Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Bachelorarbeit
FS16 Studiengang Informatik

Instrumentenerkennung in Single-Source Audio Streams

Autoren	Yannick Streit Leotrim Zulfiu
Hauptbetreuung	Thilo Stadelmann
Nebenbetreuung	Philipp Ackermann
Industriepartner	Studer Professional Audio GmbH
Datum	10. Juni 2016

Abstract

In recent years the field of *deep learning* has experienced groundbreaking results in computer vision and related fields, mostly due to advances in the computational power of graphics processing units (GPU).

First papers have been published in the field of music information retrieval that use deep learning methods for the task of musical instrument classification. However, traditional two-step architectures, which focus on manual feature extraction and subsequent classification, are still considered state of the art. Feature learning algorithms, especially in the form of convolutional neural networks (CNN), combine feature extraction and classification in an end-to-end system, and are capable of learning features relevant for classification directly from the input data.

In this bachelor thesis, a system for the automatic identification of musical instruments in monophonic audio signals is implemented using a CNN with spectrograms as input. The system is to be used for the automatic configuration of audio mixing consoles, assisting the sound engineer during soundchecks of music groups. A dataset composed of multiple sources, consisting of the classes acoustic and electric guitar, electric bass, drums, and male as well as female vocals is used for training and testing.

With the proposed architecture, the implemented system is able to classify instruments with an accuracy of over 90%. A direct comparison with the state of the art is difficult due to the different datasets used. Related work achieves similar or lower accuracy in the classification of orchestral instruments. Yet, our experiments show that CNNs with spectrograms as input can achieve very good results while still offering plenty of room for improvement.

Zusammenfassung

In den letzten Jahren wurden im Forschungsgebiet *Deep Learning*, aufgrund der heutigen Rechenmöglichkeiten, getrieben durch Fortschritte in der Grafikprozessorhardware (GPU), bahnbrechende Ergebnisse in der Bilderkennung und verwandten Bereichen erzielt.

Im Bereich Music Information Retrieval wurden erste Arbeiten veröffentlicht, welche Deep-Learning-Ansätze für die Klassifikation von Musikinstrumenten einsetzen. Klassische Zweischrittverfahren, welche durch die manuelle Wahl von geeigneten Merkmalen und anschliessender Klassifikation geprägt sind, gehören heute jedoch nach wie vor zum State-of-the-Art. So genannte *Feature-Learning-Verfahren*, insbesondere in Form von Convolutional Neural Networks (CNN), vereinen Merkmalsextraktion und Klassifikation in einem geschlossenen System, welches relevante Merkmale direkt aus den Eingabedaten lernt und für die Klassifikation verwendet.

In dieser Bachelorarbeit wird ein System zur automatischen Erkennung von Musikinstrumenten in monophonen Audiosignalen basierend auf einem CNN und Spektrogrammen als Eingabedaten entwickelt. Dieses soll später zur automatischen Konfiguration von Audio-Mischpulten eingesetzt werden, um Toningenieure beim Soundcheck von Musikgruppen zu unterstützen. Für das Training des Systems wurde ein eigener Datensatz aus verschiedenen Quellen erstellt. Dieser besteht aus den sechs Klassen akustische und elektrische Gitarre, elektrischer Bass, Schlagzeug sowie männliche und weibliche Stimme.

Mit der gewählten Architektur ist das entwickelte System in der Lage, die Instrumentenklassen mit einer Genauigkeit von über 90% zu unterscheiden. Ein direkter Vergleich mit dem State-of-the-Art ist schwierig, da unterschiedliche Daten verwendet werden. Verwandte Arbeiten erreichen eine vergleichbare oder niedrigere Genauigkeit in der Klassifikation von Orchesterinstrumenten. Die Untersuchungen zeigen, dass CNNs im Zusammenhang mit Spektrogrammen sehr gute Ergebnisse erzielen können und gleichzeitig viel Optimierungspotenzial bieten.

Vorwort

Wir bedanken uns bei unseren Betreuern Thilo Stadelmann und Philipp Ackermann, welche uns während der Arbeit stetig unterstützten und wertvolle Ratschläge gaben. Der Dank gilt ebenso den hilfsbereiten Mitarbeitenden des Instituts, welche uns, wann immer wir Hilfe brauchten, zur Seite standen. Wir danken auch Peter Glättli und Roman Riedi von der Studer Professional Audio GmbH für die sehr interessante und informative Firmenbesichtigung und für die Unterstützung und Motivation.

Wir beide sind sehr glücklich Theorie und Praxis in einem solch interessanten Gebiet der Informatik erforscht zu haben und werden weiterhin Neuigkeiten und Fortschritte in Machine Learning verfolgen.

Zu guter Letzt danken wir unseren Familien und Freunden, welche uns immer zur Seite standen und motivierten.

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Winterthur, 8. Juni 2016

Unterschriften:

.....

.....

.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Abstract bzw. dem Management Summary mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Inhaltsverzeichnis

1. Einleitung

1.1. Ausgangslage und Motivation

Die Firma Studer Professional Audio GmbH ist ein führender Hersteller professioneller Audiomischpulte für den Broadcasting-Bereich. Ein digitales Mischpult hat diverse Audio-Ein- und -Ausgänge (Kanäle), an denen jeweils einzelne Audioquellen angeschlossen sind – etwa ein Mikrofon oder ein einzelnes Instrument. Zur Unterstützung des Toningenieurs beim initialen Soundcheck ist eine Technologie gesucht, welche die an jedem Kanal anliegende Audioquelle anhand des Eingangssignals live erkennt und die Information dem Mischpult bereitstellt. Mit dieser Information kann das Mischpult Voreinstellungen, wie Kanalname und Signalverarbeitungswerkzeuge, z. B. Kompressoren, Gates, Limiters, Equalizers, Effekte etc., automatisch vornehmen.

Als Anwendungsfall ist der Auftritt einer Rock-/Pop-Gruppe denkbar: Die Musiker bereiten ihre Instrumente auf der Bühne vor, während sich der Toningenieur mit der Verkabelung der Instrumente und Mikrofone beschäftigt. Anschliessend wird das Mischpult manuell konfiguriert, sodass jedes Instrument einen Kanal zugewiesen bekommt. Als erstes benennt der Toningenieur gemäss dem angeschlossenen Instrument die einzelnen Kanäle und weist diesen typischerweise Voreinstellungen (engl. Presets) zu. Sobald beide Parteien bereit sind, beginnt der Soundcheck, bei welchem der Toningenieur unter anderem Lautstärke und Signaleigenschaften der einzelnen Instrumente anpasst, um den bestmöglichen Bühnensound zu erreichen.

Die Konfiguration des Mischpults nach der Verkablung ist ein aufwändiger manueller Prozess, welcher automatisiert werden kann, indem das an einem Kanal anliegende Instrument anhand des Signals vom System erkannt wird. Im Idealfall könnte das Mischpult die angeschlossenen Instrumente während des Soundchecks selbstständig erkennen, die Kanäle beschriften und entsprechende Voreinstellungen laden.

Die automatische Erkennung von Musikinstrumenten wurde bisher in verschiedenen Anwendungsbereichen eingesetzt, beispielsweise bei der Kennzeichnung von Musikstücken (engl. Music Tagging). Dabei handelt es sich um die Ergänzung von Audiosignalen durch Metadaten, z. B. vorkommende Instrumente, Genre, Künstler etc. Eine weitere Anwendung ist die automatische Transkription von Musikstücken.

Systeme zur automatischen Instrumentenerkennung wurden bis vor wenigen Jahren in zwei einzelnen Schritten gelöst. In der ersten sogenannten Feature-Engineering-Phase werden Merkmale, welche die zu klassifizierenden Instrumente charakterisieren, definiert, die anschliessend durch einen Algorithmus zur Klassifikation verwendet werden. Die Wahl von geeigneten Merkmalen ist besonders aufwändig und erfordert Expertise über das Problemgebiet.

In den vergangenen Jahren sind Verfahren aufgekommen, welche in der Lage sind, die für ein Problem relevanten Merkmale aus den Daten selbstständig zu lernen. Diese sogenannten Feature-Learning-Verfahren gehören im Bereich der Sprach- und Bilderkennung zum Stand der Technik und zeigen überdurchschnittlich gute Resultate. Der Kern dieser Verfahren ist die automatische Merkmalsextraktion, die den Vorteil mit sich bringt, dass die aufwändige Feature-Engineering-Phase entfällt. Erste vielversprechende Resultate zeigen, dass Feature-Learning zum State-of-the-Art im Gebiet der automatischen Instrumentenerkennung werden kann.

1.2. Zielsetzung

Das Ziel dieser Arbeit ist die Implementierung eines prototypischen Softwaremoduls, welches die Quelle aus einem monophonen Audiosignal erkennt und klassifiziert. Effektiv soll dem Signal ein Label des entsprechenden Instruments zugewiesen werden. Dabei sollen in einer ersten Version die gängigsten Rock/Pop-Instrumente (Gitarre, Bass, Schlagzeug und Gesang) erkannt werden. Die Integration dieses Softwaremoduls in die Firmware der betroffenen Mischkonsolen ist kein Bestandteil der Arbeit.

1.3. Aufbau dieser Arbeit

Kapitel 2 vermittelt die notwendigen Grundlagen dieser Arbeit und erklärt die wesentlichen Konzepte und Methoden. Für vertiefte theoretische Details wird auf weiterführende Literatur verwiesen.

Kapitel 3 stellt das Konzept und die Implementation des Systems zur automatischen Musikinstrumentenerkennung, basierend auf Feature-Learning-Verfahren, vor.

Kapitel 4 zeigt Durchführung und Auswertung diverser Experimente zur Evaluation des implementierten Systems.

Kapitel 5 reflektiert und überprüft die erzielten Ergebnisse.

In Kapitel 6 werden die Ergebnisse in Bezug auf die Aufgabenstellung aufgezeigt und weiterführende Arbeiten vorgeschlagen.

2. Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen und der Stand der Technik im Bereich der automatischen Erkennung von Musikinstrumenten vermittelt.

2.1. Audiosignale

Dieser Abschnitt basiert auf dem Werk *Fundamentals of Music Processing* von Meinard Müller [1].

Schwingungen und Frequenzen

Die Bezeichnung *Audio* bezieht sich auf Übertragung, Empfang und Wiedergabe von Tönen, die vom Menschen wahrgenommen werden können. Ein Audiosignal ist eine Repräsentation von Tönen und codiert die gesamte Information, die für die Wiedergabe eines Tons benötigt wird, beispielsweise zeitliche und dynamische Eigenschaften, Frequenzen, Tonhöhe, Lautstärke und Klangfarbe. Diverse Merkmale wie Startzeitpunkt, Tonhöhe und Notendauer sind nicht explizit im Audiosignal gegeben, was den Vergleich von verschiedenen Signalen erschwert.

Töne werden in Form von Luftdruckschwingungen von einer Schwingungsquelle, beispielsweise der Saite einer Gitarre, zu einem Empfänger übermittelt. Die beim Anregen des Objekts entstehenden Schwingungen bringen die umliegenden Luftmoleküle in Bewegung, wodurch der lokale Luftdruck verändert wird. Beim Ziel können diese von einem Menschen als Ton wahrgenommen oder von einem Mikrofon zu einem elektrischen Signal umgewandelt werden. Abbildung 2.1 zeigt den zeitlichen Verlauf der Luftdruckveränderung.

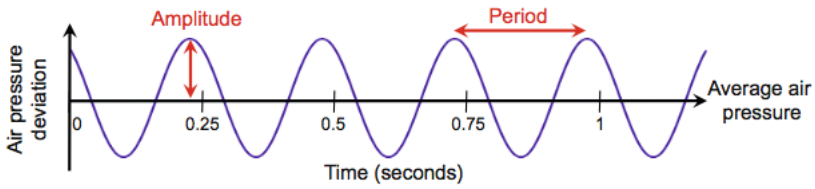


Abbildung 2.1.: Schwingungsverlauf eines Sinusoid mit einer Frequenz von 4 Hz [1, S. 21]

Eine Schwingung ist periodisch, falls sich die Hoch- und Tiefdruckstellen regelmässig wiederholen. Die Periode ist durch die benötigte Zeit eines Zyklus definiert (vgl. Abbildung 2.1). Die Frequenz (gemessen in Hertz) steht reziprok zur Periode.

Ein Sinusoid ist der einfachste Typ einer periodischen Schwingung und ist durch die Frequenz, Amplitude (höchste Abweichung vom Mittelwert) und Phase (Nullpunktstelle der Periode) definiert. Grundsätzlich ist die Tonhöhe proportional zur Frequenz. Der menschliche Hörbereich liegt zwischen 20 Hz und 20'000 Hz, wobei die Tonhöhe vom Menschen logarithmisch wahrgenommen wird. Die Distanz der Note A3 (220 Hz) zur Note A4 (440 Hz) wird demnach als gleich zur Distanz der Note A4 zur Note A5 (880 Hz) wahrgenommen. Die Note A4 wird Grundfrequenz genannt.

Töne in der realen Welt sind weit entfernt von puren Tönen mit einer wohldefinierten Frequenz. Das Spielen einer Note auf einem Instrument kann in einer komplexen Mischung aus zeitlich verändernden Frequenzen resultieren. Ein solcher musikalischer Ton kann durch eine Überlagerung von mehreren puren

Tönen beschrieben werden, wobei jeder einzelne dieser Sinusoiden eine eigene Frequenz, Amplitude und Phase besitzt. Ein Partialton (engl. partial) ist jeder Sinusoid, aus welchem ein musikalischer Ton zusammengesetzt ist. Die Frequenz des tiefsten Partialtons wird als fundamentale Frequenz des Tons bezeichnet, welche üblicherweise die Tonhöhe angibt. Ein harmonischer Partialton (engl. harmonic partial) ist ein ganzzahliges Vielfaches der fundamentalen Frequenz. Als Oberton werden alle Partialtöne ausser der fundamentalen Frequenz bezeichnet.

Ein Spektrogramm visualisiert die Zusammensetzung von einzelnen Frequenzkomponenten (Frequenzspektrum) eines Audiosignals über die Zeit. Abbildung 2.2 zeigt das Spektrogramm einer Aufnahme der C-Dur-Tonleiter, gespielt auf einem Klavier. Dabei sind die jeweiligen Partialtöne deutlich ersichtlich.

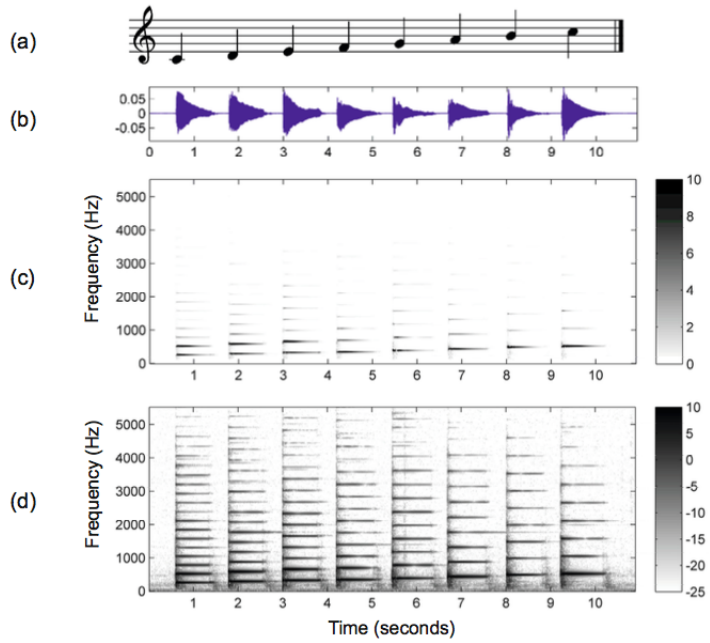


Abbildung 2.2.: Verschiedene Repräsentationen der Aufnahme einer auf einem Klavier gespielten C-Dur-Tonleiter: (a) Partitur (b) Wellenform (c) Spektrogramm (d) Spektrogramm in dB [1, S. 57]

Fourier-Analyse und -Transformation

Die Grundidee der Fourier-Analyse besteht darin, ein Signal mit Sinusoiden von verschiedenen Frequenzen zu vergleichen. Mit der Fourier-Transformation wird das Signal in all seine Frequenzkomponenten zerteilt. Eine wichtige Eigenschaft ist, dass das ursprüngliche Signal aus den einzelnen Frequenzkomponenten rekonstruiert werden kann. Dabei enthalten das ursprüngliche und das transformierte Signal denselben Informationsgehalt. Die beiden Repräsentationen haben allerdings unterschiedliche Bedeutungen über das Signal. Das Audiosignal zeigt zu welchem Zeitpunkt verschiedene Noten gespielt werden, verbirgt jedoch Information über die Frequenzen. Das transformierte Signal zeigt hingegen, welche Noten (Frequenzen) gespielt werden, jedoch ohne Zeitdimension. Abbildung 2.3 zeigt das Beispiel einer Fourier-Transformation eines Audiosignals.

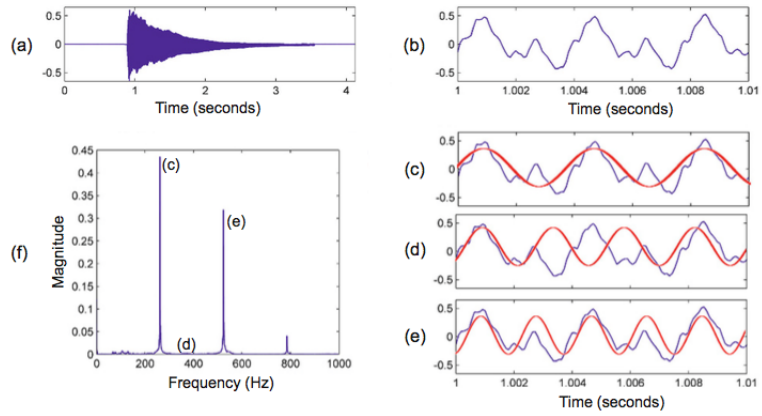


Abbildung 2.3.: (a) Wellenform der Note C4 (261.6 Hz), gespielt auf einem Klavier. (b) 10 ms Bereich des Signals. (c-e) Vergleich der Wellenform mit Sinusoiden von verschiedenen Frequenzen. (f) Transformiertes Signal. [1, S. 41]

2.2. Merkmale von Musikinstrumenten

Der Klang eines Musikinstruments besitzt eine Vielzahl an Eigenschaften, die das Instrument charakterisieren und die Unterscheidung zu anderen Instrumenten ermöglichen. Töne aus harmonischen Instrumenten, wie Gitarre oder Klavier, sind reichhaltig an harmonischen Partialtönen. Perkussive Instrumente, wie Trommeln und Becken, haben hingegen keine variable Tonhöhe und besitzen besonders viele unharmonische Partialtöne. Auch der zeitliche Verlauf solcher transienten Töne unterscheidet sich grundlegend.

Wichtige Eigenschaften eines Klangs sind die Lautstärke (engl. Loudness), die Tondauer (engl. Duration), die Tonhöhe (engl. Pitch) und die Klangfarbe (engl. Timbre). Die Lautstärke eines Tons ist definiert durch den physikalisch messbaren Schalldruck. Die Tondauer beschreibt die Dauer zwischen dem Anschlag und Zeitpunkt, an dem der Ton nicht mehr wahrgenommen werden kann. Die Tonhöhe beschreibt die subjektive Wahrnehmung der gehörten Frequenz. Zur Unterscheidung von Instrumenten spielt jedoch, unabhängig von den genannten Eigenschaften, die Klangfarbe eine wesentliche Rolle. Diese setzt sich aus dem Grundton, den Obertönen und Rauschanteilen zusammen und kann durch den temporalen und spektralen Verlauf, der An- oder Abwesenheit von Geräuschen und der Energieverteilung über die Partialtöne eines Tons erklärt werden (vgl. Müller [1]).

Abbildung 2.4 zeigt den zeitlichen Verlauf des Signals und das zugehörige Frequenzspektrum des gespielten Tons C4 auf den Instrumenten Klavier, Trompete, Geige und Flöte. Dabei sind die unterschiedlichen Spektren ersichtlich.

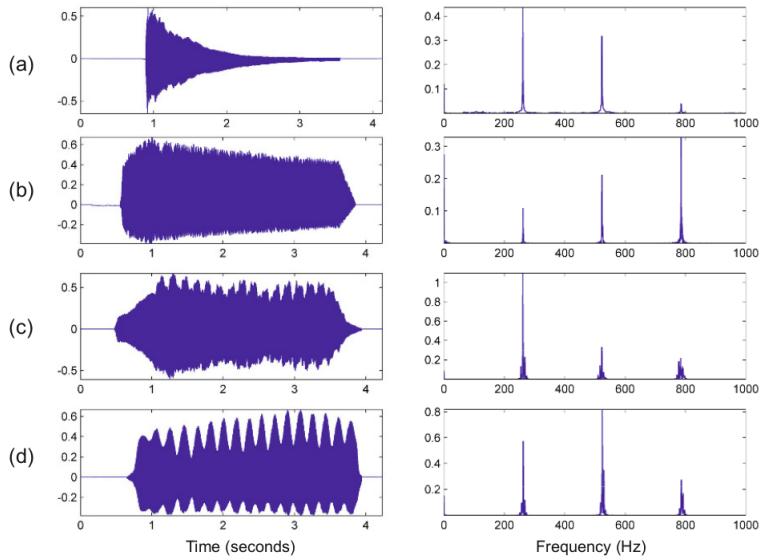


Abbildung 2.4.: C4 Ton gespielt auf (a) Piano (b) Trompete (c) Geige (d) Flöte [1, S. 59]

Instrumentenmerkmale können in drei Kategorien unterteilt werden, die bestimmte Aspekte von Audiosignalen gesondert beschreiben (vgl. Eisenberg [2, S. 56]):

- Zeitbereichsmerkmale
- Spektrale Merkmale
- Harmonische Merkmale

Im Folgenden werden die bekanntesten Merkmale dieser Kategorien vorgestellt. Für detaillierte Erklärungen wird auf das Werk von Gunnar Eisenberg [2] verwiesen.

Zeitbereichsmerkmale können direkt aus dem zeitlichen Signal berechnet werden. Die Nulldurchgangsrate (engl. Zero-Crossing Rate) gibt an, wie viele Nulldurchgänge das Signal innerhalb eines Zeitblocks erfährt. Für monophone Signale ist dieses Merkmal sehr aussagekräftig, da auf die Frequenz und somit auf den gespielten Ton geschlossen werden kann. Ein weiteres Zeitbereichsmerkmal ist der Effektivwert, der den zeitlichen Energieverlauf eines Signals beschreibt. Die Hüllkurve beschreibt einen ähnlichen Aspekt des Signals, lässt sich jedoch mit deutlich weniger Aufwand berechnen. Dabei werden die Hauptspitzen des Signals erfasst und zwischen diesen Punkten eine Interpolation durchgeführt.

Der zeitliche Verlauf eines einzelnen Instrumententons wird in vier Zeitbereiche unterteilt (siehe Abbildung 2.5). In der Anstiegsphase (engl. Attack) beginnt die Erzeugung des Tons. Die Anregung kann entweder impulsartig erfolgen, wie z. B. bei Trommeln, oder langsam ansteigen, wie bei Blas- oder Streichinstrumenten. In der Abklingphase (engl. Decay) nimmt die Lautstärke ab und das Signal pendelt sich in einen stabilen Zustand ein. In der Haltephase (engl. Sustain) breitet sich die Energie konstant aus und die zugeführte Leistung wird gleichmässig in Schallleistung umgewandelt. Im letzten Zeitbereich, der Ausschwingphase (engl. Release), wird die Energiezufuhr beendet und das Signal nimmt bis zum vollständigen Ausklingen ab.

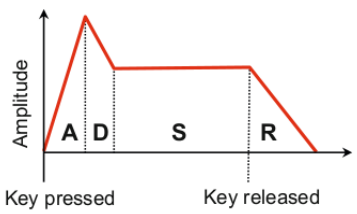


Abbildung 2.5.: Instrumentensignal als ADSR-Schema [1, S. 27]

Spektrale Merkmale werden aus dem Spektrogramm des Zeitsignals gewonnen. Der *Spectral Centroid* beschreibt Eigenschaften der Klanghelligkeit und der Klangschärfe. Ein Klang wird als hell wahrgenommen, falls im Signal viele energiereiche Obertöne und hohe Frequenzen vorhanden sind. Andernfalls wird der Klang als dumpf oder dunkel wahrgenommen. *Spectral Spread* gibt das Spektrum eines Tons an und bewertet es hinsichtlich des Spectral Centroid. *Spectral Skewness* gibt an, ob das Spektrum höhen- oder tiefenlastig ist. Die *Mel-Scale Frequency Cepstrum Coefficients* (MFCC) beschreiben in kompakter Form die wesentlichen Merkmale des Audiosignals. Mithilfe der Mel-Skala werden Aspekte der logarithmischen Tonhöhenempfindung des Menschen berücksichtigt, welche beschreibt, dass tiefe Töne leichter voneinander unterscheidbar sind als hohe Töne.

Die harmonischen Merkmale beschreiben die Harmonizität der Frequenzen, die in einem Signal vorkommen. Die Grundfrequenzerkennung gehört zu den harmonischen Merkmalen und kann sich für die Parametrisierung von Klassifikationsalgorithmen eignen. Die Rauschartigkeit ist ein Merkmal, welches das Verhältnis der Energie der nichtharmonischen Frequenzen im Verhältnis zur Gesamtenergie beschreibt. Weitere harmonische Merkmale sind *Harmonic Spectral Deviation* und *Harmonic Inner Ratio*.

Ein Merkmalsvektor, der n Merkmale zusammenfasst, lässt sich eindeutig als Punkt im n-dimensionalen Hyperraum beschreiben. Eine zu grosse Dimensionalität der Merkmalsvektoren stellt für Klassifikationsaufgaben ein bedeutendes Problem dar. Die für das Training der Modelle benötigte Anzahl der Datensätze steigt überproportional mit der Anzahl der möglichen Zustände im Hyperraum. Die Klassifikationsergebnisse verschlechtern sich drastisch, falls die Anzahl der Trainingsdaten bei steigender Dimensionalität nicht erhöht wird. Um dieses Problem zu umgehen gibt es eine Menge von Verfahren zur Dimensionsreduktion, beispielsweise *Principal Component Analysis* (PCA), *Linear Discriminant Analysis* (LDA) oder *Matrixfaktorisierung* (engl. Matrix Decomposition).

2.3. Klassische automatische Instrumentenerkennung

Im Bereich Music Information Retrieval (MIR) wird das Ziel verfolgt, inhaltsbeschreibende Metadaten aus Audiosignalen zu gewinnen. Dabei werden monophone und polyphone Signale unterschieden. In monophonen Signalen kommt ausschliesslich eine Audioquelle, beispielsweise ein Musikinstrument, vor. Polyphone Signale hingegen bestehen aus einer Mischung von unterschiedlichen Quellen. Die Untersuchung von polyphonen Signalen bringt eine zusätzliche Herausforderung, da die im Signal vorkommenden Quellen getrennt werden müssen. Diese Arbeit befasst sich mit der Instrumentenerkennung in monophonen Audiosignalen, wie sie bei einem Audio-Mischpult im Normalfall vorliegen.

Inhaltsbasierte Klassifikationsprobleme im Bereich von MIR verfolgten bisher eine Zweischritt-Architektur (vgl. Li et al. [3]). In einem ersten Schritt (sog. Feature-Engineering-Phase) werden die für das Problem relevanten Merkmale aus dem Signal manuell extrahiert, die anschliessend durch einen Klassifikationsalgorithmus einer Klasse zugeordnet werden. Viele dieser Algorithmen beruhen auf Wahrscheinlichkeitsmodellen, z. B. das *Gaussian Mixture Model* (GMM) oder das *Hidden Markov Model* (HMM), wobei sich die Entscheidung meist über die Bayes'sche Klassifikation ergibt [2, S. 112]. Die Übersetzung der relevanten Merkmale auf algorithmische Methoden stellt dabei eine grosse Schwierigkeit dar, da diese häufig von der Problemstellung abhängen.

Kaminsky und Materka [4] untersuchten 1995 in einer der ersten Arbeiten im Bereich der automatischen Instrumentenerkennung die Klassifikation der Musikinstrumente Gitarre, Klavier, Marimba und Akkordion in monophonen Audiosignalen. Ihr Ziel war die automatische Transkription von Musikstücken durch die Instrumentenerkennung zu unterstützen. Mit dem selbst aufgenommenen Datensatz erreichten sie eine Genauigkeit von über 90%. Als Eingabedaten für ein neuronales Netz verwendeten sie die RMS-Leistung (Zeitbereichsmerkmal) der einzelnen Aufnahmen.

1998 untersuchten Martin und Kim [5] die Klassifikation von 15 Orchesterinstrumente in monophonen Audiosignalen mithilfe von spektralen und temporalen Merkmalen. Zur Klassifikation verwendeten sie sowohl GMM als auch *k-Nearest-Neighbors* (k-NN), wobei sie in der Klassifikation von einzelnen Instrumenten eine Genauigkeit von ca. 70% und 90% in der Klassifikation von Instrumentenfamilien erreichten. Im selben Jahr untersuchte Brown [6] die automatische Erkennung der beiden ähnlich klingenden Musikinstrumente Oboe und Saxophon mittels spektralen Merkmalen und GMM zur Klassifikation.

Eronen und Klapuri [7] verwendeten im Jahre 2000 *cepstrale* Koeffizienten und temporale Merkmale für die Klassifikation von 30 Orchesterinstrumenten und erreichten eine Genauigkeit von 95% bei Instrumentenfamilien und 81% bei einzelnen Instrumenten. Weitere Untersuchungen von geeigneten Merkmalen und Klassifikationsalgorithmen, beispielsweise SVM, GMM und HMM, folgten in [8], [9] und [10]. Die meisten dieser Arbeiten erreichten eine Genauigkeit von unter 85%, wobei aufgrund der unterschiedlichen Daten und Instrumentenklassen keine direkten Vergleiche gemacht werden können.

Viele dieser Papers verwendeten isolierte Töne als Grundlage. 2004 untersuchten Krishna und Sreenivas [11] die Klassifikation von Musikinstrumenten in Solo-Phrasen. Sie verwendeten lineare spektrale Frequenzen (LSF) und GMM und erreichten eine Genauigkeit von ca. 84% bei der Klassifikation von 14 verschiedenen Instrumenten. Essid et al. erforschten die Erkennung von fünf Musikinstrumenten mittels Solo-Phrasen in [12] und [13], wobei MFCCs im Zusammenhang mit GMM und PCA benutzt wurden.

Guignard und Kehoe untersuchten 2015 in [14] die Klassifikation von den Musikinstrumenten akustische und elektrische Gitarre, elektrischer Bass, Schlagzeug, Bass Drum, Snare Drum, Cymbals und Hi-Hat in monophonen Audiosignalen. Sie benutzten spektrale Merkmale und SVM als Klassifikationsalgorithmus und erreichten eine Genauigkeit von 93% mit dem verwendeten Datensatz.

2.4. Feature Learning und End-to-End-Systeme

In den vergangenen Jahren sind im Gebiet *Machine Learning* Verfahren unter dem Namen *Deep Learning* aufgekommen, die in der Lage sind, relevante Merkmale aus Daten selbst zu lernen. Dies hat zur Folge, dass die manuelle Wahl von geeigneten Merkmalen (Feature-Engineering-Phase) und die dazu erforderliche Expertise, entfällt.

Deep-Learning-Verfahren bilden End-to-End-Systeme, da die Abbildung eines Inputs zum gewünschten Output allein aus den Daten gelernt wird (vgl. [15]). Diese bringen einige Vorteile gegenüber den klassischen Zweischrittverfahren (vgl. [16]):

- Die manuelle Wahl von geeigneten Merkmalen und dessen Interpretation wird automatisiert.
- Die gelernten Merkmale sind direkt auf die Problemstellung zugeschnitten, was zu besseren Klassifikationsergebnissen führen kann.
- Die gelernten Merkmale geben Einblicke in die für eine Problemstellung relevante Information.

Deep-Learning-Verfahren haben traditionelle Ansätze in den Bereichen Bilderkennung (engl. Computer Vision), Text- und Sprachverarbeitung (engl. Natural Language Processing) übertroffen und gehören heute in diesen Gebieten zum State-of-the-Art (vgl. [17], [18], [19]). 2012 wurde von Humphrey und LeCun [16] vorgeschlagen, Feature Learning in MIR-Problemstellungen anzuwenden und sich von den klassischen Zweischrittverfahren zu distanzieren.

Die grundlegende Architektur von Deep-Learning-Verfahren sind mehrschichtige vorwärts gerichtete neuronale Netze. Abhängig vom Problem existieren unterschiedliche Architekturen. Insbesondere *Recurrent Neural Networks* (RNN) und *Convolutional Neural Networks* (CNN) haben in multimedialen Information-Retrieval-Problemstellungen bemerkenswerte Ergebnisse erzielt (vgl. [20]).

Dieleman et al. [15] wendeten 2014 CNNs für die Kennzeichnung von Musikstücken, z. B. bei der Erkennung von vorkommenden Instrumenten, Genre, Emotionen etc., erfolgreich an. Dabei wurden Spektrogramme und rohe Audiodaten als Eingabedaten untersucht. Die Resultate zeigten, dass Spektrogramme als Eingabedaten bessere Klassifikationsergebnisse erzielen. Li et al. [3] entwickelten 2015 ein System basierend auf einem CNN zur automatischen Instrumentenerkennung in polyphonen Audiosignalen und zeigten, dass Deep-Learning-Verfahren bessere Ergebnisse erzielen können als traditionelle Zweischrittverfahren. Als Eingabedaten für das CNN wurden rohe Audiodaten verwendet. Park et al. [21] untersuchten im selben Jahr die Klassifikation von 20 Musikinstrumenten in monophonen Audiosignalen mittels CNNs, wobei der verwendete Datensatz aus isolierten Einzeltönen bestand. Ihre Experimente erreichten eine Genauigkeit von über 90%. Als Eingabedaten wurde eine Kombination von Spektrogrammen und *Multiresolution Recurrent Points* (MRP) verwendet.

Im Mai 2016 wurden zwei weitere Papers im Bereich der automatischen Instrumentenerkennung veröffentlicht. Han et al. [20] untersuchten die Erkennung von vorkommenden Instrumenten in polyphonen Audiosignalen mithilfe von CNNs und Mel-Spektrogrammen und erreichten Ergebnisse, die den State-of-the-Art übertreffen. Lostanlen und Cella [22] erreichten eine Genauigkeit von 74% in der Klassifikation der Instrumente Piano, Violine, Gitarre, Klarinette, Flöte, Trompete, Saxofon und weiblicher Gesang in monophonen Audiosignalen. Die Daten stammten grösstenteils aus der MedleyDB-Datenbank [23]. Als Eingabedaten für das CNN wurden *Constant-Q-Spektrogramme* verwendet, welche vergleichbare Eigenschaften wie Mel-Spektrogramme besitzen.

Die jüngsten vielversprechenden Erkenntnisse zeigen, dass CNNs, insbesondere im Zusammenhang mit Spektrogrammen als Eingabedaten, vergleichbare Ergebnisse zum State-of-the-Art in der Klassifikation von Musikinstrumenten erzielen und diese gar übertreffen können.

Spektrogramme als Audiorepräsentation

Ein Spektrogramm beschreibt die Zusammensetzung der Frequenzkomponenten des Audiosignals und ist vereinfacht eine Matrix, bestehend aus vielen Zahlen. Die Herausforderung besteht darin, von all diesen Zahlen auf ein einzelnes Instrumentenlabel zu schliessen.

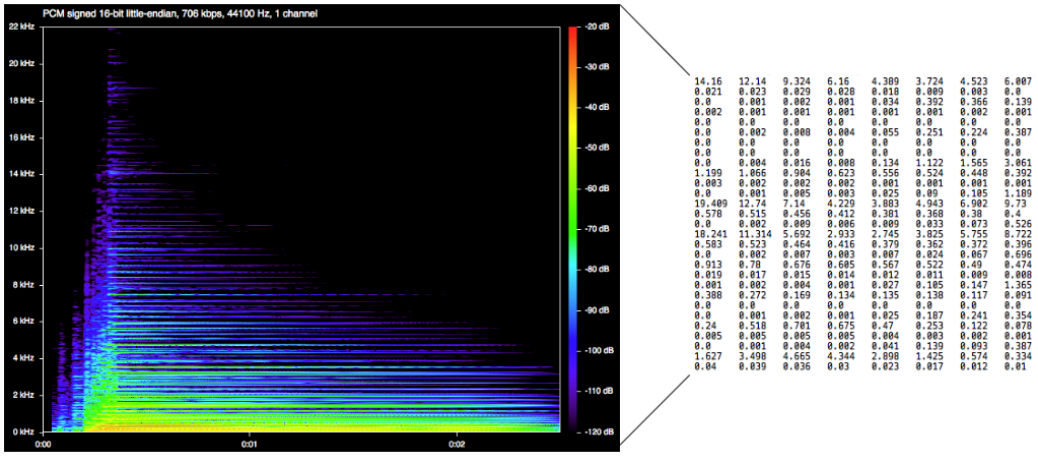


Abbildung 2.6.: Spektrogramm und Ausschnitt der Zahlenmatrix eines E1-Minor-Akkords, gespielt auf einer unverzerrten Fender-Strat-E-Gitarre

Für den Menschen kann sich die Aufgabe der Klassifikation einer kleinen Menge an Musikinstrumenten als trivial erweisen. Ein Algorithmus muss jedoch einige Aspekte beachten:

- **Tonhöhenvariation**: Ein harmonisches Instrument deckt häufig einen breiten Notenbereich ab. Ein klassisches Klavier kann beispielsweise 88 unterschiedliche einzelne Töne erzeugen.
- **Spieldynamik**: Ein Instrument kann auf diverse Arten gespielt werden. Je nach Musikgenre kann die Spieldynamik extrem variieren. Zudem können Instrumente sowohl Einzeltöne als auch Akkorde produzieren, z. B. das Klavier.
- **Klangvariation**: Der Klang eines Instruments kann unter anderem durch Effekte verändert werden. Für E-Gitarren existieren Verzerrer, Delays, Hall-Effekte und eine Menge weiterer Effektgeräte.
- **Störgeräusche**: Eine Instrumentenspur kann Störgeräusche und Fragmente anderer Instrumente aufweisen, was die Klassifikation erschwert.

2.5. Deep Convolutional Neural Networks

Anwendungen von CNNs gehen bis in die frühen neunziger Jahre zurück, als diese in der Spracherkennung und Textverarbeitung eingesetzt wurden. 1989 führten LeCun et. al. [24] erstmals CNNs für die automatische Erkennung von handgeschriebenen Zahlen ein, welche gegen Ende der neunziger Jahre für die Verarbeitung von über 10% der Schecks der Vereinigten Staaten eingesetzt wurden. Bereits seit dem Jahre 2000 werden CNNs für die Erkennung von Objekten in Bildern eingesetzt, wobei das Gebiet seit wenigen Jahren einen Durchbruch, aufgrund von Fortschritten in der Grafikprozessorhardware (GPU), erlebt.

CNNs sind biologisch durch die Ähnlichkeiten zur Funktionsweise des visuellen Kortex inspiriert und nutzen hierarchische Eigenschaften von Bildern aus, in welchen komplexe Objekte aus einfachen Strukturen zusammengesetzt sind. Beispielsweise bilden Kombinationen von Kanten und Ecken Formen, welche wiederum zu ganzen Objekten zusammengesetzt werden.

CNNs verarbeiten Daten, welche in Form von Arrays vorliegen, z. B. Bilder als zweidimensionale Arrays, die aus Pixel-Werten und drei Farbkanälen (RGB) bestehen. Ein Spektrogramm kann ebenfalls als ein mehrdimensionales Array dargestellt werden (vgl. [25]).

Im Folgenden werden drei Kernkonzepte von CNNs – *Kernel-Convolution*, *Pooling* und neuronale Netze – eingeführt.

Kernel Convolution

Eine *Convolution* beschreibt in der Mathematik eine Operation, die zwei Funktionen entgegennimmt und eine Funktion als Produkt der zwei Eingabefunktionen produziert. In der Bildverarbeitung wird die Convolution-Operation als Filter verwendet, wobei ein Eingangsbild durch einen Effekt verändert wird. Der verwendete Filter (auch *Kernel* genannt) verursacht je nach Parameter unterschiedliche Effekte auf dem Zielbild. Beispiele solcher Filter sind Unschärfe- oder Kantendetektionsfilter. Abbildung 2.7 zeigt ein Beispiel einer Convolution-Operation, in welcher eine Kantenhervorhebung stattfindet.

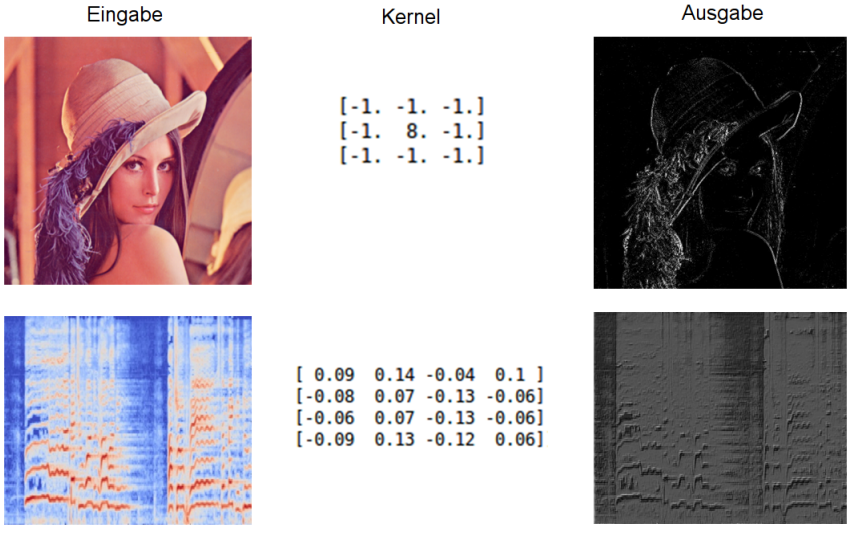


Abbildung 2.7.: Beispiel einer Convolution-Operation (Bild oben links aus [26])

Pooling

Eine Pooling-Operation ist eine Form von verlustbehafteter Kompression (engl. Subsampling). Bei *Max-Pooling* wird eine Matrix in gleich grosse, nicht-überlappende Rechtecke partitioniert. Für jedes dieser Rechtecke wird anschliessend der maximale Wert ermittelt und in die Ausgangsmatrix gespeichert. Bei *Average-Pooling* wird nicht der maximale, sondern der durchschnittliche Wert ermittelt. Abbildung 2.8 zeigt ein Beispiel einer Max-Pooling-Operation.

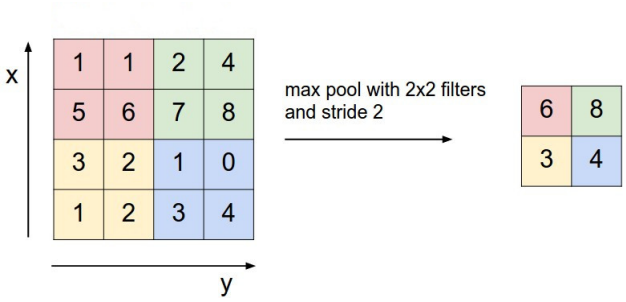


Abbildung 2.8.: Beispiel von Max-Pooling [27]

Neuronale Netze

Dieser Abschnitt gibt einen Einblick in die wesentlichen Konzepte von neuronalen Netzen und hebt die essentiellen Aspekte für diese Arbeit hervor. Für detaillierte Erklärungen wird auf das Werk *Neural Networks and Deep Learning* von Michael Nielsen [28] verwiesen.

Neuronale Netze werden für Klassifikations- und Regressionsaufgaben eingesetzt und haben eine mehrschichtige Architektur. Eine Schicht besteht aus mehreren Neuronen, wobei die einzelnen Neuronen mit allen Neuronen der nachfolgenden Schicht verbunden sind. Sämtliche Verbindungen zwischen zwei Neuronen besitzen Gewichtungen, welche als 'Verbindungsintensität' interpretiert werden können.

Die erste Schicht bildet den sogenannten *Input Layer* und nimmt einen Merkmalsvektor entgegen. Die einzelnen Werte werden durch die nachfolgenden Schichten (sogenannte *Hidden Layers*) verarbeitet. Die letzte Schicht bildet den *Output Layer*, welcher die transformierte Information auf die Neuronen dieser Schicht verteilt. Bei Klassifikationsaufgaben entspricht dies den einzelnen Klassen. Abbildung 2.9 zeigt ein Beispiel eines neuronalen Netzes mit einem einzigen Hidden Layer und drei Eingangsmerkmalen, welche auf zwei Output-Klassen transformiert werden.

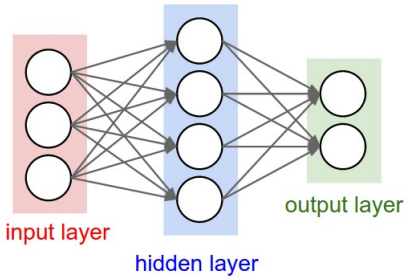


Abbildung 2.9.: Beispiel eines neuronalen Netzes [29]

Bei überwachtem Lernen (engl. Supervised Learning) wird ein neuronales Netz iterativ trainiert. Ziel des Trainings ist, dass das Netz lernt, unbekannte Daten korrekt zu klassifizieren.

Zu Beginn des Trainings werden die Gewichtungen mit zufälligen Werten initialisiert. Die einzelnen Trainingsdaten werden vorwärts durch das Netz verarbeitet, was in der letzten Schicht zu einem Klassifikationsergebnis führt. Für jedes Datum wird die Abweichung des vorausgesagten zum tatsächlichen

Ergebnis berechnet. Mithilfe des *Backpropagation-Algorithmus* wird diese Abweichung rückwärts auf die einzelnen Gewichtungen verteilt, mit dem Ziel, dass der Klassifikationsfehler in der nächsten Iteration sinkt. Mit dem fertig trainierten Netz können anschliessend Voraussagen (engl. Predictions) auf unbekannten Daten gemacht werden.

Architektur und Funktionsweise von CNNs

Bilder sind nicht als Input für gewöhnliche neuronale Netze geeignet. Bei einem Bild von 200 x 200 Pixeln und drei Farbwerten besässe alleine der Input Layer bereits 120'000 Gewichtungen (vgl. [30]). Dieser Vernetzungsgrad ist aufgrund der Rechenintensität deutlich zu hoch. CNNs verfolgen daher einen anderen Ansatz.

Die Architektur eines CNN besteht aus mehreren Schichten, wobei jede Schicht mit der nächsten verbunden ist. Die ersten Schichten bilden die so genannten *Convolution-Layers* und bestehen aus subsequenten Convolution- und Pooling-Operationen. Jeder Convolution-Layer besitzt eine Anzahl an lernbaren Filtern. Jeder dieser Filter wird über die Input-Matrix verschoben und berechnet das Skalarprodukt an jeder Position. Die resultierenden *Activation Maps* beschreiben somit die Reaktion des angewandten Filters an sämtlichen Positionen der Matrix und dienen der nachfolgenden Schicht als Input (vgl. Abbildung 2.10).

Während des Trainings werden die Filter so verändert, dass sie auf ein bestimmtes Muster reagieren, beispielsweise eine Kante, eine bestimmte Farbe oder abstraktere Muster, wie Kreise und ähnliche Formen. Je höher (in Abbildung 2.10 rechts) die Schicht, umso abstraktere Muster werden erkannt.

Nach einer Convolution-Operation findet üblicherweise eine Max-Pooling-Operation statt. Dadurch schrumpft die Matrix je nach Pooling-Konfiguration, was zu einer geringeren Rechenlast führt. Durch die gleichbleibende Filtergrösse werden räumlich immer grössere und somit abstraktere Muster gelernt.

Die höheren Schichten nach den Convolution Layers werden als *Dense-* oder *Fully Connected Layers* bezeichnet und bilden ein neuronales Netz, wobei die Activation Maps der letzten Convolution-Schicht direkt mit dem ersten Dense Layer verbunden sind (vgl. [30]). Abbildung 2.10 zeigt die Beispielarchitektur eines CNN für die Verarbeitung von Mel-Spektrogrammen.

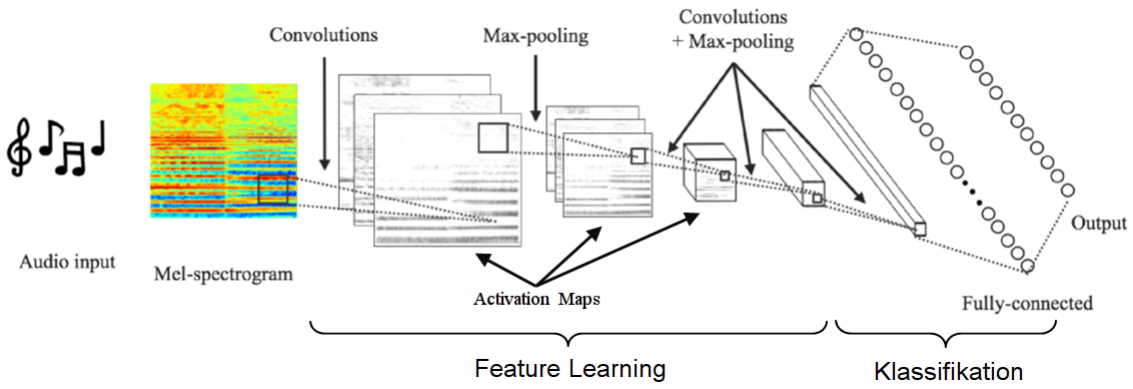


Abbildung 2.10.: CNN-Architektur [31]

Die Convolution Layers können als Feature-Learning-Komponente betrachtet werden, da in diesem Teil relevante Merkmale aus den Eingangsdaten zur Klassifikation extrahiert werden. Die Fully Connected Layers können als verantwortliche Komponente für die Klassifikation betrachtet werden.

Verstellbare Hyperparameter

An einem CNN können die folgenden Hyperparameter konfiguriert werden:

- Anzahl Convolution-Filter
- Filter-Dimension
- Max-Pooling-Dimension
- Verschiebung (engl. Stride)

Aktuell existiert kein Verfahren zur Festlegung einer optimalen Architektur für ein gegebenes Problem. Üblicherweise wird die Architektur eines CNN mittels mehreren Experimenten ermittelt. Der folgende Ausschnitt einer Diskussion mit dem Titel '*How Do You Choose Your Convolutional Neural Net Architecture?*' behandelt das Problem der Wahl einer geeigneten Architektur.

Ein anonymer Benutzer stellte die Frage:

> „*I am convinced this is at least half 'black magic' and intuition, but what sort of heuristics do people use when experimenting with different layer architectures?* [...]“ [32]

Daraufhin antwortete Sander Dieleman:

> „*I agree that it involves a lot of intuition, experience and trial and error. My approach is typically to start with a relatively shallow architecture (say 2 convolutional layers + pooling layers, a fully-connected hidden layer and an output layer). I choose the filter sizes and pooling windows in such a way that the representation at the output of the convolutional part of the network is relatively small (i.e. a 4x4 feature map for example). Then, keep adding layers until performance stops improving :) I usually end up with 4 or 5 convolutional layers (with some of them followed by pooling layers), two fully connected hidden layers + output layer.* [...]
>
> *For the number of units on each layer, you typically want to start small so you can evaluate more quickly. That usually means having fewer units in the first layer and an increasing number as you go up in the network (the first few layers get larger input feature maps, so they do more work, hence it's faster to have fewer units there). Then just increase the number of units until you start overfitting / run out of patience :)* [...]“ [32]

3. Implementation

In diesem Kapitel werden die wichtigsten Aspekte der Implementation des Systems zur automatischen Instrumentenerkennung beschrieben.

3.1. Datensatz

Die Audiodaten für das Training des CNN stammen grösstenteils aus der MedleyDB-Datenbank [23], welche einzelne Instrumentenspuren von diversen Liedern bereitstellt. Zusätzlich beinhaltet der verwendete Datensatz diverse Studioaufnahmen, welche die Firma Studer zur Verfügung stellte, Aufnahmen aus der ENST-Drums-Datenbank [33] und Daten aus dem Fraunhofer Institute for Digital Media Technology (IDMT). Von allen Daten wurden ausschliesslich Spuren aus den relevanten Instrumentenklassen extrahiert. Die Einkanal-Aufnahmen liegen mit einer Sampling-Rate von 22'050 Hz vor.

In diversen Aufnahmen des Datensatzes waren Ruhephasen von mehreren Sekunden vorhanden, in denen kein Instrument gespielt wurde oder auschliesslich Geräusche hörbar waren. Um unerwünschte Nebeneffekte durch diese Daten während der Trainingsphase zu vermeiden, wurden Ruhephasen, die unter einem bestimmten Schwellwert lagen, aus dem Datensatz entfernt. Zudem wurde die Lautstärke aller Daten auf einen Richtwert normalisiert. Abbildung 3.1 zeigt die Gesamtdauer aller Aufnahmen pro Instrumentenklasse. Dabei ist ersichtlich, dass verhältnismässig wenig Audiomaterial der Klassen Electric Guitar (Direct Input), Hihat, Kick Drum, Snare Drum, High Tom und Low Tom vorhanden ist.

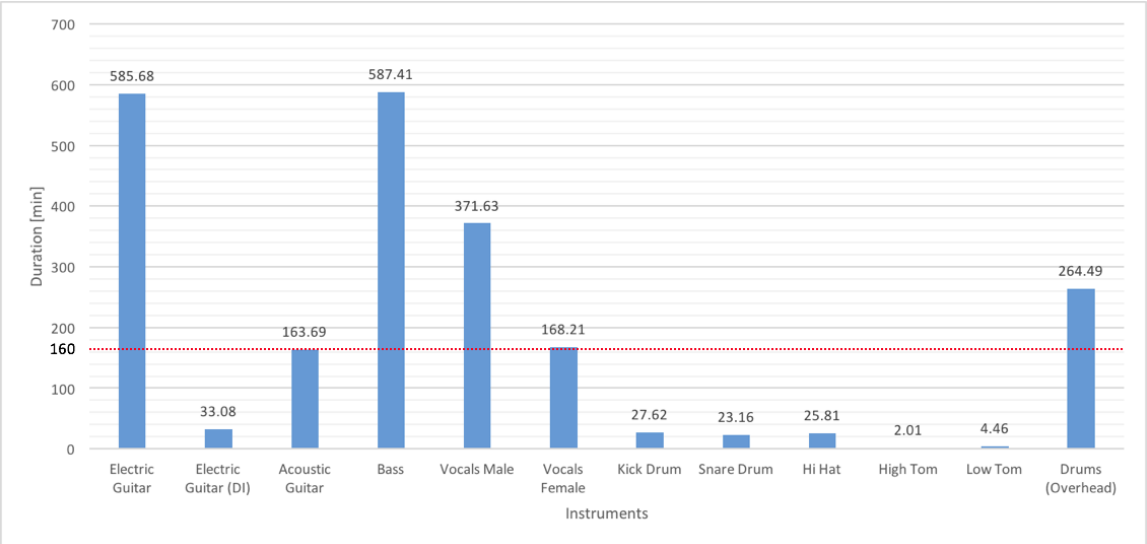


Abbildung 3.1.: Datenmenge pro Instrumentenklasse in Minuten

Aufgrund der geringen Datenmenge dieser Instrumentenklassen, und weil Feature-Learning-Verfahren eine grosse Menge an Trainingsdaten erfordern, wurde beschlossen, sie für das zu entwickelnde System auszulassen. Dadurch ergeben sich die sechs zu erkennenden Instrumentenklassen:

- Akustische Gitarre
- Elektrische Gitarre
- Elektrischer Bass
- Weiblicher Gesang
- Männlicher Gesang
- Schlagzeug

Bei der Klasse Schlagzeug handelt es sich um Aufnahmen von Overhead-Mikrofonen, die das gesamte Schlagzeug-Kit aufnehmen. Der fertige Datensatz beinhaltet gleich viel Audiomaterial (160 Minuten) pro Instrument.

3.2. Verwendete Hard- und Software

Für das Training des CNN wurde ein Server des Instituts zur Verfügung gestellt. Die folgenden Python-Bibliotheken wurden für die Datenaufbereitung und für die Implementation des CNN verwendet:

- Theano 0.8.2
- Lasagne 0.1
- Nolearn 0.6adev
- Librosa 0.4.2
- SciPy 0.17.0
- NumPy 1.11.0
- PyDub 0.16.4

Das Training von CNNs erfordert spezielle Hochleistungshardware, um die Berechnungen effizient durchzuführen. Grafikkarten eignen sich wegen einer tiefen Latenz durch die Parallelisierung besser als CPUs (vgl. [34]). Das Training kann dabei mehrere Stunden in Anspruch nehmen. Ein trainiertes Netz kann je nach Aufwand der Datenaufbereitung auf einer vergleichsweise weniger leistungsfähigen Hardware durchgeführt werden und benötigt deutlich weniger Zeit für die Klassifikation.

Der für das Training verwendete Server besitzt eine Nvidia GeForce GTX Titan X Grafikkarte, welche die Programmierschnittstelle CUDA in der Version 7.5 anbietet. Diese wird von der Bibliothek Theano angesteuert.

3.3. Konzept

Dieses Kapitel beschreibt wesentliche Konzepte des implementierten Systems.

CNN-Eingabedaten

Das CNN verwendet Spektrogramme der Instrumentenaufnahmen als Eingabedaten. Die Spektrogrammerzeugung wird in Kapitel 3.4 näher behandelt. Abbildung 3.2 zeigt zufällige Spektrogramme der sechs Instrumentenklassen, dabei ist der Unterschied von Auge deutlich erkennbar. Beispielsweise ist ersichtlich, dass ein Bass eher tiefe Frequenzen aufweist. Dahingegen tritt bei der elektrischen Gitarre ein reichhaltiges Spektrum an Obertönen in der Mitte des Frequenzbandes auf.

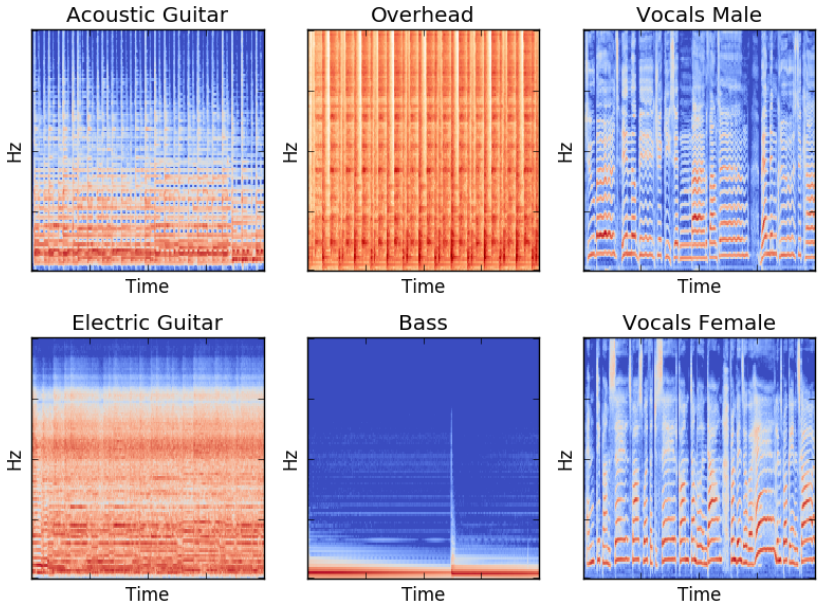


Abbildung 3.2.: Spektrogramme der zu erkennenden Instrumentenklassen

CNN-Ausgangsarchitektur

Abbildung 3.3 zeigt die vorgeschlagene Architektur aus Dieleman et al. [15], wo die Tauglichkeit von CNNs für MIR-Problemstellungen untersucht und als Ausgangsarchitektur für diese Arbeit gewählt wurde. Die genannte Arbeit befasst sich mit der allgemeinen Kennzeichnung von Musikstücken, z. B. vorkommende Instrumente, Genre, Spielweisen, Emotionen etc.

Die ersten zwei Convolution-Schichten bilden die Grundlage für die Merkmalsextraktion. Die Convolution-Filter und die Max-Pooling-Operationen sind eindimensional in Zeitrichtung. Die Abkürzung *S* (Size) steht für die Grösse und die Abkürzung *St* (Stride) für die Verschiebung der Max-Pooling-Matrix.

Die Schichten fünf und sechs bilden das neuronale Netz (Dense Layers). Die jeweiligen Neuronen sind mit einer linearen Aktivierungsfunktion (Rectified Linear Function) konfiguriert. Die Ausgabeschicht besitzt sechs Neuronen (ein Neuron pro Instrument), die eine Soft-Max-Funktion verwenden, um die Wahrscheinlichkeiten der Klassen auf die Ausgangsneuronen zu verteilen. Der Fehler zwischen der vorhergesagten und der tatsächlichen Ausgabe wird mithilfe der Kreuzentropiefunktion (engl. Cross Entropy Function) berechnet.

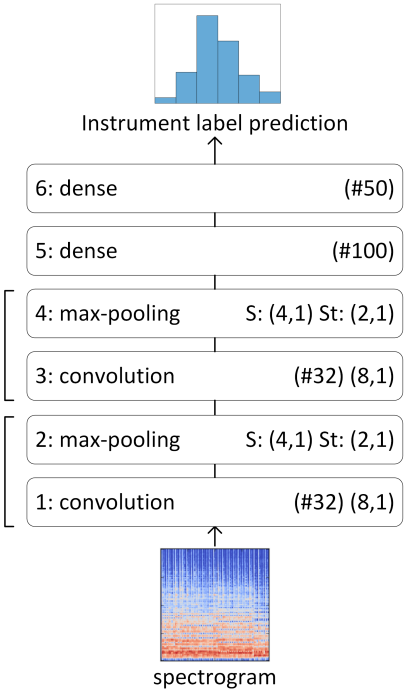


Abbildung 3.3.: CNN-Grundarchitektur

Aufteilung des Datensatzes

Der gesamte Datensatz wird in zwei Sätze aufgeteilt: Das Trainings- und das Testset. Diese Trennung dient dazu, das trainierte CNN mithilfe der unbekannten Daten aus dem Testset zu evaluieren. Ein bestimmter Teil des Trainingssets wird für Validierungszwecke während der Trainingsphase verwendet, um nach jeder Trainingsiteration die Genauigkeit des Netzes zu ermitteln.

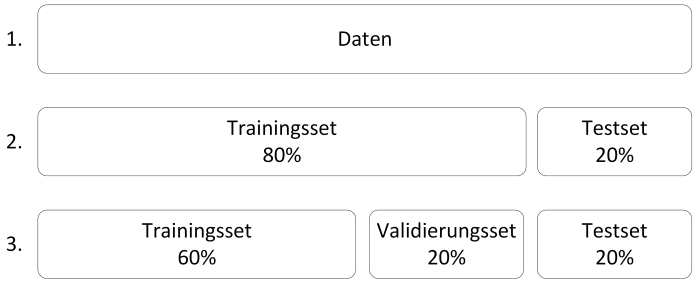


Abbildung 3.4.: Aufteilung des Datensatzes

3.4. Datenaufbereitung

Aus den Daten, die in der Dateistruktur im Wave-Format vorliegen, werden mittels dem *data_ preparation.py*-Script zehnsekündige Spektrogramme generiert, welche als Eingabedaten für das CNN dienen. Die resultierenden Spektrogramme werden anschliessend einem Trainings- und einem Testset zugewiesen.

Für die Generierung eines Spektrogramms wird ein Signal zuerst in gleich grosse Abschnitte (FFT-Window) unterteilt. Anschliessend wird für jeden Abschnitt das Spektrum berechnet. Die zusammengesetzten Spektrum-Vektoren ergeben ein Spektrogramm, wobei die horizontale Achse die Zeit und die vertikale Achse die Frequenz repräsentiert. Die lineare Frequenzskala der Spektrogramme wird mithilfe der Formel (3.1) in die Mel-Skala umgerechnet, welche einerseits die Dimensionalität verringert und andererseits die logarithmische Tonhöhenwahrnehmung des menschlichen Gehörs abbildet (vgl. [35]).

$$M(f) = 1125 \cdot \ln(1 + \frac{f}{700}) \tag{3.1}$$

Die Spektrogramme werden mithilfe der Python-Bibliothek *librosa* mit den folgenden Konfigurationen generiert:

- FFT-Zeitfenster: 2048
- Hop-Länge: 210
- Sampling-Rate: 22'050 Hz
- Maximale Frequenz: 11'025 Hz
- Anzahl Mel-Koeffizienten: 128

Das Spektrogramm wird gemäss [15] dynamisch komprimiert. Die Kompression erfolgt durch die Formel (3.2), wobei die Konstante C den Kompressionsgrad steuert. Der Wert wurde entsprechend auf 10'000 gesetzt.

$$f(x) = \log(1 + C \cdot x) \tag{3.2}$$

Abbildung 3.5 zeigt den Ablauf der Spektrogrammerzeugung und der Aufteilung der Daten.

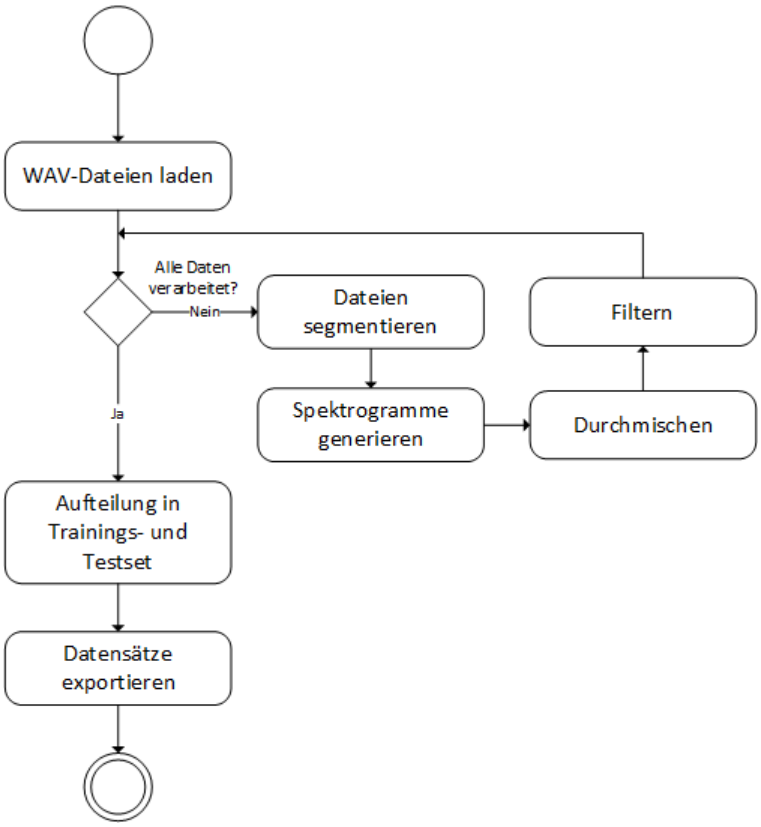


Abbildung 3.5.: Ablauf der Spektrogrammerzeugung

1. **WAV-Dateien laden**: Die WAV-Dateien werden mithilfe der Python-Bibliothek *librosa* eingelesen.
2. **WAV-Dateien segmentieren**: Die eingelesenen Dateien werden in zehnsekündige Segmente geteilt. Unvollständige Segmente werden mit Nullen ergänzt, um die gewünschte Segmentgrösse zu erreichen.
3. **Spektrogramme generieren**: Aus den einzelnen Segmenten werden mithilfe der Python-Bibliothek *librosa* Spektrogramme generiert. Das Script *spectrogram_utils.py* stellt die hierzu benötigten Funktionen zur Verfügung.
4. **Spektrogramme durchmischen**: Die Spektrogramme werden durchmischt, um die zeitliche Abhängigkeit der WAV-Dateien zu eliminieren.
5. **Spektrogramme filtern**: Spektrogramme mit einem geringen Dateninhalt werden aus dem Datensatz entfernt. Hierzu wird die Anzahl an Nullen pro Spektrogramm summiert. Falls ein definierter Schwellwert überschritten wird, wird das Spektrogramm nicht zum Datensatz hinzugefügt.
6. **Aufteilung in Trainings- und Testset**: Die Spektrogramme werden gemäss einem definierten Verhältnis in Trainings- und Testset aufgeteilt.
7. **Datensätze exportieren**: Aufgrund der hohen Datenmenge können die Daten nicht im Pickle-Format gespeichert werden. Stattdessen wird die *Save*-Funktion der Python-Bibliothek *numpy* verwendet.

3.5. Trainingsphase

Dieser Abschnitt beschreibt die Trainingsphase des CNN. Abbildung 3.6 visualisiert den Trainingsablauf.

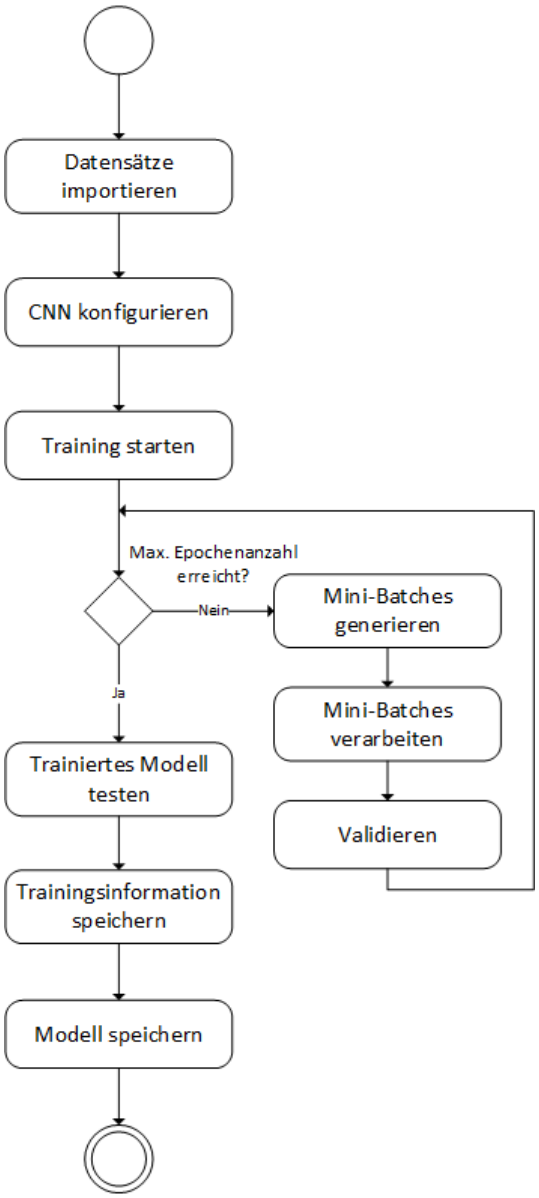


Abbildung 3.6.: Übersicht des Trainingablaufs

- **Datensatz laden**: Trainings- und Testset werden importiert.
- **CNN konfigurieren**: In diesem Schritt werden die Anzahl der zu unterscheidenden Instrumentenklassen, die Anzahl der Trainingsepochen und die Architektur des CNN definiert.
- **Training starten**: Das Training des CNN wird gestartet.
- **Mini-Batches generieren**: Die Daten aus dem Trainingsset werden in Batches aufgeteilt.
- **Mini-Batches verarbeiten**: Das Netz verarbeitet die Batches und passt die konfigurierbaren Parameter mithilfe des *Backpropagation-Algorithmus* an.

- **Validieren**: Nachdem alle Batches der aktuellen Epoche verarbeitet wurden, wird die Genauigkeit mithilfe des Validierungssets ermittelt.
- **Trainiertes Modell testen**: Das trainierte Modell wird durch das Testset evaluiert.
- **Trainingsinformationen speichern**: Die Informationen zum Training werden gespeichert. Hierzu gehören Accuracy-, Train-Loss- und Validation-Loss-Plots, die Modellkonfiguration und die Trainingsaufzeichnung.
- **Trainiertes Modell speichern**: Abschliessend kann das trainierte Modell im Pickle-Format gespeichert werden.

Batch-Iterator

Für die Batch-Generierung wurden zwei eigene Iteratoren implementiert. Der *Train-Batch-Iterator* extrahiert aus den zehnsekündigen Spektrogrammen eines Batches zweisekündige Ausschnitte einer zufälligen Position. Durch diese Zufälligkeit steigt die Trainingsdatenmenge und gleichzeitig sinkt die Wahrscheinlichkeit, dass ausschliesslich Merkmale der Trainingsdaten gelernt werden. Die Spektrogramme des Batches werden ebenfalls zufällig gewählt, mit dem Ziel dass möglichst alle Instrumentenklassen in einem Batch vorkommen. In einer Epoche wird somit nur ein Ausschnitt der Trainingsdaten verarbeitet.

Beim *Test-Batch-Iterator* werden hingegen aus einem zehnsekündigen Spektrogramm der Reihe nach zweisekündige Ausschnitte extrahiert und zum aktuellen Batch hinzugefügt, damit jede Epoche exakt gleich validiert wird und Aussagen über das Lernverhalten des CNN möglich sind. Anders als beim Train-Batch-Iterator werden pro Epoche die Daten des gesamten Validierungssets verarbeitet.

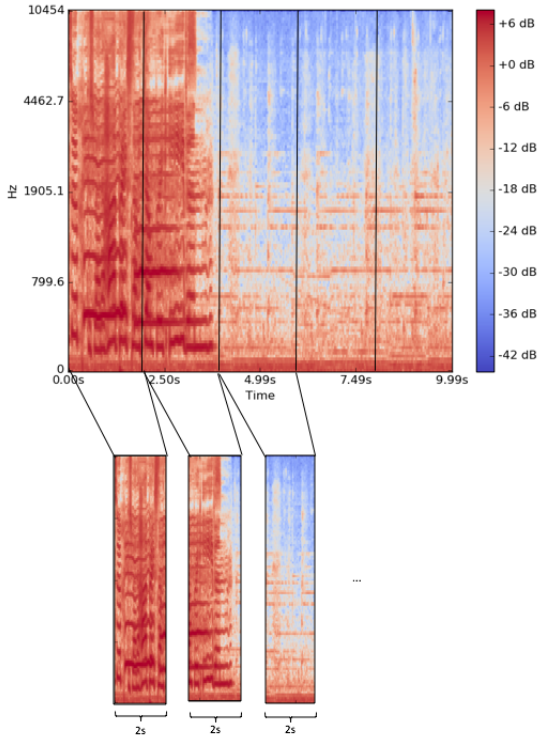


Abbildung 3.7.: Batch-Generierung mittels Test-Batch-Iterator am Beispiel einer Aufnahme von weiblichem Gesang

3.6. Implementationsdetails

In diesem Kapitel wird auf ausgewählte Details des implementierten Systems eingegangen.

Datenstruktur

Im Kontext des überwachten Lernens werden Merkmale üblicherweise in einem Vektor `X` und die entsprechenden Labels in einem Vektor `y` gespeichert. `X` beinhaltet folglich die Spektrogramme und `y` für jeden Eintrag die entsprechende Instrumentenklasse als Zahl (vgl. Abbildung 3.8).

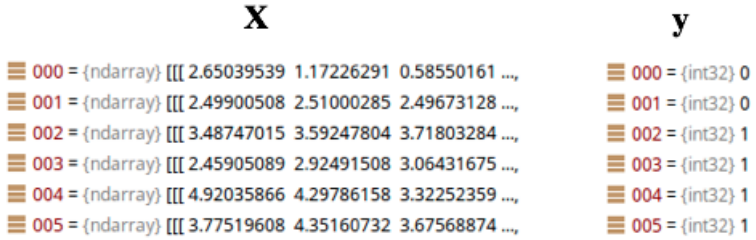


Abbildung 3.8.: Daten- und Label-Vektoren

Voraussage einzelner Instrumentenspuren

Nolearn stellt Funktionen für die Voraussage von unbekannten Daten zur Verfügung. Die Funktion `net.predict_proba(X)` gibt für jedes Spektrogramm eine Wahrscheinlichkeitsverteilung der vorausgesagten Instrumentenklassen. Abbildung 3.9 zeigt die Verteilung einer möglichen Voraussage eines Spektrogramms. Bei einem nicht eindeutigen Resultat können mithilfe der Wahrscheinlichkeiten Massnahmen getroffen werden.

Die Funktion `net.predict(X)` gibt die Instrumentenklasse mit der höchsten Wahrscheinlichkeit zurück.

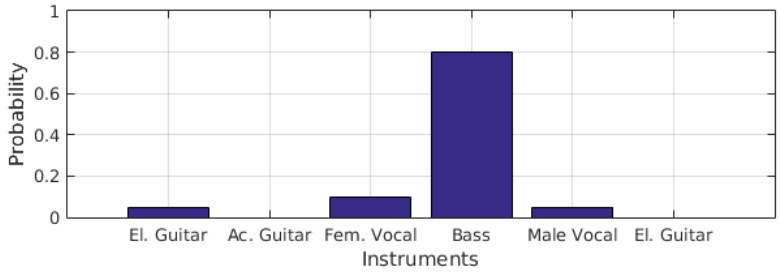


Abbildung 3.9.: Wahrscheinlichkeitsverteilung einer Voraussage

Konfigurationsobjekte und Trainingsreihen

Ein Konfigurationsobjekt beinhaltet die Parameter einer Trainingseinheit und wird als *Dictionary* gespeichert. Beim Start eines Trainingsdurchlaufs wird ein solches Konfigurationsobjekt mitgegeben und beinhaltet die folgenden Daten:

- Netzarchitektur
- Epochenanzahl
- Instrumentenklassen
- Flag für die Speicherung der Plots
- Flag für die Speicherung des trainierten Modells
- Experimentname
- Datensatzgrösse

```
default_config = {
    'net': [
        # convolution layers
        # ...
        # dense layers
        # ...
        # output layer
        # ...
    ],
    'epochs': 2000,
    'instruments': ['vocals_male', 'acoustic_guitar', 'electric_guitar',
                    'overhead', 'vocals_female', 'bass'],
    'save_plots': True,
    'save_model': False,
    'experiment_name': 'filter_size',
    'set_size': 1.0
}
```

Abbildung 3.10.: Beispiel eines Konfigurationsobjekts (in Python)

Die Dauer eines Trainingsdurchlaufs ist abhängig von den Parametern im Konfigurationsobjekt und kann bis zu mehreren Stunden in Anspruch nehmen. Um Experimentreihen mit unterschiedlichen Konfigurationen effizient durchführen zu können, wurde ein Script implementiert, welches mehrere Trainingsdurchläufe mit den entsprechenden Konfigurationsobjekten startet und die Resultate strukturiert abspeichert. Zudem wurde eine E-Mail-Funktion implementiert, welche nach jedem abgeschlossenem Trainingsdurchlauf signalisiert, dass die Resultate eingesehen werden können, und dass der nächste Trainingsdurchlauf der Experimentreihe automatisch gestartet wird.

4. Experimentelle Untersuchungen

In diesem Kapitel wird die Performance des Systems mithilfe verschiedener Experimente analysiert, mit dem Ziel, das System optimal für die Problemstellung zu parametrisieren. Dabei werden sowohl Modell- als auch Trainingseigenschaften untersucht, ausgewertet und gegebenenfalls angepasst. Die Experimente wurden, wenn nicht anders vermerkt, jeweils mit 5'000 Epochen mit dem gesamten Datensatz durchgeführt. Die untersuchte Messgrösse ist die Genauigkeit (engl. Accuracy), welche die Anzahl korrekter Klassifikationen geteilt durch die Gesamtzahl der Klassifikationen angibt.

4.1. Performance der Ausgangsarchitektur

In den ersten Experimenten wurde untersucht, ob sich die Ausgangsarchitektur des CNN für die Klassifikation von Musikinstrumenten mit Mel-Spekrogrammen als Eingabedaten eignet und wie hoch die Accuracy ist. Dazu wurden in einem ersten Schritt Experimente mit jeweils zwei ähnlich und zwei unterschiedlich klingenden Instrumenten durchgeführt, welche erste Aussagen über die Performance des Systems ermöglichen. Im zweiten Schritt wurden Experimente mit allen sechs Instrumentenklassen durchgeführt.

Unterschiedlich klingende Instrumente

Die Wahl der Instrumente in diesem Experiment beschränkte sich auf Bass und weiblichen Stimme, welche für den Menschen einfach zu unterscheiden sind und somit auch eine höhere Erkennungsrate als bei ähnlich klingenden Instrumenten zu erwarten ist.

Im Loss- und Accuracy-Plot in Abbildung 4.1 ist ersichtlich, dass das System in der Lage ist die zwei Instrumente korrekt zu klassifizieren. Sowohl der Validierungsfehler (engl. Validation-Loss) als auch der Trainingsfehler (engl. Training-Loss) nehmen mit der Anzahl Epochen ab, wobei letzteres grosse Schwankungen aufweist, was mit hoher Wahrscheinlichkeit an den zufälligen Daten pro Epoche, verursacht durch den *Train-Batch-Iterator*, zurückzuführen ist.

Auf dem Validierungsset wurde in den letzten 3'000 Epochen durchschnittlich eine Accuracy von 99.5% erreicht. Das trainierte System erreichte mit dem Testset eine Accuracy von über 99.6%. Die falsch klassifizierten Daten wurden im Nachhinein untersucht wobei festgestellt wurde, dass die Mehrheit der betroffenen zweisekündigen Spektrogramme fehlerhaft ist. Beispielsweise waren fremde Instrumente oder nicht erkennbare Geräusche im Signal enthalten. In wenigen Fällen wurden hohe Basstöne als weibliche Stimme klassifiziert.

Die Resultate zeigen, dass Mel-Spektrogramme als Eingabedaten für die Klassifikation geeignet sind und dass die Ausgangsarchitektur bereits eine sehr hohe Accuracy erzielt.

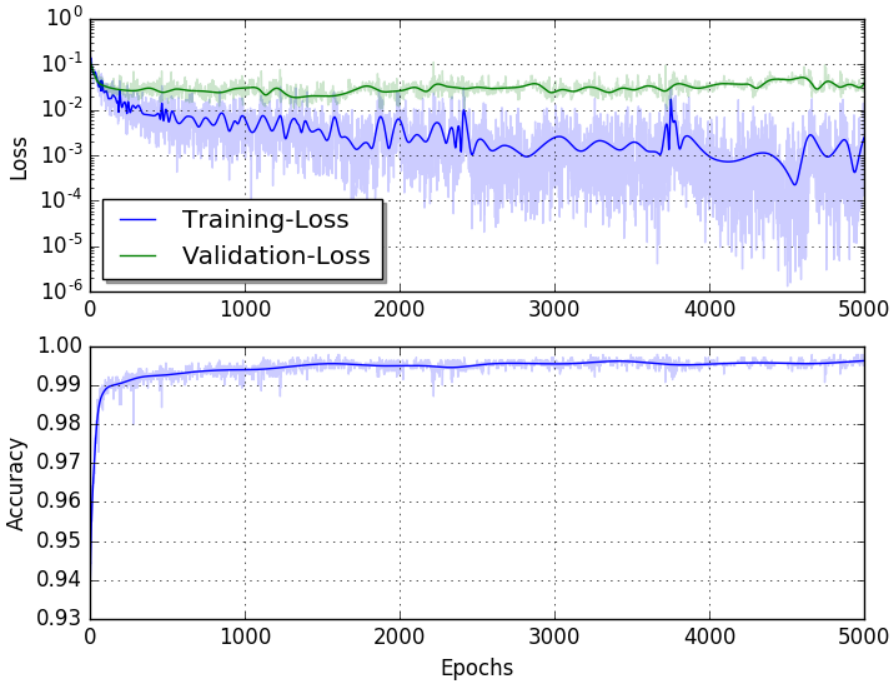


Abbildung 4.1.: Trainings-, Validierungsfehler und Accuracy des ersten Experiments mittels Bass und weiblichen Stimme

Ähnlich klingende Instrumente

Die Performance des Systems wurde in zwei Experimenten mittels ähnlich klingenden Instrumenten untersucht. Für das erste Experiment wurden die Daten der akustischen und elektrischen Gitarre verwendet. Im zweiten Experiment wurde die Klassifikation der weiblichen und männlichen Stimme analysiert. Das Resultat sollte erwartungsgemäss eine geringere Accuracy als beim Experiment mit unterschiedlich klingenden Instrumenten erreichen, da eine Unterscheidung der gewählten Instrumente je nach Spielweise und Variation auch für den Menschen ein schwieriges Problem darstellt.

Instrumente		**Accuracy**
Akustische Gitarre	Elektrische Gitarre	99.6%
Weibliche Stimme	Männliche Stimme	96.1%

Tabelle 4.1.: Testset-Accuracy mit ähnlich klingenden Instrumenten

Die Ergebnisse zeigen, dass für beide Experimente eine hohe Accuracy von über 95% erreicht wird. Die beiden Stimmen sind für das System schwieriger zu klassifizieren als es mit der akustischen und elektrischen Gitarre der Fall ist. Im Accuracy-Plot (siehe Abbildung 4.2) ist zudem ersichtlich, dass bei den Gitarren von Anfang an eine höhere Accuracy erreicht wird, was darauf hinweist, dass die Merkmale schneller gelernt werden. Im Gegensatz dazu werden die Merkmale der beiden Stimmen nur langsam gelernt, wobei erst nach ca. 3'000 Epochen eine konstante Accuracy erreicht wird.

Die Untersuchung der falsch klassifizierten Daten des Testsets ergab, dass mehrheitlich elektrische Gitarren als akustische Gitarren klassifiziert wurden, insbesondere wenn sie unverzerrt gespielt wurden. In einem Fall wurden auf der elektrischen Gitarre Flageolett-Töne gespielt, welche als Töne einer akustischen Gitarre interpretiert wurden und auch vom Menschen nur schwierig zu unterscheiden sind.

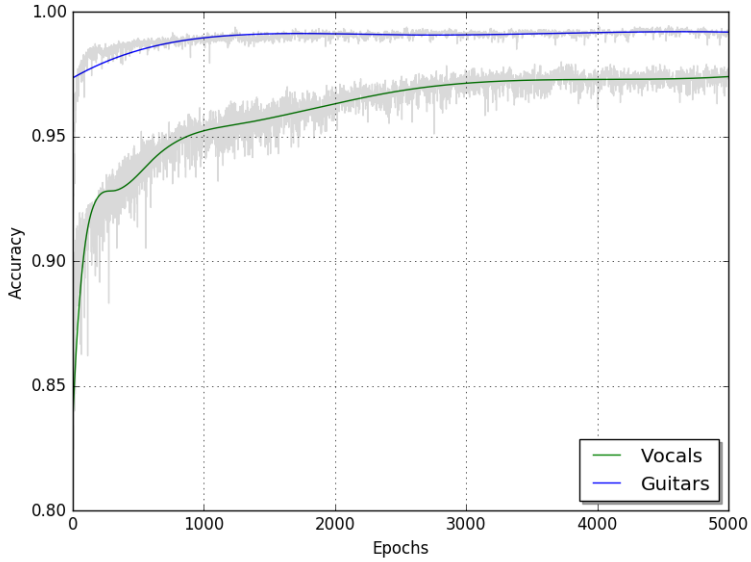


Abbildung 4.2.: Validation-Accuracy im Experiment mit ähnlich klingenden Instrumenten

Bei den falsch klassifizierten Stimmen wurden diverse fehlerhafte Daten entdeckt, z. B. waren oft ausschliesslich Instrumente einer anderen Klasse, Störgeräusche oder Ruhephasen hörbar. In anderen Fällen waren Mischungen von Stimme und Instrumenten vorhanden, die das System falsch klassifizierte. Zum Beispiel enthielt der Datensatz Musikspuren, in denen gleichzeitig gesungen und im Hintergrund Klavier gespielt wurde.

Die Experimente bestätigen, dass das System sowohl unterschiedlich als auch ähnlich klingende Instrumente robust unterscheiden kann.

Gesamter Datensatz

In diesem Experiment wurde das CNN mit allen sechs Instrumentenklassen trainiert und die Resultate ausgewertet.

Die Auswertung zeigt, dass das System auch mit einer grösseren Menge an Instrumenten eine hohe Accuracy von 97.5% auf dem Testset und eine durchschnittliche Accuracy von 97.2% in den letzten 2'000 Epoche auf dem Validierungset erreicht. In Abbildung 4.3 ist ersichtlich, dass ab der Epoche 2'000 eine konstante Validation-Accuracy erreicht wird. Die Confusion-Matrix 4.2 fasst die Klassifikationsresultate des Experiments zusammen. Die Werte sind auf zwei Nachkommastellen gerundet.

Instrument	Klassifiziert als					
	a	b	c	d	e	f
a: Schlagzeug	99.63	0.24	0	0	0	0.12
b: akustische Gitarre	0	99.44	0.55	0	0	0
c: elektrische Gitarre	0	0	98.57	0.95	0.23	0.47
d: Bass	0	0.68	2.94	96.37	0	0
e: Weibliche Stimme	0.11	0	0.33	0	95.96	3.59
f: Männliche Stimme	0.33	0.33	0.79	0	4.41	94.12

Tabelle 4.2.: Confusion-Matrix der Klassifikationsresultate mit dem Testset (in Prozent)

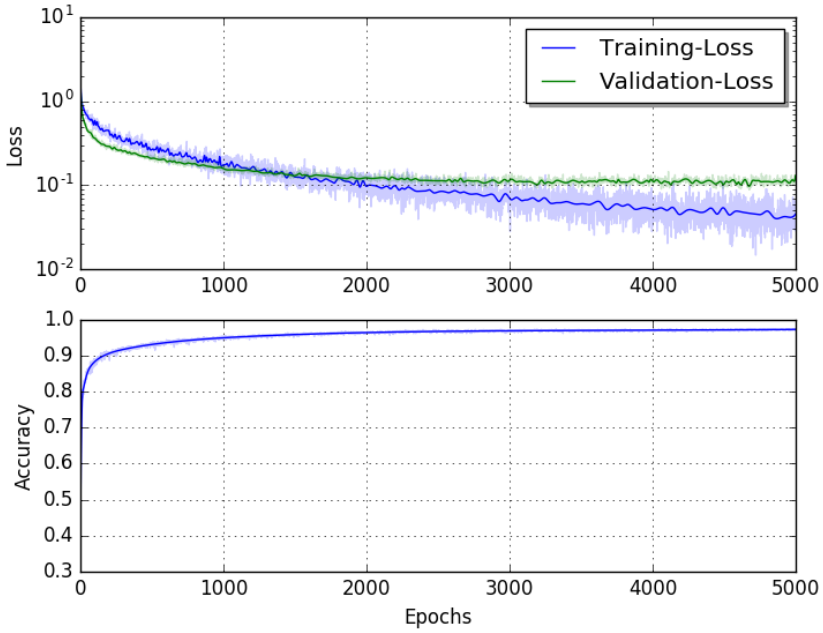


Abbildung 4.3.: Trainings-, Validierungsfehler und Accuracy im Experiment mit dem gesamten Trainingsset

Die Untersuchung ergab, dass das System insbesondere Schwierigkeiten bei der Unterscheidung von männlicher und weiblicher Stimme hat, was in einem früheren Experiment bereits erkannt wurde. Ein falsch klassifiziertes Bass-Segment, in welchem Anweisungen des Toningenieurs zu hören sind, wurde erstaunlicherweise korrekt als männliche Stimme erkannt. In der weiteren Analyse der falsch klassifizierten Daten wurden Bassspuren, in denen hohe Töne gespielt werden, als elektrische Gitarre klassifiziert. In einem anderen Fall wurde in einer Instrumentenspur gleichzeitig Bass und Gitarre gespielt, was vom System als elektrische Gitarre gewertet wurde.

Die Auswertung der Resultate zeigt, dass das System mit einer hohen Genauigkeit von über 97% die sechs Instrumente korrekt klassifiziert. Viele der Fehler sind qualitätsbedingt und andere sind gar für den Menschen schwierig zu klassifizieren.

4.2. Optimale Segmentlänge

In der Literatur beträgt die Segmentlänge der Eingabedaten für das CNN häufig zwischen 1 und 3 Sekunden, beispielsweise in [3], [15], [36].

Bei einer zu kurzen Segmentlänge könnten relevante temporale Merkmale verloren gehen, da nur kurze Ausschnitte betrachtet werden. Zudem steigt die Wahrscheinlichkeit, dass ganze Segmente ausschliesslich fehlerhafte Daten, z. B. Störgeräusche, beinhalten und somit falsch klassifiziert werden. Bei langen Segmenten könnten sich zu viele Variationen innerhalb eines Segments, z. B. der Übergang einer Strophe in ein Refrain, negativ auf das Klassifikationsergebnis auswirken. Ein weiterer Nachteil von zu langen Segmenten ist, dass entsprechend viel Audiomaterial vorhanden sein muss, was bei einem Soundcheck nicht garantiert ist. Aus diesem Grund wurden Experimente mit verschiedenen Segmentlängen durchgeführt. Für die Segmentlänge wurden die Werte 1, 2, 4, 5 und 8 Sekunden untersucht.

Auswertung

Abbildung 4.4 zeigt die Validation-Accuracy der durchgeführten Experimente. Die Testset-Accuracy-Werte des trainierten Netzes sind in der Tabelle 4.3 ersichtlich. Für das Experiment wurden alle sechs Instrumentenklassen verwendet.

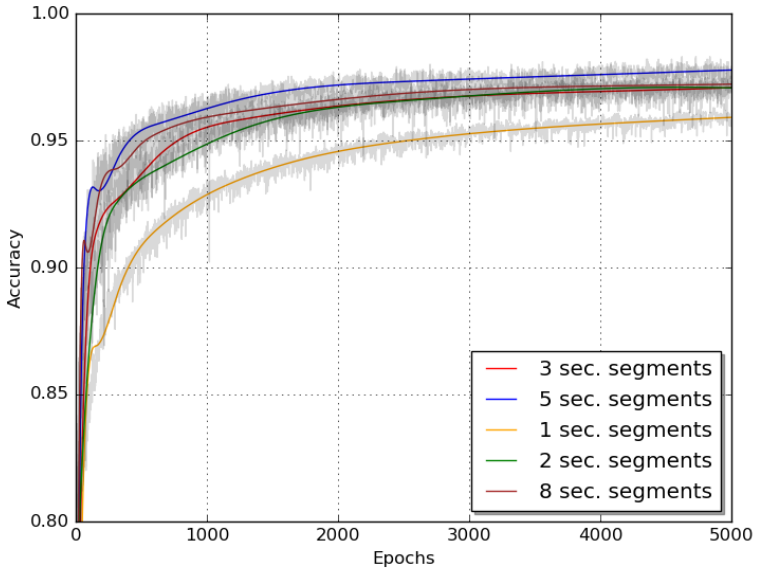


Abbildung 4.4.: Validation-Accuracy der Trainingsverläufe mit unterschiedlichen Segmentlängen

Segmentlänge	Accuracy
1 Sekunde	96.18%
2 Sekunden	97.40%
3 Sekunden	97.60%
5 Sekunden	97.80%
8 Sekunden	97.17%

Tabelle 4.3.: Testset-Accuracy der verschiedenen Segmentlängen

Die Accuracy-Werte des Testsets weisen keine grossen Unterschiede auf. Daher konnte in diesem Experiment auf keine allgemeingültige optimale Segmentlänge geschlossen werden. Der Verlauf der verschiedenen Validation-Accuracy-Werte zeigt jedoch ein schnelleres Lernverhalten während den ersten 2'000 Epochen bei einer Segmentlänge von 5 Sekunden. Nach 1'000 Epochen ist beispielsweise die Accuracy bei 5 Sekunden-Segmenten ca. 4% höher als bei 1 Sekunden-Segmente. Die langsame Lernrate bei kurzen Segmenten könnte dadurch verursacht werden, dass nicht genügend charakteristische Merkmale pro Segment vorhanden sind. Somit müssen mehr Trainingsepochen für eine vergleichbare Accuracy durchgeführt werden. Für die folgenden Experimente werden auf Grund der schnelleren Lernrate Segmente einer Länge von 5 Sekunden verwendet.

4.3. Convolution- und Pooling-Dimension

In den Arbeiten von Dieleman et al. [15], [35] werden eindimensionale Convolution-Filter in Zeit-Richtung vorgeschlagen. Lukic et al. [36] hingegen verwenden zweidimensionale Filter. In diesem Experiment wurden fünf verschiedene Konfigurationen von Convolution-Filter und Pooling-Richtungen untersucht. Sowohl ein-, als auch zweidimensionale Filter, welche Zeit- und Frequenzrichtung berücksichtigen, wurden untersucht. Tabelle 4.4 zeigt die erzielte Testset-Accuracy mit den unterschiedlichen Konfigurationen.

Filterdimension	Pooling-Dimension	Accuracy
8 x 1	4 x 1	97.54%
1 x 8	1 x 4	97.37%
4 x 4	4 x 4	97.16%
8 x 8	4 x 4	97.65%

Tabelle 4.4.: Untersuchungen der Testset-Accuracy mittels unterschiedlichen Convolution-Filter- und Pooling-Dimensionen (Zeit x Frequenz)

Die Ergebnisse zeigen, dass die Richtung und Dimension der Filter und des Pooling mit den untersuchten Konfigurationen die Resultate nicht stark beeinflussen. Die Accuracy-Plots weisen eine ähnliche Lernrate auf. Für die weiteren Experimente wird aus diesem Grund weiterhin die Konfiguration der Ausgangsarchitektur (8 x 1, 4 x 1) verwendet.

4.4. Trainingsdatenmenge

Feature-Learning-Verfahren erfordern eine grosse Datenmenge. Um die relevanten Merkmale aus den Daten zu lernen, müssen möglichst viele Variationen der einzelnen Klassen im Trainingsset vorhanden sein. Die Menge der erforderlichen Daten ist sehr schwierig einzuschätzen, daher wurde in diesem Experiment die Veränderung der Klassifikationsergebnisse bei sinkender Trainingsdatenmenge untersucht.

Auswertung

In der Tabelle 4.5 ist ersichtlich, dass eine niedrige Datenmenge zu einer deutlich geringeren Testset-Accuracy führt. Dies bestätigt die Annahme, dass eine grosse Datenmenge zu besseren Ergebnissen führt und weist darauf hin, dass eine bestimmte Trainingsdatenmenge erforderlich ist, um genügend gute Merkmale für die Klassifikation zu lernen. Abbildung 4.5 zeigt, dass die Lernrate bei allen drei Trainingsdurchläufe ab ca. 300 Epochen nicht weiter steigt.

Datenmenge	Dauer pro Instrument	Accuracy
50%	47 Minuten	31.66%
75%	70.5 Minuten	57.08%
100%	94 Minuten	97.54%

Tabelle 4.5.: Testset-Accuracy mit verschiedenen Datenmengen

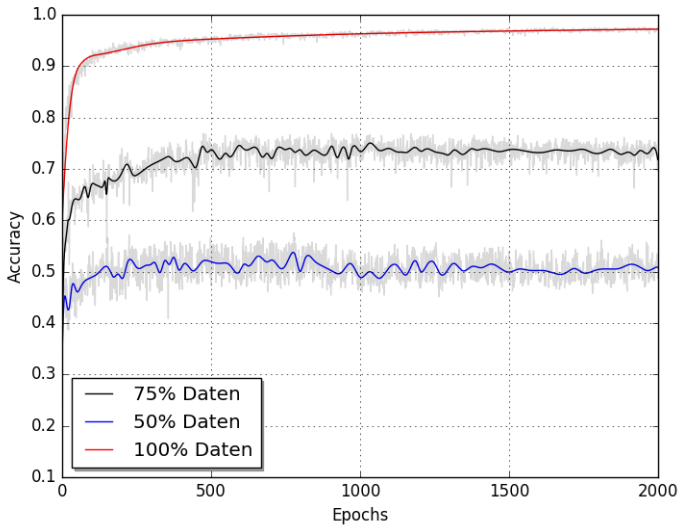


Abbildung 4.5.: Validation-Accuracy der Trainingsverläufe mit unterschiedlichen Datenmengen

Im Training- und Validation-Loss-Plot in Abbildung 4.6 ist ersichtlich, dass der Validierungsfehler ca. ab Epoche 500 zu steigen beginnt, während der Trainingsfehler weiterhin abnimmt. Dieses Verhalten wird als Überanpassung (engl. Overfitting) bezeichnet und tritt auf, wenn sich das Modell zu stark an die Trainingsdaten anpasst (vgl. [37]). Die Validierungsdaten dienen bekanntlich der epochenweisen Überprüfung des Lernverhaltens und verändern das Modell nicht. Daher lernt das Modell in diesem Falle nicht die für das Problem relevanten Merkmale, sondern versucht einzig den Trainingsfehler zu reduzieren. Um diesem Phänomen vorzubeugen sind unterschiedliche Lösungsansätze bekannt, beispielsweise das rechtzeitige Abbrechen des Trainings oder die Einführung von so genannten Dropout-Schichten, in welchen zufällige Gewichte des Netzes verändert werden (vgl. [38]).

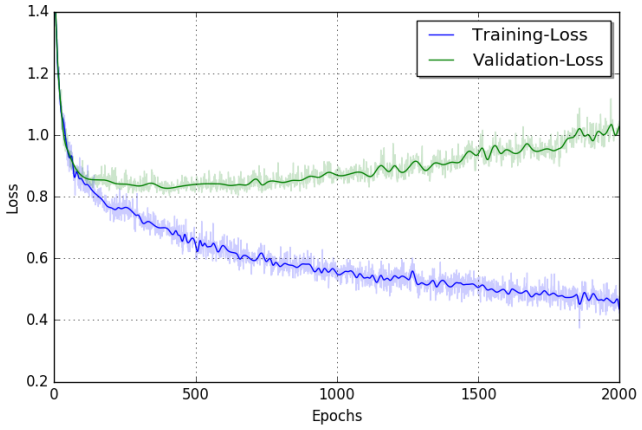


Abbildung 4.6.: Trainings- und Validierungsfehler mit 50% der Datenmenge

Die Auswertung zeigt die Wichtigkeit der Trainingsdatenmenge bei Feature-Learning-Verfahren. Mit dem kompletten Datensatz erreicht das System eine Testset-Accuracy von ca. 97%. Mit 75% der Datenmenge sinkt die Accuracy bereits auf unter 60% (vgl. Tabelle 4.5). Bei einer grösseren Datenmenge ist anzunehmen, dass sich die Ergebnisse weiter verbessern würden.

5. Resultate

In diesem Kapitel werden die Klassifikationsergebnisse des implementierten Systems evaluiert und interpretiert.

5.1. Evaluation mithilfe eines zweiten Testsets

Die Daten aus dem Testset, welches für die Evaluation des Systems verwendet wurden, stammen aus denselben Quellen wie die Trainingsdaten. Bei der Datenaufbereitung (Kapitel 3.4) wurden die Rohdaten des Datensatzes eingelesen und nach der Spektrogrammerzeugung zufällig durchmischt. Dies impliziert, dass verschiedene Teile einzelner Aufnahmen sowohl im Trainings- als auch im Testset enthalten sind, was die Ergebnisse verfälschen könnte. Um die allgemeine Performance des Systems anhand komplett unbekannter Daten zu überprüfen, wurde ein zweites kleineres Testset aus unterschiedlichen Quellen zusammengestellt. Tabelle 5.1 listet die Gesamtdauer pro Instrumentenklasse auf.

Instrument	**Dauer [min]**
Acoustic Guitar	14.8
Electric Guitar	13.1
Bass	13.1
Drums	10.8
Vocals Male	4.0
Vocals Female	1.9

Tabelle 5.1.: Gesamtdauer pro Instrumentenklasse des zweiten Testsets

Die Aufnahmen des zweiten Testsets beinhalten sehr unterschiedliche Spielarten, Genres und Aufnahmequalitäten und sind dadurch deutlich vielfältiger als diejenigen des Trainingssets. Bei einem gewöhnlichen Soundcheck ist ebenfalls mit extremen Variationen der einzelnen Instrumenten zu rechnen, daher ist es sinnvoll das System unter ähnlichen Bedingungen zu evaluieren. Tabelle 5.2 zeigt die erreichte Accuracy mit den beiden Testsets.

Testdaten	**Accuracy**
Normales Testset	97.2%
Zweites Testset	90.7%

Tabelle 5.2.: Accuracy der beiden Testsets

Das System ist in der Lage auch komplett unbekannte Daten zu klassifizieren. Dies bedeutet, dass die richtigen Merkmale während des Trainings gelernt wurden. Die Confusion-Matrix 5.3 fasst die Klassifikationsresultate des zweiten Testsets zusammen. Die Werte sind auf zwei Nachkommastellen gerundet.

Bei der Analyse der falsch klassifizierten Daten fällt auf, dass viele der Basssegmente, die als elektrische Gitarre klassifiziert wurden, verzerrt sind. Die Unterscheidung ist auch für den Menschen schwierig, da die beiden Instrumente auf diese Weise sehr ähnlich klingen. Dies weist darauf hin, dass die Bassaufnahmen der Trainingsdaten nicht genügend vielfältig sind. Mit zusätzlichen Trainingsdaten, welche ein möglichst breites Spektrum der Bassvariationen abdecken, sind bessere Resultate anzunehmen.

Instrument	Klassifiziert als					
	a	b	c	d	e	f
a: Schlagzeug	99.18	0	0	0	0	0.81
b: akustische Gitarre	0	95.93	3.48	0.58	0	0
c: elektrische Gitarre	0	3.35	88.59	1.34	0	6.71
d: Bass	5.48	1.37	8.90	84.24	0	0
e: Weibliche Stimme	0	0	0	0	66.66	33.33
f: Männliche Stimme	10.87	0	2.17	0	0	86.96

Tabelle 5.3.: Confusion-Matrix der Klassifikationsresultate mit dem zweiten Testset (in Prozent)

Alle Segmente einer Spur der Klasse *männliche Stimme* wurden als Schlagzeug erkannt, obwohl auf der Spur ausschliesslich männlicher Gesang zu hören ist. Dies könnte auf die schlechte Qualität der Aufnahme zurückzuführen sein, welche sehr grelle Klänge aufweist, die im Spektrogramm in Form von hohen Frequenzen ersichtlich sind. 6% der Segmente der Klasse *elektrische Gitarre* wurden fälschlicherweise als männliche Stimme erkannt. Eine genauere Untersuchung zeigte, dass in vielen dieser Segmenten einzelne Gitarrentöne gespielt werden. Es ist denkbar, dass in den Trainingsdaten mehrheitlich Gitarrenakkorde vorkommen, die als Begleitung der Musikstücke dienen. Somit erkennt das System andere Spielarten dieses Instruments nicht genügend zuverlässig.

Von der Klasse *weibliche Stimme* sind deutlich weniger Testdaten vorhanden als von den anderen Instrumenten, daher sind allgemeine Aussagen über die Gründe der falschen Klassifikation schwierig (vgl. Tabelle 5.1).

Viele der falsch klassifizierten Daten beruhen auf Datenqualitätsproblemen. Es ist wahrscheinlich, dass Spektrogramme anfällig auf die Aufnahmequalität sein könnten und dadurch die Klassifikationsergebnisse beeinflusst werden. Es ist anzunehmen, dass mit einer grösseren Menge an Trainingsdaten, welche die in Kapitel 2.4 beschriebenen Variationen berücksichtigen, bessere Ergebnisse erzielt werden könnten.

5.2. Resultate vergleichbarer Arbeiten

Die drei meist genutzten Datensätze für MIR Problemstellungen sind die *McGill University Master Samples*, die *University of Iowa Musical Instrument Samples Database*, und die *RWC Database*, wobei bisher keines dieser Datensätze als Standard festgelegt wurde. Demzufolge verwenden, in der Literatur vorgeschlagene Verfahren, unterschiedliche Daten, was einen direkten Vergleich der erzielten Ergebnisse erschwert (vgl. [39]).

Guignard und Kehoe [14] untersuchten 2015 die Klassifikation mittels spektraler Merkmale und SVMs von den Instrumenten akustische- und elektrische Gitarre, elektrischer Bass, Schlagzeug, Bass Drum, Snare Drum Cymbals und Hi-Hat in monophonen Audiosignalen und erreichten eine Genauigkeit von 93.1%. Dabei wurde ein eigener deutlich kleinerer Datensatz von ungefähr 20 Minuten pro Instrument verwendet. Lostanlen und Cella [22] erreichten eine Genauigkeit von 74% bei der Erkennung von acht verschiedenen Instrumentenklassen des MedleyDB-Datensatzes. Aktuell sind keine weiteren Arbeiten bekannt, welche vergleichbare Instrumente klassifizieren.

5.3. Fazit

Die Untersuchungen zeigen, dass CNNs im Zusammenhang mit Mel-Spektrogrammen für die Erkennung von Musikinstrumenten in monophonen Audiosignalen sehr gute Ergebnisse erzielen. Das System erreicht eine Genauigkeit von über 90% auf den beiden Testsets. Eine Ausweitung des Trainingssets durch vielfältigere Daten, beispielsweise unterschiedliche Spielweisen, Genres, Instrumententypen etc. würde die Accuracy des Systems vermutlich deutlich verbessern.

6. Diskussion und Ausblick

In diesem Abschnitt werden die Ergebnisse in Hinblick auf die Aufgabenstellung aufgezeigt und mögliche weiterführende Arbeiten vorgeschlagen und empfohlen.

6.1. Integration in Audio-Mischpulte

Die Ausgabe einer Klassifikation ist ein symbolisches Label des erkannten Instruments. Diese Metainformation kann von der Mischpult-Firmware weiterverarbeitet und für die Steuerung und Konfiguration einzelner Komponenten, z. B. Kanalbezeichnung und Voreinstellungen, verwendet werden. Für die Generierung der Spektrogramme muss das System auf die Eingangssignale des Mischpults zugreifen können.

Die Quellen der Spektrogramme dieser Arbeit sind WAV-Dateien. Für eine Erkennung in 'Echtzeit' müsste ein Zusatzmodul implementiert werden, welches das Eingangssignal on-the-fly segmentiert, die Spektrogramme erzeugt und anschliessend klassifiziert. Für die Ausführung der implementierten Software sind eine Python-Umgebung und die abhängigen Bibliotheken (vgl. Kapitel 3.2 *Verwendete Hard- und Software*) erforderlich.

6.2. Ausweitung der Trainingsdatenmenge

Die Beschaffung von Instrumentensamples erwies sich als besonders aufwändig, da heute keine standardisierte Datensätze für MIR-Problemstellungen vorhanden sind. Die aktuellen Trainingsdaten sind nicht vollständig und erweiterbar. Mit vielfältigeren Trainingsdaten, welche ein möglichst breites Spektrum der Instrumentvariationen abdecken, könnten mit hoher Wahrscheinlichkeit bessere Klassifikationsergebnisse erzielt werden.

6.3. Ausweitung der Instrumentenklassen

Bei genügend Trainingsdaten könnte in Betracht gezogen werden, die Anzahl der Instrumentenklassen auszuweiten. Auf diese Weise könnten zusätzlich die einzelnen Schlagzeugkomponenten, z. B. Bass-Drum, Snare-Drum, Hi-Hat etc. klassifiziert werden, welche bei Rock-/Pop-Musikgruppen meistens vorhanden sind.

6.4. Weitere Systemevaluationen

Ein direkter Vergleich zwischen dem implementierten System und einem klassischen Verfahren, mit exakt denselben Daten, die in dieser Arbeit verwendet wurden, wäre für die Beurteilung der Ergebnisse sinnvoll. Als Merkmale könnten beispielsweise MFCCs verwendet werden und GMM als Klassifikationsalgorithmus. Dies könnte weitere Einsichten in die Qualität der gelernten Merkmale geben. Zudem wäre ein Kreuzvalidierungsverfahren (engl. Cross Validation) für eine genauere Evaluation des aktuellen Systems denkbar.

6.5. Analyse der gelernten Merkmale

In einer weiterführenden Arbeit wäre es besonders interessant, die vom CNN gelernten Merkmale der Convolution-Schichten zu analysieren, um eine Einsicht in die relevante Information für die Erkennung von Musikinstrumenten zu erhalten. Abbildung 6.1 zeigt die 32 gelernten Convolution-Filter der ersten Schicht, in welchen teilweise deutliche Strukturen zu erkennen sind.

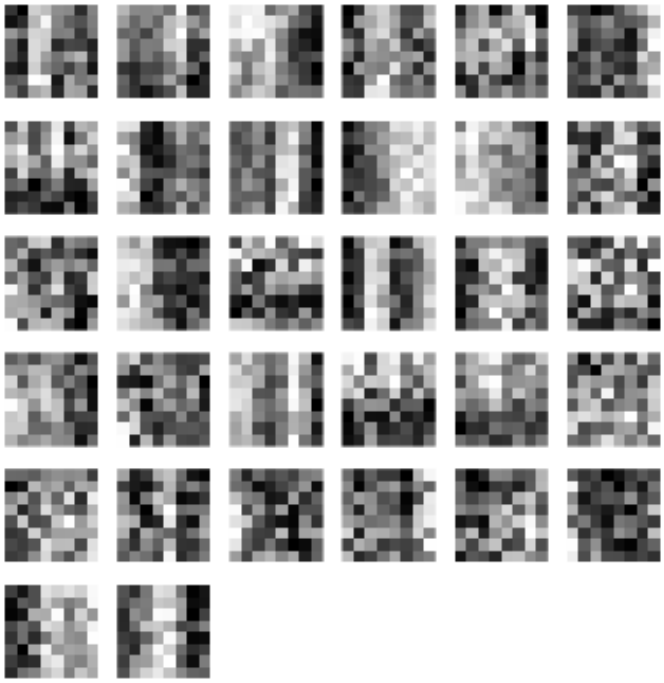


Abbildung 6.1.: Gelernte Filter der ersten Convolution-Schicht

6.6. Architekturuntersuchungen

Die gewählte Ausgangsarchitektur aus [15] erzielte in den durchgeführten Untersuchungen sehr hohe Accuracy-Werte. Es wäre denkbar, alternative Architekturen einer geringeren Komplexität für die Erkennung von Musikinstrumenten in zukünftigen Experimenten zu untersuchen.

6.7. Alternative Audiorepräsentationen

Alternativ zu Mel-Spektrogrammen könnten weitere Audiorepräsentationen als Eingabedaten für das CNN untersucht werden, beispielsweise rohe Audiodaten, Chromagramme oder Constant-Q-Spektrogramme.

7. Verzeichnisse

Literaturverzeichnis

[1] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[2] G. Eisenberg, *Identifikation und Klassifikation von Musikinstrumentenklängen in monophoner und polyphoner Musik*. PhD thesis, Technische Universität Berlin, 2008.

[3] P. Li, J. Qian, and T. Wang, "Automatic instrument recognition in polyphonic music using convolutional neural networks," *arXiv preprint arXiv:1511.05520*, 2015.

[4] I. Kaminsky and A. Materka, "Automatic source identification of monophonic musical instrument sounds," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, pp. 189–194, IEEE, Nov 1995.

[5] K. D. Martin and Y. E. Kim, "2pMU9. Musical instrument identification: A pattern-recognition approach," in *Proc. 136th meeting of ASA*, (Norfolk, USA), 1998.

[6] J. C. Brown, "Computer identification of musical instruments using pattern recognition with cepstral coefficients as features," *The Journal of the Acoustical Society of America*, vol. 105, no. 3, pp. 1933–1941, 1999.

[7] A. J. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*, pp. 753–756, 2000.

[8] J. Marques and P. J. Moreno, "A study of musical instrument classification using gaussian mixture models and support vector machines," *Cambridge Research Laboratory Technical Report Series CRL*, vol. 4, 1999.

[9] M. Eichner, M. Wolff, and R. Hoffmann, "Instrument classification using hidden markov models," in *International Symposium on Music Information Retrieval (ISMIR)*, 2006.

[10] A. Zlatintsi and P. Maragos, "Musical instruments signal analysis and recognition using fractal features," in *Signal Processing Conference, 2011 19th European*, pp. 684–688, IEEE, Aug 2011.

[11] A. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 4, pp. iv–265, IEEE, 2004.

[12] S. Essid, G. Richard, and B. David, "Musical instrument recognition on solo performances," in *Signal Processing Conference, 2004 12th European*, pp. 1289–1292, IEEE, 2004.

[13] C. Joder, S. Essid, and G. Richard, "Temporal integration for audio classification with application to musical instrument classification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 1, pp. 174–186, 2009.

[14] L. Guignard and G. Kehoe, "Learning instrument identification," *Acoustic Guitar*, vol. 146, pp. 10–0, 2015.

[15] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference*, pp. 6964–6968, IEEE, May 2014.

[16] E. J. Humphrey, J. P. Bello, and Y. Lecun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *Proc. ISMIR*, 2012.

[17] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[18] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, pp. 82–97, Nov 2012.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[20] Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *arXiv preprint arXiv:1605.09507*, 2016.

[21] T. Park and T. Lee, "Musical instrument sound classification with deep convolutional neural network using feature fusion approach," *arXiv preprint arXiv:1512.07370*, 2015.

[22] V. Lostanlen and C.-E. Cella, "Deep convolutional networks on the pitch spiral for music instrument recognition," *arXiv preprint arXiv:1605.06644*, 2016.

[23] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDB: a multitrack dataset for annotation-intensive MIR research," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014.

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, no. 4, pp. 541–551, 1989.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] Anon., "The usc-sipi image database." `http://sipi.usc.edu/database/`, n.d. [Online; accessed 6-June-2016].

[27] A. Karpathy and J. Johnson, "Cs231n: Convolutional neural networks for visual recognition." `http://cs231n.github.io/assets/cnn/maxpool.jpeg`, 2016. [Online; accessed 6-June-2016].

[28] M. Nielsen, "Neural networks and deep learning." `http://neuralnetworksanddeeplearning.com`, 2016. [Online; accessed 7-June-2016].

[29] A. Karpathy and J. Johnson, "Cs231n: Convolutional neural networks for visual recognition." `http://cs231n.github.io/assets/nn1/neural_net.jpeg`, 2016. [Online; accessed 7-June-2016].

[30] A. Karpathy and J. Johnson, "Cs231n: Convolutional neural networks for visual recognition." `http://cs231n.github.io/convolutional-networks/`, 2016. [Online; accessed 7-June-2016].

[31] Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *arXiv preprint arXiv:1605.09507*, 2016.

[32] Anon., "How do you choose your convolutional neural net architecture?." `https://groups.google.com/forum/#!topic/theano-users/zZws4T0_geU`, 2014. [Online; accessed 8-June-2016].

[33] O. Gillet and G. Richard, "Enst-drums: an extensive audio-visual database for drum signals processing," in *International Society for Music Information Retrieval Conference (ISMIR 2006)*, 2006.

[34] L. Brown, "Accelerate machine learning with the cudnn deep neural network library." `https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/`, 2014. [Online; accessed 1-June-2016].

[35] S. Dieleman, "Recommending music on spotify with deep learning." `http://benanne.github.io/2014/08/05/spotify-cnns.html`, 2014. [Online; accessed 30-May-2016].

[36] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks." unpublished, 2016.

[37] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[39] T. Li, M. Ogihara, and G. Tzanetakis, *Music data mining*. CRC Press, 2011.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Begriff	**Bedeutung**
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FFT	Fast Fourier Transformation
GMM	Gaussian Mixture Model
GPU	Graphics Processor Unit
HMM	Hidden Markov Model
IDMT	Fraunhofer Institute for Digital Media Technology
k-NN	K-Nearest-Neighbors
LDA	Linear Discriminant Analysis
LSF	Line Spectral Frequencies
MFCC	Mel-Scale Frequency Cepstrum Coefficients
MIR	Music Information Retrieval
MRP	Multiresolution Recurrent Points
PCA	Principal Component Analysis
RGB	Rot Grün Blau - Farbkanäle
RNN	Recurrent Neural Networks
SVM	Support Vector Machines
WAV	Wave-Dateiformat

A. Anhang

A.1. Projektmanagement

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Instrumentenerkennung in Single-Source Audio Streams

BA16_stdm_5

BetreuerInnen:	Thilo Stadelmann, stdm Philipp Ackermann, acke
Fachgebiete:	Datenanalyse (DA) Digitale Signalverarbeitung (DSV) Software (SOW)
Studiengang:	IT
Zuordnung:	Institut für angewandte Informationstechnologie (InIT)
Gruppengrösse:	2

Kurzbeschreibung:
Die Firma Studer Professional Audio GmbH aus Regensdorf/ZH ist ein führender Hersteller professioneller Audiomischpulte für den Broadcasting-Bereich (Radio, Fernsehen etc.) mit dem Anspruch höchster Qualität und Innovation. Ein digitales Studer-Mischpult hat ca. 5'000 Audio-Ein- und Ausgänge (Kanäle), an denen jeweils einzelne Audioquellen angeschlossen sind (etwa ein Mikrofon oder einzelnes Instrument).

Zur Unterstützung des Toningenieurs beim initialen Soundcheck ist eine Technologie gesucht, welche die an jedem Kanal anliegende Audioquelle anhand des Eingangssignals live erkennt (z.B. "Snaredrum", "E-Gitarre", "Flöte") und diese Klassifikation an das Mischpult zurückgibt, so dass dieses den Kanal beispielsweise anhand eines Templates (Kanalname, Gain, Equalizer-Einstellungen, Aux-Mix etc.) grob voreinstellen kann.

Inhalt:

- Ziel: Implementierung eines prototypischen Softwaremoduls, welches die an einem Audioeingang eines digitalen Mischpults anliegende Signalquelle erkennt und diese Information dem Mischpult live bereitstellt
- Recherchieren und Darstellen des wissenschaftlichen State of the Art bzgl. der Erkennung von Musikinstrumenten aus single-source Audiodaten
- Zusammenfassen relevanter wissenschaftlicher Publikationen
- Bewerten der Ansätze hinsichtlich Fitness und Machbarkeit in Bezug auf obigen Use Case
- Erstellen einer Übersicht über existierende Implementierungen, beispielsweise Open-Source Bibliotheken oder kommerzielle Toolkits
- Definieren (zusammen mit Studer) der zu erkennenden Menge von Instrumenten und Aufbau entsprechender Klassifikatoren
- Effizientes Implementieren des optimalen Ansatzes aus dem Rechercheergebnis
- Design eines Experiments (Setup, Daten, Metriken) zur Evaluation des Ergebnisses

Voraussetzungen:
Es sind keine speziellen Vorkenntnisse in Signalverarbeitung, Musikanalysie oder Maschinellem Lernen notwendig. Wir erwarten lediglich Engagement, Lust auf wissenschaftliches Arbeiten und Freude am Experimentieren.

Support durch Studer

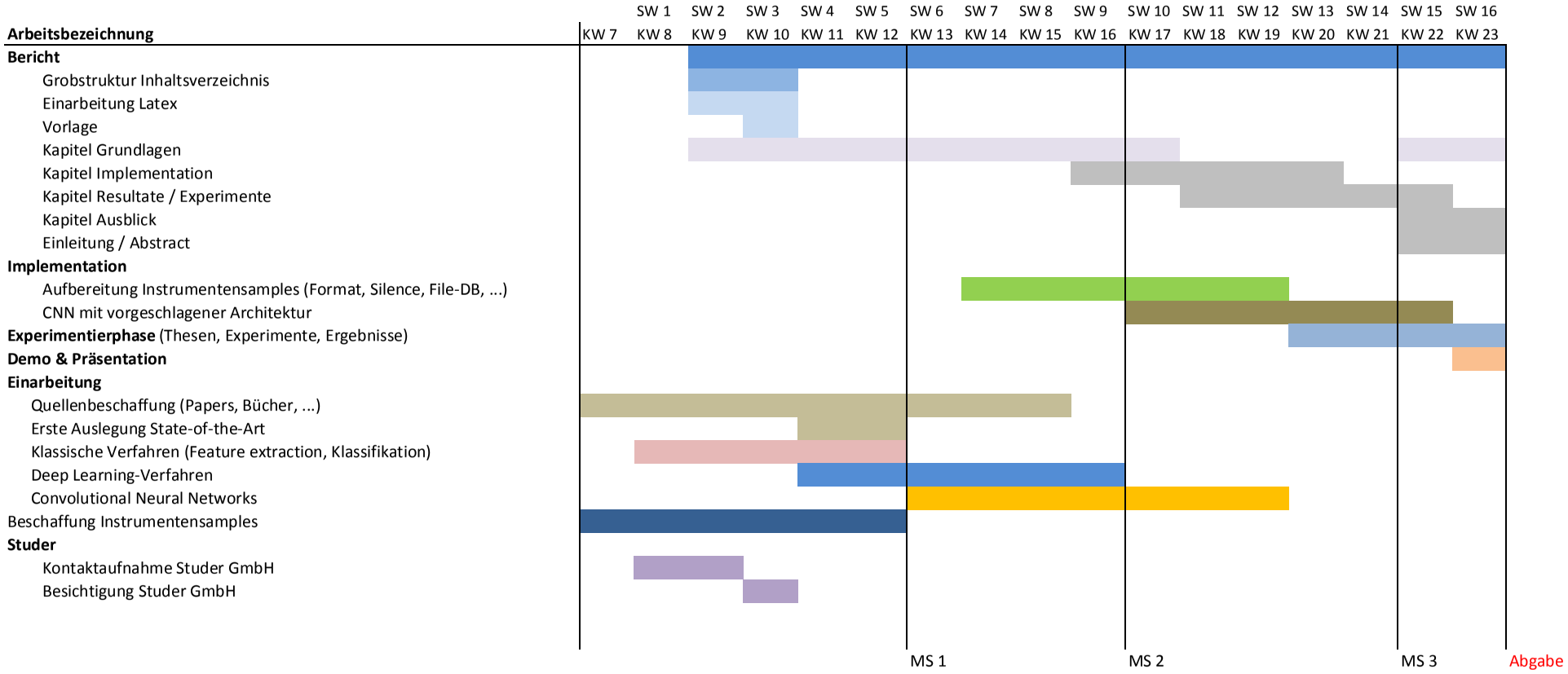
- Mitwirkung an Definition der vom Prototypen zu erkennenden Instrumentenmenge
- Bereitstellung von Audiodaten in Ausreichender Menger und Variabilität pro zu erkennendem Instrument

Die Arbeit ist vereinbart mit:
Yannick Streit (streiyan)
Leotrim Zulfiu (zulfileo)

Projektplan - Bachelorarbeit

Projektstart: 15.02.2016
Projektabgabe: 10.06.2016
Projektmitglieder: Y. Streit, L. Zulfiu
Projektbetreuer: T. Stadelmann, P. Ackermann

Milestones	
MS 1	25.03.2016
MS 2	22.04.2016
MS 3	27.05.2016
Abgabe	10.06.2016



A.2. Weiteres

Inhalte auf dem Datenträger

In der folgenden Grafik ist die Ordnerstruktur des Datenträgers dargestellt.

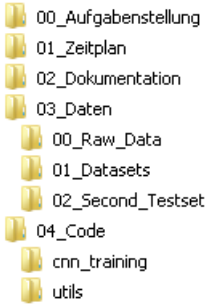


Abbildung A.1.: Ordnerstruktur auf dem Datenträger

Erläuterungen

README.txt enthält weitere Informationen zur Verwendung der Software.

- **03_Daten**
  - *Raw_Data:* Beinhaltet alle verwendeten Instrumentenspuren (ein Ordner pro Instrument)
  - *Datasets:* Enthält Trainings– und Testset normalisiert und vorbereitet im Numpy–Format (Arrays mit Spektrogrammen und Labels)
  - *Second_Testset:* Beinhaltet das zweite Testset in Rohform (ein Ordner pro Instrument, wobei die Spuren in WAV-Dateiformat abgespeichert sind)
- **04_Code**
  - *cnn_training:* Enthält die benötigten Scripts für das Trainieren des CNN
  - *utils:* Enthält unterschiedliche Scripts, beispielsweise um Plots zu generieren

Instructions for Usage

In this bachelor thesis, a system for the automatic identification of musical instruments in monophonic audio signals is implemented using a CNN with spectrograms as input.

Installation

Following dependencies are needed to run the software. Mainly the libraries Theano, Lasagne and NoLearn are required. It is recommended to create a virtual environment and start all scripts with the corresponding python interpreter.

- audioread (2.1.2)
- backports.shutil-get-terminal-size (1.0.0)
- cycler (0.10.0)
- decorator (4.0.9)
- gdbn (0.1)
- gnumpy (0.2)
- ipython (4.2.0)
- ipython-genutils (0.1.0)
- joblib (0.9.4)
- Lasagne (0.1)
- librosa (0.4.2)
- matplotlib (1.5.1)
- nolearn (0.6a0.dev0)
- numpy (1.11.0)
- pathlib2 (2.1.0)
- pexpect (4.1.0)
- pickleshare (0.7.2)
- pip (8.1.1)
- ptyprocess (0.5.1)
- pydotplus (2.0.2)
- pydub (0.16.4)
- pyparsing (2.1.1)
- python-dateutil (2.5.3)
- pytz (2016.4)
- scikit-learn (0.17.1)
- scikits.samplerate (0.3.3)
- scipy (0.17.0)
- setuptools (20.10.1)
- simplegeneric (0.8.1)
- six (1.10.0)
- tabulate (0.7.5)
- Theano (0.8.2)

- traitlets (4.2.1)
- wheel (0.29.0)

Usage

All scripts are started from the home code directory (04_Code). It is recommendable to do `export PYTHONPATH=.` since all imports in the scripts are absolute from the top `__init__.py`.

Settings

In this config file several values as the segmentation length or mini-batch size which are used in all other scripts can be adjusted.

Data Preparation

1. Set the right paths in `utils/data_preparation.py`
2. Start the script from the home code directory like `python utils/data_preparation.py` (use the corresponding python interpreter e.g. the one in the virtual environment)
3. The train- and testset can be found in the configured dataset directory

Training

1. Set the right paths in `cnn_training/cnn_training.py`
2. Start the script with from the home code directory like `python utils/data_preparation.py` (use the corresponding python interpreter)
3. After training the plots and train history are generated in the configured result directory path