Visualisierung kontinuierlicher, multimodaler Schmerz Scors am Beispiel akustischer Signale

Masterarbeit

Franz Anders
HTWK Leipzig

Januar 2017

Abstract

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

8

2 Grundlagen der medizinischen Schrei-Forschung

2.1 Schmerz Scores

Bei erwachsenen Menschen wird der Schmerzgrad typischerweise durch eine Selbsteinschätzung des Patienten unter der Leitung gezielter Fragen des Arztes vorgenommen. Bei Kindern unter 3 Jahren ist diese Selbsteinschätzung nicht möglich. Schmerz drückt sich in Veränderungen des psychologischen, körperlichen und biochemischen Verhaltens des Säuglings aus. Die für den Arzt am leichtesten feststellbaren Verhaltensänderungen sind von außen wahrnehmbaren Merkmale, wie zum Beispiel ein Verkrampfen des Gesichtsausdrucks, erhöhte Körperbewegungen oder lang anhaltendes Weinen. Um eine weitesgehend objektive Schmerzfeststellung zu ermöglichen, wurden sogenannte *Pain-Scores* entwickelt, die durch ein Punktesystem den insgesamten Schmerzgrad des Babies quantifizieren.[23] Es existieren *eindimensionale* Pain-Scores, die den Schmerz nur Aufgrund der Beobachtung eines Merkmals beurteilen, so wie beispielsweise die reine Beurteilung des Gesichtsausdruckes. *Mehrdimensionale* (auch *multimodale*) Pain-Scores beziehen mehrere Faktoren in das Scoring mit ein.[1]. Tabelle 2.1 zeigt das Scoring-System „Neonatal Infant Pain Scale“(NIPS) als Beispiel für eine multimodale Pain-Score. Der Säugling wird anhand der aufgeführten Kategorien bewertet und alle vergebenen Punkte aufsummiert. Ein insgesamter Wert von > 3 zeigt Schmerz an, ein Wert von > 4 großen Schmerz.[11]

Tabelle 2.1: NIPS-Scoring

NIPS	**0 points**	**1 point**	**2 points**
Facial Expr.	Relaxed	Contracted	-
Cry	Absent	Mumbling	Vigorous
Breathing	Relaxed	Different than basal	-
Arms	Relaxed	flexed/stretched	-
Legs	Relaxed	flexed/stretched	-
Alertness	Sleeping	uncomfortable	-

In den meisten mehrdimensionalen Scoring-Systeme werden die Schreigeräusche mit einbezogen. Tabelle 2.2 zeigt eine Übersicht über eine ausgewählte Menge an multimodalen Pain-Scores. Alle Pain-Scores sind für Kleinkinder bis 3 Jahren gedacht. In der Übersicht wird nicht wiedergegeben, welche weiteren Merkmale jeweils in das Scoring mit einbezogen werden, oder welche Insgesamtpunktzahlen auf welche Schmerzintensität hinweisen. Es soll an dieser Stelle nur verdeutlicht werden, welche unterschiedlichen Ansätze zur Bewertung des Schreiens aus medizinischer Sicht im Zusammenhang mit Pain-Scores existieren. Folgende Beobachtungen lassen sich aus der Übersicht ziehen:

1. Die zu beobachtenden Eigenschaften des Weinens werden mit subjektiv behafteten Werten charakterisiert. Beispielsweise wird im N-PASS-System ist ein Schmerz-Schrei

als „High-pitched or silent-continuous crying“ beschrieben. Es wird nicht fest definiert, was als „crying“ gilt oder welche Tonhöhe als „high-pitched“ ist. Auch die Erstquellen geben keine festen Definitionen.

2. Es gibt verschiedene Ansätze zur Bewertung des Weinens. Bei CRIE ist die Tonhöhe, bei BIIP die Länge und bei COMFORT die Art des Weinens entscheidend.

3. Die Beschreibungen sind kurz und prägnant gehalten, der Arzt hat in keinem der Modelle auf mehr als drei Parameter des Schreiens zu achten. Die Begründung liegt darin, dass bei allen Modellen a.) das Schreien nur eines von mehreren Faktoren ist, und b.) Die Schmerzbestimmung in einem vorgegebenen Zeitrahmen durchführbare sein muss.

System	**P.**	**Description**
FLACC[29]	0	No cry (awake or asleep)
	1	Moans or whimpers; occasional complaint
	2	Crying steadily, screams or sobs, frequent complaints
N-PASS[24]	-2	No cry with painful stimul
	-1	Moans or cries minimally with painful stimuli
	0	Appropiate Crying
	1	Irritable or Crying at Intervals. Consolable
	2	High-pitched or silent-continuous crying. Not consolable
BIIP[9]	0	No Crying
	1	Crying <2 minutes
	2	Crying >2 minutes
	3	Shrill Crying >2 minutes
CRIES[3]	0	If no cry or cry which is not high pitched
	1	If cry high pitched but baby is easily consoled
	2	If cry is high pitched and baby is inconsolable
COVERS[15]	0	No Cry
	1	High-Pitched or visibly crying
	2	Inconsolable or difficult to soothe
PAT[12]	0	No Cry
	1	Cry
DAN[4]	0	Moans Briefly
	1	Intermittent Crying
	2	Long-Lasting Crying, Continuous howl
COMFORT[21]	0	No crying
	1	Sobbing or gasping
	2	Moaning

	3	Crying
	4	Screaming
MBPS[22]	0	Laughing or giggling
	1	Not Crying
	2	Moaning quiet vocalizing gentle or whimpering cry
	3	Full lunged cry or sobbing
	4	Full lunged cry more than baseline cry

Tabelle 2.2: Übersicht über Pain-Scores

2.2 Schmerz-Schrei aus medizinischer Sicht

Die Frage ist: Woher kommen diese unterschiedlichen Bewertungen des Weinens in Tabelle 2.2? Gibt es eine Pain-Score, die aus wissenchaftlicher Sicht „recht hat“? Dieser Fragestellung unterliegen unterliegen zwei grundlegendere Fragen: 1.) Ist es überhaupt möglich, anhand der akustischen Eigenschaften den Grund für den Schrei abzuleiten, also beispielsweise Hunger, Einsamkeit oder Schmerz? Anders formuliert: Gibt es überhaupt so etwas wie einen Schmerz-Schrei? 2.) Ist es möglich, anhand der akustischen Eigenschaften den Schweregrad des Unwohlseins abzuleiten (also beispielsweise den Grad des Schrei-Versursachenden Schmerzes)?

Die Annahme, dass es möglich ist, aus dem Schreien den Grund abzuleiten, wird als „Cry-Types Hypothesis“ bezeichnet. Die berühmtesten Befürworter dieser Hypothese ist eine skandinavische Forschungsgruppe, auch bezeichnet als „Scandinavian Cry-Group“, die diese Idee in dem Buch „Infant Crying: Theoretical and Research Perspectives“[2] publik machte. Die Annahme ist, dass die verschiedenen Ursachen *Hunger, Freude, Schmerz, Geburt und Anderes* klare Unterschiede hinsichtlich ihrer akustischen Merkmale aufweisen, welche an einem Spektogramm ablesbar seien. Entsprechende Beispiele werden in dem Buch gegeben. Nur einige Jahre Später zeigte Müller et al [7] in einem Paper, dass bei leichter Veränderung der Bedingungen der Experimente die Unterscheidung nicht möglich ist. Die Gegenhypothese ist, dass Weinen „nichts als undifferenziertes Rauschen“ sei. 50 Jahre später liegt kein anerkannter Beweis für die eine oder andere Hypothese vor. Es gibt nur starke Hinweise dafür, dass die Plötzlichkeit des Eintretens des Schreigrundes hörbar ist. Ein plötzliches Ereignis, wie ein Nadelstich oder ein lautes Geräuch, führen auch zu einem plötzlich beginnenden Schreien. Ein langsam einretendes Ereignis, wie ein langsam immer stärker werdender physischer Schmerz oder langsam eintretender Hunger führen auch zu einem langsam eintretenden Weinen. Da keine Einigung herrscht, wird empfohlen, den Grund aus dem Kontext abzuleiten.[28]

Die Zweite Frage nach der Ableitung der Stärke des Unwohlseins aus den akustischen Eigenschaften des Geschreis wird in der Fachsprache unter dem Begriff *Cry as a graded Signal* subsumiert. Je „stärker“ das Weinen, desto höher das Unwohlsein (*Level of Distress (LoD)*) des Säuglings. Tatsächlich bemessen wird dabei der von dem Beobachter vermutete Grad des Unwohlsein des Babies, und nicht der tatsächliche Grad, da dieser ohne die Möglichkeit der direkten Befragung des Kindes nie mit absoluter Sicherheit bestimmt werden kann. Dieser vermutete LoD wird entweder durch das subjektive Empfinden der Beobachter oder durch Pain-Scores festgestellt. Ein hohes Level of Distress hat vor allem

eine schnelle Reaktion der Aufsichtspersonen zur Beruhigung des Babies zur Folge, womit dem Geschrei eine Art Alarm-Funktion zukommt. Es gibt starke Hinweise darauf, dass das Level of Distress anhand objektiv messbarer Eigenschaften des Audiosignals bestimmt werden kann. So herrscht beispielsweise weitesgehend Einigung darüber, dass ein „lang“ anhaltendes Geschrei auf einen hohen Level of Distress hinweist. Insofern aus dem Kontext des Schreiens Schmerz als wahrscheinlichste Ursache eingegrenzt werden kann, kann aus einem hohen Level of Distress ein hoher Schmerz abgeleitet werden. [28] und [27]

Es herrscht wiederum keine Einigung darüber, welche akustischen Eigenschaften im Detail ein hohes Level of Distress anzeigen. Carlo V Bellieni et al [4] haben festgestellt, dass bei sehr hohem Schmerz in Bezug auf die DAN-Scala (siehe Tabelle 2.2) die Tonhöhe des Geschreis steigt. Qiaobing Xie et al [27] haben festgestellt, dass häufiges und „verzerrtes“ Schreien (ohne feststellbares Grundfrequenz, da der Ton stimmlos erzeugt wird) auf einen hohen Level of Distress hinweist.[28] Diese Uneinigkeit hat wahrscheinlich zu den verschiedenen Bewertungen in den Pain-Scores geführt. 2.2.

2.3 Phyisio-Akustische Modellierung des Weinens

Das Ziel dieses Kapitels ist die Schaffung eines einheitlichen Vokabulares, auf den sich bezogen wird, um das Schreien eines Babys zu beschreiben. Die hier vorgestellten Begriffe stammen sowohl aus dem Buch „A Physioacoustic Model of the Infant Cry “ H Golub und M Corwin [13] als auch aus dem Paper „Rythmic organization of the Sound of Infant Cry “ von Zeskind et al.[25]

Die Lautäußerung eines Neugeborenen, umgangssprachlich auch als „Weinen“ oder „Schreien“ bezeichnet, lässt sich im allgemeinen beschreiben als das „rythmische Wiederholen eines beim ausatmen erzeugen Geräusches, einer kurzen Pause, einem Einatmungs-Geräusch, einer zweiten Pause, und dem erneuten Beginnen des Ausatmungs-Geräusches.“[34].

Das Vokabular, welches insbesondere von H Golub und M Corwin geschaffen wurde, ist sehr umfassend. An dieser Stelle wird eine Auswahl grundlegender Begrifflichkeiten vorgestellt, die in dieser Arbeit gebraucht werden. Sie werden in Abbildung 2.1 veranschaulicht.

Expiration beschreibt den Klang, der bei einem einzelnen, ununterbrochenem Ausatmen mit Aktivierung der Stimmbänder durch das Baby erzeugt wird. [25]. Der von Golub et al [13] verwendete Begriff **Cry-Unit** wird in dieser Arbeit synonym verwendet. Umgangssprachlich ist handelt es sich um einen einzelnen, ununterbrochenen *Schrei*.

Inspiration beschreibt den Klang, der beim Einatmen durch das Baby erzeugt wird.

Burst beshreibt die Einheit von einer Expiration und der darauf folgenden Inspiration. Das heisst, dass die zeitliche Dauer eines Bursts sowohl das Expiration-Geräusch, das Inspiration-Geräusch als auch die beiden Pausen zwischen diesen Geräuschen umfasst. Praktisch ergibt sich das Problem, dass vor allem bei stärkerem Hintergrundrauschen die Inspiration-Geräusche häufig weder hörbar noch auf dem Spektrogramm erkennbar sind. Daher wird die Zeitdauer eines Bursts oder Cry-Unit vom Beginn einer Expiration bis zum Beginn der darauf folgenden Expiration definiert und somit allein von den Expirations auf die Bursts geschlossen. Implizit wird somit eine Inspiration zwischen zwei Expirations angenommen.

Cry die insgesamte klangliche Antwort zu einem spezifischen Stimulus. Eine Gruppe mehre-

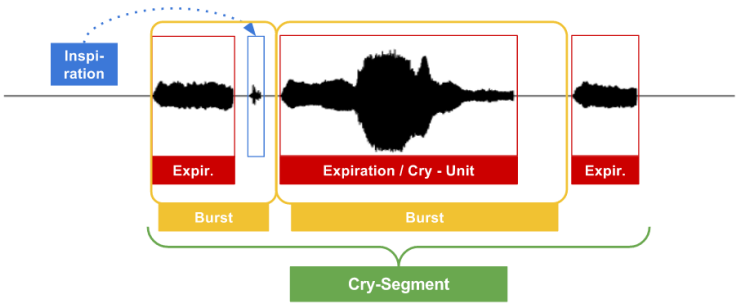

Abbildung 2.1: Veranschaulichung des Grundvokabulars

rer Cry-Units.[13] In dieser Arbeit wird ein *Cry* auch als **Cry-Segment** bezeichnet, um Verwechslungen zu vermeiden.

Weiterhin wurden von H Golub und M Corwin [13] Cry-Units in eine der folgenden drei Kategorien eingeführt:

Phonation beschreibt eine Cry-Unit mit einer „vollen Vibration der Stimmbänder“ mit einer Grundfrequenz zwischen 250 und 700 Hz. Entspricht umgangssprachlich einem Weinen mit einem „klaren, hörbaren Ton“.

Hyper-Phonation beschreibt eine Cry-Unit mit einer „falsetto-artigem Vibration der Stimmbänder“ mit einer Grundfrequenz zwischen 1000 und 2000 Hz. Entspricht umgangssprachlich einem Weinen mit einem „sehr hohen, aber klaren, hörbaren Ton“.

Dysphonation beschreibt eine Cry-Unit ohne klar feststellbare Tonhöhe, produziert durch Turbulenzen an den Stimmbändern. Entspricht umgangsprachlichen dem „Brüllen oder Krächzen“.

Eine Cry-Unit gehört dabei mindestens einer dieser Kategorien an, kann aber auch in seinem zeitlichen Verlauf die Kategorie wechseln. H Golub und M Corwin [13] stellen weiterhin eine Reihe an charakteristischen Eigenschaften vor, die in Bezug auf ein Cry-Segment berechnet werden.

Latency-Period beschreibt die Dauer zwischen dem zufügen eines Schmerz-Stimulus und dem beginn des ersten Cry-Bursts des Segmentes

Duration beschreibt die insgesamte Zeitdauer des Cry-Segmentes. Es wird keine genaue Definition gegeben, wodurch Beginn und Ende definiert werden. Das Segment endet dort, wo es „scheint, aufzuhören“.

Maximum-Pitch beschreibt die höchste festgetellte Grunfrequenz des Segmentes.

... und viele weitere, die in [13] nachgelesen werden können, aus Platzgründen an dieser Stelle jedoch nicht vollständig genannt werden.

3 System zur Visualisierung akustischer Schmerz-Scores

Das Ziel dieser Arbeit ist die Ableitung des Schmerz-Scores aus einem Audiosignal sowie die darauf folgende Visualisierung dieser Schmerz-Scores. Folgende Anforderungen werden an das System gestellt:

1. Das System muss dazu in der Lage sein, aus den akustischen Eigenschaften des Weinens die Schmerz-Score abzuleiten
2. Das System muss dazu in der Lage sein, die Schmer-Score zu visualiseren.
3. Die Verarbeitungspipeline muss genug Flexibilität bieten, um beliebige Pain-Scores einzubinden.
4. Die Analyse muss auch bei nicht-optimalen akustischen Bedingungen Einsatzfähig sein.
5. Die Methoden müssen kontinuierlich eingesetzt werden können. Das heißt, dass zu einem Analysezeitpunkt nur Informationen verwendet werden können, die nicht in der Zukunft liegen.

Im folgenden wird ein Überblick über bereits verföffentlichte Ansätze zur Analyse von akustischen Signalen Neugeborener oder sonstiger automatisierter Systeme zur Ableitung der Pain-Score gegeben.

3.1 Literatur-Übeblick

Bei der Aufgabenstellung handelt es sich grob betrachtet um einen Klassifikations/Regressions-Aufgabe, bei der aus den Eigenschaften des Audiossignals mit den Kindlichen Lautäußerungen eine Schluss gezogen werden soll. In dieser Aufgabenstellung ist der Schluss eine Pain-Score. An dieser Stelle wird ein Überblick über Veröffentlichungen gegeben, in denen ähnliche Aufgabenstellungen bearbeitet worden.

Der Großteil der Veröffentlichungen stellt Systeme Klassifikation einzelner Cry-Units vor, entweder bezüglich der Wein-Ursache (Hunger, Angst, Schmerz...) oder zur Diagnose bestimmer Krankheiten. Diese Methoden sind nicht für die kontinuierliche Analyse geeignet, sondern haben das Ziel, bezüglich einer bereits vorliegenden Cry-Unit eine möglichst hohe Klassifizierungs-Accuracy zu erzielen. Probleme wie Hintergrundrauschen, Berechnungsaufwand oder kontextuelle Informationen werden selten mit in Betracht gezogen. Beispiele für solche Systeme sind die von Abdulaziz et al [35] oder Furh et al [31].

Várallyay stellt in seiner Dissertation „Analysis of the Infant Cry with Objective Methods“ [33] Methoden zur automatisierten Analyse kindlicher Lautäußerungen vor. Das eigentliche Ziel der Dissertation ist die Erforschung der Unterschiede zwischen den Lautäußerungen gesunder und tauber Neugeborener. Die automatisierte Verarbeitungs-Pipeline der Audiosi-

gnale ist dabei ein „Nebenprodukt“ zur schnelleren Auswertung der Signale. Die Auswertung muss nicht kontinuierlich erfolgen. In der vorgestellten Verarbeitungspipeline wird das Eingangssignal in Zeitfenster weniger Millisekunden zerlegt und jedes Fenster auf Basis der Fenstereigenschaften als Stimmhaft oder nicht-Stimmhaft klassifiziert. Die stimmhaften Signalfenster werden zu *Segmenten* zusammengefasst (in Kapitel 2.3 als Cry-Unit bezeichnet). Auf Basis der Segmente werden Auswertungen bezüglich der Zeit-Bereiches (Durchschnittliche Segmentlänge, Pausenlängen etc.), des Frequenz-Bereiches (Grund-Frequenz, Formanten-Frequenzen etc.) und des Melodie-Verlaufes (Melodie-typ) angestellt. Analysiert wurden Signale mit einer Länge von 10 bis 100 Sekunden, die Lautäußerungen von Babies mit oder ohne Hörbehinderung beinhalten. Aus den Auswertungsergebnisse stellt Várallyay die wichtigsten Unterscheidungsmerkmale zwischen tauben und gesunden Babies fest. In der Dissertation [33] wird ein Überblick über das Vorgehen und die Ergebnisse gegeben. Die Verarbeitungsschritte werden detailllierter in einzelnen Veröffentlichungen beschrieben, auf die der Autor dieser Arbeit jedoch kein Zugriff gewährt wurde.

Cohen et al haben 2012 in dem Paper „Infant Cry Analysis and Detection “ [5] ein System zur Analyse der akustischen Signale von Neugeborenen vorgestellt. Dieses System klassifziert die Audio-Signale in eine der drei Klassen *Cry, No Cry* und *No Activity*. Mit *Cry* sind Lautäußerungen gemeint, die eine potentiell Gefahr für das Baby anzeigen, wie z.B. wie Schmerz oder Hunger. *No Cry* meint, dass das Baby zwar Laute von sich gibt, diese aber keine potentielle Gefahr anzeigen. emphNo Activity meint keinerlei Lautäußerung. Die Verarbeitungs-Pipeline wird detailliert vorgestellt und ist für die kontinuierliche Verarbeitung mit einer gewissen Verzögerungszeit spezialisiert. Das Signal wird in überlappende *Segmente* à 10 Sekunden zerlegt. Die Stimmaktivität in dem Segment wird algorithmisch festgestellt. Wenn Aktivität vorliegt, wird das Segment in Sections à 1 Sekunden zerlegt und die Stimmaktivität für jede Section analysiert. Wird genügend Stimmaktivität für eine Section festgestellt, wird die Section in *Frames*) à 32 Millisekunden zerlegt und Features für jedes Signalfenster errechnet. Mit Hilfe eines Predictors werden die Frames in *Cry, No-Cry, No-Activity* klassifiziert, wobei Kontextuelle Informationen der umliegenden Frames mit einbezogen werden. Aus den Klassen der Frames wird auf die Klasse der Section geschlossen, und aus den Klassen der Sections auf die Klasse des 10 Sekunden langen Segments. Das System hat in Bezug auf diese Arbeit den Vorteil, dass ebenfalls die kontinuierliche Verarbeitung im Vordergrund steht. Der Nachteil an dieser Methode ist, dass die zeitliche Einheit, für die die Klassifizierung vorgenommen wird, auf unflexibel auf 10 Sekunden festgelegt ist. Daher müsste diese Verarbeitungspipeline abgewandelt werden, um Anstelle der Ableitung der drei genannten Klassen einer Pain-Score zu verwenden, die einen längeren Beobachtungszeitraum als 10 Sekunden benötigt.

Pal et al haben 2006 in dem Paper „Emotion detection from infant facial experessions and cries“ [26] ein System zur Emotions-Detektion bei Neugeborenen aus Aufnahmen des Gesichtsausdruck und akustischen Aufnahmen des Weinens vorgestellt. Die zu erkennenden Emotionen sind *Traurigkeit, Wut, Hunger, Angst und Schmerz*. Es wird nicht erwähnt, ob die Analyse kontinuierlich oder nicht-kontinuierlich erfolgt. Bei der Verarbeitung der akustischen Signale werden die Features *Grund-Tonhöhe* und die *Frequenz der ersten drei Formanten* extrahiert und mit einem Klassifikations-Algorithmus klassifiziert. Es werden keinerlei Details genannt, inwiefern die Features aus kurzen Signalfenstern oder längeren Signalabschnitten errechnet werden, welche Vorverarbeitungsschritte angewandt werden und ob die Klassfizierung auf Ebene der Signalfenster oder über längere Zeitabschnitte

hingweg geschieht. Die Veröffentlichung liefert Ideen über mögliche Features, bietet jedoch keinen Einblick in die Verarbeitungspipeline.

Zamzi et al haben 2016 in dem Paper „An Approach for Automated Multimodal Analysis of Infants' Pain" [10] ein System zur automatisierten und kontinuierlichen mutlimodalen Analyse von Neugeborenen zur Ableitung des Schmerzes vorgestellt. Das System trägt den Namen *MPAS*. Der Insgesamte Schmerzgrad wird aus den Analyseergebnissen der monomdaler Schmerzindikatoren für *Gesichtsausdruck, Körperbewegung, Vitalfunktionen und Weinen* errechnet. Das Ziel des Projektes kommt der Aufgabenstellung dieser Masterarbeit am nächsten, da es ebenfalls um die Ableitung von Schmerz in einem multimodalen Verbund geht. Es wird jedoch nicht die Anfoderung gestellt, Flexibilität in der Wahl der Pain-Score zu gewährleisten . Während in der Veröffentlichung die Analyse der ersten drei genannten Schmerzindikatoren angekündigt wird, werden daraufhin die Methoden zur Analyse der akustischen Signale *nicht* erläutert. Auch die ersten Validierungs-Ergebnisse beziehen sich nur auf den Gesichtsausdruck, Körperbewegung und Vitalfunktionen. Es ist nicht klar, ob die Miteinbeziehung akutischer Signale fallen gelassen wurde. Die Ausführungen konzentrieren sich dazu vermehrt auf die Methoden zur Kombination der Auswertungsergebnisse der monomodalen Schmerzindikatoren. Die Verarbeitungs-Pipelines der monomdalen Schmerzindikatorn werden nur grob vorgestellt.

3.2 Verarbeitungs-Pipeline

In Kapitel 3.1 wurden verschiedene Systeme vorgestellt, deren Problemstellungen dem Thema dieser Masterarbeit ähneln. Keine der präsentierten Verarbeitungs-Pipelines eignet sich, um mit nur leichten Anpassungen übernommen werden zu können: Entweder sind die Verarbeitungsschritte nicht für die kontinuierliche Verarbeitung konzipiert [35] [31] [33], nicht genügen abstrahiert, um für andere Klassifizierungen als die ursprünglich geplanten abgewandelt werden zu können [5], oder stellen die Verarbeitungs-Pipeline nicht vor [26] [10].

In dieser Arbeit wird die folgende Verarbeitungs-Pipeline vorgestellt. Sie wird in in Abbildung 3.1 visualisiert.

1. **Pre-Processing**. Vorverarbeitung des Signals. An dieser Stelle geschieht eine Anpassung der Lautstärke mit Hilfe eines Audiocompressors zur besseren Kontrolle der Signalenergie. Das Pre-Processing wird in Kapitel 3.3 vorgestellt.
2. **Voice-Activity-Detection**. Das Audiosignal wird in einander überlappende Zeitfenster weniger Millisekunden zerschnitten. Mit Hilfe eines Klassifizierungs-Algorithmus werden die Zeitfenster in als *Stimmhaft* oder *nicht Stimmhaft* markiert. Ununterbrochene Reihen von Stimmhaften Signalfenstern werden zu *Cry-Units* zusammengefasst. Das Ergebnis der Voice-Activity-Detection sind Markierungen der Anfangs- und Endzeitpunkte *Cry-Units*, die die Basis aller darauf folgenden Auswertungen bilden. Diese Idee ist aus der Dissertation von Várallyay [33, S. 16 - 17] übernommen, welcher Cry-Units als *Segments* bezeichnet. Die Voice-Activity-Detection wird in Kapitel 3.4 vorgestellt.
3. **Segmentierung** (engl *Segmenting*), das Zusammenfassen mehrer Cry-Units zu Segmenten, welche in Kapitel 2.3 als *Cry* bezeichnet werden. Dieser Schritt ist Notwendig, weil die Ableitung der Schmerz-Scores nicht aus den Informationen einer Cry-Unit, sondern aus dem Verbund mehrerer Cry-Units geschieht. Keine der in Kapitel 3.1 vorgestellten

Veröffentlichungen beschreibt ein Verfahren, welches adaptiert werden können, entweder weil der Input der Algorithmen bereits auf die Länge der Segmente beschnitten wurde, oder weil ein eventuell verwendetes Verfahren nicht beschrieben wird. Daher wird ein simpler Algorithmus für die Segmentierung vorgeschlagen, welcher für die kontinuierliche Auswertung implementiert werden kann. Die Segmentierung wird in Kapitel 3.5 vorgestellt.

4. **Feature-Extraction**, das heißt die Berechnung von Eigenschaften für jedes Segment, die für die Ableitung der Pain-Scores von Interesse sind. Diese Eigenschaften weden in Regelmäßigen Zeitintervallen innheralb des Segmentes abgefragt. Diese Feature-Extraktion ist ein notwendiger Vorbereitungsschritt für die Anschließende Klassifikation/Regression, welcher in allen in Kapitel 3.1 vorgestellten Veröffentlichungen durchgeführt wird. Die Feauture-Extraction wird in Kapitel 3.6 vorgestellt.
5. **Ableitung der Pain Score** aus den Features des Segmentes. Während es sich in allen in Kapitel 3.1 vorgestellten Veröffentlichungen um Klassifikationsaufgaben handelte, wird hier eine Regression vorgenommen. Die Feature-Extraction wird in Kapitel 3.7 vorgestellt.
6. **Visualisierung** der errechneten Pain-Score. In dieser Arbeit werden mehrere Versionen eines Systems vorgeschlagen, welche den zeitlichen Verlauf auf Ampel-Farben abbilden, welche die höhe der Schmerz-Score codieren. die Visualisierung wird in Kapitel 3.8

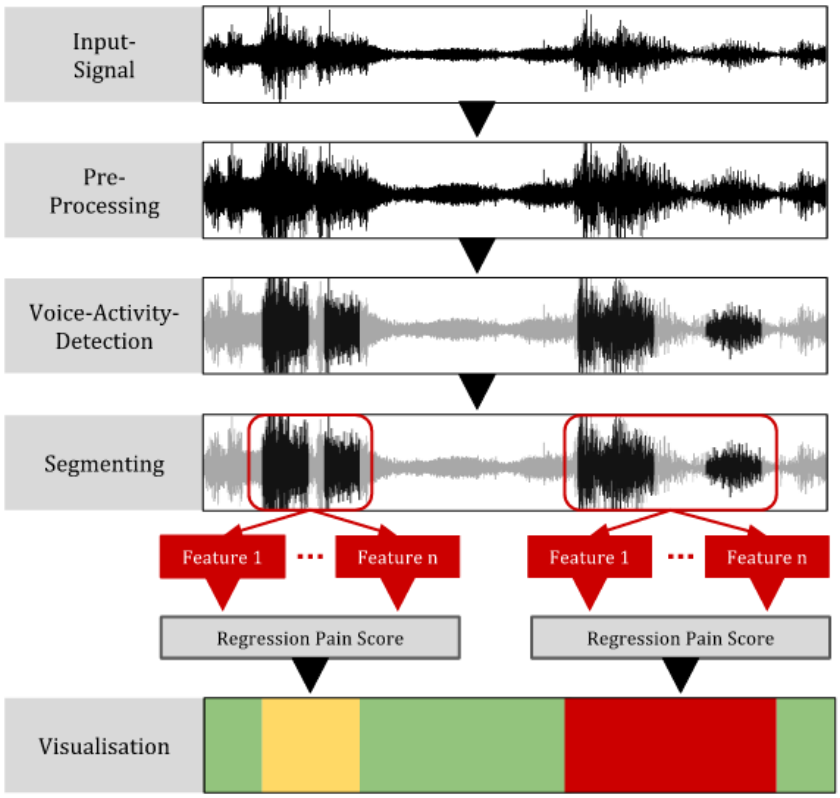

Abbildung 3.1: Die Verarbeitungs-Pipeline des vorgestellten Systems

3.3 Preprocessing

Beim Preprocessing wird das Signal so vorverarbeitet, dass Störeinflüsse auf die darauf folgenden Verarbeitungsschritte von vorneherein minimiert werden. Welches Pre-Processing durchgefüht wird, ist Abhägig von der konrketen Aufgabenstellung. So werden beispielsweise bei einigen Algorithmen zur Voice-Activity-Detection, also dem markieren stimmhafter Signalabschnitte, Tiefpass, Hochpass- und Bandpassfilter eingesetzt, um diejenigen Frequenzanteile herauszufiltern, die von der Stimme nicht produziert werden können [19] [30] [18]. Bei einigen Pitch-Detection-Algorithmen wird *Centerclipping* eingesetzt, also das 0-Setzen von Samples mit $x[i] < 0.5 \cdot$ Maximalaussteuerung.[8]

In dieser Arbeit wurde sich für eine Vorverarbeitung entschieden, bei der das Signal hinsichtlich seiner Dynamik im Zeitbereich eingegrenzt wird. Dies ist ein typischer Vorverarbeitungschritt bei Sprachaufnahmen. Hintergrund ist, dass sehr kurz, aber sehr laute Pegelspitzen weit über dem Durchschnittspegel des Gesamtsignals den Maximalwert des Signals unnötig begrenzen und die Signalenergie so gering halten. Da die Testsignale, die in dieser Arbeit verwendet werden, aus inhomgenen Quellen stammen und sehr unterschiedliche Lautstärken haben, wird so gewährleistet, dass sie zumindest ähnliche Energien haben. An dieser Stelle werden (noch) keine Frequenanteile herausgefiltert, um keine Frequenzen zu verlieren, die in den späteren Verareitungsschritten wieder Voice-Activity-Detection 3.4 oder der Feature-Extraction eventuell noch benötigt werden.

Die Dynamikeinschränkung wird mit Hilfe eines Audiokompressor umgesetzt. Ein Audiokompressor verringert Signalspitzen, die über einen festgelegten *Schwellwert (Threshold)* liegen, um ein festgelegtes *Verhältnis (Ratio)*. Ein Threshold von 0.3 mit Ratio von 0.5 bedeutet beispielsweise, dass alle Signalspitzen, die den Wert 0.3 überschreiten oder -0.3 unterschreiten, um 50% verringert werden. Ein Kompressor kann auf die Überschreitung des Thresholds erst nach einer als *Attack* bezeichneten Verzögerung reagieren, und bei erneuten Verlassen des Thresholdes mit einer als *Release* bezeichneten Verzögerung nachwirken. Signalspitzen werden so verringert und die Lautstärke-Dynammik eingeschränkt. Die tatsächliche Erhöhung der Signalenergie geschieht im Anschluss durch die Anhebung der insgesamten Signallautstärke, wie Beispielsweise der Normalisierung des Signals auf den Maximalpegel. Abbildung 3.2 zeigt die Parameter eines solchen Audio-Kompressors.

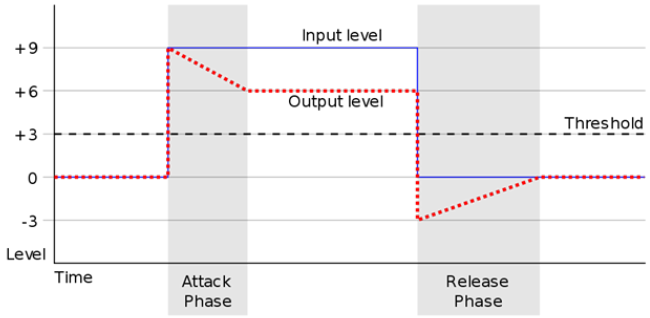

Abbildung 3.2: Parameter eines Audio-Kompressors

Der entwickelte Kompressor automatisiert die Einstellung von Threshold und Ratio auf Grundlage des Root-Mean-Square (RMS) des Signales x der Länge N. Der RMS-Wert ist ein Maß für die durchschnittliche Signalenergie und wird wie nach Formel 3.1 berechnet.

Threshold und Ratio werden nach den Formeln 3.2 und 3.3 berechnet, wobei der Parameter r_a den Ziel-RMS-Wert anbgibt und mit dem Wert.

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x[n]^2} \tag{3.1}$$

$$\mathrm{THold}(x) = \left[\frac{\mathrm{RMS}(x[])}{r_a}\right]^2 \tag{3.2}$$

$$\mathrm{Ratio}(x) = \left[\frac{\mathrm{RMS}(x[])}{r_a}\right]^2 \tag{3.3}$$

Abbildung 3.3 zeigt das ein Signal vor und nach dem Preprocessings. Zu sehen ist, dass die Lautstärke der einzelnen Schrei-Einheiten nach der Anpassung einheitlicher ist.

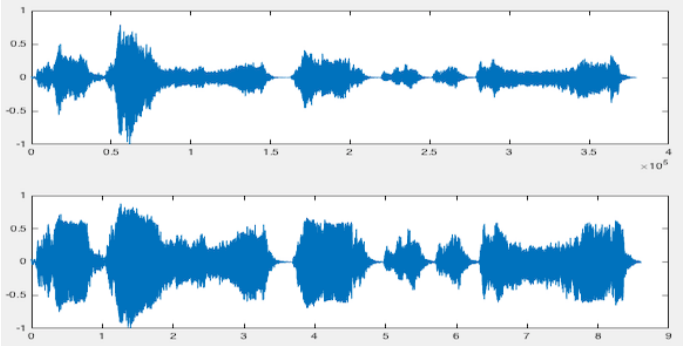

Abbildung 3.3: Ergebnis des Preprocessings

3.4 Voice Activity Detection

Das Ziel ist, in einem Audiosignal diejenigen Stellen zu markieren, in denen Stimme enthalten ist. Abbilung 3.4 visualisiert ein Beispiel für eine solche Markierung: Zu sehen ist der Zeitbereich eines Audiosignales mit drei klar erkennbaren Cry-Units. Die Rote Linie, die das Signal überspannt, bildet die Zeiteinheiten des Eingangssignales in die binären Kategorien *Stimmhaft* und *Stille* ab.

Die Erkennung des Vorhandenseins von Stimme in einem Signal wird als *Voice Activity Detection (VAD)* oder auch *Speech Detection* bezeichnet. Das Ziel ist die Unterscheidung von denjenigen Zeiträumen im Signal, in denen Stimme enthalten ist, von den Zeiträumen

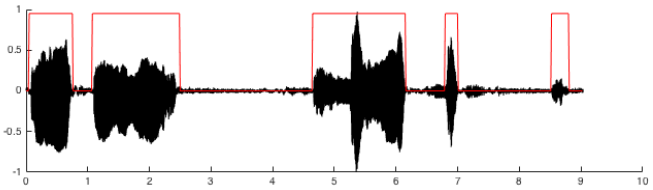

Abbildung 3.4: Markierung von Schreigeräuschen im Audiosignal

ohne Stimme. Die größte Herausforderung für VAD-Algorithmen ist die robuste Erkennung bei Signalen mit Rauschen unbekannter Stärke und Natur. [18, S. 1] [32, S. 1]

Der Grundlegende Aufbau eines VAD-Algorithmus ist wie folgt:

1. **Windowing**: Unterteilung des Signals in (einander überlappende) Fenster, für die Entscheidung durchgeführt werden soll.
2. **Feature-Extraction** aus den einzelnen Fenstern
3. **Thresholding / Klassifizierung** über die Präsens oder Nicht-Präsens von Stimme für jedes Zeitfenster auf Grundlage der Extrahierten Features mit Hilfe von Entscheidungsregeln wie Grenzwerten.
4. **Decision-Smoothing**, das nachträgliche Hinzufügen oder Entfernen von Entscheidungen mit Hilfe von kontextuellen Informationen der umliegenden Entschiedungen.[16, S. 8 - 9] [18, S. 1 - 2]

Der an dieser Stelle entwickelte Ansatz ist eine Kombination aus den Ideen, die von Moattar et al [20], Kristjansson et al [32], Waheed et al [19], Ahmadi et al [30] und Shen et al[17] vorgestellt wurden.

3.4.1 Windowing

Das Signal $x[\,]$ wird nach den in Kapitel **??** beschriebenen Verfahren nach Gleichung **??** in die Signalfenster $x_0[\,] \ldots x_m[\,]$ zerlegt, bezeichnet als „Windowing“. Die Signalfenster werden zunächst im Zeitbereich belassen. Es wurde sich für die Waheed et al [19] vorgestellte Fensterlänge von 25 ms entschieden, als Kompromiss zwischen den von Moattar et al[20] empfohlenen 10 ms und den von Ahmadi et al [30] empfohlenen 40 ms. Die Fenster überlappen einander um 50%, das heisst 12.5 ms.

3.4.2 Feature Extraction

Für jedes Signalfenster $x_0[\,]...x_m[\,]$ à 25 ms werden die folgenden Features aus den Kategorien **Zeit-Bereich**, **Frequenz-Bereich**, **Cesptrum** und **Auto-Korrelation** berechnet.

Zeit-Bereich

Im Zeit-Bereich werden die beiden Features *Root-Mean-Square*-Wert *[RMS]* und *Zero-Crossing-Rate [ZCR]* berechnet.

Moattar et al [20] bezeichnen den Energiegehalt eines Signals als das für die VAD am häufigsten Angewandte Feature. Daher wird der RMS-Wert eines Signalfensters nach Gleichung 3.1 verwendet. Hintergrund ist, dass der Energiegehalt eines Stimmsignals typischerweise Höher ist als der des Hintergrundrauschens. Bei geringen Signal-to-Noise-Ratios ist diese Bedingung jedoch nicht immer gegeben. Als zweites Feature des Zeitbereiches wird die in verwendete *Zero-Crossing-Rate* berechnet. Die ZCR nach Formel 3.4 gibt an, wie häufig ein Vorzeichenwechsel im Signal vorkommt. Eine höhere ZCR weist auf Stille hin, da Rauschen typischerweise einen höheren ZCR als Signale mit einer Periodizität aufweist.

Problematisch ist dieses Kriterium bei Signalen, bei denen gar kein Hintergrundrauschen vorliegt, da solche Signalfenster eine ZCR von 0 aufweisen. [30]

$$\mathrm{ZCR}(x_i[\,]) = \sum_{1}^{N-1} |\mathrm{sng}(x_i[n]) - \mathrm{sng}(x_i[n-1])| \tag{3.4}$$

Autokorrelation

Neben den in Kapitel 3.4.2 geannten „einfachen“ Features des Zeitbereiches wird zur VAD die Autokorrelation verwendet. Wie in Kapitel **??** ausgeführt, weisen stimmhafte Signale eine höhere Periodizität als das Hintergrundrauschen auf. Daher eignet sich die in Kapitel **??** vorgestellte Autokorrelation, um diese Periodizität festzustellen. Es werden die Features *Maximum Autocorrelation Peak* [*aMax*] und (*Autocorrelation Peak Count*) [*aCount*] berechnet.

Beide Features werden von Kristjansson et al [32, S. 1 - 2] zur VAD erprobt. Die *höchste Magnitude der Autokorrelation* (*Maximum Autocorrelation Peak*) wird nach der Formel 3.5 definiert und bestimmt die höchste Magnitude im Autokorrelations-Signal. Eine höherer [*aMax*]-Wert spricht für eine dominante Grundfrequenz im Signal. Das zweite Feature ist die *Anzahl an Autokorrelations-Spitzen* nach Formel 3.6. Ein höherer [*aCount*]-Wert spricht für das vorhandensein dominanter Obertonwellen im Signal. Aus Kapitel 2.3 geht hervor, dass die Grundfrequenz von Neugeborenen zwischen $200 - 2000\,\mathrm{Hz}$ liegt, weshalb auch nur in Lags dieses Bereichs die Autokorrelation durchgeführt wurde.

$$\mathrm{aMax}(x_i[\,]) = \max_k \mathrm{mag}\{\text{NA-Corr}_k(x_i[\,])\} \tag{3.5}$$

$$\mathrm{aCount}(x_i[\,]) = \operatorname*{count}_k \mathrm{mag}\{\text{NA-Corr}_k(x_i[\,])\} \tag{3.6}$$

Frequenz-Bereich

Aus dem **Frequenz-Bereich** werden die drei Features *unnormalisierte spektrale Entropie* [$SEnt_u$], *normalisierte spektrale Entropie* [$SEnt_n$] und *dominanteste Frequenzkomponenten* [f_{Dom}] berechnet.

Als Vorbereitungsschritt werden die Signalfenster des Zeit-Bereiches $x_0[\,] \dots x_m[\,]$ zunächst mit der in Kapitel **??** vorgestellten Short Time Fourier Transformation in die *Frequenz-Fenster* $X_0[\,] \dots X_m[\,]$ transformiert. Das heißt, dass $X_i[\,] = \mathrm{DFT}(w[\,] \cdot x_i[\,])$. Es wurde eine 2048 Punkte Lange FFT und eine Hamming-Window als Fensterfunktion verwendet.

Kristjansson et al [32, S. 2] verwenden die *spektrale Entropie* Voice Activity Detection. Dabei wird das Spektrum des Frequenzfensters X_i als Wahrscheinlichkeitsverteilung betrachtet. Die Entropie als Maß zur „Unreinheit“ wird in Kapitel **??** erläutert. Die *normalisierte spektrale Entropie* wird nach der Formel 3.8 berechnet. Das Signal px_i ergibt sich durch die Normalisierung des N-Punkte langen Spektrums nach Formel 3.7. Neben der in [32] vorgestellten normalisierten spektralen Entropie wird zusätzlich die *unnormalisierte Spektrale Entropie* nach Formel 3.9 berechnet. Bei dieser wird das Spektrum nicht

normalisiert, das heißt, es gilt $px_i[f] = X_i[f]$. Somit hat Energie des Signals einen größeren Einfluss die höhe des Features. Bei der normalisierten spektralen Entropie ist zu erwarten, dass Frequenzfenster mit Hintergrundrauschen eine höhere Entropie haben als Fenster mit Stimme . Bei der unnormalisierten spektralen Entropie ist zu erwarten, dass Signalfenster mit Stimme eine höherer Spektrale Entropie haben als Fenster mit Stille.[1]

In die Berechnungen wurden nur die Frequenzen im Bereich von 200 - 8000 Hz mit einbezogen, da aus Kapitel **??** die tiefst Mögliche Frequenz kindlicher Lautäußerung bei 200 Hz liegt und nach Shen et al [17] Stimme keine Informationen oberhalb von 8000 Hz übertragen.

$$px_i[n] = \frac{X_i[n]}{\sum_{k=1}^{N} X_i[k]} \tag{3.7}$$

$$\mathrm{SEnt}_n(px_i[\,]) = -\sum_{k=1}^{N} px_i[k] \cdot \log(px_i[k]) \tag{3.8}$$

$$\mathrm{SEnt}_u(X_i[\,]) = -\sum_{k=1}^{N} X_i[k] \cdot \log(X_i[k]) \tag{3.9}$$

Moattar et al [20, S. 2550] stellen die *dominanteste Frequenzkomponente* zur Voice-Activity-Detection vor. Für jedes Frequenzfenster X_i wird diejenige Frequenz nach Formel 3.10 berechnet, welches die höchste Amplitude hat. Es wird dabei, im Gegensatz zur spektralen Entropie, der gesamte Frequenzraum betrachtet! Ein stimmhaftes Signal hat typischerweise eine höhere f_{Dom} als ein nicht stimmhaftes Signal, bedingt durch die hohe Amplitude der Grundfrequenz.

$$f_{Dom}(X_i[\,]) = \arg\max_k \{X_i[k]\} \tag{3.10}$$

Cepstrum

In Kapitel **??** wurde das Cepstrum vorgestellt und Erläutert, wie Peaks im oberen Quefrency-Bereich auf das Vorhandensein eines periodischen, obertonreichen Signals, wie Stimme, hinweist. Aus dem Cepstrum-Bereich werden die Features *Upper Cepstrum Peak* [$Ceps_{mag}$] und *Upper Cepstrum Peak Location* [$Ceps_{loc}$] berechnet.

Ahmadi et al [30] sowie Kristjansson et al[32] schlagen vor, die *höchste Magnitude im oberen Quefrency-Bereich* (Upper Cepstrum Peak) als Feature zu verwenden. Formel 3.11 definiert die Berechnung. $c_i[\,]$ ist das Cepstrum des iten Frequenzfenster $X_{[}\,]$. Wie in Kapitel 2.3 erläutert, kann die Grundfrequenz nur zwischen 200 und 2000 Hz liegen, was einem Quefrency-Bereich von 5 - 40 ms entspricht. Folglich werden bei der Berechnung nach Formel 3.11 nur Quefrency-Werte in diesem Bereich betrachtet werden. Eine hoher $Ceps_{mag}$-Wert weist auf das Vorhandensein von Stimme für das aus dem Fenster x_i Berechneten Cepstrum x_i hin. Als zweites Features wird die Quefrency der höchsten Amplitude des Cepstrum

[1] Ein Hinweis: Kristjansson et al [32, S. 2] verwenden zur Entropie-Berechnung den Logarithmus zur Basis 10, anstatt zur Basis 2. Es ist nicht klar, ob es sich dabei um einen Fehler handelt. Zur Featureberechnung in dieser Arbeit wurde, wie in dem Paper beschrieben, ebenfalls der Logarithmus zur Basis 10 verwendet!

(Upper Cepstrum Peak Location) nach Formel 3.12 berechnet. Bei Signalfenstern mit Stille ist es wahrscheinlicher, dass sich die höchste Ampltiude am Mindest- oder Maximum-Wert des durchsuchten Qufrency-Bereiches befindet.

$$Ceps_{mag}(c_i) = \max_k \text{mag}\{\text{mag}(c[k])\} \tag{3.11}$$

$$Ceps_{loc}(c_i) = \arg\max_k \{\text{mag}(c[k])\} \tag{3.12}$$

Abbildung 3.5 visualisiert alle vorgestellten Features, die für die Voice Activity Detection eingesetzt werden. Der oberste Plot zeigt das Audiosignal aus Abbildung 3.4 mit einem Signal/Rauschabstand von 20 dB. Die Rote Linie über dem Signal zeigt, welche Zeitabschnitte stimmhaft, und welche nicht stimmhaft sind.

Konstruktion des Feature-Vektors

Für jedes Signalfenster à 25 ms werden die 9 vorgestellten Features *RMS, ZCR,* $SEnt_u$*,* $SEnt_n$*,* f_{Dom}*,* $Ceps_{mag}$*,* $Ceps_{loc}$*, aMax* und *aCount* berechnet. Wie beschrieben, sollten Signalfenster mit Stimme einen höheren Wert des jeweiligen Features erzeugen als Signalfenster mit Stille (oder, abhängig vom Feature wie der ZCR, einen tieferen).

Abbildung 3.6 zeigt in (a) die RMS-Werte eines Signals mit einem Signal-Rausch-Abstand (SNR) von 50 dB. Die Zeiträume mit Stille haben einen weitaus niedrigeren RMS-Wert als die Signalteile mit Stimme. In (b) ist das selbe Signal mit einem Signal-Rausch-Abstand von 3 dB zu sehen. Nun liegen die RMS-Werte des Rauschens nur noch knapp unter denen des Sprachsignals. Zu sehen ist, dass starkes Hintergrundrauschen ähnlich hohe Feature-Werte erzeugen kann wie die Stimme.

In [20] und [19] wird die Idee präsentiert, den Wert des jeweiligen Features zu messen, der in den stimmlosen Segmenten durch das Hintergrundrauschen erzeugt wird. Jedoch ist für ein Signalfenster x_i zum Zeitpunkt der Berechnung des Features noch nicht bekannt, ob es sich um ein Zeitfenster mit Stille oder Stimme handelt. Genau diese Entscheidung wird später auf Basis der Features getroffen. Man kann jedoch davon ausgehen, dass kein Baby-Schrei länger eine bestimmte Zeitraum t_{max} dauert, bevor das Babie Luft holen muss und somit ein Zeitfenster mit Stille entsteht. Der Mindestwert eines Features in einem Zeitraum t_{max} ist somit der Wert des Features, der mindestens durch das Hintergrundrauschen erzeugt wird. Er wird nach Formel berechnet, wobei t_{xi} die Länge eines Signalfensters x_i ist (in diesem Fall 25 ms). Das *Differenzfeature* wird definiert nach Formel 3.14 als die Differenz des für das aktuelle Signalfenster berechneten Features und dem Mindestwert dieses Feature der letzten t_{max} Sekunden.

$$\text{MinF}(Feat(x_i)) = \min_{k=i-z\ldots i}(\text{Feat}(x_k)), z = \frac{2t_{max}}{t_{xi}} \tag{3.13}$$

$$\text{DiffF}(Feat(x_i)) = Feat(x_i) - MinF(Feat(x_i)) \tag{3.14}$$

Der Featurevektor, der schlussendlich für jedes Signalfesnter $x_i[]$ berechnet wurde, besteht neben den 9 vorgestellten Features, zusätzlich aus den Differenzfeatures für einen Zeitraum

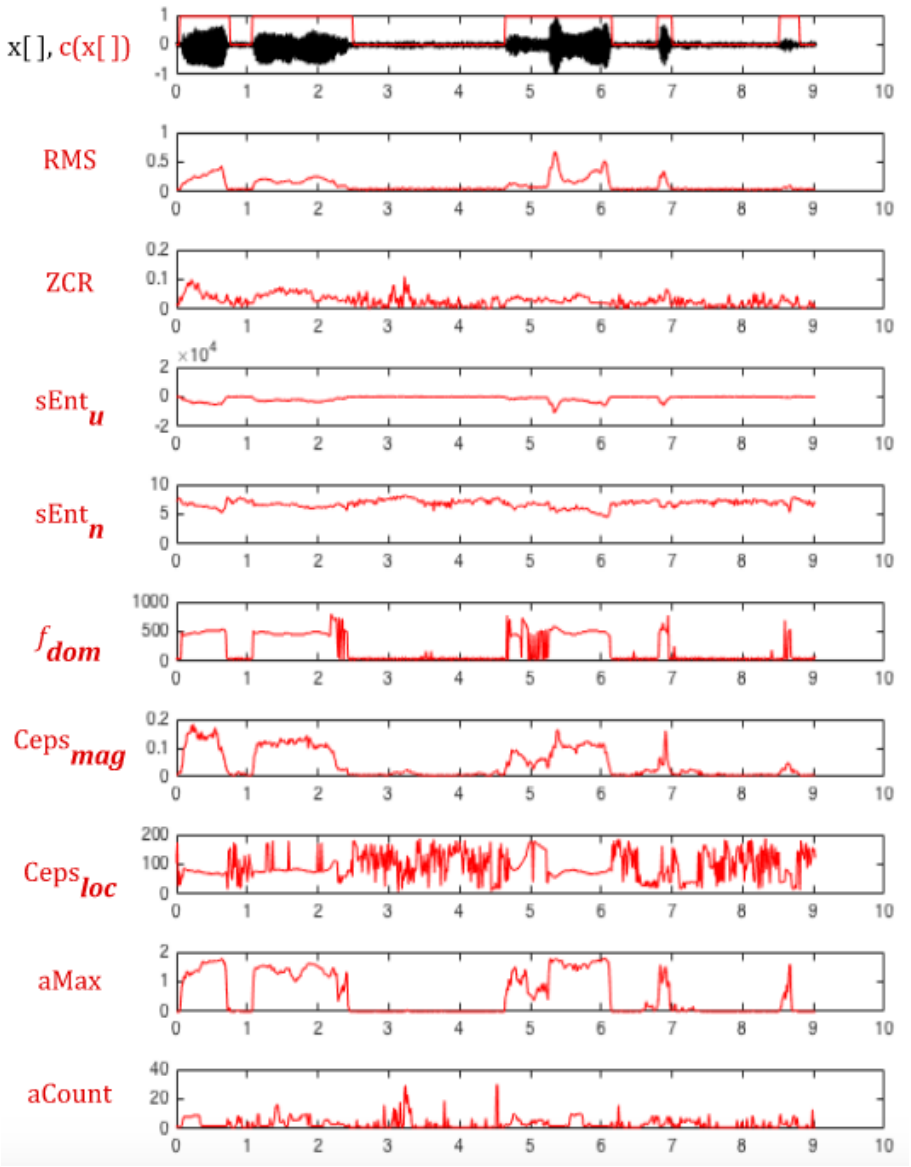

Abbildung 3.5: Übersicht über alle Features, die für die Voice Activity Detection verwendet werden.

von $t_{max} = 4$ Sekunden. Features, deren Wert bei Stimmhaften Signalfenstern geringer ist als bei stimmlosen, werden an der x-Achse gespiegelt, um Formel 3.13 anwenden zu können (das Betrifft die Features ZCR, $\mathrm{SEnt_u}$ und aCount). Das einzige Feature, welches nicht als Differenzfeature dem Feautrevektor beigefügt wurde, ist der *Upper Cepstral Peak Location*-Feature $[Ceps_{loc}]$, da es bei Stille sowohl einen höheren als auch einen niedrigeren Wert annehmen kann. Der Feature-Vektor V des Signalfensters x_i wird nach Formel 3.15

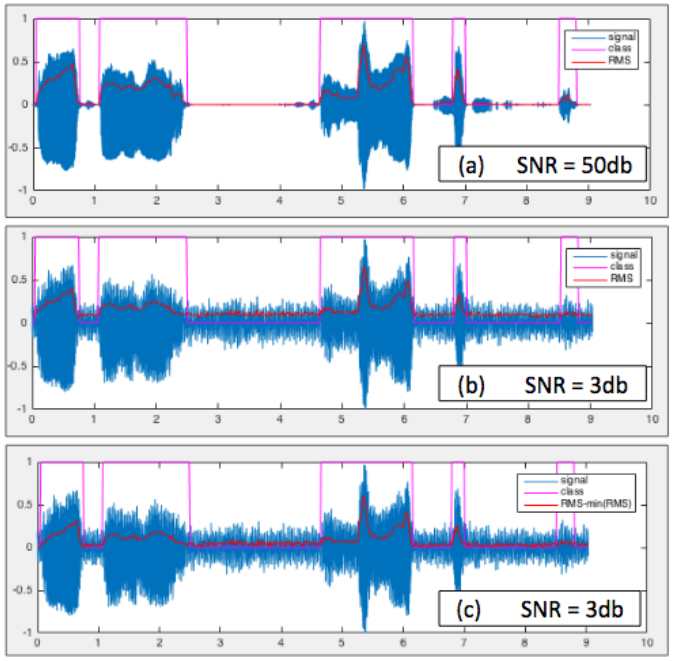

Abbildung 3.6: Differenz-Signal des RMS-Wertes

berechnet umfasst 17 Features, wobei 9 absolute Features und 8 Differenzfeatures verwendet werden.

$$V(x_i) = \Big(\text{RMS}(x_i), ..., \text{ aCount}(x_i) \\ \text{DiffF}(\text{RMS}(x_i)) ....\text{DiffF}(- \text{ aCount}(x_i))\Big) \tag{3.15}$$

3.4.3 Thresholding

Finden der Grenzwerte

Für jedes Signalfenster $x_1 ... x_n$ liegt nun ein Featurevektor $v_1 ... v_n$ vor. Das Ziel ist nun, Grenzwerte für die Features zu finden, bei deren Überschreitung das Signalfenster als *Stimmhaft* kategorisiert wird. Abbildung 3.7 verdeutlicht das Prinzip für das Feature $Ceps_{mag}$. Eine binäre Kategorisierung nach dem Muster $C(x_i) = \{1, \text{wenn } \text{Ceps}_{mag}(x_i) \geq 0.2, 0 \text{ sonst}\}$ würde auf den ersten Blick eine weitgehend richtige Kategorisierung vornehmen.

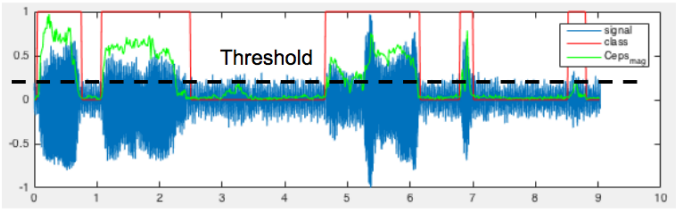

Abbildung 3.7: Thresholding eines Feature-Signales

Es sind Kombinationen von Grenzwerten in Form von Entscheidungsbäumen denkbar,

wenn mehrere Features in die Kategorisierung einfließen. Ein Beispiel wird in Listing 3.1 dargestellt, bei dem die Kategorsierung hierarchisch zuerst nach einem Grenzwert für Ceps_{mag} und danach für den RMS-Wert entschieden wird.

Listing 3.1: Beispiel eines CART-Entscheidungsbaums

```
if Ceps_mag(x_i) > 0.2
|    if RMS(x_i) < 0.13
|    |    C(x_i) = 0
|    |else
|    |    C(x_i) = 1
|else
|     C(x_i) = 1
```

Zur Festlegung der Grenzwerte wird ein *Classification And Regression Tree* (CART)-Algorithmus verwendet. CART-Algorithmen sind Klassifizierungsalgorithmen, bei denen Entscheidungsbäume wie in Listing 3.1 konstruiert werden. Sie haben den Vorteil, dass die Klassifizierung in Form von Regeln mit Grenzwerten dargestellt werden können, die für den Menschen nachvollziehbar sind (im Gegensatz zu z.B. Neuronalen Netzen). Der durch den CART-Algorithmus konstruierte Entscheidungsbaum wird auch als *Klassifizierungs-Modell* bezeichnet.[6]

Einer der bekanntesten CART-Algorithmen ist der ID3-Algorithmus, entwickelt von J. Ross Quinlan an der University of Sidney. Als Trainingsdatensatz S wird eine Menge an Instanzen verwendet, deren Klassenzugehörigkeit bereits bekannt ist. Der ID-3 Algorithmus funktioniert nur für diskrete Features und Klassen. Ein Beispiel für ein diskretes Feauture ist [*Lautstärke = {laut, leise}*]. Ein Beispiel für ein numerisches Feature, welches vom ID-3 Algorithmus nicht akzeptiert wird, ist [*Lautstärke* = $\mathbb{N}$]. Der Algorithmus funktioniert folgendermaßen:

1. Gehören alle Instanzen des Datensatzes S nur einer Klasse an?
   a) Ja: Markiere diesen Knoten als Blatt und STOP.
   b) Nein: Erzeuge einen neuen Knoten. Wähle das Feature F, welches den höchsten *Informationsgewinn* bringt. Das heisst, dass dieses Feature die stärkste Unterscheidung zwischen den einzelnen Klassen ermöglicht. Der Informationsgewinn $H(S)$ eines Features F mit den Untermengen X, welche aus den Instanzen der selben Klasse bestehen, wird mit Hilfe der Entropie nach Formel 3.16 berechnet.

$$H(S) = -\sum_{x \in X} F(x) \log_2 F(x) \tag{3.16}$$

2. Unterteilung den Trainingsdatensatz S in die Untermengen $S_1 ... S_n$, wobei eine Untermenge für jeden möglichen Attributbwert des ausgewählten Features gebildet wird.
3. Wiederhole die Schritte 1. und 2. für alle Untermengen $S_1 ... S_n$ [6]

Der C.45-Algorithmus ist eine Erweiterung des ID-3 Algorithmus, der neben anderen Erweiterungen ebenfalls mit numerischen Attributwerten umgehen kann. Dafür werden für ein Feature alle Instanzen des Datensatzes S nach den Werten dieses Features geordnet und derjenige Grenzwert gesucht, der den höchsten Informationsgewinn bringt. Der Datensatz wird daraufhin an den Knoten in genau 2 Untermengen aufgespalten.[14]

Zum Finden der Grenzwerte der in Kapitel 3.4.2 beschriebenen Features wurde die Implementierung des C.45-Algorithmus *REPTree*[2] der Open Source Data-Mining-Bibliothek *Weka*[3] verwendet. Diese hat den Vorteil, dass die Tiefe des Entscheidungsbaumes festlegbar ist und somit die Komplexität des Baumes begrenzt werden kann. Ein Nebeneffekt ist, dass so Overfitting, also die Überanpassung des Entscheidungsbaumes auf den Trainingsdatensatz, vermieden wird.

Trainings- und Testdatensätze

Zum Training des REPTree-Algorithmus musste in Trainingsdatensatz S erstellt werden. Dazu wird zunächst eine Menge an Audiosignalen benötigt. Es wurden 6 Audiosignale mit Weinen von Babies von der freien Online-Sound-Bibliothek `https://www.freesound.org/` heruntergeladen und zu Segmenten à 10 Sekunden beschnitten. Diese Audiosignale sind weitestgehend Rausch-frei. Die Segmente der Audiosignale wurden händisch Kategorisiert in die Klassen {1 = Stimme, 0 = Still}. Weiterhin wurden 3 verschiedene Rauschsignale heruntergeladen. Es handelt sich um "realistische"Rauschsignale mit Krankenhausatmosphären. Jedes dieser 3 Rauschsignale wurde mit den 6 Weinsignalen überlagert, einmal mit einem Signal-Rausch-Abstand von 50 dB (fast unhörbares Rauschen), und einmal mit einem Signal-Rausch-Abstand von 3 dB (sehr starkes Rauschen). Außerdem wurde ein Testsignal erzeugt, welches eine siebte Tonaufnahme eines Kinderweinens enthält, dass mit einem vierten Rauschsignal mit einem SNR von 7 dB überlagert wurde. Dieses Signal spielt eine Sonderrolle, da es nur zur Verifikation verwendet wird und enthält daher nur ein Audiosignal (Siehe Kapitel 3.4.3) So wurden vier Mengen an Audiosignalen erzeugt:

$\mathbf{A}_{50\,\mathrm{dB}}$ enthält $3 \cdot 6 = 18$ Audiosignale, bei dem alle 6 Wein-Signale mit den 3 Rauschsignalen bei einem Signal-Rausch-Abstand von 50 dB überlagert wurden

$\mathbf{A}_{3\,\mathrm{dB}}$ enthält $3 \cdot 6 = 18$ Audiosignale, bei dem alle 6 Wein-Signale mit den 3 Rauschsignalen bei einem Signal-Rausch-Abstand von 3 dB überlagert wurden

$\mathbf{A_{50+3\,dB}} = \{A_{50\,\mathrm{dB}} \cup A_{3\,\mathrm{dB}}\} = 32$ Audiosignale

$\mathbf{A}_{7\,\mathrm{dB}*}$ enthält 1 Audiosignal, bei dem ein siebstes Wein-Signale mit einem vierten Rauschsignal bei einem Signal-Rausch-Abstand von 7 dB überlagert wurde.

Im nächsten Schritt werden die eigentlichen Trainingsdatensätze $S_{SNR,Feat}$ gebildet, in dem Audiosignale dieser Signalmengen (1) wie in Kapitel 3.3 vorverarbeitet werden, (2) wie in Kapitel 3.4.1 in die Signalfenster à 25 ms zerlegt werden und (3) für jedes Audiosignal die durch Gleichung 3.15 definierte Featurevektoren berechnet werden. Außerdem wird jedem Featurevektor die Klasseninformation *Stimme/Stille* zum Training des REPTree-Algorithmus mitgegeben.

Es ist rechnerisch zu Aufwendig, alle genannten Features in einem kontinuierlichen System zur Voice Activity Detection zu berechnen. Daher werden Untermengen der Features in den Datensätzen gebildet. Das Ziel ist es, diejenige Untermenge an Features zu finden, die sich am besten für die Voice-Activity-Detection sowohl bei niedrigem als auch bei starkem Hintergrundrauschen eignet. Die Untermengen werden in Bezug auf die Methode gebildet, durch die die Features berechnet werden. Das heißt, dass beispielsweise die Untermenge

[2] Dokumentation von REPTree: `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html`

[3] Download von WEKA: `http://www.cs.waikato.ac.nz/ml/weka/`

Zeit die in Kapitel 3.4.2 beschriebenen Features *RMS* und *ZCR* sowie die dazugehören Differenzfeatures *FDiff(RMS)* und *FDiff(ZCR)* beinhaltet.

Die 9 Untermengen sind: { Zeitbereich, Frequenzbereich, Cepstrum, Autokorrelation, Zeit + Frequenzbereich, Zeit + Cepstrum, Zeit + Autokorrelation, Frequenz + Cepstrum, Frequenz + Autokorrelation }.

So enthält beispielsweise der Datensatz $S_{3\,\mathrm{dB},Zeit}$ die Featurevektoren des Zeitbereiches für die Audiosignale mit einem Signal-Rausch-Abstand von 325 dB. Die Audiosignal-Mengen [$\mathrm{A}_{50\,\mathrm{dB}}$], [$\mathrm{A}_{3\,\mathrm{dB}}$] [$\mathrm{A}_{50+3\,\mathrm{dB}}$] und [$\mathrm{A}_{7\,\mathrm{dB}^*}$] wurden in Datensätze umgewandelt. Es werden schlussendlich $4 \cdot 9 = 36$ Datensätze gebildet.

Training und Ergebnis

Der REPTree-Algorithmus entwirft einen Entscheidungsbaum, der für den angegebenen Trainingsdatensatz eine möglichst hohe Accuracy gewährleistet (unter den gegebenen Einschränkungen eines CART-Algoirthmus). Wird dem REPTree-Algorithmus beispielsweise der Datensatz $S_{3\,\mathrm{dB},Zeit}$ (also der Datensatz mit einem SNR von 3 dB unter Verwendung der Zeit-Features, siehe Kapitel 3.4.3) als Input gegeben, entwirft der Algorithmus einen Entscheidungsbaum auf Basis dieses Datensatzes. Wird das gebildete Modell daraufhin für die Klassifikation der $\mathrm{A}_{3\,\mathrm{dB}}$-Signalmenge verwendet, kann man aus dem Klassifikationsergebnis die Accuracy des Modells für diesen Signal-Rausch-Abstand berechnen.

Der Datensatz $S_{SNR,Feat}$, der als Input für den REPTree-Algorithmus und somit zur Bildung des Entscheidungsbaums verwendet wird, wird als *Trainings-Datensatz* bezeichnet. Die Signalmenge A_{SNR}, auf den das Modell angewandt wird und für den die Accuracy berechnet wird, wird als *Test-Datensatz* bezeichnet. Wird also beispielsweise der Trainings-Datensatz $S_{50\,\mathrm{dB},Zeit}$ und als Test-Signalmenge $A_{3\,\mathrm{dB}}$ verwendet, so kann man berechnen, wie gut sich ein Modell unter Verwendung der Zeit-Features zur Klassifizierung niedriger SNRs eignet, obwohl es für hohe SNRs entworfen wurde.

Das Ziel ist, den Entscheidungsbaum für eine Feature-Untermenge zu finden, die eine möglichst hohe Klassifikations-Accuracy für sowohl hohe als auch niedrige SNR gewährleistet. Die Frage ist, ob ein Entschiedungsbaum, der für einen niedrigen SNR gebildet wird, auch für hohe SNR gut funktioniert, oder ob das Gegenteil zutreffend ist. Daher werden die Entscheidungsbäume sowohl auf Basis verschiedener SNRs als auch verschiedener Feature-Untermengen gebildet. Die Modelle werden daraufhin gegen die Signale mit den hohen und niedrigen SNRs getestet.

Es wurden, wie in Kapitel 3.4.3 beschrieben, $3 \cdot 9 = 27$ Trainings-Datensätze erzeugt ([3 SNR-Werten: 3 dB, 50 dB und 50+3 dB] $\times$ [9 Feature-Untermengen]. Der Datensatz mit einem SNR von 7 dB wird *nicht* zum Training verwendet). Mit diesen 27 Trainingsdatensätze werden mit Hilfe des REPTree-Algorithmus 27 Klassifikationsbäume erzeugt. Jeder Klassifikationsbaum wurde gegen die 3 Testdatensätze $\mathrm{A}_{3\,\mathrm{dB}}$, $\mathrm{A}_{50\,\mathrm{dB}}$ und $\mathrm{A}_{7\,\mathrm{dB}^*}$ getestet und die Accuracy berechnet. Das Signal $\mathrm{A}_{7\,\mathrm{dB}^*}$ erfüllt dabei eine Sonderrolle, da weder das Signal des Weinens noch das Rausch-Signal in den Trainingsdatensätzen enthalten sind und somit verifiziert wird, ob der Datensatz nur „auswendig gelernt“ wird oder das Modell auf neue Anwedungsfälle übertragen werden kann. Um Overfitting des Modells zu vermeiden und die Komplexität des Entscheidungsbaumes zu verringern, wurde die maximale Tiefe des REPTree auf 2 gesetzt. Die Ergebnisse sind in Tabelle .1 zu sehen.

Die Features, welche zu den höchsten Accuracy-Werten führten, sind die des *Cepstrum*-Bereiches, genauer gesagt das DiffF(Ceps_{mag})-Feature, da es vom REPTree-Algorithmus als einziges Feature dieses Bereiches für die Entscheidungsbäumen ausgewählt wurde. Die Entscheidungsbäume, die mit dem DiffF(Ceps_{mag})-Feature entworfen wurden, erreichten eine durchschnittliche Accuracy (das heißt, gemittelt über die Testsignale $A_{3\,\text{dB}}$, $A_{50\,\text{dB}}$ und $A_{7\,\text{dB}*}$) von mindestens 91,45%. Der nächstbeste Entscheidungsbaum mit einer Accuracy von 86,96% wurde unter Verwendung der Features des Zeitbereiches und der rechnerisch aufwendigeren Autokorrelation auf dem Datensatz $S_{50+3\,\text{dB,Zeit+Correlation}}$ entworfen. Sobald der Cepstrum-Bereich in Verbindung mit den Features der Bereiche *Zeit* und *Frequenz* verwendet wurde, wurde das DiffF(Ceps_{mag})-Feature vom REPTree-Algorithmus bevorzugt, so dass die Features der anderen beiden Bereiche keine Anwendung mehr in den entsprechenden Bäumen fanden.

Auf Basis der Datensätze $S_{3\,\text{dB,Ceps}}$, $S_{3\,\text{dB,Zeit+Ceps}}$, $S_{3\,\text{dB,Freq+Ceps}}$, $S_{50+3\,\text{dB,Ceps}}$, $S_{50+3\,\text{dB,Zeit+Ceps}}$ sowie
$S_{50+3\,\text{dB,Freq+Ceps}}$ wurde der selbe Entscheidungsbaum erzeugt, der in Listing 3.2 zu sehen ist. Auf Basis der Datensätze $S_{50\,\text{dB,Ceps}}$ und $S_{50\,\text{dB,Zeit+Ceps}}$ wurde der Entscheidungsbaum in Listing 3.3 erzeugt. Es ist zu sehen, dass (a) beide Entscheidungsbaum einen einfachen Grenzwert für das DiffF(Ceps_{mag})-Feature setzen, und zweitens (b) sich die beiden Modelle nur im konkreten Wert des Grenzwertes unterscheiden.

Da das Modell aus Listing 3.2 eine durchschnittliche Accuracy von 92,22% und das Modell aus Listing 3.3 eine unwesentlich geringere Accuracy von 91,45% erreicht, wurden für beide Modelle die Specificity und Sensitivity berechnet, um eine Entscheidung für eines der beiden Modelle fällen zu können. Dazu wurden die Signalmengen $A_{3\,\text{dB}}$, $A_{50\,\text{dB}}$ und $A_{7\,\text{dB}*}$ in Frames à 100 Windows zerlegt und für jedes Zeitfenster die Senstivity, Specificity und Accuracy bezüglich der beiden Modelle berechnet. Die Ergebnisse werden als Boxplots in Abbildung .1 dargestellt. Der Unterschied zwischen den Modellen ist am Stärksten beim Testing gegen die Signale mit einem SNR von 3 dB und 7 dB zu sehen. Das Modell mit dem Grenzwert von 0.03 erzielt in beiden Fällen eine höhere Specificity, aber geringere Senstivitiy als das Modell mit dem Grenzwert bei 0.02. Es wurde sich für das Modell für mit einem Grenzwert von 0.02 entschieden, da durch die höhere Senstivity mehr Wein-Signale erkannt werden, die in späteren Verarbeitungsschritten immernoch als False-Positives erkannt und verworfen werden können. Einmal im Prozess der VAD als Stimmlos markierte Fenster werden jedoch nicht weiter verarbeitet und gehen somit „verloren“.

Listing 3.2: Entscheidungsbaum für die VAD mit einem Cepstrum-Grenzwert von 0.02

```
if FDiff(Ceps_mag(x_i)) < 0.02
|    C(x_i) = 0
|else
|    C(x_i) = 1
```

Listing 3.3: Entscheidungsbaum für die VAD mit einem Cepstrum-Grenzwert von 0.03

```
if FDiff(Ceps_mag(x_i)) < 0.03
|    C(x_i) = 0
|else
|    C(x_i) = 1
```

Der Finale Funktion zur Klassifikation eines Signalfensters C(x) in 0 = *Stille* oder 1 = *Stimme* ist somit durch Gleichung 3.17 gegeben.

$$C(x) = \begin{cases} 1, & \text{if Ceps}_{mag}(x) >= 0.02 \\ 0, & \text{otherwise} \end{cases} \tag{3.17}$$

3.4.4 Markierung der Cry-Units

Das Ergebnis der Voice-Activity-Detection ist eine Zurdnung aller Signalfenster $x_1 ... x_n$ zu den Klassen $C(x_i) = 0$ *Stille* oder $C(x_i) = 1$ *Stimme*. In [19] wird die Idee vorgestellt, zusammenhängende und ununterbrochene Ketten als *stimmhaft* klassifizierter Signalfesnter zu *Stimm-Segmenten* zusammenzufassen, welche in diesem Zusammenhang eine *Cry-Units* entsprechen. Abbildung 3.8 veranschaulicht diese Gruppierung. Formel 3.18 gibt die Definition des Datentypes *Cry-Unit* [CU]. Eine Cry-Unit definiert sich durch einen Anfangszeitpunkt *start*, einen Endzeitpunkt *end* und der Liste seiner Signalfenster $windows = [x_1 ... x_n]$.

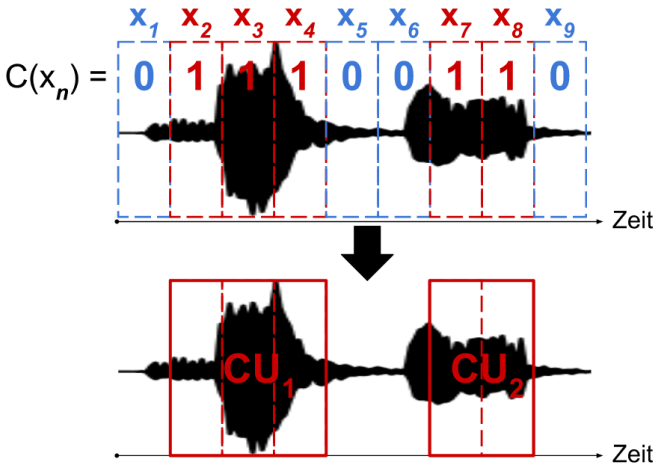

Abbildung 3.8: Zusammenfassung klassifizierter Signalfenster zu Cry-Units

Algorithmus 1 zeigt in Pseudo-Code, wie in der Liste aller Signalfenster $X_{windows} = [x_1 ... x_n]$ eine Liste von Cry-Units $CU_{all} = [cu_1 ... cu_m]$ markiert wird. Die Funktion C(x) ist die Klassifikations-Funktion der Signalfenster in Stille/Stimme nach Gleichung 3.17. Die Funktion getTimeOf(x) liefert die Anfangszeitpunkt des Signalfensters x.

$$CU = (windows = [x_1 ... x_n], start \in Zeit, end \in Zeit) \tag{3.18}$$

$$\lambda(CU) = CU.end - CU.start \tag{3.19}$$

$$\text{d}(CU_i, CU_j) = CU_j.start - CU_i.end \tag{3.20}$$

Die Dauer eine Cry-Unit $\lambda(CU)$ wird nach Formel 3.19 berechnet. Der (Stille)-Zeitraum zwischen zwei Cry-Units d(CU_i, CU_j), wird nach Formel 3.20 berechnet. Diese Zusammenhänge werden in Abbildung 3.9 visualisiert.[19]

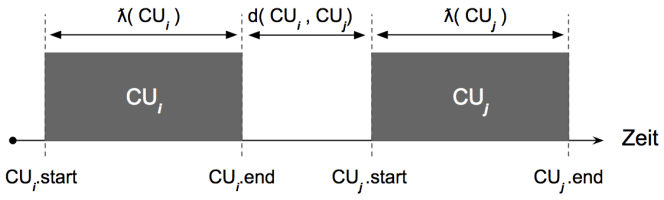

Abbildung 3.9: Beziehung zwischen agrenzenden Segmenten

Algorithm 1 Gruppierung von x-Windows zu Cry-Units

```
1:  function turnWindowsIntoCryUnits(X_windows)
2:      CU_all ← []
3:      cu_i ← ([], 0, 0)
4:      for i = 1...length(X_windows) do
5:          c_i ← C(x_i)
6:                                                  ▷ Start of Cry-Unit
7:          if c_i == 1 ∧ isEmpty(cu_i.windows) then
8:              cu_i.start ← getTimeOf(x_i)
9:              cu_i.windows ← [cu_i.windows, x_i]
10:         end if
11:                                                 ▷ Inside Cry-Unit
12:         if c_i == 1 ∧ ! isEmpty(cu_i.windows) then
13:             cu_i.windows ← [cu_i.windows, x_i]
14:         end if
15:                                                 ▷ End of Cry-Unit
16:         if c_i == 0 ∧ ! isEmpty(cu_i.windows) then
17:             cu_i.end ← getTimeOf(x_i)
18:             CU ← [CU, cu_i]
19:             cu_i.windows ← []
20:         end if
21:     end for
22:                                 ▷ End last Cry-Unit by force if still open.
23:     if ! isEmpty(cu_i.windows) == 0 then
24:         cu_i.end ← getTimeOf(X_windows[end])
25:         CU_all ← [CU_all, cu_i]
26:     end if
        return CU_all
27: end function
```

3.4.5 Decision Smoothing

Abbildung 3.10 zeigt ein Audiosignal mit einem Signal-Rausch-Abstand von 3 dB, bei dem die Klassifizierung nach dem Entscheidungsbaum aus Listing 3.2 durchgeführt wurde. Die rote Linie zeigt die tatsächliche Klassifizierung, und die grüne Linie die gefundene Klassifizierung nach dem vorgestellten Algorithmus. Die tatsächlichen/gefunden Cry-Units sind klar zu erkennen als die Bereiche, die von der roten/grünen Linie überspannt werden. Es ist zu sehen, dass False-Negatives und False-Positives in der Klassifizierung enthalten

sind. Im folgenden werden drei charakteristische Arten falscher Klassifizierungen näher erläutert:

False Negatives nach (a) : Eine korrekt erkannte, längere Cry-Unit wird zu früh beendet. Oft werden kurz nach dem Ende sehr kurze Cry-Units erkannt, die eigentlich noch zu der längeren, vorhergehenden Cry-Unit gehören.

False Positives nach (b): Kurze Cry-Units werden in eigentlichen Stille-Bereichen erkannt.

False Negatives nach (c): Eine Cry-Unit zerfällt in zwei kürzere Cry-Units, da einige Signalfenster in der Mitte als Stille erkannt wurden.

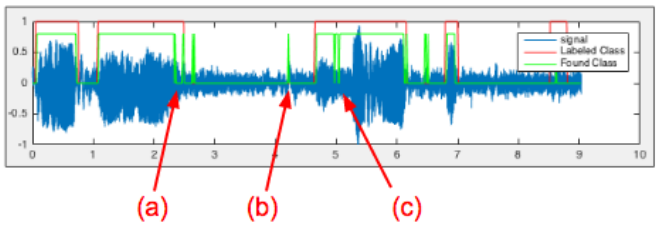

Abbildung 3.10: Klassifizierung vor dem Decision Smoothing

Im Process des **Decision Smoothing** werden kontextuelle Informationen genutzt, um nachträglich False-Positives und False-Negatives zu entfernen. Es werden dazu die in [19] präsentierten Ideen verwendet. Es werden zwei Parameter eingeführt: λ_{min}, die Mindestlänge einer akzeptierten Cry-Unit, und d_{min}, die Mindestlänge eines akzeptierten Stille-Segmentes. Das Decision Smoothing wird nach den folgenden Entscheidungsregeln durchgeführt:

- ist $\lambda(CU_i) \leq \lambda_{min}$?
  - wenn $\lambda(CU_{i-1}) > \lambda_{min}$ und $d(CU_{i-1}, CU_i) \leq d_{min}$, dann vereinige CU_i mit CU_{i-1} . $\Rightarrow$ behebt False-Negatives des Types (a)
  - ansonsten entferne $CU_i \Rightarrow$ behebt False-Negatives des Types (b)
- wenn $\lambda(CU_i > \lambda_{min}$ und $d(CU_{i-1}, CU_i) \leq d_{min}$, dann vereinige CU_i mit CU_{i-1} . $\Rightarrow$ behebt False-Negatives des Types (c)

Die Entscheidungsregeln greifen Algorithmus greifen nur auf die aktuellen und die letzten bekannte Cry-Unit um, um eine kontinuierliche Analyse zu gewährleisten, weshalb die Entscheidungsregeln jedoch auch komplex sind. Bei einer offline-Analyse können die Entscheidungsregeln vereinfacht werden, da False-Negative Type (a) und (c) mit der selben Regeln abgefragt werden können. Algorithmus 2 zeigt in Pseudo-Code, wie das Decision-Smoothing durchgeführt wird. Input der Funktion ist die Liste aller Cry-Units CU_{all}, die durch Algorithmus 1 entstanden ist, sowie die Grenzwerte λ_{min}, d_{min}. Ausgang der Funktion ist die Liste aller Cry-Units nach dem Decision-Smoothing $CU_{smoothed}$.

Algorithm 2 Decision-Smoothing of VAD

```
1:  function DECISIONSMOOTHING(CU_all, λ_min, d_min)
2:      CU_smoothed ← [CU_all[1]]
3:      for i = 2...length(CU_all) do
4:          cu_i ← CU_all[i]
5:          cu_{i−1} ← CU_smoothed[end]
6:          if λ(cu_i) > λ_min then
7:              if d(cu_{i−1}, cu_i) > d_min then
8:                  CU_smoothed ← [CU_smoothed, cu_i]
9:              else
10:                                              ▷ Erase False-Negative Type (c)
11:                 cu_i ← vereinige(cu_i, cu_{i−1})
12:                 CU_smoothed ← [CU_smoothed[1 : end − 1], cu_i]
13:             end if
14:         else
15:                                              ▷ Erase False-Negative Type (a)
16:             if λ(cu_i) > λ_min ∧ d(cu_{i−1}, cu_i) ≤ d_min then
17:                 cu_i ← vereinige(cu_i, cu_{i−1})
18:                 CU_smoothed ← [CU_smoothed[1 : end − 1], cu_i]
19:             else
20:                                  ▷ Don't accept cu_i. Erases False-Positives (b)
21:             end if
22:         end if
23:     end for
        return CU_smoothed
24: end function
```

Abbildung 3.11 zeigt das Signal vor und nach dem Decision-Smoothing. Die Parameter wurden experimentell mit $\lambda_{min} = 50ms$ und $d_{min} = 50ms$ bestimmt.

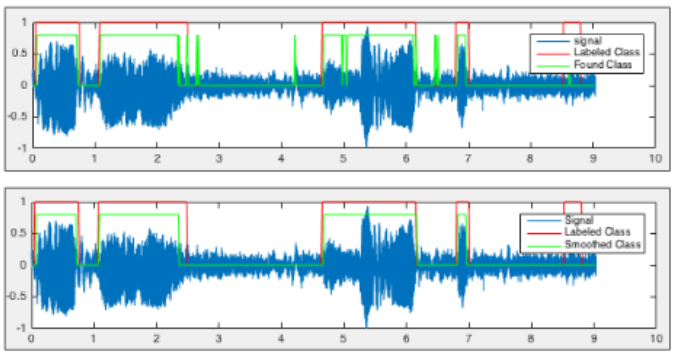

Abbildung 3.11: Klassifizierung vor und nach dem Decision Smoothing

3.5 Segmentierung

Das Ergebnis der Voice-Activiy-Detection ist eine Liste an Cry-Units $CU_1...CU_n$. Das Ziel ist nun, diese Cry-Units zu Cry-Segmenten zu gruppieren. Ein Cry-Segment definiert sich

nach Golub et al [13] als „die komplette klangliche Antwort auf einen spezifischen Stimulus. Sie kann mehrere Cry-Units entahlten “. Die Defintion lässt folgende Fragen offen:

- Beginnt das Segment bereits bei Zuführung des Stimulus, oder erst ab der ersten Cry-Unit?
- Wodurch definiert sich der Beginn, wenn ohne Zuführung eines Stimulus das Baby beginnt, zu weinen?
- Endet ein Cry-Segment mit Ende der letzten „Cry-Unit“, oder erstreckt es sich bis zu Beginn des nächsten Cry-Segmentes?

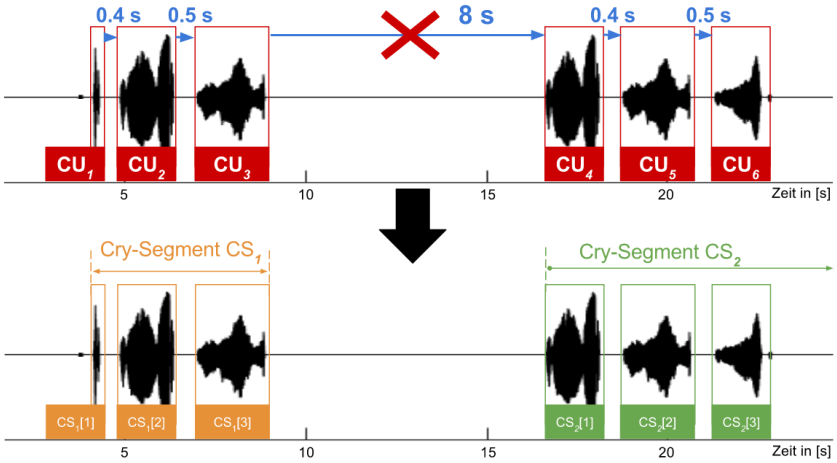

Abbildung 3.12: Ergebnis der Segmentierung

Die Zusammenfassung von Cry-Units zu Cry-Segmentes unterliegt einer gewissen subjektiven Einschätzung, welche Cry-Units als Zuammengehörig angesehen werden, insbesondere, wenn kein erkennbarer Stimulus vorliegt. Abbildung 3.13 verdeutlicht das Problem.

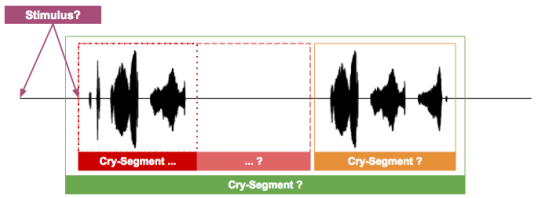

Abbildung 3.13: Mögliche Segmentierungen eines Signals

Um das Problem zu objektivieren, wird es mathematisch formuliert. Eine *Cry-Segment* $[CS]$ wird als Datentyp nach Formel 3.21 definiert. Ein Cry-Segment ist folglich eine Liste aufeinander folgenden Cry-Units, die gruppiert werden. Der Start-Zeitpunkt eines Cry-Segmentes wird nach Formel 3.22 als der Startzeitpunkt der ersten Cry-Unit des Segmentes definiert. Die Begründung für diese Entscheidung liegt darin, dass rein aus dem Audiomaterial der Zeitpunkt des Stimulus nicht festgestellt werden kann und das Segment mit Sicherheit erst bei der ersten feststellbaren Cry-Unit beginnt. Das Ende eines Segmentes wird definiert als das Ende der letzten Cry-Unit nach Gleichung 3.23. Die Begründung liegt darin, dass das Ende der Reaktion auf den Stimulus ebenfalls rein aus dem Audiosignal

abgeleitet werden kann und somit der einzig feststellbare Indikator die letzte Cry-Unit des Segmentes ist.

$$CS = [cu_1, ..., CU_n] \tag{3.21}$$

$$start(CS) = CS[1].start \tag{3.22}$$

$$end(CS) = CS[end].end \tag{3.23}$$

Wurde bei der kontinuierlichen Analyse des Signals ein Segment geschlossen, führt die Markierung einer neuen Cry-Unit zur Eröffnung eines neuen Segmentes, dessen Start-Zeitpunkt der Start-Punkt dieser Cry-Unit ist. Die Frage ist, welches Kriterium zum schließen dieses Segments führt. Laut Golub et al [13] ein Cry-Segment „die komplette klangliche Antwort auf einen spezifischen Stimulus “. Eine mögliche und objektiv messbare Interepration dieses Endes ist, dass nach dem Auftreten von Cry-Units eine längere Stille mit einer Abwesenheit von Cry-Units festgestellt wird, da das Baby „aufgehört hat, zu weinen “. Übertragen auf die in 3.4.4 vorgestellte Terminologie heißt das, dass ein Segment beendet und ein neues begonnen wird, wenn die Distanz (Zeitraum der Stille) zwischen zwei benachbarten Cry-Units $\mathrm{d}(CU_i, CU_{i+1})$ einen gewissen Grenzwert $t_{seg-max}$ überschreitet. Gleichung 3.24 formalisiert diesen Zusammenhang. Daraus lässt sich schlussfolgern, dass die Distanzen zwischen allen benachbarten Cry-Unit eines Segmentes unter diesem Grenzwert $t_{seg-max}$ liegen. Gleichung 3.25 formalisiert diese Nebenbedingung an die Cry-Units eines Segmentes.

$$d(cu_i, cu_{i+1}) > t_{seg-max} \rightarrow CS_n = [CS_n, cu_i] \wedge CS_{n+1} = [cu_{i+1}] \tag{3.24}$$

$$\forall i = 1...\mathrm{length}(CS) - 1 : \mathrm{d}(CS[i], CS[i+1]) \leq t_{seg-max} \tag{3.25}$$

Die einfachste Art, $t_{seg-max}$ festzulegen, ist, einen festen Grenzwert von t s zuzuweisen. Abbildung 3.12 visualisiert die so resultierende Segmentierung an einem Beispiel. Jeder Grenzwert mit $t_{seg-max} > 0.5\,\mathrm{s}$ würde zu der gezeigten Segmentierung führen.

Algorithmus 3 zeigt die Segmentierung nach diesem Prinzip in Pseudo-Code. Input des Algorithmus ist die Liste aller Cry-Units $CU_{all} = [cu_1...cu_n]$, die nach dem Decision-Smoothing nach Algorithmus 2 entstanden ist. Das Ergebnis des Algorithmus ist die Liste, die alle gefundene Cry-Segmente $[cs_1...cs_n]$ enthält.

Algorithm 3 Gruppierung von Cry-Units zu Cry-Segments

```
1: function segmentCryUnits(CU_all, t_seg−max)
2:     CS_all ← []
3:     cs_i ← [CU_all[1]]
4:     for i = 2...length(CU_all) do
5:         cu_i ← CU_all[i]
6:         cu_{i−1} ← CU_all[i − 1]
7:         if d(cu_{i−1}, cu_i) < t_seg−max then
8:             cs_i ← [cs_i, cu_i]
9:         else
10:            CS_all ← [CS_all, cs_i]
11:            cs_i ← [cu_i]
12:        end if
13:    end forreturn CS_all
14: end function
```

Algorithmus 3 kann zwar kontinuierlich durchgeführt werden, da er jeweils nur auf die aktuelle gefundene und eine vergangene Cry-Unit zurückgreift, hat in dieser Form jedoch den nachteil, dass das Ende eines Segmentes später als notwendig festgestellt wird. Angenommen, ein Grenzwert von $t_{seg-max} = 20\,\mathrm{s}$ wurde festgelegt

Bei einer kontinuierlichen durchgeführten Segmentierung wird das erste Segment dann eröffnet, sobald die erste Cry-Unit durch die VAD markiert wurde, und diese Cry-Unit dem Segment hinzugefügt. Die Dauer der Stille nach dieser Cry-Unit wird kontinuierlich gemessen. Wird ein nächste Cry-Unit festgestellt, bevor die Stille $d_{seg-max}$ übersteigt, so wird diese Cry-Unit dem Segment hinzugefügt und das Messen der Stille nach dieser Cry-Unit beginnt von vorne. Dieser Prozess wird so lange wiederholt, bis die Dauer der Stille nach einer hinzugefügten Cry-Unit $d_{seg-max}$ übersteigt. Dann wird das Segment beendet und der Endzeitpunt des Segmentes auf den Endzeitpunkt der letzten Cry-Unit gesetzt. Abbildung 3.12 zeigt die resultiderende Segmentierung für Beispielsignal mit $d_{seg-max} = 3\,\mathrm{s}$. Tatsächlich würde in dem Beispiel jeder Grenzwert $d_{seg-max} > 0.5\,\mathrm{s}$ zur gezeigten Segmentierung führen.

Es gibt verschiedene Möglichkeiten, die höchst mögliche Pause $\mathrm{d}_{seg-max}$ zu definieren. Der Einfachste Fall, der auch in Abbildung 3.12 angenommen wurde, ist das Setzen eines global festlegten Grenzwertes. Weitere Möglichkeiten sind, $\mathrm{d}_{seg-max}(CS)$ als Funktion des Segmentes selber zu gestalten. So könnte beispielsweise ein längeres Segment eine höhere maximal-Pause erzeugen.

Schlussendlich konnten in der Fachliteratur keine konkreten Hinweise zur Bestimmung von $\mathrm{d}_{seg-max}$ gefunden werden. Daher wurde entschieden, dem Arzt, der das System benutzt, selber einstellen zu lassen. Es wird mit einem Festen

3.6 Feature-Extraction

3.7 Ableitung der Schmerz-Scores

3.8 Visualisierung

4 Zusammenfassung

Literaturverzeichnis

[1] K J S Anand. *Pain in Neonates and Infants*. Elsevier, 2007.

[2] Zachariah Boukydis Barry Lester. *Infant Crying: Theoretical and Research Perspectives*. Springer, 1985.

[3] Judy Bildner. *CRIES Instrument Assessment Tool of Pain in Neonates*. City of Hope Pain, 1997. Online unter `http://prc.coh.org/pdf/CRIES.pdf`.

[4] R Sisto & Giuseppe Buonocore Carlo Bellieni, Franco Bagnoli. Cry features reflect pain intensity in term newborns: An alarm threshold. *Pediatric Research*, 5:142–146, 1. Online unter `https://www.researchgate.net/publication/297827342_Cry_features_reflect_pain_intensity_in_term_newborns_An_alarm_threshold`.

[5] Rami Cohen and Yizhar Lavner. Infant Cry Analysis and Detection. In *27th Convention of Electrical and Electronics Engineers in Israel*. IEEE, 2012. Online unter `https://www.researchgate.net/publication/261116332_Infant_cry_analysis_and_detection`.

[6] Douglas Dankel. The ID3 Algorithm , 1997. Online unter `http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-pa/2.htm`.

[7] H. Hollien & T Murry E Müller. Perceptual responses to infant crying: identification of cry types. *Journal of Child Language*, 1(1):89–95, 1974. Online unter `https://www.cambridge.org/core/journals/journal-of-child-language/article/perceptual-responses-to-infant-crying-identification-of-cry-types/4F0F8088116FCE381851D8D560697A5F`.

[8] B. Simak E. Verteletskaya. Performance Evaluation of Pitch Detection Algorithms, 2009. Online unter `http://access.feld.cvut.cz/view.php?cisloclanku=2009060001`.

[9] Jan Hamers & Peter Gessler Eva Cignac, Romano Mueller. Pain assessment in the neonate using the Bernese Pain Scale for Neonates. *Early Human Development*, 78(2):125–131, 2004. Online unter `http://www.sciencedirect.com/science/article/pii/S0378378204000337`.

[10] Dmitry Goldgof Rangachar Kasturi Terri Ashmeade Ghada Zamzmi, Chih-Yun Pai and Yu Sun. An Approach for Automated Multimodal Analysis of Infants' Pain. In *23rd International Conference on Pattern Recognition*, Cancun, Mexico, 2016.

[11] Health Facts For You. *Using Pediatric Pain Scales Neonatal Infant Pain Scale (NIPS)*, 2014. Online unter `https://www.uwhealth.org/healthfacts/parenting/7711.pdf`.

[12] Hodgkinson. Neonatal Pain Assessment Tool , 2012. Online unter `http://www.rch.org.au/uploadedFiles/Main/Content/rchcpg/hospital_clinical_guideline_index/PAT%20score%20update.pdf`.

[13] Michael J Corwin Howard L Golub. A Physioacoustic Model of the Infant Cry. In *Infant Crying - Theoretical and Research Perspectives*, chapter 3, pages 59 – 82. Plenung,

1985.

[14] Giorgio Ingargiola. Building Classification Models: ID3 and C4.5. Online unter http://cis-linux1.temple.edu/~giorgio/cis587/readings/id3-c45.html.

[15] Donna Geiss Laura Wozniak & Charles Hall Ivan Hand, Lawrence Noble. COVERS Neonatal Pain Scale: Development and Validation. *International Journal of Pediatrics*, 2010, 2010. Online unter https://www.hindawi.com/journals/ijpedi/2010/496719/.

[16] J Gorriz & J Segura J Ramorez. Voice Activity Detection. Fundamentals and Speech Recognition System Robustness. *Robust Speech Recongnition and Understanding*, page 460, 2007. Online unter http://cdn.intechopen.com/pdfs/104/InTech-Voice_activity_detection_fundamentals_and_speech_recognition_system_robustness.pdf.

[17] Jeih-weih Hung & Lin-shan Lee Jia-lin Shen. Robust Entropy-based Endpoint Detection for Speech Recognition in Noisy Environments. 1998. Online unter https://www.researchgate.net/publication/221489354_Robust_entropy-based_endpoint_detection_for_speech_recognition_in_noisy_environments.

[18] Carol Espy-Wilson & Tarun Pruthi Jonathan Kola. Voice Activity Detection. *MERIT BIEN*, 2011. Online unter http://www.ece.umd.edu/merit/archives/merit2011/merit_fair11_reports/report_Kola.pdf.

[19] Kim Weaver & Fathi M. Salam Khurram Waheed. A robust Algorithm for detecting speech segments using an entropic contrast. *IEEE*, 2003. Online unter http://ieeexplore.ieee.org/document/1187039/.

[20] M M Homayounpour M H Moattar. A simple but efficient real-time Voice Activity Detection Algorithm. Signal Processing Conference, IEEE, August 2009. Online unter http://ieeexplore.ieee.org/document/7077834/?arnumber=7077834&tag=1.

[21] Hans M Koot Dick Tibboel Jan Passchier & Hugo Duivenvoorden Monique van Dijk, Josien de Boer. The reliability and validity of the COMFORT scale as a postoperative pain instrument in 0 to 3-year-old infants. *Pain*, 84(2):367—-377, 2000. Online unter http://www.sciencedirect.com/science/article/pii/S0304395999002390.

[22] Taddio Nulman. A revised measure of acute pain in infants. *J Pain Symptom Manage*, 10:456–463, 1995. Online unter http://geriatricphysio.yolasite.com/resources/Modified%20Behavioral%20Pain%20Scale%20(MBPS)%20in%20infants.pdf.

[23] J L Mathew P J Mathew. Assessment and management of pain in infants. *Postgrad Med J*, 79:438–443, 2003. Online unter http://pmj.bmj.com/content/79/934/438.full.

[24] Steven Creech & Marc Weiss. Patricia Hummel, Mary Puchalski. N-PASS: Neonatal Pain, Agitation and Sedation Scale – Reliability and Validity. *Pediatrics/Neonatology*, 2(6), 2004. Online unter http://www.anestesiarianimazione.com/2004/06c.asp.

[25] Susan Parker-Price & Ronald Barr Philip Zeskind. Rythmic organization of the Sound of Infant Cry. *Dev Psychobiol*, 26(6):321–333, 1993. Online unter https://www.ncbi.nlm.nih.gov/pubmed/8119482.

[26] Ananth N. Iyer Pritam Pal and Robert E. Yantorno. Emotion detection from infant facial experessions and cries. In *Acoustics, Speech and Signal Processing*. IEEE, 2006.

[27] R Ward & C Laszlo Qiaobing Xie. Automatic Assessment of Infants' Levels-of-Distress from the Cry Signals. *IEEE Transanctions on Speech and Audio Processing*, 4(4):253–

265, 1996. Online unter http://ieeexplore.ieee.org/document/506929/.

[28] Brian Hopkins & James Green Ronald Barr. *Crying as a Sign, a Symptom, and a Signal*. Mac Keith Press, 2000.

[29] J R Shayevitz & Shobha Malviya Sandra Merkel, Terri Voepel-Lewis. The FLACC: A Behavioral Scale for Scoring Postoperative Pain in Young Children. *Pediatric Nursing*, 23(3):293–7, 1996. Online unter https://www.researchgate.net/publication/13998379_The_FLACC_A_Behavioral_Scale_for_Scoring_Postoperative_Pain_in_Young_Children.

[30] Andreas Spanias Sassan Ahmadi. Cepstrum-Based Pitch Detection Using a New Statistical V/UV Classification Algorithm. *IEEE Transactions on Speech and Audio Detection*, 7(3):333–338, 1999. Online unter http://ieeexplore.ieee.org/document/759042/.

[31] Henning Reetz & Carla Wegener Tanja Fuhr. Comparison of Supervised-learning Models for Infant Cry Classification. *InternatIonAl Journal of HealtH Professions*, 2015. Online unter https://www.degruyter.com/view/j/ijhp.2015.2.issue-1/ijhp-2015-0005/ijhp-2015-0005.xml.

[32] Sabine Deligne & Peder Olsen Trausti Kristjansson. Voicing Features for Robust Speech Detection. In *Interspeech Lisboa*, September 2005. Online unter http://papers.traustikristjansson.info/wp-content/uploads/2011/07/KristjanssonRobustVoicingEurospeech2005.pdf.

[33] Gyorgy Ivan Varallyay. *Analysis of the Infant Cry with Objective Methods*. PhD thesis, Budapest University of Technology and Economics, 2009. Online erhältlich unter: https://pdfs.semanticscholar.org/5c38/b368dc71d67cbfabc3077a50536b086d8eec.pdf.

[34] P H Wolff. The role of biological rhythms in early psychological development. *Bulletin of the Menninger Clinic*, 31:197–218, 1967.

[35] Syed Ahmad Yousra Abdulaziz, Sharrifah Mumtazah. Infant Cry Recognition System: A Comparison of System Performance based on Mel Frequency and Linear Prediction Coefficients. In *Information Retrieval & Knowledge Management*, 2010. Online unter http://ieeexplore.ieee.org/document/5466907/.

Appendices

Tabelle .1: Accuracy-Werte der Grenzwertfindung mit REPTree

$S_{Training}$	3 dB				50 dB				50+3 dB			
A_{Test}	3 dB	50 dB	7 dB*	Mean	3 dB	50 dB	7 dB*	Mean	3 dB	50 dB	7 dB*	Mean
Zeit	77.81%	79.02%	86.04%	80,96%	49.33%	94.70%	48.66%	64,23%	77.54%	92.47%	84.38%	84,80%
Freq	82.05%	89.28%	82.71%	84,68%	70.52%	94.37%	55.06%	73,31%	81.75%	91.22%	74.90%	82,62%
Ceps	88.98%	94.72%	92.96%	**92,22%**	86.83%	94.68%	92.83%	**91,45%**	88.98%	94.72%	92.96%	**92,22%**
Corr	80.45%	73.47%	84.89%	79,60%	73.07%	87.14%	77.98%	79,39%	77.90%	84.88%	82.84%	81,87%
Zeit+Freq	82.05%	89.28%	82.71%	84,68%	70.52%	94.37%	55.06%	73,31%	81.75%	91.22%	74.90%	82,62%
Zeit+Ceps	88.98%	94.72%	92.96%	**92,22%**	86.83%	94.68%	92.83%	**91,45%**	88.98%	94.72%	92.96%	**92,22%**
Zeit+Corr	80.45%	73.47%	84.89%	79,60%	49.33%	94.70%	48.66%	64,23%	80.32%	92.35%	88.22%	86,96%
Freq+Ceps	88.98%	94.72%	92.96%	**92,22%**	70.65%	94.75%	55.06%	73,49%	88.98%	94.72%	92.96%	**92,22%**
Freq+Corr	82.05%	89.28%	82.71%	84,68%	70.52%	95.60%	95.60%	87,24%	81.75%	94.42%	74.90%	83,69%

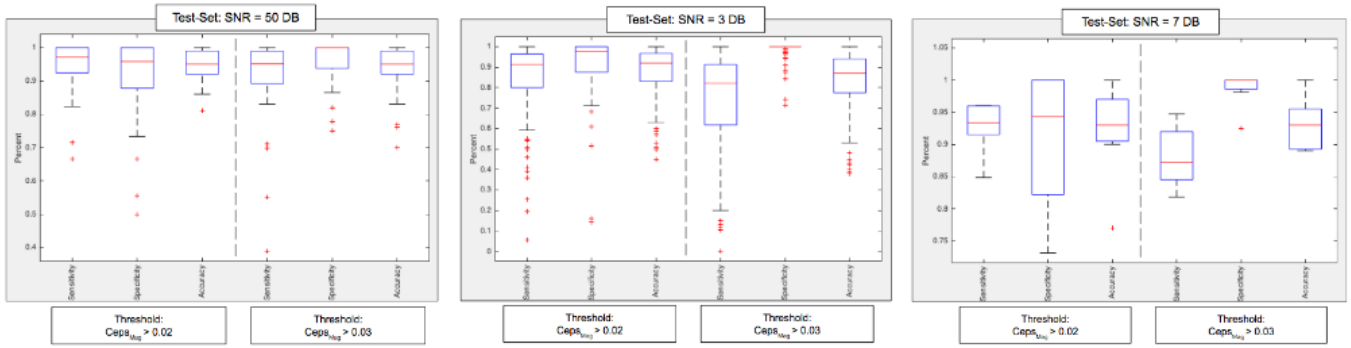

Abbildung .1: Boxplot-Auswertung über Sensitivity, Specificity und Accuracy der beiden VAD-Modelle