wurde untersucht in welchem Umfang die Wahl der Parameter einer Merkmalsextraktion und die Wahl verschiedener Klassifikatoren Einfluss auf die Genauigkeit der Erkennung akustischer Szenen haben. Als Vorverarbeitungsschritt wurden aus den Audiodaten MFCCs erzeugt. Zudem wurde untersucht welchen Einfluss die Zuordnung mehrere Klassenlabel auf die Klassifikationsgenauigkeit hat. Dabei wurde gezeigt, dass bei der Verwendung von kNN-basierten Klassifikatoren kein gravierender Unterschied zwischen der Klassifikation von Singlelabel- und Multilabel-Datensätzen zu erkennen ist. Klassifikatoren, die auf einem naiven Bayesansatz aufbauen, profitieren jedoch von der Einschränkung auf eine Singlelabel-Klassifikation und erreichen dabei eine Classification-Accuracy von ungefähr 78%.

Es wird vermutet, dass sich die Klassifikationsgenauigkeit weiter steigern lässt indem die semantische Bedeutung der einzelnen Klassenlabel berücksichtigt wird, so dass sich Klassenlabel gegenseitig bedingen oder ausschließen können. Ebenso kann die Verwendung mehrerer verschiedener Merkmalsextraktionen die Klassifikationsgenauigkeit steigern. Es ist zu untersuchen in wie weit an dieser Stelle ein Kompromiss zwischen Klassifikationsgenauigkeit und Systemauslastung zu finden ist.

In dieser Arbeit wurden 13 MFCCs zu einem Merkmalsvektor zusammengefasst. Erste Untersuchungen haben gezeigt, dass bei der Benutzung des BRkNN-Klassifikators die Klassifikationsgenauigkeit durch Reduktion der Anzahl der verwendeten MFCCs nicht verringert wird. Dadurch kann der Aufwand für die Klassifikation unbekannter Audiodaten gesenkt werden. Weitere Untersuchungen sind nötig, um dies zu bestätigen.

Es wurde das Programm Earacle für die Android-Plattform entwickelt, dass ein System zur Erkennung akustischer Szenen implementiert. Um den Benutzer nicht in der Wahl der Klassenlabel einzuschränken, werden Multilabel-Klassifikatoren verwendet. Earacle erlaubt es, Klassifikatoren mit selbst aufgenommenen Audiodaten zu trainieren und anschließend unbekannte Audiodaten zu klassifizieren. Earacle stellt eine Schnittstelle bereit über die es möglich ist die Klassifikationsfähigkeit von Earacle in anderen Android-Programmen zu nutzen. Um möglichst gute Ergebnisse bei der Klassifikation von Umgebungen zu erhalten, können somit die Ergebnisse der Klassifikation von Audiodaten mit Informationen fusioniert werden, die aus anderen Sensoren gewonnen werden.

Bei den benutzten Bibliotheken zur Merkmalsextraktion und Klassifikation handelt es sich jeweils um Java-Bibliotheken, die nicht auf die Benutzung unter Android optimiert sind. Durch eine Anpassung dieser Bibliotheken können Systemressourcen geschont werden. Ebenso ist ein Geschwindigkeitszuwachs durch eine Umstellung der Merkmalsextraktion von Java auf C++ zu erwarten. Die Einbindung von nativem C++-Code in eine Android-Anwendung ist mit Hilfe des Java-Native-Interface (JNI) möglich.

Mit Hilfe von Earacle ist es nun zum Beispiel möglich eine Anwendung zu erstellen, die zu jeder akustischen Szene die aktuellen Einstellungen des Mobilgeräts speichert und diese bei erneuter Erkennung der gleichen Szene wieder herstellt. Ein solches System kann durch die Einbindung von zusätzlichen Sensoren, wie zum Beispiel den Beschleunigungssensoren und GPS-Informationen, weiter ausgebaut werden, so dass eine feinere Abstimmung auf den Benutzer erfolgen kann.

Literaturverzeichnis

[1] Android. http://www.android.com/, 22.9.2011.

[2] Android Reference. http://developer.android.com/reference/android/package-summary.html, 22.9.2011.

[3] Android Reference, AudioRecord. http://developer.android.com/reference/android/media/AudioRecord.html#AudioRecord%28int,%20int,%20int,%20int,%20int%29, 22.9.2011.

[4] Open Handset Alliance. http://www.openhandsetalliance.com/, 22.9.2011.

[5] A.S. Bregman. *Auditory scene analysis: The perceptual organization of sound*. The MIT Press, 1994.

[6] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2):211–225, 2009.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[8] V. Hamacher, J. Chalupper, J. Eggers, E. Fischer, U. Kornagel, H. Puder, and U. Rass. Signal processing in high-end hearing aids: state of the art, challenges, and future trends. *EURASIP Journal on Applied Signal Processing*, 2005:2915–2929, 2005.

[9] B. Logan et al. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, volume 28, page 5. Citeseer, 2000.

[10] P. Lukowicz, J. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. Recognizing workshop activity using body worn microphones and accelerometers. *Pervasive Computing*, pages 18–32, 2004.

[11] C. McKay, I. Fujinaga, and P. Depalle. Jaudio: A feature extraction library.

[12] D. O'shaughnessy. *Speech communications: human and machine*. Universities Press, 1987.

[13] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational auditory scene recognition. In *Acoustics, Speech, and Signal Processing, 1993.*

ICASSP-93., 1993 IEEE International Conference on, volume 2, pages II–II. IEEE, 2002.

[14] V.T.K. Peltonen, A.J. Eronen, M.P. Parviainen, and A.P. Klapuri. Recognition of everyday auditory scenes: potentials, latencies and cues. *PREPRINTS-AUDIO ENGINEERING SOCIETY*, 2001.

[15] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

[16] N. Sawhney and P. Maes. Situational awareness from environmental sounds. *url: http://web.media.mit.edu/˜nitin/papers/Env_Snds/EnvSnds.html*.

[17] E. Spyromitros, G. Tsoumakas, and I. Vlahavas. An empirical study of lazy multilabel classification algorithms. *Artificial Intelligence: Theories, Models and Applications*, pages 401–406, 2008.

[18] S.S. Stevens, J. Volkmann, and E.B. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 1937.

[19] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[20] G. Tsoumakas, I. Katakis, and I. Vlahavas. A review of multi-label classification methods. 2006.

[21] G. Tsoumakas, J. Vilcek, E. Spyromitros, and I. Vlahavas. Mulan: a java library for multi-label learning. *Journal of Machine Learning Research (accepted for publication conditioned on minor revisions)*, 2010.

[22] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML 2007*, pages 406–417, 2007.

[23] M.L. Zhang and Z.H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[24] S. Zulkernain, P. Madiraju, S.I. Ahamed, and K. Stamm. A mobile intelligent interruption management system. *Journal of Universal Computer Science*, 16(15):2060–2080, 2010.

Erklärung

Ich versichere, dass ich die vorliegende wissenschaftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, wurden unter Angabe der Quelle als Entlehnung deutlich gemacht. Das gleiche gilt auch für beigegebene Skizzen und Darstellungen. Diese Arbeit hat in gleicher oder ähnlicher Form meines Wissens nach noch keiner Prüfungsbehörde vorgelegen.

Bielefeld, im September 2011

Tobias Rodehutskors