JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
**Filip Korzeniowski**

Submitted at
**Institute of Computational Perception**

Supervisor and
First Examiner
**Gerhard Widmer**

Second Examiner
**Juan Pablo Bello**

September 2018

# Harmonic Analysis of Musical Audio using Deep Neural Networks

Doctoral Thesis
to obtain the academic degree of
Doktor der technischen Wissenschaften
in the Doctoral Program
Technische Wissenschaften

**JOHANNES KEPLER UNIVERSITY LINZ**
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

# Statutory Declaration

# Acknowledgments

I want to thank fortune for being born in a place and time in the world where I did not suffer war, famine, or illness; where I could enjoy a blissful childhood, was not forced to work to survive, and had access to the best education. These are the key circumstances which brought me where I am now.

None of these fortunate happenstances would matter, if I did not have the love, protection and guidance of my parents and my family. They supported me in any way they could, for which I will be forever grateful. I did not choose the family I was born into—they join the list of lucky circumstances for which I lack the words to express my gratitude for.

I was also fortunate enough to find another fellow human who helped me to drudge[1] through all these years and complete this thesis. Thank you, Kathrin, for the moral anchors you provide, for the support you offer every day, and for all that I have learned from you. This little paragraph cannot convey the appreciation you deserve; know that I cherish the privilege to have met you.

I am also indebted to my friends from the institute. They made these years a unique experience. It is rare to find a group of people who are so fun, challenging, knowledgeable, and good-hearted in one place at the same time. Without them, this thesis would not exist, and I would not be the person I am (for better or for worse).

Finally, I want to express my gratitude to my supervisor Gerhard Widmer, who enabled me to conduct research for such a long time—trusting that something will come out of it in the end—and teaching me not only how to do research, but also how to communicate it.

Thank you.

The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 670035, project "Con Espressione").

[1]In a pleasant way.

# Harmonic Analysis of Musical Audio using Deep Neural Networks

## Abstract

In this thesis, I consider the automatic extraction of harmonic information from musical audio. Obtaining such information automatically is relevant not only for theoretical analyses, but also for commercial applications such as music tutoring programs or lead sheet generators. I focus on two aspects of harmony—chords and the global key—and tackle the problem of extracting them using deep neural networks.

My work on chord recognition constitutes the main part of this thesis. To recognise chords in the audio, I first develop data-driven feature extraction methods (or, acoustic models) that outperform hand-engineered ones. I then focus on modelling chord sequences, and show that doing so on a frame-by-frame basis (as common in existing chord recognition systems) prevents learning musical relationships between chords—regardless of the complexity or power of a sequence model. I also show that such models instead need to operate on higher-level chord symbol sequences in order to be useful. I continue by systematically exploring such chord sequence models based on recurrent neural networks and show their superiority to finite-context models. Finally, I devise a probabilistic model that integrates these chord sequence models with acoustic models using various models of chord duration, and evaluate how the performance of each model influences the final chord recognition results.

The second part of this thesis concerns key classification. Here, I develop a convolutional neural network based on traditional key classification pipelines to create a key classifier that performs better than existing, hand-designed methods. I then evaluate how well the model generalises over datasets of different musical genres (a problem existing systems have not solved), and propose adaptations in training and network structure that enable learning a genre-agnostic model that outperforms genre-specific models on many available datasets.

# Harmonische Analyse von Audioaufnahmen mit Hilfe tiefer neuronaler Netze

## Kurzfassung

Diese Arbeit befasst sich mit der automatischen Extraktion harmonischer Information aus Audioaufnahmen von Musik. Automatische Methoden zur Extraktion solcher Information sind nicht nur für die theoretische Musikanalyse relevant. Auch kommerzielle Anwendungen wie Musiklernprogramme oder Generatoren von Leadsheets sind ohne solcher Methoden nicht denkbar. In der Arbeit werden zwei Aspekte von Harmonie betrachtet – Akkorde und globale Tonart –, welche mit mit Hilfe tiefer neuronaler Netze extrahiert werden.

Akkorderkennung bildet den Hauptbestandteil dieser Dissertation. Um Akkorde in Audioaufnahmen zu erkennen, werden zuerst datengetriebene Merkmalsextraktoren (sogenannte akustische Modelle) entwickelt. Danach rückt die Modellierung von Akkordsequenzen in den Fokus. Hierbei zeigt die Arbeit, dass die Modellierung auf Basis von kurzen Zeitschritten in der Zeitdomäne (wie in bestehenden Akkorderkennungssystemen üblich) das Erkennen von musikalischen Zusammenhängen verhindert. Des Weiteren wird gezeigt, dass solche Modelle direkt auf Akkordebene arbeiten müssen, um sinnvoll zu funktionieren. In weiterer Folge werden Akkordsequenzmodelle, die auf rekurrenten neuronalen Netzen basieren, systematisch evaluiert, und ihre Überlegenheit gegenüber Modellen mit endlichem Kontext gezeigt. Abschließend präsentiert die Arbeit ein probabilistisches Modell, welches die erwähnten Akkordsequenzmodelle, mit Hilfe verschiedener Modelle der Akkorddauer, mit akustischen Modellen verbindet. Hierbei wird insbesondere betrachtet, wie sich die Qualität der einzelnen Modelle auf das Endergebnis auswirkt.

Der zweite Teil dieser Arbeit beschäftigt sich mit der Klassifizierung der Tonart. Hierfür wird, basierend auf klassischen Methoden der Tonartklassifizierung, ein faltendes neuronales Netz entwickelt, welches Tonarten besser klassifizieren kann als bestehende Methoden. In weiterer Folge wird evaluiert, wie gut dieses Modell auf andere Musikrichtungen generalisiert (ein Problem, an dem bestehende Methoden scheitern), und Änderungen der Netzwerkstruktur und der Lernprozedur präsentiert, die das Erlernen eines genre-unabhängigen Modelles ermöglichen. Dieses Modell erreicht schließlich bessere Ergebnisse als auf bestimmte Musikrichtungen spezialisierte Modelle.

# Contents

# List of Figures

# List of Tables

Quid ergo est bonum? Rerum scientia. Quid malum est? Rerum imperitia.

—Seneca

# 1 Introduction

This thesis is concerned with the automatic extraction of harmonic information from musical audio signals. *Harmonic information* refers to symbols in Western music theory that designate combinations of pitch intervals between multiple notes. Here, I will distinguish two levels of abstraction: chords and key. *Chords* are local harmonic descriptors, and can be understood as the interplay of multiple note pitches. The *Key* of a musical piece (or a segment thereof) is a higher-level descriptor of its harmonic centre. It thus gives meaning to the harmonic progression of chords in a piece.

Extracting harmonic information from musical audio is key to the computational understanding of music. It describes when tension is built, and when it is released; it structures musical pieces into meaningful parts; it provides the back-drop for the content that is ostensibly important to the listener, such as melody and vocals. Thus, if we do not consider the harmonic content of a piece, our (or, the computer's) understanding of it is only superficial.

Computational harmonic analysis facilitates many practical applications. For producers of electronic music, it can find musical samples that fit well harmonically to their tracks. For musicians, it can suggest scales for improvising over a given chord progression, it can help to automatically create lead sheets for songs they want to play[1], and it can assist students in mastering their instrument[2]. Furthermore, it can serve as a building block in other tasks such as music similarity estimation or cover song identification. While keeping the practical relevance in mind, this thesis focuses on the technical task itself: building computational models that extract harmonic information (chords and key) from audio signals of music.

[1]https://chordify.net

[2]https://yousician.com

I assume the reader to be familiar with the basics of audio signal processing and machine learning, in particular artificial neural networks. Schlüter (2017) gives a much better introduction to these topics than I could provide. I recommend reading it, be it only as a refresher for someone who already knows the fundamentals.

This thesis consists of two parts. The first part, which forms the main body of the text, addresses the subject of *Chord Recognition*. The second part is concerned with *Key Classification*. I will briefly introduce both in the following; for a more thorough description, see Chapters 2 and 9.

## 1.1 Chord Recognition

The goal of chord recognition is to *segment* the audio and *label* these segments with a chord symbol. This symbol should correspond to the harmonic interpretation of an expert listener. This short description bears the imprint of subjectivity: harmonic interpretations often differ among musical experts (de Clercq and Temperley, 2011; Humphrey and Bello, 2015), which complicates the building and evaluation of chord recognition models. The reason for this is that only a subset of all pitches that are perceived to sound simultaneously are deemed relevant for the local harmony. Which subset this is, and which pitches are considered to sound simultaneously, is subject to interpretation.

The task is often called *Chord Estimation* in the literature. However, I deem the term *Recognition* to be a better fit: the target is categorical, uncertainty about the prediction is usually ignored, and the process of transforming the input into the target resembles pattern recognition systems in general, and speech recognition in particular. Indeed, chord recognition systems more often than not resemble adapted models from speech recognition. The main distinction is that in chord recognition, start and end times of the labelled segments are vital, while in speech recognition usually only the sequence of recognised words matters.

Chord recognition systems follow the scheme shown in Figure 1.1. They feature an *acoustic model* that extracts features from a context of audio, and often also predicts a chord label for the centre frame of this context. These predictions are then processed by a *temporal model*, which incorporates more temporal context and outputs homogeneous labelled chord segments. For example, many chord recognition systems are based on chroma features modelled by Gaussian mixtures as acoustic model, with a hidden Markov model as the temporal model (Cho et al., 2010).

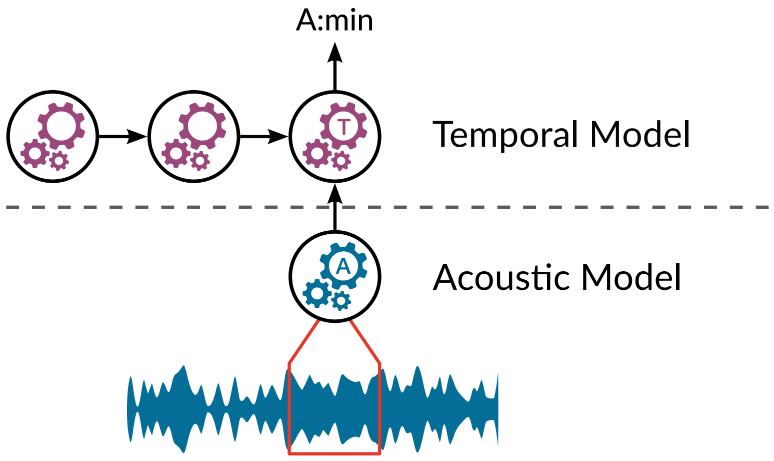


**Figure 1.1:** Overview of a chord recognition system. The acoustic model extracts features and predicts chord labels for an audio context given by a sliding window. The temporal model casts these predictions into chord segments, taking into account the temporal dimension.

## 1.2 Key Classification

The goal of key classification is to find the *global key* of a piece of musical audio. Again, this global key should be an aggregated harmonic representation over the whole piece, as interpreted by an expert listener. As in chord recognition, this is a subjective undertaking; however, to my knowledge, there are no studies that examine how this subjectivity affects computational key classification models.

In the literature, this task is commonly referred to as *Key Estimation*. However, considering the same arguments that favour the term Chord Recognition, I deem *Classification* to more precisely describe the task: given a low-level input representation, we assign a categorical label to the entirety of the input. This is, per definitionem, a classification scenario.

In this work, I focus on finding the *global* key of a piece. This is adequate for some musical styles (e.g. electronic dance music), while over-simplifying for others (e.g. classical music, where key modulations are common). Although the models I develop in this thesis can serve as a basis for systems that are able to track key modulations, I will also show in Chapter 11 that doing so will reach a glass ceiling, and argue that models considering local context will be necessary.

## 1.3 Contributions

This thesis improves the automatic extraction of harmonic information from musical audio in multiple ways. In chord recognition, its main contribution is in analysing the interplay between acoustic models (frame-wise feature extractors or classifiers) and temporal models (sequence models that cast frame-wise predictions into chord sequences). Its overall contributions to chord recognition can be summarised as follows:

i) The thesis presents acoustic models based on deep neural networks that compute interpretable features: Chapter 3 introduces a neural network that extracts chroma features, and is better able to suppress harmonically irrelevant spectral content than hand-crafted approaches; Chapter 4 presents a model for chord recognition that learns musically interpretable mid-level features.

ii) It demonstrates why most temporal models used for chord recognition are limited to simple smoothing of the predictions of the acoustic model, regardless of their capacity. It shows that the conceptual mistake that leads to this problem is that existing temporal models operate on the temporal level of audio frames, and that in order to overcome this problem, we need to learn harmonic language models on the level of chord symbols (Chapter 5).

iii) The thesis systematically evaluates language models of harmony based on recurrent neural networks, and shows that the power of such models lies in the dynamic adaptation to the song currently processed, which simpler finite-context models are unable to do (Chapter 6).

iv) It introduces a probabilistic model for chord recognition that combines acoustic models with meaningful language models of harmony and models of harmonic rhythm, and shows that considering both improves results (Chapters 7 and 8).

In key classification, the main contribution of this thesis is a machine learning model that overcomes the genre-specificity of existing approaches. In summary:

i) The thesis presents a machine-learning-based key classification system which tries to mimic traditional approaches, but is end-to-end differentiable, and thus can be trained purely from data (Chapter 10). This model beats state-of-the-art methods, but shares their downside: its performance degrades the more musical genres it is trained on.

ii) The thesis then shows that using a more general classification model based on modern neural network architectures, in conjunction with an adapted learning scheme that increases data variability, results in a model that does not suffer from this generalisation problem. Indeed, a single, general model presented in Chapter 11 out-performs other methods that are specialised to specific datasets.

## 1.4 Publications

The contributions outlined above stem from the following peer-reviewed publications:

S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. Madmom: A new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia (ACMMM)*, Amsterdam, Netherlands, Oct. 2016.

F. Korzeniowski and G. Widmer. Feature Learning for Chord Recognition: The Deep Chroma Extractor. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, Aug. 2016a.

F. Korzeniowski and G. Widmer. A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, Sept. 2016b.

F. Korzeniowski and G. Widmer. End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, Aug. 2017a.

F. Korzeniowski and G. Widmer. On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017b.

F. Korzeniowski and G. Widmer. Automatic Chord Recognition with Higher-Order Harmonic Language Modelling. In *2018 26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, Sept. 2018a.

F. Korzeniowski and G. Widmer. Genre-Agnostic Key Classification With Convolutional Neural Networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Sept. 2018b.

F. Korzeniowski and G. Widmer. Improved Chord Recognition by Combining Duration and Harmonic Language Models. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Sept. 2018c.

F. Korzeniowski, D. R. W. Sears, and G. Widmer. A Large-Scale Study of Language Models for Chord Prediction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, Apr. 2018.

# Part I

# Chord Recognition

# 2
# Chord Recognition: An Overview

## 2.1 The State of the Art

Chord Recognition methods recognise and transcribe musical chords from audio recordings, a labour-intensive task that requires extensive musical training if done manually. Since its inception by Fujishima (1999), chord recognition meets great interest (see McVicar et al. (2014) for a review), because harmonic content is a highly descriptive feature of (Western) music that forms the basis of many kinds of applications: theoretical, such as computational harmonic analysis of music; practical, such as automatic lead-sheet creation for musicians[1] or music tutoring systems[2]; and finally, as basis for higher-level tasks such as cover song identification or key classification.

What is a chord? A chord can be defined as multiple notes perceived simultaneously in harmony. This does not require the notes to be *played* simultaneously; a melody or a chord arpeggiation can imply the perception of a chord, even if intertwined with out-of-chord notes. Through this perceptual process, the identification of a chord is subject to interpretation and debate even among trained experts. This inherent subjectivity is evidenced by diverse ground-truth annotations for the same songs and discussions about proper evaluation metrics (de Clercq and Temperley, 2011; Humphrey and Bello, 2015).

For the sake of this thesis—which is in computer science, and not music theory or philosophy—I will ignore the ramifications of this vague definition. Instead, I will adopt a simplifying (but easier to formalise) stance: I assume that the interplay between notes can be mapped to a single chord symbol, that

[1]https://chordify.net/
[2]https://yousician.com

the correct chord symbol was identified by an expert, and that it is available as ground truth. This makes the task one of *segmentation* and *labelling*, similar to speech recognition. The key difference is that we are interested in both the labels and the timestamps of the segments, whereas speech recognition only the label sequence matters.

Computational systems that extract high-level information from signals face two key problems: (i) how to extract meaningful information from noisy sources, and (ii) how to process this information into sensible output. For chord recognition, this translates to *acoustic modelling*—how to predict a chord label for each position or frame in the audio—and *temporal modelling*—how to cast this information into meaningful segments of chords.

Acoustic models extract frame-wise chord classifications, typically in the form of a distribution over chord labels. In traditional chord recognition systems, these models were hand-crafted and split into *feature extraction* and *pattern matching*. Feature extraction transforms audio signals into representations that emphasise harmonic content; usually, this is some form of pitch-class profiles, e.g. Mauch and Dixon (2010b); Müller et al. (2009); Ueda et al. (2010). Pattern matching assigns chord labels to such representations, but works on single frames or local context only; here, template matching (Fujishima, 1999) and Gaussian mixtures (Cho, 2014) are popular choices.

The line between feature extraction and pattern matching blurs with the advent of end-to-end audio processing methods (specifically, deep neural networks). Some approaches use neural networks solely for feature extraction: linear regression (Chen et al., 2012), feed-forward neural networks (Korzeniowski and Widmer, 2016a) (Chapter 3) and convolutional neural networks (Humphrey et al., 2012) learn to transform a general time-frequency representation into a manually defined one that is useful for chord recognition, like chroma vectors or a Tonnetz representation. However, neural networks can also learn to extract discriminative, hierarchical features (Bengio et al., 2013) jointly with a simple "pattern matcher" (logistic regression) in their final layer (Humphrey and Bello, 2012; McFee and Bello, 2017), and Korzeniowski and Widmer (2016b) (Chapter 4).

Temporal models process the predictions of an acoustic model and cast them into chord segments. They provide coherence to the possibly volatile predictions of the acoustic model. They also facilitate the introduction of higher-level musical knowledge—we know from music theory that certain chord progressions are more likely than others—to potentially improve the obtained chord segmentation. A number of works (e.g. (Mauch and Dixon, 2010b; Ni et al., 2012a; Pauwels and Martens, 2014)) implemented hand-designed temporal models for chord recognition. These models are usually first-order Dynamic Bayesian Networks that operate at the beat or time-frame level. They are designed to incorporate musical knowledge, with parameters set by hand or trained from data.

A different approach is to learn temporal models fully from data, without any imposed structure. Here, natural choices include hidden Markov models (Cho et al., 2010; Sheh and Ellis, 2003), conditional random fields (Burgoyne et al., 2007; Korzeniowski and Widmer, 2016b) (see also Chapter 4), or recurrent neural networks (Boulanger-Lewandowski et al., 2013; Sigtia et al., 2015), where the vast majority of chord recognition systems relies on hidden Markov models.

Research has focused on improving frame-wise predictions of acoustic models (Humphrey et al., 2012; Korzeniowski and Widmer, 2016a; McFee and Bello, 2017; Ueda et al., 2010), while only few works have explored improvements in temporal models (Boulanger-Lewandowski et al., 2013; Mauch and Dixon, 2010a; Pauwels and Martens, 2014). This trend was reinforced through the insight that the capabilities of existing temporal models are limited: as shown by Chen et al. (2012) and Cho and Bello (2014), temporal models enforce continuity of individual chords rather than provide information about chord transitions—i.e., they mainly model the chord's duration. Chen et al. (2012) also observed that in popular music "chord progressions are less predictable than it seems", and thus knowing chord history does not greatly narrow the possibilities for the next chord.

## 2.2 Contributions

Steady advances notwithstanding, chord recognition results have seemed to stagnate in recent years. The study in (Humphrey and Bello, 2015) offers possible explanations, including invalid harmonic assumptions, limitations of evaluation measures, conflicting problem definitions, and the subjectivity inherent to this task. All of these criticisms are certainly valid. However, at present there is still a large performance gap between human annotators (measured via inter-annotator agreement) and automatic systems (Humphrey and Bello, 2015), although they are subjected to the same constraints. I hope that the contributions that form this thesis serve as stepping stones to overcome this apparent glass ceiling.

I invite you, the reader, to join me in exploring various aspects of chord recognition. In the first chapters, we will consider acoustic modelling. In Chapter 3, we will develop a strong and interpretable harmonic feature extractor based on neural networks. The model will compute chroma vectors that contain only harmonically relevant information, while suppressing spectral information that is irrelevant to harmony. This improves upon hand-crafted chroma extraction methods that focus on specific nuisance factors, but cannot capture them all. In Chapter 4, we will apply modern deep learning techniques to create an acoustic model for chord recognition, and combine it with conditional random fields for sequence decoding. We will then show that the automatically learned mid-level features have a musical interpretation. This concludes my work on acoustic models for chord recognition.

We will then focus on temporal models, and investigate their alleged inability to model chord transitions. We will find an explanation in the temporal level they have been applied on—audio frames, instead of chord sequences—, and describe the experiments that lead to this explanation in Chapter 5. We will then conduct a systematic study of harmonic language models based on recurrent neural networks in Chapter 6, and demonstrate their superiority compared to statistical finite-context models. We will also see that RNN-based harmonic language models model chord sequences differently: they automatically adapt to the processed pieces by remembering chord sub-sequences they have observed in the past.

Finally, in Chapter 7, we will develop a probabilistic model that allows the integration of meaningful harmonic language models with the frame-wise predictions of acoustic models. This model binds together the two branches of research we have followed in the preceding chapters (acoustic modelling and harmonic language modelling) using a third component: chord duration models. While we employ only simple language and duration models in the beginning, in Chapter 8, we will replace them using recurrent neural networks, and analyse their properties and effect on the final chord recognition results.

# 3 An Interpretable Harmonic Feature: The Deep Chroma Extractor

Let us first take a step towards better features for chord recognition by introducing a data-driven approach to extract chromagrams that specifically encode content relevant to harmony. The method learns to discard irrelevant information like percussive noise, overtones or timbral variations automatically from data. It is thus able to compensate a broader range of interferences than hand-crafted approaches.

This work has been previously released in Korzeniowski and Widmer (2016a).

## 3.1 Chromagrams

The most popular feature used for chord recognition is the *Chromagram*. A chromagram comprises a time-series of chroma vectors, which represent harmonic content at a specific time in the audio as $\mathbf{c} \in \mathbb{R}^{12}$. Each $c_i$ stands for a pitch class, and its value indicates the current salience of the corresponding pitch class. Chroma vectors are computed by applying a filter bank to a time-frequency representation of the audio. This representation results from either a short-time Fourier transform (STFT) or a constant-q transform (CQT), the latter being more popular due to a finer frequency resolution in the lower frequency area.

Chromagrams are concise descriptors of harmony because they encode tone quality and neglect tone height. In theory, this limits their representational power: without octave information, one cannot distinguish e.g. chords that comprise the same pitch classes, but have a different bass note (like G vs. G/5, or A:sus2 vs. E:sus4). In practice, we can show that given chromagrams derived from ground truth annotations, using logistic regression we can recognise ≈ 97%

of chords (reduced to major/minor) in the Beatles dataset. This result encourages us to create chroma features that contain harmony information, but are robust to spectral content that is harmonically irrelevant.

Chroma features are noisy in their basic formulation because they are affected by various interferences: musical instruments produce overtones in addition to the fundamental frequency; percussive instruments pollute the spectrogram with broadband frequency activations (e.g. snare drums) and/or pitch-like sounds (tom-toms, bass drums); different combinations of instruments (and different, possibly genre-dependent mixing techniques) create different timbres and thus increase variance (Cho and Bello, 2014; McVicar et al., 2014).

Researchers have developed and used an array of methods that mitigate these problems and extract cleaner chromagrams: Harmonic-percussive source separation can filter out broadband frequency responses of percussive instruments (Ono et al., 2008; Ueda et al., 2010), various methods tackle interferences caused by overtones (Cho and Bello, 2014; Mauch and Dixon, 2010a), while (Müller et al., 2009; Ueda et al., 2010) attempt to extract chromas robust to timbre. See Cho and Bello (2014) for a recent overview and evaluation of different methods for chroma extraction. Although these approaches improve the quality of extracted chromas, it is very difficult to hand-craft methods for all conceivable disturbances, even if we could name and quantify them.

The approaches mentioned above share a common limitation: they mostly operate on single feature frames. Single frames are often not enough to decide which frequencies salient in the spectrum are relevant to harmony and which are noise. This is usually countered by contextual aggregation such as moving mean/median filters or beat synchronisation, which are supposed to smooth out noisy frames. Since they operate only *after* computing the chromas, they address the symptoms (noisy frames) but do not tackle the cause (spectral content irrelevant to harmony). Also, Cho and Bello (2014) found that they blur chord boundaries and details in a signal and can impair results when combined with more complex chord models and post-filtering methods.

It is close to impossible to find the rules or formulas that define harmonic relevance of spectral content manually. We thus resort to the data-driven approach of deep learning. Deep learning was found to extract strong, hierarchical, discriminative features (Bengio et al., 2013) in many domains. Deep learning based systems established new state-of-the-art results in computer vision, speech recognition, and MIR tasks such as beat detection (Böck et al., 2014), tempo estimation (Böck et al., 2015) or structural segmentation (Ullrich et al., 2014).

We want to exploit the power of deep neural networks to compute harmonically relevant chroma features. The proposed chroma extractor learns to filter harmonically irrelevant spectral content from a *context of audio frames*. This way, we circumvent the necessity to temporally smooth the features by allowing the chroma extractor to use context information directly. Our method computes cleaner chromagrams while retaining their advantages of low dimensionality and intuitive interpretation.

## 3.2 Related Work

A number of works used neural networks in the context of chord recognition. Humphrey and Bello (2012) applied Convolutional Neural Networks to classify major and minor chords end-to-end. Boulanger-Lewandowski et al. (2013) and Sigtia et al. (2015) explored Recurrent Neural Networks as a post-filtering method, where the former used a deep belief net, the latter a deep neural network as underlying feature extractor. All these approaches train their models to directly predict major and minor chords, and following (Bengio et al., 2013), the hidden layers of these models learn a hierarchical, discriminative feature representation. However, since the models are trained to distinguish major/minor chords only, they consider other chord types (such as seventh, augmented, or suspended) mapped to major/minor as intra-class variation to be robust against, which will be reflected by the extracted internal features. These features might thus not be useful to recognise other chords.

We circumvent this by using chroma templates derived from chords as distributed (albeit incomplete) representation of chords. Instead of directly classifying a chord label, the network is required to compute the *chroma representation* of a chord given the corresponding spectrogram. We expect the network to learn which salience in the spectrogram is responsible for a certain pitch class to be harmonically important, and compute higher values for the corresponding elements of the output chroma vector.

Approaches to directly learn a mapping from spectrogram to chroma include those by İzmirli and Dannenberg (2010) and Chen et al. (2012). However, both learn only a linear transformation of the time-frequency representation, which limits the mapping's expressivity. Additionally, both base their mapping on a single frame, which comes with the disadvantages we outlined in the previous section.

In an alternative approach, Humphrey et al. (2012) apply deep learning methods to produce Tonnetz features from a spectrogram. Using other features than the chromagram is a promising direction, and was also explored by Chen et al. (2012) for bass notes. Most chord recognition systems however still use chromas, and more research is necessary to explore to which degree and under which circumstances Tonnetz features are favourable.

## 3.3 Method

To construct a robust chroma feature extractor, we use a deep neural network (DNN). DNNs consist of $L$ hidden layers $h_l$ of $U_l$ computing units. These units compute values based on the results of the previous layer, such that

$$h_l(\mathbf{x}) = \sigma_l \left( \mathbf{W}_l \cdot h_{l-1}(\mathbf{x}) + \mathbf{b}_l \right), \tag{3.1}$$

where $\mathbf{x}$ is the input to the net, $\mathbf{W}_l \in \mathbb{R}^{U_l \times U_{l-1}}$ and $b_l \in \mathbb{R}^{U_l}$ are the weights and the bias of the $l^{\text{th}}$ layer respectively, and $\sigma_l$ is a (usually non-linear) *activation function* applied point-wise.

We define two additional special layers: an *input layer* that is feeding values to $h_1$ as $h_0(\mathbf{x}) = \mathbf{x}$, with $U_0$ being the input's dimensionality; and an *output layer* $h_{L+1}$ that takes the same form as shown in Eq. 3.1, but has a specific semantic purpose: it represents the output of the network, and thus its dimensionality $U_{L+1}$ and activation function $\sigma_{L+1}$ have to be set accordingly.[1]

The weights and biases constitute the model's parameters. They are trained in a supervised manner by gradient methods and error back-propagation in order to minimise the loss of the network's output. The loss function depends on the domain, but is generally some measure of difference between the current output and the desired output (e.g. mean squared error, categorical cross-entropy, etc.)

In the following, we describe how we compute the input to the DNN, the concrete DNN architecture and how it was trained.

### 3.3.1 Input Processing

We compute the time-frequency representation of the signal based on the magnitude of its STFT $\mathbf{S}$. The STFT gives significantly worse results than the constant-q transform if used as basis for traditional chroma extractors, but we found in preliminary experiments that our model is not sensitive to this phenomenon. We use a frame size of 8 192 with a hop size of 4 410 at a sample rate of 44 100 Hz. Then, we apply a triangular filter bank $\mathbf{B}^{\triangle}_{\log}$ to convert the linear frequency scale of the magnitude spectrogram to a logarithmic one, and apply an element-wise logarithmic compression. We call the result the *logarithmic quarter-tone spectrogram*

$$\mathbf{Q} = \log\left(1 + \mathbf{B}^{\triangle}_{\log} \cdot |\mathbf{S}|\right). \tag{3.2}$$

[1]For example, for a 3-class classification problem one would use 3 units in the output layer and a softmax activation function such that the network's output can be interpreted as probability distribution of classes given the data.

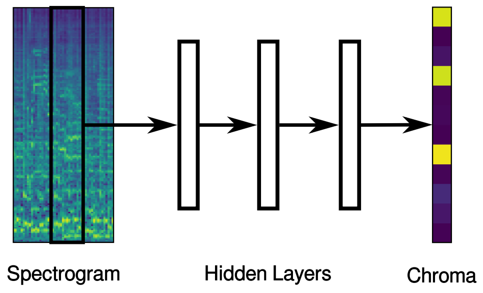


**Figure 3.1:** Model overview. At each time 15 consecutive frames of the input quarter-tone spectrogram $S_{\log}$ are fed to a series of 3 dense layers of 512 rectifier units, and finally to a sigmoid output layer of 12 units (one per pitch class), which represents the chroma vector for the centre input frame.

The quarter-tone spectrogram contains only bins corresponding to frequencies between 30 Hz and 5 500 Hz and has 24 bins per octave. This results in a dimensionality of 178 bins. To be concise, we will refer to $\mathbf{Q}$ as "spectrogram" in rest of this chapter.

Our model uses a context window around a target frame as input. Through systematic experiments on the validation folds (see Sec.3.4.1) we found a context window of ±0.70 s to work best. Since we operate at 10 fps, we feed our network at each time 15 consecutive frames, which we will denote as *super-frame* $\mathbf{x}_t$, where $t$ is the index of the central frame.

### 3.3.2 Model

We define the model architecture and set the model's hyper-parameters based on validation performance in several preliminary experiments. Although a more systematic approach might reveal better configurations, we found that results do not vary by much once we reach a certain model complexity.

Our model is a deep neural network with 3 hidden layers of 512 rectifier units (Glorot et al., 2011) each. Thus, $\sigma_l(\mathbf{x}) = \max(0, \mathbf{x})$ for $1 \leq l \leq L$. The output layer, representing the chroma vector, consists of 12 units (one unit per pitch class) with a sigmoid activation function $\sigma_{L+1}(\mathbf{x}) = {}^1/_{1+\exp(-\mathbf{x})}$. The input layer represents the input super-frame and thus has a dimensionality of 2 670. Fig. 3.1 shows an overview of our model.

### 3.3.3 Training

To train the network, we propagate back through the network the gradient of the loss $\mathscr{L}$ with respect to the network parameters. Our loss is the binary cross-entropy between each pitch class in the predicted chroma vector $\hat{\mathbf{y}}_t = h_{L+1}(\mathbf{x}_t)$

and the target chroma vector $\mathbf{y}_t$, which is derived from the ground truth chord label at time $t$. For a single data instance,

$$\mathscr{L}\left(\hat{\mathbf{y}}_t, \mathbf{y}_t\right) = \frac{1}{12}\sum_{i=1}^{12} -y_{t,i}\log(\hat{y}_{t,i}) - (1 - y_{t,i})\log(1 - \hat{y}_{t,i}). \qquad (3.3)$$

We learn the parameters with mini-batch training (batch size 512) using the ADAM update rule (Kingma and Ba, 2015). We also tried simple stochastic gradient descent with Nesterov momentum and a number of manual learn rate schedules, but could not achieve better results (to the contrary, using ADAM training usually converged earlier). To prevent over-fitting, we apply dropout (Srivastava et al., 2014) with probability 0.5 after each hidden layer and early stopping if validation accuracy does not increase after 20 epochs.

## 3.4 Experiments

To evaluate the chroma features our method produces, we set up a simple chord recognition task. We ignore any post-filtering methods and use a simple, linear classifier (logistic regression) to match features to chords. This way we want to isolate the effect of the feature on recognition accuracy. As it is common, we restrict ourselves to distinct only major/minor chords, resulting in 24 chord classes and a 'no chord' class.

Our compound evaluation dataset comprises the Beatles (Harte, 2010), Queen and Zweieck (Mauch et al., 2009) datasets (which form the "Isophonics" dataset used in the MIREX[2] competition), the RWC pop dataset[3] (Goto et al., 2002), and the Robbie Williams dataset (Di Giorgi et al., 2013). The datasets total 383 songs or approx. 21 hours and 39 minutes of music.

We perform 8-fold cross validation with random splits. For the Beatles dataset, we ensure that each fold has the same album distribution. For each test fold, we use six of the remaining folds for training and one for validation.

As evaluation measure, we compute the Weighted Chord Symbol Recall (WCSR), often called Weighted Average Overlap Ratio (WAOR), of major and minor chords using the mir_eval library (Raffel et al., 2014).

### 3.4.1 Compared Features

We evaluate our extracted features $\mathbf{C}^D$ against three baselines: a standard chromagram $\mathbf{C}$ computed from a constant-q transform, a chromagram with frequency

[2]http://www.music-ir.org/mirex

[3]Chord annotations available at https://github.com/tmc323/Chord-Annotations

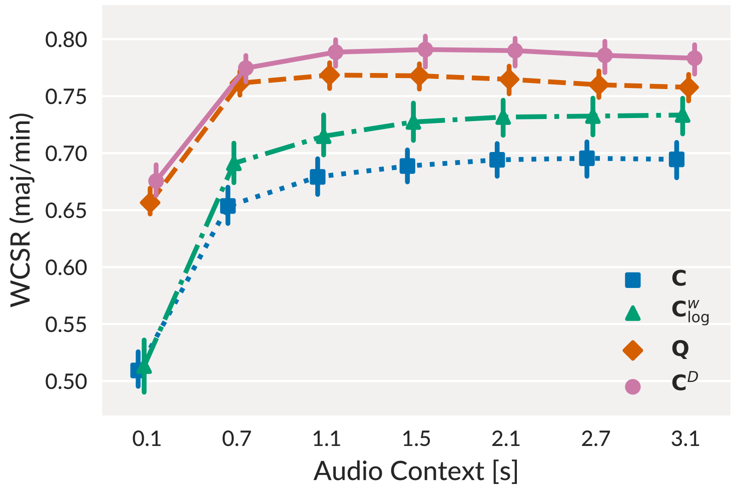


**Figure 3.2:** Validation WCSR for Major/minor chord recognition of different methods given different audio context sizes. Whiskers represent 0.95 confidence intervals.

weighting and logarithmic compression of the underlying constant-q transform $\mathbf{C}^w_{\log}$, and the quarter-tone spectrogram $\mathbf{Q}$. The chromagrams are computed using the librosa library. Their parametrisation follows closely the suggestions in (Cho and Bello, 2014), where $\mathbf{C}^w_{\log}$ was found to be the best chroma feature for chord recognition.

Each baseline can take advantage of context information. Instead of computing a running mean or median, we allow logistic regression to consider multiple frames of each feature. This is a more general way to incorporate context, because running mean is a subset of the context aggregation functions possible in our setup. Since training logistic regression is a convex problem, the result is at least as good as if we used a running mean. Note that this applies only to the baseline methods. For our DNN feature extractor, "context" means the amount of context the DNN sees. Then, the logistic regression sees only one frame of the feature the DNN computed.

We determined the optimal amount of context for each baseline experimentally using the validation folds, as shown in Fig. 3.2. The best results achieved were 79.0 % with 1.5 s context for $\mathbf{C}^D$, 76.8 % with 1.1 s context for $\mathbf{Q}$, 73.3 % with 3.1 s context for $\mathbf{C}^w_{\log}$, and 69.5 % with 2.7 s context for $\mathbf{C}$. We fix these context lengths for testing.

|  | *Btls* | *Iso* | *RWC* | *RW* | *Total* |
|---|---|---|---|---|---|
| $\mathbf{C}$ | $71.0_{\pm 0.1}$ | $69.5_{\pm 0.1}$ | $67.4_{\pm 0.2}$ | $71.1_{\pm 0.1}$ | $69.2_{\pm 0.1}$ |
| $\mathbf{C}^{w}_{\text{log}}$ | $76.0_{\pm 0.1}$ | $74.2_{\pm 0.1}$ | $70.3_{\pm 0.3}$ | $74.4_{\pm 0.2}$ | $73.0_{\pm 0.1}$ |
| $\mathbf{Q}$ | $78.0_{\pm 0.2}$ | $76.5_{\pm 0.2}$ | $74.4_{\pm 0.4}$ | $77.8_{\pm 0.4}$ | $76.1_{\pm 0.2}$ |
| $\mathbf{C}^{D}$ | $\mathbf{80.2}_{\pm 0.1}$ | $\mathbf{79.3}_{\pm 0.1}$ | $\mathbf{77.3}_{\pm 0.1}$ | $\mathbf{80.1}_{\pm 0.1}$ | $\mathbf{78.8}_{\pm 0.1}$ |

**Table 3.1:** Cross-validated WCSR on the Maj/min task of compared methods on various datasets. Best results are bold-faced ($p < 10^{-9}$). Small numbers indicate standard deviation over 10 experiments. "Btls" stands for the Beatles, "Iso" for Isophonics, and "RW" for the Robbie Williams datasets. Note that the Isophonics dataset comprises the Beatles, Queen and Zweieck datasets.

## 3.5 Results

Table 3.1 presents the results of our method compared to the baselines on several datasets. The chroma features $\mathbf{C}$ and $\mathbf{C}^{w}_{\text{log}}$ achieve results comparable to those (Cho and Bello, 2014) reported on a slightly different compound dataset. Our proposed feature extractor $\mathbf{C}^{D}$ clearly performs best, with $p < 10^{-9}$ according to a paired t-test. The results indicate that the chroma vectors extracted by the proposed method are better suited for chord recognition than those computed by the baselines.

To our surprise, the raw quarter-tone spectrogram $\mathbf{Q}$ performed better than the chroma features. This indicates that computing chroma vectors in the traditional way mixes harmonically relevant features found in the time-frequency representation with irrelevant ones, and the final classifier cannot disentangle them. This raises the question of why chroma features are preferred to spectrograms in the first place. We speculate that the main reason is their much lower dimensionality and thus ease of modelling (e.g. using Gaussian mixtures).

Artificial neural networks often give good results, but it is difficult to understand what they learned, or on which basis they generate their output. In the following, we will try to dissect the proposed model, understand its workings, and see what it pays attention to. To this end, we compute saliency maps using guided back-propagation (Springenberg et al., 2015), adapting code freely available[4] for the Lasagne library (Dieleman et al., 2015). Leaving out the technical details, a saliency map can be interpreted as an attention map of the same size as the input. The higher the absolute saliency at a specific input dimension, the stronger its influence on the output, where positive values indicate a direct relationship, negative values an indirect one.

[4]https://github.com/Lasagne/Recipes/

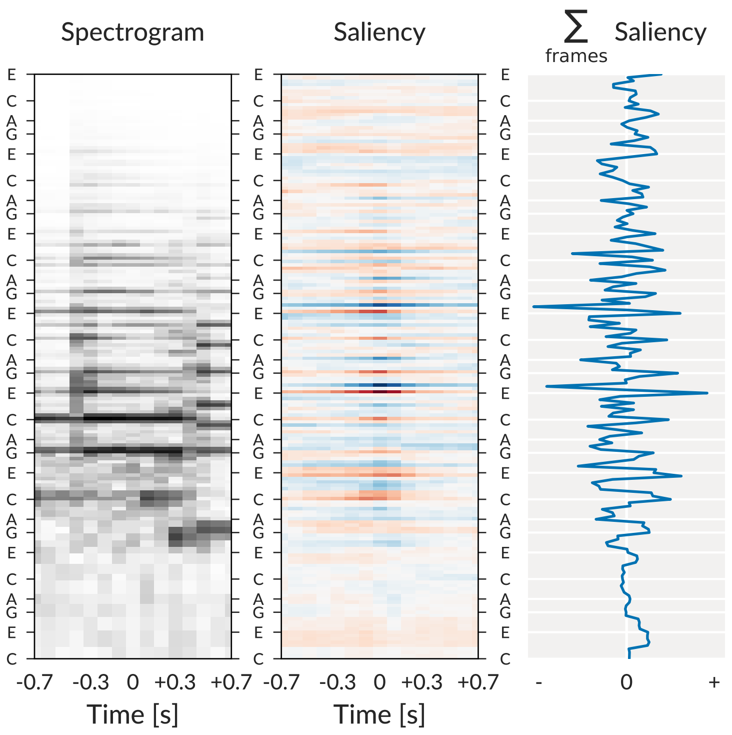


**Figure 3.3:** Input example (C major chord) with corresponding saliency map. The *left image* shows the spectrogram frames fed into the network. The *centre image* shows the corresponding saliency map, where red pixels represent positive, blue pixels negative values. The stronger the saturation, the higher the absolute value. The *right plot* shows the saliency summed over the time axis, and thus how each frequency bin influences the output. Note the strong positive influences of frequency bins corresponding to *c*, *e*, and *g* notes that form a C major chord.

Fig. 3.3 shows a saliency map and its corresponding super-frame, representing a C major chord. As expected, the saliency map shows that the most relevant parts of the input are close to the target frame and in the mid frequencies. Here, frequency bins corresponding to notes contained in a C major chord (c, e, and g) showing positive saliency peeks, with the third, e, standing out as the strongest. Conversely, its neighbouring semitone, f, exhibits strong negative saliency values. Fig. 3.4 depicts average saliencies for two chords computed over the whole Beatles corpus.

Fig. 3.5 shows the average saliency map over all super-frames of the Beatles dataset summed over the frequency axis. It thus shows the magnitude with which individual frames in the super-frame contribute to the output of the neural network. We observe that most information is drawn from a ±0.30 s window around the centre frame. This is in line with the results shown in Fig. 3.2, where the proposed method already performed well with 0.70 s of audio context.

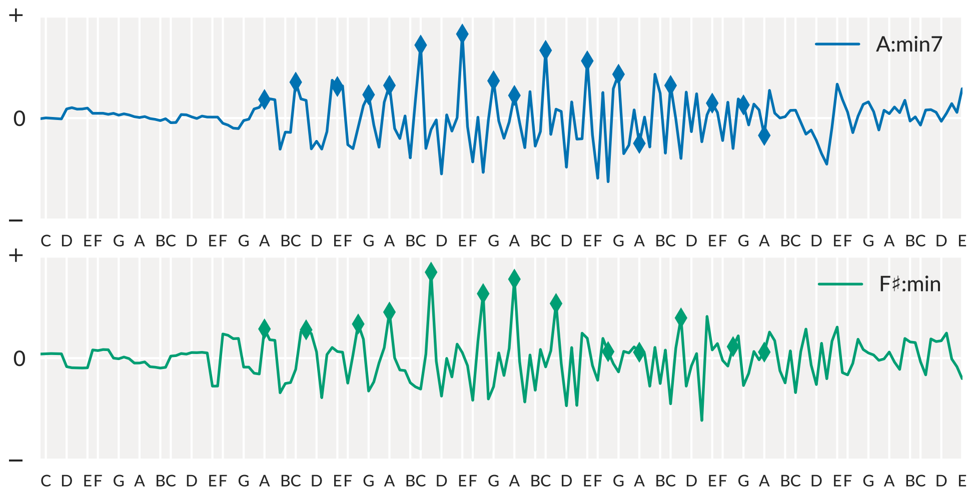


**Figure 3.4:** Average saliency map summed over the time axis for A:min7 and F:min chords computed on the Beatles dataset. As expected, we observe mostly positive peaks for frequency bins corresponding to notes present in the chords (a, c, e, g for A:min7; f, a, c for F:min).

Fig. 3.6 shows the average saliency map over all super-frames of the Beatles dataset, and its sum over the time axis. We observe that frequency bins below 110 Hz and above 3 136 Hz (wide limits) are almost irrelevant, and that the net focuses mostly on the frequency range between 196 Hz and 1 319 Hz (narrow limits). In informal experiments, we could confirm that recognition accuracy drops only marginally if we restrict the frequency range to the wide limits, but significantly if we restrict it to the narrow limits. This means that the secondary information captured by the additional frequency bins of the wide limits is also crucial.

To allow for a visual comparison of the computed features, we depict different chromagrams for the song "Yesterday" by the Beatles in Fig. 3.7. The images show that the chroma vectors extracted by the proposed method are less noisy and chord transitions are crisper compared to the baseline methods.

## 3.6 Conclusions and Future Work

In this chapter, we presented a data-driven approach to learning a neural-network-based chroma extractor for chord recognition. The proposed extractor computes cleaner chromagrams than state-of-the-art baseline methods, which we showed quantitatively in a simple chord recognition experiment and examined qualitatively by visually comparing extracted chromagrams.

We inspected the learned model using saliency maps and found that a frequency range of 110 Hz to 3 136 Hz seems to suffice as input to chord recognition

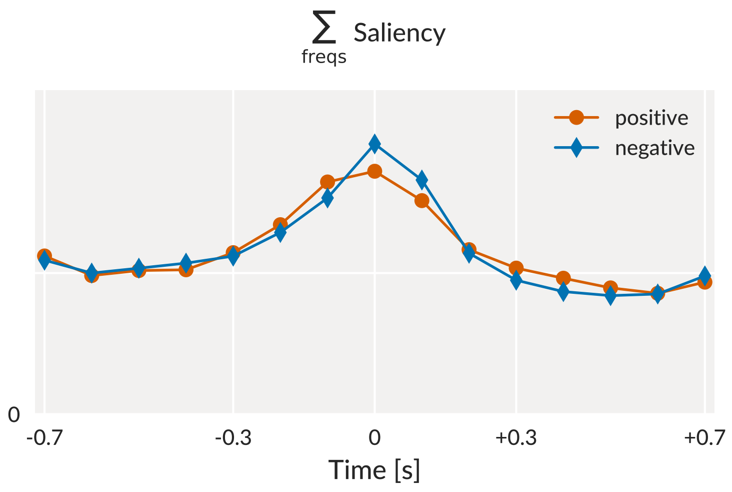


**Figure 3.5:** Average positive and negative saliencies of all input frames of the Beatles dataset, summed over the frequency axis. Most of the important information is within ±0.30 s around the centre frame, and past data seems to be more important than future data. Around the centre frame, the network pays relatively more attention to what should be *missing* than present in a given chroma vector, and vice versa in areas further away from the centre. The differences are statistically significant due to the large number of samples.

methods. Using saliency maps and preliminary experiments on validation folds we also found that a context of 1.50 s is adequate for local harmony estimation.

There are plenty possibilities for future work to extend and/or improve our method. To achieve better results, we could use DNN ensembles instead of a single DNN. We could ensure that the network sees data for which its predictions are wrong more often during training, or similarly, we could simulate a more balanced dataset by showing the net super-frames of rare chords more often. To further assess how useful the extracted features are for chord recognition, we shall investigate how well they interact with post-filtering methods; since the feature extractor is trained discriminatively, Conditional Random Fields (Lafferty et al., 2001) would be a natural choice.

Finally, we believe that the proposed method extracts features that are useful in any other MIR applications that use chroma features (e.g. structural segmentation, key estimation, cover song detection). To facilitate respective experiments, we provide source code for our method as part of the *madmom* audio processing framework (Böck et al., 2016). Information and source code to reproduce our experiments can be found at https://fdlm.github.io/post/deepchroma/.

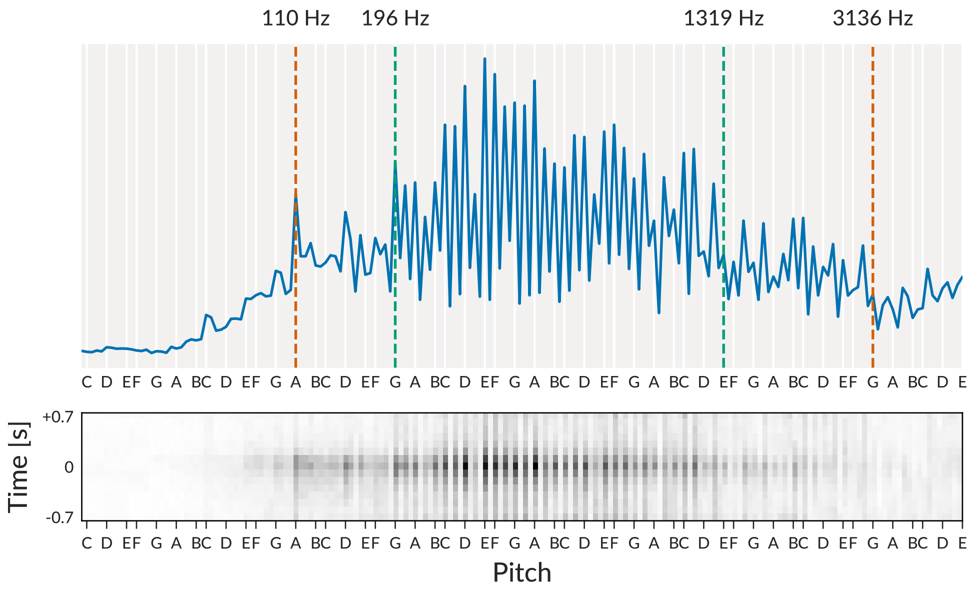


**Figure 3.6:** Average saliency of all input frames of the Beatles dataset (bottom image), summed over the time axis (top plot). We see that most relevant information can be collected in barely 3 octaves between G3 at 196 Hz and E6 at 1 319 Hz. Hardly any harmonic information resides below 110 Hz and above 3 136 Hz. The plot is spiky at frequency bins that correspond to clean semitones because most of the songs in the dataset seem to be tuned to a reference frequency of 440 Hz. The network thus usually pays little attention to the frequency bins between semitones.

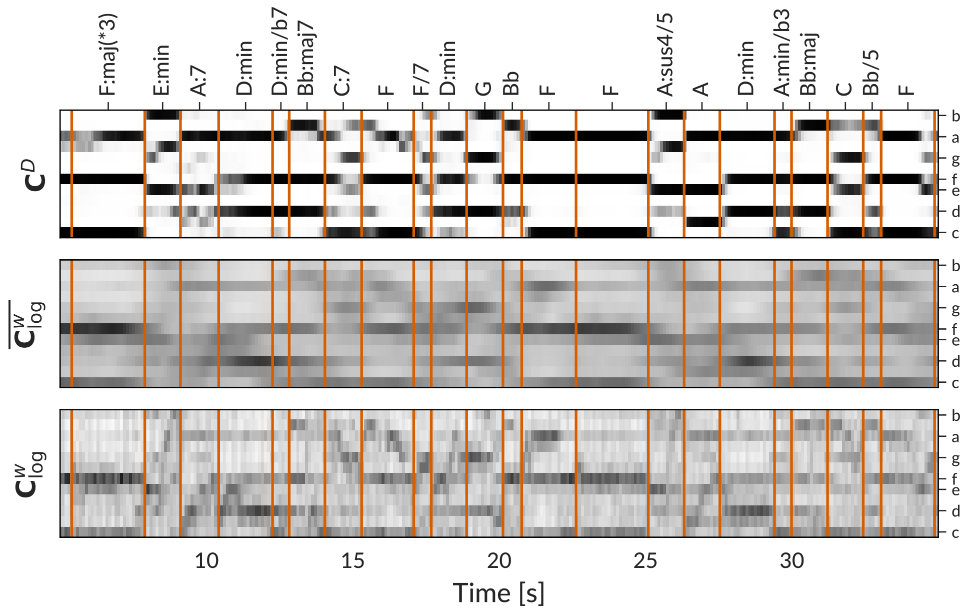


**Figure 3.7:** Excerpts of chromagrams extracted from the song "Yesterday" by the Beatles. The *lower image* shows chroma computed by the $\mathbf{C}^w_{\log}$ without smoothing. We see a good temporal resolution, but also noise. The centre image shows the same chromas after a moving average filter of 1.50 s. The filter reduced noise considerably, at the cost blurring chord transitions. The upper plot shows the chromagram extracted by our proposed method. It displays precise pitch activations and low noise, while keeping chord boundaries crisp. Pixel values are scaled such that for each image, the lowest value in the respective chromagram is mapped to white, the highest to black.

# 4 A Fully Convolutional Acoustic Model

In this chapter, we present an end-to-end chord recognition system that combines a fully convolutional neural network (CNN) for feature extraction with a conditional random field (CRF) for chord sequence decoding. Fully convolutional neural networks replace the stack of dense layers traditionally used in CNNs for classification with *global average pooling* (GAP) (Lin et al., 2014), which reduces the number of trainable parameters and improves generalisation. Similarly to (Humphrey and Bello, 2012), we train the CNN to directly predict chord labels for each audio frame, but instead of using these predictions directly, we use the hidden representation computed by the CNN as features for the subsequent pattern matching and chord sequence decoding stage. We call the feature-extracting part of the CNN *acoustic model*.

For pattern matching and chord sequence decoding, we connect a CRF to the acoustic model. Combining neural networks with CRFs gives a fully differentiable model that can be learned jointly, as shown in (Do and Arti, 2010; Peng et al., 2009). For the task at hand, however, we found it advantageous to train both parts separately, both in terms of convergence time and performance.

This work has been previously released in Korzeniowski and Widmer (2016b).

## 4.1 Feature Extraction

Feature extraction is a two-phase process. First, we convert the signal into a time-frequency representation in the pre-processing stage. Then, we feed this representation to a CNN and train it to classify chords. We take the activations of a hidden layer in the network as high-level feature representation, which we then use to decode the final chord sequence.

### 4.1.1 Pre-processing

The first stage of our feature extraction pipeline transforms the input audio into a time-frequency representation suitable as input to a CNN. We use the same processing steps as in Chapter 3. CNNs consist of fixed-size filters that capture local structure, which requires the spatial relations to be similarly distributed in each area of the input. To achieve this, we compute the magnitude spectrogram of the audio and apply a filter bank with logarithmically spaced triangular filters. This gives us a time-frequency representation in which distances between notes (and their harmonics) are equal in all areas of the input. Finally, we logarithmise the filtered magnitudes to compress the value range. Mathematically, the resulting time-frequency representation of an audio recording is defined as

$$\mathbf{Q} = \log\left(1 + \mathbf{B}_{\log}^{\triangle} \left|\mathbf{S}\right|\right),$$

where $\mathbf{S}$ is the short-time Fourier transform (STFT) of the audio, and $\mathbf{B}_{\log}^{\triangle}$ is the logarithmically spaced triangular filter bank. To be concise, we will refer to $\mathbf{Q}$ as *spectrogram* in the remainder of this chapter.

We feed the network spectrogram frames with context, i.e. the input to the network is not a single column $\mathbf{q}_i$ of $\mathbf{Q}$, but a matrix

$$\mathbf{X}_i = \left[\mathbf{q}_{i-C}, \dots, \mathbf{q}_i, \dots, \mathbf{q}_{i+C}\right],$$

where $i$ is the index of the target frame, and $C$ is the context size.

We chose the parameter values based on our previous study on data-driven feature extraction for chord recognition (see Chapter 3) and a number of preliminary experiments. We use a frame size of 8 192 with a hop size of 4 410 at a sample rate of 44 100 Hz for the STFT. The filter bank comprises 24 filters per octave between 65 Hz and 2 100 Hz. The context size $C = 7$, thus each $\mathbf{X}_i$ represents 1.50 s of audio. Our choice of parameters results in an input dimensionality of $\mathbf{X}_i \in \mathbb{R}^{105\times15}$.

### 4.1.2 Acoustic Model

To extract discriminative features from the input, in Chapter 3, we used a simple deep neural network (DNN) to compute *chromagrams*, concise descriptors of harmonic content. From these chromagrams, we used a simple classifier to predict chords in a frame-wise manner. Despite the network being simple conceptually, due to the dense connections between layers, the model had 1.9 million parameters.

Here, we use a CNN for feature extraction. CNNs differ from traditional deep neural networks by including two additional types of computational layers: *convolutional layers* compute a 2-dimensional convolution of their input with a set of fixed-sized, trainable kernels per feature map, followed by a (usually non-linear) activation function; *pooling layers* sub-sample the input by aggregating over a local neighbourhood (e.g. the maximum of a $2 \times 2$ patch). The former can be re-formulated as a dense layer using a sparse weight matrix with tied weights. This interpretation indicates the advantages of convolutional layers: fewer parameters and better generalisation.

CNNs typically consist of convolutional lower layers that act as feature extractors, followed by fully connected layers for classification. Such layers are prone to over-fitting and come with a large number of parameters. We thus follow (Lin et al., 2014) and use *global average pooling* (GAP) to replace them. To further prevent over-fitting, we apply *dropout* (Srivastava et al., 2014), and use *batch normalisation* (Ioffe and Szegedy, 2015) to speed up training convergence.

Table 4.1 details our model architecture, which consists of 900k parameters, roughly 50 % of the original DNN. Inspired by the architecture presented in (Simonyan and Zisserman, 2014), we opted for multiple lower convolutional layers with small $3 \times 3$ kernels, followed by a layer computing 128 feature maps using $12 \times 9$ kernels. The intuition is that these bigger kernels can aggregate harmonic information for the classification part of the network. We will denote the output of this layer as $\mathbf{F}_i$, the *features* extracted from input $\mathbf{X}_i$.

We target a reduced chord alphabet in this work (major and minor chords for 12 semitones) resulting in 24 classes plus a "no chord" class. This is a common restriction used in the literature on chord recognition (McVicar et al., 2014). The GAP construct thus learns a weighted average of the 128 feature maps for each of the 25 classes using the $1 \times 1$ convolution and average pooling layer. Applying the softmax function then ensures that the output sums to 1 and can be interpreted as a probability distribution of class labels given the input.

Following Bengio et al. (2013), the activations of the network's hidden layers can be interpreted as hierarchical feature representations of the input data. We will thus use $\mathbf{F}_i$ as a feature representation for the subsequent parts of our chord recognition pipeline.

### 4.1.3 Training and Data Augmentation

We train the acoustic model in a supervised manner using the Adam optimisation method (Kingma and Ba, 2015) with standard parameters, minimising the categorical cross-entropy between true targets $\mathbf{y}$ and network output $\hat{\mathbf{y}}$.

| *Layer Type* | *Parameters* | *Padding* | *Output Size* |
|---|---|---|---|
| Input | | | 105 × 15 |
| Conv-Rectify | 32 × 3 × 3 | yes | 32 × 105 × 15 |
| Conv-Rectify | 32 × 3 × 3 | yes | 32 × 105 × 15 |
| Conv-Rectify | 32 × 3 × 3 | yes | 32 × 105 × 15 |
| Conv-Rectify | 32 × 3 × 3 | yes | 32 × 105 × 15 |
| Pool-Max | 2 × 1 | | 32 × 52 × 15 |
| Conv-Rectify | 64 × 3 × 3 | no | 64 × 50 × 13 |
| Conv-Rectify | 64 × 3 × 3 | no | 64 × 48 × 11 |
| Pool-Max | 2 × 1 | | 64 × 24 × 11 |
| Conv-Rectify | 128 × 12 × 9 | no | 128 × 13 × 3 |
| Conv-Linear | 25 × 1 × 1 | no | 25 × 13 × 3 |
| Pool-Avg | 13 × 3 | | 25 × 1 × 1 |
| Softmax | | | 25 |

**Table 4.1:** Proposed CNN architecture. Batch normalisation is performed after each convolution layer. Dropout with probability 0.5 is applied at horizontal rules in the table. All convolution layers use rectifier units (Glorot et al., 2011), except the last, which is linear. The bottom three layers represent the GAP, replacing fully connected layers for classification.

Including a regularisation term, the loss is defined as

$$\mathscr{L} = -\frac{1}{D}\sum_{i=1}^{D} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + \lambda\, |\theta|_2 ,$$

where $D$ is the number of frames in the training data, $\lambda = 10^{-7}$ the $l_2$ regularisation factor, and $\theta$ the network parameters. We process the training set in mini-batches of size 512, and stop training if the validation accuracy does not improve for 5 epochs.

We apply two types of data manipulations to increase the variety of training data and prevent model over-fitting. Both exploit the fact that the frequency axis of our input representation is linear in pitch, and thus facilitates the emulation of pitch-shifting operations. The first operation, as explored by Humphrey and Bello (2012), shifts the spectrogram up or down in discrete semitone steps by a maximum of 4 semitones. This manipulation does not preserve the label, which we thus adjust accordingly. The second operation emulates a slight detuning by shifting the spectrogram by fractions of up to 0.4 of a semitone. Here, the label remains unchanged. We process each data point in a mini-batch with randomly selected shift distances. The network thus almost never sees exactly the same input during training, which significantly reduces over-fitting.

## 4.2 Chord Sequence Decoding

Using the predictions of the pattern matching stage directly (in our case, the predictions of the CNN) often gives good results in terms of frame-wise accuracy. However, chord sequences obtained this way are often fragmented. The main purpose of chord sequence decoding is thus to smooth the reported sequence. Here, we use a linear-chain CRF (Lafferty et al., 2001) to introduce inter-frame dependencies and find the optimal state sequence using Viterbi decoding.

### 4.2.1 Conditional Random Fields

Conditional random fields are probabilistic energy-based models for structured classification. They model the conditional probability distribution

$$P\left(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}\right) = \frac{\exp\left[E\left(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}\right)\right]}{\sum_{\mathbf{y}'_{1:T}} \exp\left[E\left(\mathbf{y}'_{1:T}, \mathbf{x}_{1:T}\right)\right]} \tag{4.1}$$

where $\mathbf{y}_{1:T}$ is the label vector sequence, and $\mathbf{x}_{1:T}$ the feature vector sequence of same length. We assume each $\mathbf{y}_t$ to be the target label in one-hot encoding. The energy function is defined as

$$E\left(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}\right) = \sum_{t=1}^{T} \left[\mathbf{y}_{t-1}^{\top}\mathbf{A}\mathbf{y}_t + \mathbf{y}_t^{\top}\mathbf{c} + \mathbf{x}_t^{\top}\mathbf{W}\mathbf{y}_t\right] + \mathbf{y}_0^{\top}\boldsymbol{\pi} + \mathbf{y}_T^{\top}\boldsymbol{\tau} \tag{4.2}$$

where $\mathbf{A}$ models the inter-frame potentials, $\mathbf{W}$ the frame-input potentials, $\mathbf{c}$ the label bias, $\boldsymbol{\pi}$ the potential of the first label, and $\boldsymbol{\tau}$ the potential of the last label. This form of energy function defines a linear-chain CRF.

From Eq. 4.1 and 4.2 follows that a CRF can be seen as generalised logistic regression. They become equivalent if we set $\mathbf{A}$, $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$ to 0. Further, logistic regression is equivalent to a softmax output layer of a neural network. We thus argue that a CRF whose input is computed by a neural network can be interpreted as a generalised softmax output layer that allows for dependencies between individual predictions. This makes CRFs a natural choice for incorporating dependencies between predictions of neural networks.

### 4.2.2 Model Definition and Training

Our model has 25 states (12 semitones × {major, minor} and a "no-chord" class). These states are connected to observed features through the weight matrix $\mathbf{W}$, which computes a weighted sum of the features for each class. This corresponds to what the global-average-pooling part of the CNN does. We will thus use the

|  | *Isophonics* | *Robbie Williams* | *RWC* |
|---|---|---|---|
| CB3 | 82.2 | - | - |
| KO1 | 82.7 | - | - |
| NMSD2 | 82.0 | - | - |
| Proposed | **82.9** | **82.8** | **82.5** |

**Table 4.2:** Weighted Chord Symbol Recall of major and minor chords achieved by different algorithms. The results of NMSD2 are statistically significantly worse than others, according to a Wilcoxon signed-rank test. Note that train and test data overlaps for CB3, KO1 and NMSD2, while the results of our method are determined by 8-fold cross-validation.

input to the GAP-part, $\mathbf{F}_i$, averaged for each of the 128 feature maps, as input to the CRF. We can pull the averaging operation from the last layer to right after the feature-extraction layer, because the operations in between (linear convolution, batch normalisation) are linear and no dropout is performed at test-time.

Formally, we will denote the input sequence as $\overline{\mathbf{F}} \in \mathbb{R}^{128 \times T}$, where each column $\overline{\mathbf{f}}_i$ is the averaged feature output of the CNN for a given input $\mathbf{X}_i$. Our CRF thus models $p\left(\mathbf{y}_{1:T} \mid \overline{\mathbf{F}}\right)$.

As with the CNN, we train the CRF using Adam, but set a higher learning rate of 0.01. The mini-batches consist of 32 sequences with a length of 1024 frames (102.3 sec) each. As optimisation criterion, we use the $l_1$-regularised negative log-likelihood of all sequences in the data set:

$$\mathscr{L} = -\frac{1}{S} \sum_{s=1}^{S} \log p\left(\mathbf{y}_{1:T}^{(s)} \mid \overline{\mathbf{F}}^{(s)}\right) + \lambda \left|\xi\right|_1 ,$$

where $S$ is the number of sequences in the data set, $\lambda = 10^{-4}$ is the $l_1$ regularisation factor, and $\xi$ are the CRF parameters. We stop training when validation accuracy does not increase for 5 epochs.

## 4.3 Experiments

We evaluate the proposed system using 8-fold cross-validation on a compound dataset that comprises the following subsets: *Isophonics*[1]: 180 songs by The Beatles, 19 songs by Queen, and 18 songs by Zweieck, totalling 10:21 hours of audio. *RWC Popular* (Goto et al., 2002): 100 songs in the style of American and Japanese pop music originally recorded for this data set, totalling 6:46 hours of

[1]http://isophonics.net/datasets

audio. *Robbie Williams* (Di Giorgi et al., 2013): 65 songs by Robbie Williams, totalling 4:30 hours of audio.

As evaluation measure, we compute the Weighted Chord Symbol Recall (WCSR), often called Weighted Average Overlap Ratio (WAOR), of major and minor chords as implemented in the "mir_eval" library (Raffel et al., 2014): $\mathscr{R} = {}^{t_c}/_{t_a}$, where $t_c$ is the total time where the prediction corresponds to the annotation, and $t_a$ is the total duration of annotations of the respective chord classes (major and minor chords, in our case).

We compare our results to the three best-performing algorithms in the MIREX competition in 2013[2]: *CB3* (Cho, 2014); *KO1* (Khadkevich and Omologo, 2011); and *NMSD2* (Ni et al., 2012b).

### 4.3.1 Results

The results presented for the reference algorithms differ from those found on the MIREX website. This is because of minor differences in the implementation of the evaluation libraries. To ensure a fairer comparison, we obtained the predictions of the compared algorithms and ran the same evaluation code for all approaches. Note that for the reference algorithms there is a known overlap between train and test set, and their obtained results might be optimistic.

Table 4.2 shows the results of our method compared to three state-of-the-art algorithms. We can see that the proposed method performs slightly better (but not statistically significant), although the train set of the reference methods overlaps with the test set.

## 4.4 Acoustic Model Analysis

Following Lin et al. (2014), the final feature maps of a GAP network can be interpreted as "category confidence maps". Such confidence maps will have a high average value if the network is confident that the input is of the respective category. In our architecture, the average activation of a confidence map can be expressed as a weighted average over the (batch-normalised) feature maps of the preceding layer. We thus have 128 weights for each of the 25 categories (chord classes).

We wanted to see whether the penultimate feature maps $\mathbf{F}_i$ can be interpreted in a musically meaningful way. To this end, we first analysed the similarity of the weight vectors for each chord class by computing their correlation. The result is shown in Fig. 4.1. We see a systematic correlation between weight vectors of chords that share notes or are close to each other in the circle of fifths. The

[2]http://www.music-ir.org/mirex/wiki/2013:MIREX2013_Results

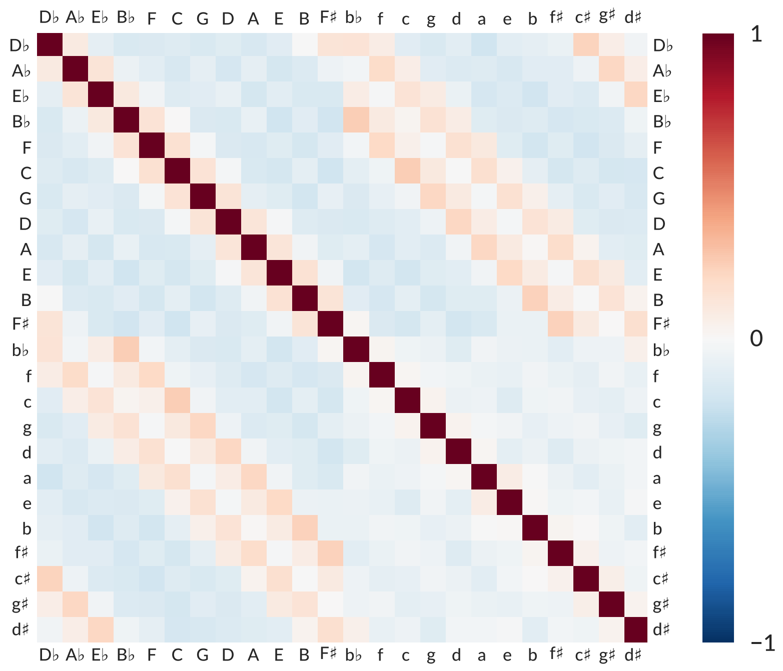
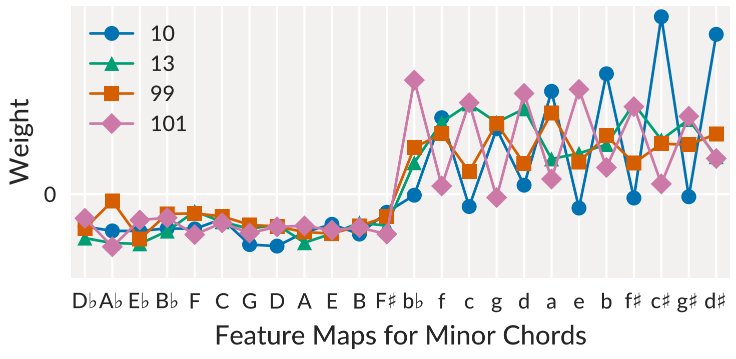


**Figure 4.1:** Correlation between weight vectors of chord classes. Rows and columns represent chords. Major chords are represented by upper-case letters, minor chords by lower-case letters. The order of chords within a chord quality is determined by the circle of fifths. We observe that weight vectors of chords close in the circle of fifths (such as 'C', 'F', and 'G') correlate positively. Same applies to chords that share notes (such as 'C' and 'a', or 'C' and 'c').

patterns within minor chords are less clear. This might be because minor chords are under-represented in the data, and the network could not learn systematic patterns from this limited amount.

Furthermore, we wanted to see if the network learned to distinguish major and minor modes independently of the root note. To this end, we selected the four feature maps with the highest connection weights to major and minor chords respectively and plotted their contribution to each chord class in Fig. 4.2. Here, an interesting pattern emerges: feature maps with high average weights to minor chords have negative connections to all major chords. High activations in these feature maps thus make *all major chords* less likely. However, they tend to be specific on *which minor chords* they favour. We observe a zig-zag pattern that discriminates between chords that are next to each other in the circle of fifths. This means that although the weight vectors of harmonically close chords correlate, the network learned features to discriminate them.



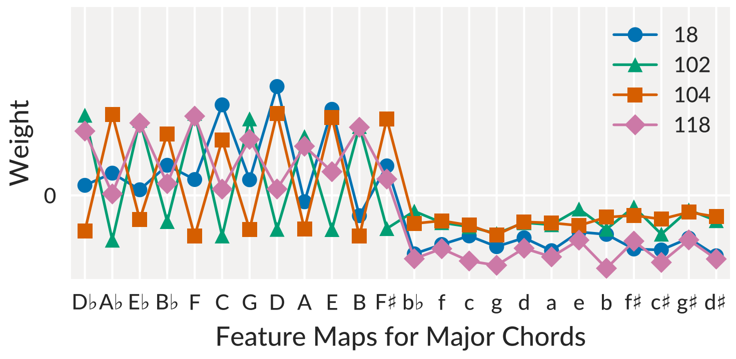


**Figure 4.2:** Connection weights of selected feature maps to chord classes. Chord classes are ordered according to the circle of fifths, such that harmonically close chords are close to each other. In the upper plot, we selected feature maps that have a high average contribution to *minor* chords. In the lower plot, those with high contribution to *major* chords. Feature maps with high average weights to minor chords show negative connections to *all* major chords. Within minor chords, we observe that two of them (10 and 101) discriminate between chords that are harmonically close (zig-zag pattern). We observe a similar pattern in the right plot.

## 4.5 Conclusion

We presented a novel method for chord recognition based on a fully convolutional neural network in combination with a CRF. The method automatically learns musically interpretable features from the spectrogram, and performs at least as good as state-of-the-art systems.

# 5 On the Futility of Frame-Level Language Modelling

Let us now focus on the second part of chord recognition systems: temporal models. As mentioned in Chapter 2, for this purpose, it is common to use Hidden Markov Models (Cho et al., 2010; Sheh and Ellis, 2003) or Conditional Random Fields (Burgoyne et al., 2007; Korzeniowski and Widmer, 2016b) with states corresponding to chord labels. However, recent research showed that such first-order models have limited capacity to to encode musical knowledge, and only ensure stability between consecutive predictions (i.e. they only smooth the output sequence). In Cho and Bello (2014), self-transitions dominate other transitions by several orders of magnitude, and chord recognition results improve if self-transitions are amplified manually. In Chen et al. (2012), employing a first-order chord transition model hardly improves recognition accuracy, given a duration model is applied.

This result bears little surprise: firstly, many common chord patterns in pop, jazz, or classical music span more than two chords, and thus cannot be adequately modelled by first-order models; secondly, models that operate at the frame level by definition only predict the chord symbol of the next frame (typically ≈10 ms away), which most of the time will be the same as the current chord symbol.

To overcome this, a number of recent papers suggested to use Recurrent Neural Networks (RNNs) as temporal models for a number of music-related tasks, such as chord recognition (Boulanger-Lewandowski et al., 2013; Sigtia et al., 2015) or multi-f0 tracking (Sigtia et al., 2016). RNNs are capable of modelling relations in temporal sequences that go beyond simple first-order connections. Their great modelling capacity is, however, limited by the difficulty to optimise their parameters: exploding gradients make training unstable, while vanishing gradients hinder learning long-term dependencies from data.

In this chapter, we argue that (neglecting the aforementioned problems) adopting and training complex temporal models *on a time-frame basis* is futile *on principle*. We support this claim by experimentally showing that they do not outperform first-order models (for which we know they are unable to capture musical knowledge) as part of a complete chord recognition system, and perform only negligibly better at modelling chord label sequences. We thus conclude that, despite their greater modelling capacity, the input representation (musical symbols on a time-frame basis) prohibits learning and applying knowledge about musical structure—the language models hence resort to what their simpler first-order siblings are also capable of: smoothing the predictions.

In the following, we describe three experiments: in the first, we judge two temporal models directly by their capacity to model frame-level chord sequences; in the second, we deploy the temporal models within a fully-fledged chord recognition pipeline; finally, in the third, we learn a *language model* at chord level to show that RNNs are able to learn musical structure if used at a higher hierarchical level. Based on the results, we then draw conclusions for future research on temporal and language modelling in the context of music processing.

This content of this chapter has been published previously in Korzeniowski and Widmer (2017b).

## 5.1 Experiment 1: Chord Sequence Modelling

In this experiment, we want to quantify the modelling power of temporal models directly. A temporal model predicts the next chord symbol in a sequence given the ones already observed. Since we are dealing with frame-level data and adopt a frame rate of 10 fps, a chord sequence consists of 10 chord symbols per second. More formally, given a chord sequence $\mathbf{y} = y_{1:T}$, a model $M$ outputs a probability distribution $P_M(y_t \mid y_{1:t-1})$ for each $y_t$. From this, we can compute the probability of the chord sequence

$$P_M(\mathbf{y}) = P_M(y_1) \cdot \Pi_{t=2}^{T} P_M(y_t \mid y_1, \dots, y_{t-1}) . \tag{5.1}$$

To measure how well a model $M$ predicts the chord sequences in a dataset $\mathcal{Y}$, we compute the average log-probability that it assigns to the sequences $\mathbf{y} \in \mathcal{Y}$:

$$\mathcal{L}(M, \mathcal{Y}) = \frac{1}{N_{\mathcal{Y}}} \sum_{\mathbf{y} \in \mathcal{Y}} \log\left(P_M(\mathbf{y})\right), \tag{5.2}$$

where $N_{\mathcal{Y}}$ is the total number of chords symbols in the dataset.

## 5.1.1 Temporal Models

We compare two temporal models in this experiment: a first-order Markov Chain and a RNN with Long-Short Term Memory (LSTM) units.

For the *Markov Chain*, $P_M(y_t \mid y_{1:t-1})$ in Eq. 5.1 is simplified to $P_M(y_t \mid y_{t-1}) = A_{y_t, y_{t-1}}$ due to the Markov property, and $P_M(y_1) = \pi_{y_1}$. Both $\pi$ and $\mathbf{A}$ can be estimated by counting the corresponding occurrences in the training set.

For the *LSTM-RNN*, we follow closely the design, parametrisation and training procedure proposed by Sigtia et al. (2015), and we refer the reader to their paper for details. The input to the network at time step $k$ is the chord symbol $y_{t-1}$ in one-hot encoding, the output is the probability distribution $P_M(y_t \mid y_{1:t-1})$ used in Eq. 5.1. (For $t = 1$, we input a "no-chord" symbol and the network computes $P(y_1)$.) As loss we use the categorical cross-entropy between the output and the one-hot encoding of the target chord symbol.

We use 2 layers of 100 LSTM units each, and add skip-connections such that the input is connected to both hidden layers and to the output. We train the network using stochastic gradient descent with momentum for a maximum of 200 epochs (the network usually converges earlier) with the learning rate decreasing linearly from 0.001 to 0. As Sigtia et al. (2015), we show the network sequences of 100 symbols (corresponding to 10 seconds) during training. We experimented with longer sequences (up to 50 seconds), but results did not improve (i.e. the network did not profit from longer contexts). Finally, to improve generalisation, we augment the training data by randomly shifting the key of the sequences each time they are shown during training.

## 5.1.2 Data

We evaluate the models on the McGill Billboard dataset (Burgoyne et al., 2011). We use songs with ids smaller than 1000 for training, and the remaining for testing, which corresponds to the test protocol suggested by the website accompanying the dataset. To prevent train/test overlap, we filter out duplicate songs. This reduces the number of pieces from 890 to 742, of which 571 are used for training and validation (59155 unique chord annotations), and 171 for testing (16809 annotations).

The dataset contains 69 different chord types. These chord types are, to no surprise, distributed unevenly: the four most common types (major, minor, dominant 7, and minor 7) already comprise 85% of all annotations. Following Cho and Bello (2014), we simplify this vocabulary to major/minor chords only, where we map chords that have a minor 3rd as their first interval to minor, and all other chords to major. After mapping, we have 24 chord symbols $(12 \text{ root notes} \times \{\text{major, minor}\})$ and a "no-chord" symbol, thus 25 classes.

| | *Markov Chain* | *Recurrent NN* |
|---|---|---|
| $\mathscr{L}(\mathcal{M}, \mathcal{Y})$ | -0.273 | -0.266 |
| $\mathscr{L}_c(\mathcal{M}, \mathcal{Y})$ | -5.420 | -5.219 |
| $\mathscr{L}_s(\mathcal{M}, \mathcal{Y})$ | -0.044 | -0.046 |

**Table 5.1:** Average log-probabilities of chords changes in the test set for the two temporal models. $\mathscr{L}(\mathcal{M}, \mathcal{Y})$ is the number for all chord symbols, $\mathscr{L}_c(\mathcal{M}, \mathcal{Y})$ for positions in the sequence where the chord symbol changes, and $\mathscr{L}_s(\mathcal{M}, \mathcal{Y})$ where it stays the same

### 5.1.3 Results

Table 5.1 shows the resulting avg. log-probabilities of the models on the test set. Additionally to $\mathscr{L}(\mathcal{M}, \mathcal{Y})$, we report $\mathscr{L}_s(\mathcal{M}, \mathcal{Y})$ and $\mathscr{L}_c(\mathcal{M}, \mathcal{Y})$. These numbers represent the average log-probability the model assigns to chord symbols in the dataset when the current symbol is the *same* as the previous one and, when it *changed*, respectively. They are computed similarly to $\mathscr{L}(\mathcal{M}, \mathcal{Y})$, but the product in Eq. 5.1 only captures $t$ where $y_t = y_{t-1}$ or $y_t \neq y_{t-1}$, respectively.

They permit us to reason about how well a model will smooth the predictions when the chord is stable, and how well it can predict chords when they change (this is where "musical knowledge" could come into play).

We can see that the RNN performs only slightly better than the Markov Chain (MC), despite its higher modelling capacity. This improvement is rooted in better predictions when the chord changes (-5.22 for the RNN vs. -5.42 for the MC). This might indicate that the RNN is able to model musical knowledge better than the MC after all. However, this advantage is minuscule and comes seldom into play: the correct chord has an avg. probability of 0.0054 with the RNN vs. 0.0044 with the MC[1], and the number of positions at which the chord symbol changes, compared to where it stays the same, is low.

In the next experiment, we evaluate if the marginal improvement provided by the RNN translates into better chord recognition accuracy when deployed in a fully-fledged system.

## 5.2 Experiment 2: Frame-Level Chord Recognition

In this experiment, we want to evaluate the temporal models in the context of a complete chord recognition framework. The task is to predict for each audio

[1]Both are worse than the random chance of $1/25 = 0.04$, because both would still favour self-transitions

|  | *None* | *MV* | *HMM* | *RNN* |
|---|---|---|---|---|
| *LogReg* | 72.3 | 72.8 | **73.4** | 73.1 |
| *DNN* | 74.2 | 75.3 | **76.0** | 75.7 |
| *ConvNet* | 77.6 | 78.1 | **78.9** | 78.7 |

**Table 5.2:** Weighted Chord Symbol Recall of the 24 major and minor chords and the "no-chord" class for the tested temporal models (columns) on different acoustic models (rows).

frame the correct chord symbol. We use the same data, the same train/test split, and the same chord vocabulary (major/minor and "no chord") as in the first experiment.

Our chord-recognition pipeline comprises spectrogram computation, an automatically learned feature extractor and chord predictor, and finally the temporal model. The first two stages are based on our previous work described in chapters 3 and 4. We extract a log-filtered and log-scaled spectrogram between 65 Hz and 2 100 Hz at 10 frames per second, and feed spectral patches of 1.50 s into one of three acoustic models: a *logistic regression classifier* (LogReg), a *deep neural network* (DNN) with 3 fully connected hidden layers of 256 rectifier units, and a *convolutional neural network* (ConvNet) with the exact architecture we presented in Chapter 4.

Each acoustic model yields frame-level chord predictions, which are then processed by one of three different temporal models.

### 5.2.1 Temporal Models

We test three temporal models of increasing complexity. The simplest one is Majority Voting (MV) within a context of 1.30 s, The others are the very same we used in the previous experiment.

Connecting the Markov Chain temporal model to the predictions of the acoustic model results in a *Hidden Markov Model* (HMM). The output chord sequence is decoded using the Viterbi algorithm.

To connect the RNN temporal model to the predictions of the acoustic model, we apply the *hashed beam search* algorithm, as introduced by Sigtia et al. (2015), with a beam width of 25, hash length of 3 symbols and a maximum of 4 solutions per hash bin. The algorithm only approximately decodes the chord sequence (no efficient and exact algorithms exist, because the output of the network depends on *all* previous inputs).

### 5.2.2 Results

Table 5.2 shows the Weighted Chord Symbol Recall (WCSR) of major and minor chords for all combinations of acoustic and temporal models. The WCSR is defined as $\mathscr{R} = {}^{t_c}/_{t_a}$, where $t_c$ is the total time where the prediction corresponds to the annotation, and $t_a$ is the total duration of annotations of the respective chord classes (major and minor chords and the "no-chord" class, in our case). We used the implementation provided in the "mir_eval" library (Raffel et al., 2014).

The results show that the complex RNN temporal model does not outperform the simpler first-order HMM. They improve compared to not using a temporal model at all, and to a simple majority vote.

The results suggest that the RNN temporal model does not display its (marginal) advantage in chord sequence modelling when deployed within the complete chord recognition system. We assume the reasons to be (i) that the improvement was small in the first place, and (ii) that exact inference is not computationally feasible for this model, and we have to resort to approximate decoding using beam search.

## 5.3 Experiment 3: Modelling Chord-level Label Sequences

In the final experiment, we want to support our argument that the RNN does not learn musical structure because of the hierarchical level (time frames) it is applied on. To this end, we conduct an experiment similar to the first one: an RNN is trained to predict the next chord symbol in the sequence. However, this time the sequence is not sampled at frame level, but at chord level (i.e. no matter how long a certain chord is played, it is reduced to a single instance in the sequence). Otherwise, the data, train/test split, and chord vocabulary are the same as in Experiment 1.

The results confirm that in such a scenario, the RNN clearly outperforms the Markov Chain (Avg. Log-P. of -1.62 vs. -2.28). Additionally, we observe that the RNN does not only learn static dependencies between consecutive chords; it is also able to adapt to a song and recognise chord progressions seen previously in this song without any on-line training. This resembles the way humans would expect the chord progressions not to change much during a part (e.g. the chorus) of a song, and come back later when a part is repeated. Figure 5.1 shows exemplary results from the test data.

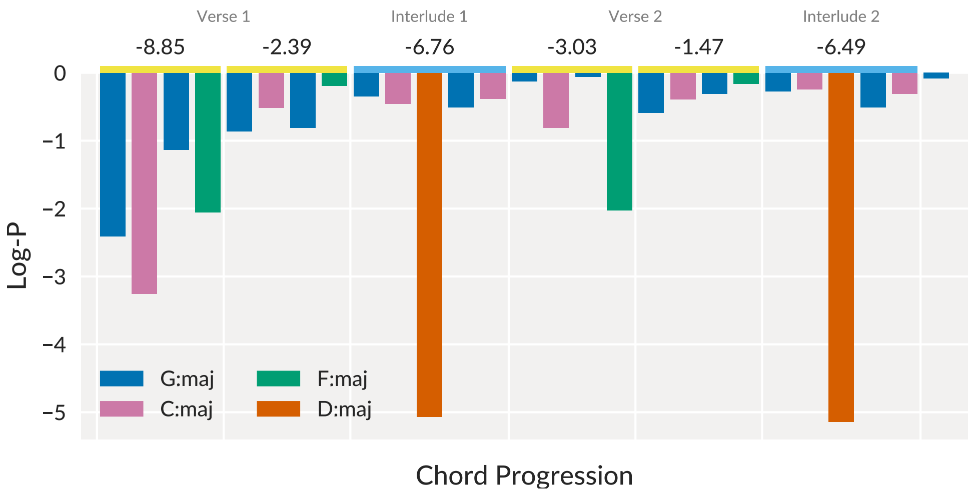


**Figure 5.1:** Log-probabilities of chords at the beginning of The Beatles' "A Hard Day's Night", as computed by a RNN language model at the chord level. Bar colours indicate the chord class. We observe that the two repeated chord sequences ("G-C-G-F" and "G-C-D-G-C", marked with yellow and light blue on the top) achieve higher probabilities as they are repeated, with the exception of one repetition of the first sequence at the beginning of Verse 2 (in this case, the network did not expect to see the "F" after several repetitions of a G-C transition). This indicates that the network was able to remember to some degree the chord progressions seen earlier in the song.

## 5.4 Conclusion

We argued that learning complex temporal models for chord recognition[2] on a time-frame basis is futile. The experiments we carried out support our argument. The first experiment focused on how well a complex temporal model can learn to predict chord sequences compared to a simple first-order one. We saw that the complex model, despite its substantially greater modelling capacity, performed only marginally better. The second experiment showed that, when deployed within a chord recognition system, the RNN temporal model did not outperform the first-order HMM. Its slightly better capability to model frame-level chord sequences was probably counteracted by the approximate nature of the inference algorithm. Finally, in the third experiment, we showed preliminary results that when deployed at a higher hierarchical level than time frames, RNNs are indeed capable of learning musical structure beyond first-order transitions.

Why are complex temporal models like RNNs unable to model frame-level chord sequences? We believe the following two circumstances to be the causes: (i) transitions are dominated by self-transitions, i.e. models need to predict self-transitions as well as possible to achieve good predictive results on the data, and

[2]We expect similar results for other music-related tasks.

(ii) predicting chord changes "blindly" (i.e. without knowledge *when* the change might occur) competes with (i) via the normalisation constraint of probability distributions.

Inferring when a chord changes proves difficult if the model can only consider the *frame-level* chord sequence. There are simply too many uncertainties (e.g. the tempo and harmonic rhythm of a song, timing deviations, etc.) that are hard to estimate. However, the models also do not have access to features computed from the input signal, which might help in judging whether a chord change is imminent or not. Thus, the models are blind to the *time* of a chord change, which makes them focus on predicting self-transitions, as we outlined in the previous paragraph.

We know from other domains such as natural language modelling that RNNs are capable of learning state-of-the-art language models (Mikolov et al., 2010). We thus argue that the reason they underperform in our setting is the frame-wise nature of the input data. For future research, we propose to focus on *language models* instead of frame-level temporal models for chord recognition. By "language model" we mean a model at a higher hierarchical level than the temporal models explored in this chapter (hence the distinction in name)—like the model used in the final experiment in Sec. 5.3. Such language models can then be used in sequence classification framework such as sequence transduction (Graves, 2012) or segmental recurrent neural networks (Lu et al., 2016).

Our results indicate the necessity of hierarchical models, as postulated by Widmer (2016): powerful feature extractors may operate at the frame-level, but more abstract concepts have to be estimated at higher temporal (and, conceptual) levels. Similar results have been found for other music-related tasks: e.g., Srinivasamurthy et al. (2016), divided longer metrical cycles into their subcomponents to improve beat tracking results; in the field of musical structure analysis (an obviously hierarchical concept), McFee and Ellis (2014) extracted representations on different levels of granularity (although their system scored lower in traditional evaluation measures for segmentation, it facilitates a hierarchical breakdown of a piece).

Music theory teaches that cadences play an important role in the harmonic structure of music, but many current state-of-the-art chord recognition systems (including our own) ignore this. Learning powerful language models at sensible hierarchical levels bears the potential to further improve the accuracy of chord recognition systems, which has remained stagnant in recent years.

# 6
# A Large-Scale Study of Chord-Level Language Models

As we saw in the previous chapter, temporal models that are learned from frame-level data cannot improve chord recognition results. However, humans seem to be able to predict the next chord in a song, given an appropriate past context. We might thus suggest another explanation for the performance gap between human chord annotators and computational systems: current state-of-the-art chord recognition systems lack a meaningful understanding of harmony and its development in music. Although we have seen work on optimising low-level and/or Markovian *temporal* models by e.g. Cho and Bello (2014), and work on low-level non-Markovian models by e.g. Boulanger-Lewandowski et al. (2013), Chapter 5 showed that such an understanding can only derive from sequential models that operate on higher temporal levels than those based on audio frames. On the level of audio frames, there is little to improve upon the current state-of-the-art.

In this chapter, we address this issue by exploring the capabilities of automatically learned chord language models to predict chord sequences. We will compare recurrent neural networks (RNNs) with higher-order $N$-gram models, and evaluate these models solely based on how well they predict chord sequences. The question on how to integrate them into a complete chord recognition system is treated in chapters 7 and 8.

The study in this chapter has been previously published in Korzeniowski et al. (2018).

| *Name* | *Reference* | *Pieces* | *Chords* |
|---|---|---|---|
| Beatles | (Harte, 2010) | 180 | 12 646 |
| Jay Chou | (Deng and Kwok, 2016) | 29 | 3 356 |
| McGill Billboard | (Burgoyne et al., 2011) | 742 | 70 197 |
| Queen | (Mauch et al., 2009) | 20 | 2 265 |
| Robbie Williams | (Di Giorgi et al., 2013) | 65 | 6 513 |
| Rock | (de Clercq and Temperley, 2011) | 201 | 18 343 |
| RWC | (Goto et al., 2002) | 100 | 12 726 |
| US Pop 2002 | (Ellis et al., 2003) | 195 | 23 309 |
| Weimar Jazz | (Pfleiderer et al., 2017) | 291 | 18 179 |
| Zweieck | (Mauch et al., 2009) | 18 | 1 822 |
| *Total* | | 1 841 | 169 356 |
| *Unique* | | 1 766 | 161 796 |

**Table 6.1:** Individual datasets used to create the compound dataset in this study. For the Rock corpus, we used the annotations by Temperley rather than those by De Clerq.

## 6.1 Data

We compiled a comprehensive set of chord annotations to perform a large-scale evaluation of different language models for chord prediction. To our knowledge, our compound dataset consists of all time-aligned chord annotations that are publicly available. Table 6.1 provides detailed information about the data. In total, we have 1 841 songs from a variety of genres and decades, with a focus on pop/rock between 1950 and 2000. After we remove duplicate songs and merge consecutive identical annotations, the dataset consists of 1 766 songs containing 161 796 unique chord annotations.

The dataset is minuscule compared to those available for natural language processing: e.g., the NYT section of the English Gigaword dataset (of which only a subset of 6.4 million words is used by Mikolov et al. (2010) for training language models) consists of 37 million words. Therefore, we leverage domain knowledge to generate additional, semi-artificial chord sequences. Assuming that chord progressions are independent of musical key (as in Roman numeral analysis), and that musical keys are uniformly distributed (a simplifying assumption), we transpose each chord sequence to all possible keys during training. This step increases the amount of training sequences by a factor of 12; however, the artificially created data are highly correlated with the existing data, and thus are not equivalent to truly having 12 times as much data available. We will refer to this process as data augmentation in the remainder of this chapter.

We focus on the major/minor chord vocabulary, and follow the standard mapping as described by Cho and Bello (2014): chords with a minor 3rd in the first interval are considered minor, the rest major. This mapping results in a vocabulary of size 25 ($12$ root notes $\times \{\text{maj}, \text{min}\}$ and the "no-chord" class). Bearing in mind the reduced vocabulary and the more repetitive nature of chord progressions compared to spoken language, we feel that the size of this dataset after data augmentation is appropriate for training and evaluating models for chord prediction.

## 6.2 Experimental Setup

Our experiments evaluate how well various models predict chords. This amounts to measuring the average probability the model assigns to the (correct) upcoming chord symbol, given the ones it has already observed in a sequence. More formally, given a sequence of chords $\mathbf{y} = y_{1:K}$, the model $M$ yields $P_M(y_k \mid y_{1:k-1})$ for each $k$. The probability of the complete chord sequence can then be computed as

$$P_M(\mathbf{y}) = P_M(y_1) \cdot \Pi_{k=2}^{K} P_M(y_k \mid y_{1:k-1}) \,. \tag{6.1}$$

The higher $P_M(\mathbf{y})$, the better $M$ models $\mathbf{y}$. To measure how well $M$ models all chord sequences in a dataset $\mathcal{Y}$, we next compute the average log-probability assigned to its sequences:

$$\mathcal{L}(M, \mathcal{Y}) = \frac{1}{N_{\mathcal{Y}}} \sum_{\mathbf{y} \in \mathcal{Y}} \log\left(P_M(\mathbf{y})\right), \tag{6.2}$$

where $N_{\mathcal{Y}}$ is the total number of chord symbols in the dataset. This equation corresponds to the negative cross-entropy between the model's distribution and the data distribution.

All models are trained and tested on the compound dataset. We use 20% of the data for testing, and 80% for training. 15% of the training data is held out for validation. All splits are stratified by dataset.

## 6.3 *n*-gram Language Models

*n*-gram language models are Markovian probabilistic models that assume a fixed-length history (of length $n - 1$) in order to predict the next symbol. Hence, they assume $P_M\left(y_k \mid y_{1:k-1}\right) = P_M\left(y_k \mid y_{(k-n+1):(k-1)}\right)$. This fixed-length history allows the probabilities to be stored in a table, with its entries computed using maximum-likelihood estimation (i.e., by counting occurrences in the training set).

With larger $n$, the sparsity of the probability table increases exponentially due to the finite number of *n*-grams in the training set. We solve this problem using Lidstone smoothing, which adds a pseudo-count $\alpha$ to each possible *n*-gram. We determine the value of $\alpha$ for each model using the validation set.

### 6.3.1 Model Selection

We find the best *n*-gram model by selecting the one with best results on the validation set. To this end, we evaluate models with $n \in \{1, 2, 3, 4, 5, 6\}$, with and without data augmentation. $n = 1$ corresponds to a model that predicts chords just by their frequency in the training data, $n = 2$ to a model that could be deployed in a simple first-order Hidden Markov Model.

Figure 6.1 presents the results of the models on the training and validation sets. As expected, data augmentation reduces over-fitting, and enables the models to achieve better results. To our surprise, validation results increased up to $N = 5$ ($\alpha = 0.3$)—we expected the sparsity problem to earlier prevent this model class from improving with increasing $N$ (sparsity is 98.3% for the best model). We will thus compare the 5-gram model trained with data augmentation to the models based on recurrent neural networks.

## 6.4 Recurrent Neural Language Models

Recurrent Neural Networks (RNNs, see Pascanu et al. (2014) for an overview) are powerful models designed for sequential modelling tasks. In their simplest form, RNNs transform input sequences $\mathbf{x}_{1:K}$ to an output sequence $\mathbf{o}_{1:K}$ through a non-linear projection into a hidden layer $\mathbf{h}_{1:K}$, parametrised by weight

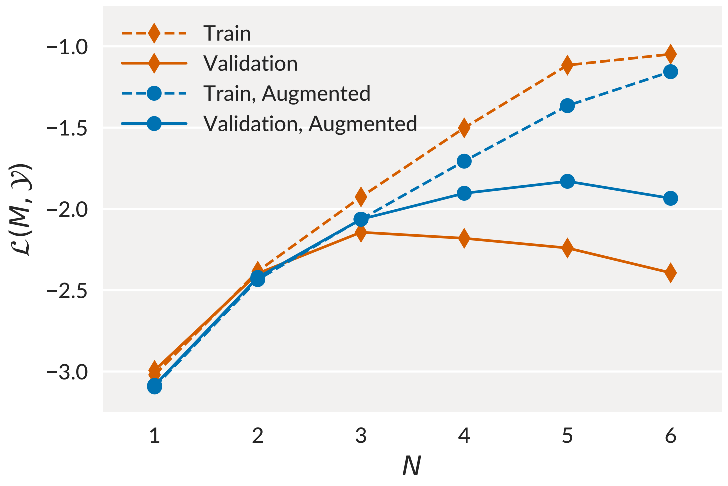


**Figure 6.1:** Average log-probability of all evaluated $N$-gram models.

matrices $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{oh}$:

$$\mathbf{h}_k = \sigma_h\left(\mathbf{W}_{hx}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_{k-1}\right) \tag{6.3}$$

$$\mathbf{o}_k = \sigma_o\left(\mathbf{W}_{oh}\mathbf{h}_k\right), \tag{6.4}$$

where $\sigma_h$ and $\sigma_o$ are the activation functions for the hidden layer (e.g. the sigmoid function), and the output layer (e.g. the softmax), respectively. We left out bias terms for simplicity.

Their use as language models was proposed by Mikolov et al. (2010). For this purpose, the input at each time step $k$ is a vector representation of the preceding symbol $y_{k-1}$. We call this the *chord embedding*, and denote it $v(y_{k-1})$. In the simplest case, this is a one-hot vector. The network's output $\mathbf{o}_k$ is then interpreted as the conditional probability over the next chord symbol $P_M\left(Y_k \mid y_{1:k-1}\right)$. During training, the categorical cross-entropy between $\mathbf{o}_k$ and the true chord symbol is minimised by adapting the weight matrices in Eq. 6.3 and 6.4 using stochastic gradient descent and back-propagation through time. Figure 6.2 provides an overview of the model structure.

Each output $\mathbf{o}_k$ depends on all previous inputs $y_{1:k-1}$ through the recurrent connection in the hidden layer. This allows the network to consider all previous chords when computing $P_M(y_k \mid y_{1:k-1})$. In practice, this capacity is limited because of the fixed size of the hidden layer, and the fragile learning procedure of back-propagation through time, which faces the well-known problems of exploding and vanishing gradients.

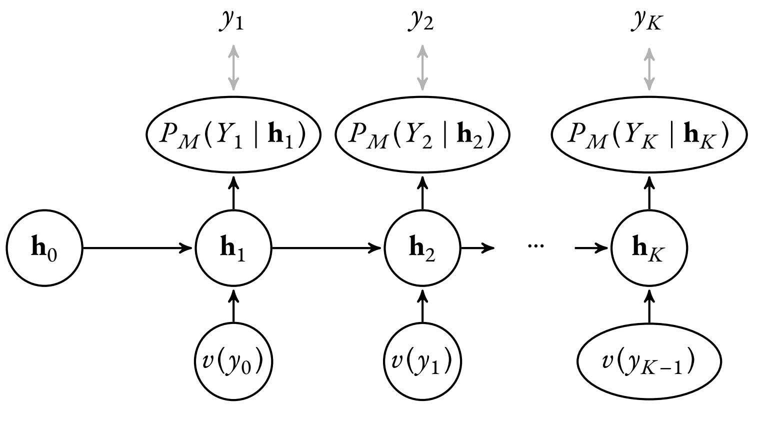


**Figure 6.2:** A simple RNN-based language model. We can easily stack more recurrent hidden layers or add skip-connections between the input and each hidden layer or the output.

### 6.4.1 Chord Embeddings

As mentioned earlier, we need to represent chord classes as vectors to use them in the RNN language model framework. In this work, we explore three possibilities: (i) using the one-hot encoding of the class, (ii) using a fixed-length vector that is learned jointly with the language model, and (iii) learning an embedding using the word2vec skip-gram model (Mikolov et al., 2013) before training the language model itself. In this case, the chord embeddings are optimised to predict the neighbouring chords.

### 6.4.2 Model Selection

RNNs have more hyper-parameters compared to *n*-gram models: we can set the number of hidden layers, the size of each hidden layer, the type and dimensionality of the input chord embedding, the activation function for each hidden layer, whether to use skip-connections, and finally, we can decide to use simple RNNs or more advanced hidden layer structures such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (GRU) (Cho et al., 2014). Additionally, the training procedure has its own hyper-parameters, such as learning rate or mini-batch size. Table 6.2 presents the hyper-parameter space we sampled from.

We fix the following hyper-parameters for all training runs: we apply data augmentation, employ stochastic gradient descent with a batch size of 4 and the ADAM update rule (Kingma and Ba, 2015), set all hidden layers to the same (but variable) size, and stop training if the validation loss does not improve within 15 epochs.

The large number of hyper-parameters prevents us from conducting an

| *Hyper Parameter* | *Sample Space* |
|---|---|
| Embedding Size | $\{4, 8, 16, 24\}$ |
| Embedding Type | {one-hot, word2vec, learned} |
| No. Hidden Layers | $N_h \in \{1, 2, 3, 4, 5\}$ |
| Hidden Layer Size | $D_h \in \{128, 256, 512, 1024\}$ |
| Skip Connections | {yes, no} |
| Learning Rate | $lr = \begin{cases} \{0.001, 0.0005\} & N_h \leq 3 \\ \{0.0005, 0.00025\} & \text{else} \end{cases}$ |
| Learning Rate (GRU) | $lr = \begin{cases} \{0.005, 0.001\} & N_h \leq 3 \\ \{0.001, 0.0005\} & \text{else} \end{cases}$ |

**Table 6.2:** Hyper-parameter space we sampled from to find good model configurations for each of the RNN types (simple, LSTM, GRU). Possible learning rate values were determined on a limited number of preliminary experiments.

exhaustive search for the optimal architecture. Instead, we use Hyperband (Li et al., 2018), a bandit-based black-box hyper-parameter optimisation scheme, to find good configurations for each considered RNN type.

In total, we considered 128 configuration for each RNN type (384 different models). The best RNN setup used one-hot input encoding, skip connections, and $N_h = 5, D_h = 256, lr = 2.5e^{-4}$. The best GRU also used one-hot input encoding, but no skip connections, and $N_h = 3, D_h = 512, lr = 1e^{-3}$. The best LSTM used a word2vec input encoding with 16 dimensions, skip connections, and $N_h = 3, D_h = 512, lr = 1e^{-3}$.

In a final step, we further improve the learned models by a fine-tuning stage. Here, we take the best models found for each RNN type, and re-start training from the epoch that gave the best results on the validation set, but use 1/10 of the original learning rate. As shown in Table 6.3, this step slightly improves the models.

## 6.5 Results and Discussion

We compare the best model of each RNN type with the 5-gram baseline. To this end, we compute $\mathscr{L}(M, \mathscr{Y})$ according to Eq. 6.2 on the test set, and present the results in Tab. 6.3 and Fig. 6.3. We see the RNN-based models easily outperform the 5-gram model, with the LSTM and GRU models performing best. Statistical significance was determined by a paired t-test with Bonferroni correction.

Recurrent neural networks have the capability to remember information over time. We thus call them *dynamic* models, compared to the *static* characteristic of

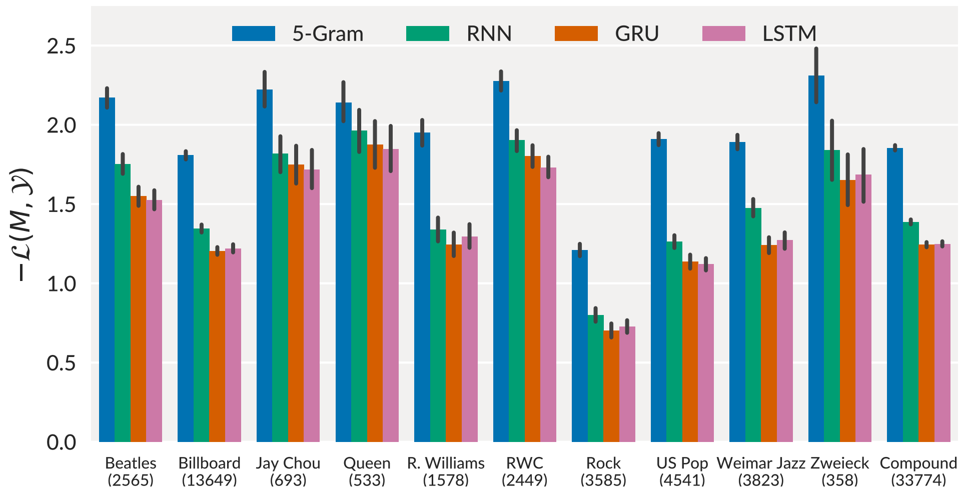


**Figure 6.3:** $-\mathscr{L}(\mathcal{M}, \mathcal{Y})$ (lower is better) of the best model for each model class on the compound test set, and split up into the individual datasets. The numbers in parentheses show the number of chord predictions in each set. Whiskers are 95% confidence intervals computed using bootstrapping. We observe a similar pattern for each set, with the LSTM and GRU performing equally on most of the datasets. We also see that chords are easier to predict in some datasets (e.g. "Rock"), while more difficult in others, (e.g. "RWC").

| | *5-gram* | *RNN* | *GRU* | *LSTM* |
|---|---|---|---|---|
| Val. | 1.830 | 1.465 | 1.328 | 1.302 |
| Val. (fine-tuned) | — | 1.417 | 1.290 | 1.272 |
| Test | 1.874 | 1.387 | **1.244** | **1.249** |

**Table 6.3:** $-\mathscr{L}(\mathcal{M}, \mathcal{Y})$ (lower is better) on the validation and test sets for the best model of each model class. All RNN-based models outperform the 5-gram model, with the GRU yielding the highest avg. log-probability. Statistically equivalent results are marked in bold.

standard models based on *n*-grams. To investigate if the RNN-based language models can leverage this capability, we examine the development of their prediction quality—the log-probability of the correct chord—over the progression of songs.

This is a function of both model capacity and song complexity: given a static model, and repetitive songs that do not change much over time, the log-probabilities assigned to the chords should remain approximately constant; however, if e.g. chords in interludes tend to deviate more from standard chord progressions, static models would yield lower log-probabilities in these cases. On the other hand, if a dynamic model could remember chord progressions from the past, the predictions would improve over time for very repetitive songs.

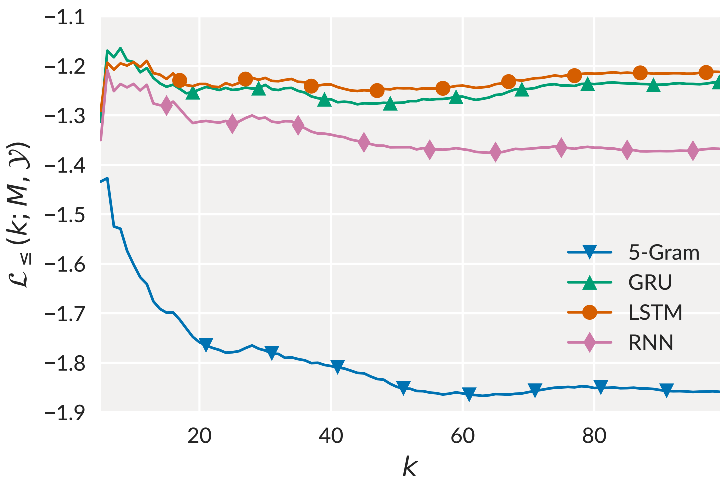


**Figure 6.4:** Avg. cumulative log-probability per chord step $\mathscr{L}_{\leq}(k; M, \mathscr{Y})$. The predictions of the static 5-gram model worsen over time, which indicates that chord progressions later in the songs deviate more from general patterns than in the beginning. The LSTM and GRU models do not suffer from this problem, however, and predictions even improve after chord 40. We conjecture that these models better remember previously seen chords and patterns, which enables them to automatically adapt as a song progresses.

To evaluate this quantitatively, we first selected from the test set songs that contained at least 100 chords, which left us with 136 pieces. We then computed the average cumulative log-probability of chords up to a position in a song as

$$\mathscr{L}_{\leq}(k; M, \mathscr{Y}) = \frac{1}{k|\mathscr{Y}|} \sum_{\mathbf{y} \in \mathscr{Y}} \sum_{\bar{k}=1}^{k} \log P_M \left( y_{\bar{k}} \mid y_{1:\bar{k}-1} \right). \tag{6.5}$$

Eq. 6.5 converges to the objective in Eq. 6.2 with increasing $k$.

Figure 6.4 presents the results for each model. To our surprise, the performance of the static 5-gram model drops significantly during the first 20 chords and continues to fall until around chord 60, after which it stagnates. This indicates that the chord progressions in the beginning of the songs are easier to predict by a static model—i.e., more closely follow the general chord progressions the model learned, while those later in the songs deviate more from common patterns.

In comparison, the RNN-based models do not suffer during the first 20 chords. Although they show similar behaviour during the first 20 chords, the LSTM and GRU models also behave differently later in the songs: both the GRU and the LSTM improve from around chord 40, whereas the performance of the simple RNN continues to drop until around chord 60 (similarly to the 5-gram model, but on a higher level).

Since predicting chords later in the songs is more difficult than at the beginning, but the LSTM and the GRU are only negligibly affected by this (they almost recover to their performance at the beginning), we argue that both the GRU and the LSTM models are better capable of adapting to the current song. One explanation for this might be that these models privilege intra-song statistics during prediction. Thus, rather than learning a global, generic model trained on the statistics of an entire corpus, we conjecture that these models also acquire and apply knowledge about the immediate past.

## 6.6 Conclusion

We presented a comprehensive evaluation of chord language models, with a focus on RNN-based architectures. We discovered the best performing hyper-parameters, and trained and evaluated the models on a large compound dataset consisting of various genres. Our results show that (i) all RNN-based models outperform *n*-gram models, (ii) gated RNN cells such as the LSTM cell or the GRU outperform simple RNNs, and (iii) that both LSTM and GRU networks seem to adapt their predictions to what they observed in the past.

We conjecture that to improve the recently stagnant chord recognition results, we need models with a better understanding of music than has been demonstrated in previous state-of-the-art systems. This work is a first step towards this goal.

# 7 Integrating Language Models with Acoustic Models

We know the importance of language models in domains such as speech recognition, where hierarchical grammar, pronunciation and context models reduce word error rates by a large margin. However, the degree to which higher-order language models improve chord recognition results yet remains unexplored. To shed light on this topic, we will introduce a probabilistic model combines the frame-wise predictions of an acoustic model with chord-level harmonic language models. We will then apply *n*-gram chord language models on top of an neural network based acoustic model. We will also evaluate to which degree this combination suffers from acoustic model over-confidence, a typical problem with neural acoustic models (Chorowski and Jaitly, 2016).

The content of this chapter is mainly based on Korzeniowski and Widmer (2018a), with the description of the probabilistic model taken from Korzeniowski and Widmer (2018c).

## 7.1 A Probabilistic Chord Recognition Model

Until now, we have discussed acoustic models and language models separately. We will now connect these two themes into an integrated chord recognition system. In the following, we will develop a probabilistic model that allows for combining an acoustic model with explicit modelling of chord transitions and chord durations. This allows us to deploy language models on the *chord level*, not the frame level.

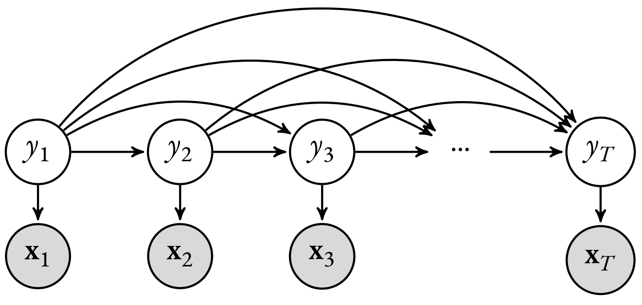


**Figure 7.1:** Generative chord sequence model. Each chord label $y_t$ depends on all previous labels $y_{1:t-1}$.

Chord recognition is a sequence labelling task, i.e. we need to assign a categorical label $y_t \in \mathcal{Y}$ (a chord from a chord alphabet) to each member of the observed sequence $\mathbf{x}_t$ (an audio frame), such that $y_t$ is the harmonic interpretation of the music represented by $\mathbf{x}_t$. Formally,

$$\hat{y}_{1:T} = \underset{\mathbf{y}_{1:T}}{\operatorname{argmax}} P\left(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}\right). \tag{7.1}$$

Assuming a generative structure as shown in Fig. 7.1, the probability distribution factorises as

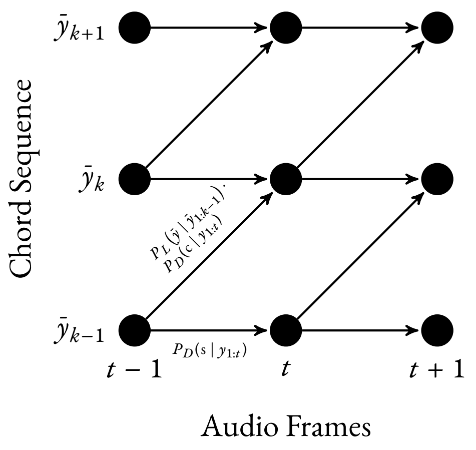
$$P(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}) \propto \prod_t \frac{1}{P(y_t)} P_A\left(y_t \mid \mathbf{x}_t\right) P_T\left(y_t \mid y_{1:t-1}\right), \tag{7.2}$$

where $P_A$ is the acoustic model, $P_T$ the temporal model, and $P(y_t)$ the label prior which we assume to be uniform, following Renals et al. (1994).

The temporal model $P_T$ predicts the chord symbol of each audio frame. As discussed in Chapter 5, this prevents both finite-context models (such as HMMs or CRFs) and unrestricted models (such as RNNs) to learn the underlying musical language of harmony, modelling the symbolic chord sequence on a frame-wise basis is dominated by self-transitions. To enable this, we disentangle $P_T$ into a *harmonic language model* $P_L$ and a *chord duration model* $P_D$, where the former models the harmonic progression of a piece, and the latter models the duration of chords.

The language model $P_L$ is defined as $P_L\left(\bar{y}_k \mid \bar{y}_{1:k-1}\right)$, where $\bar{y}_{1:k} = C\left(y_{1:t}\right)$, and $C\left(\cdot\right)$ is a sequence compression mapping that removes all consecutive duplicates of a chord (e.g. $C\left((C, C, F, F, G)\right) = (C, F, G)$). The frame-wise labels $y_{1:t}$ are thus reduced to chord *changes*, and $P_L$ can focus on modelling these.

The duration model $P_D$ is defined as $P_D\left(s_t \mid y_{1:t-1}\right)$, where $s_t \in \{\mathrm{c}, \mathrm{s}\}$ indicates whether the chord changes (c) or stays the same (s) at time $t$. $P_D$ thus



$\bar{y}_{k+1}$
$\bar{y}_k$
$\bar{y}_{k-1}$
Chord Sequence
$P_L(\bar{y} \mid \bar{y}_{1:k-1}) \cdot$
$P_D(\mathrm{c} \mid y_{1:t})$
$P_D(\mathrm{s} \mid y_{1:t})$
$t-1$
$t$
$t+1$
Audio Frames

**Figure 7.2:** Chord-time lattice representing the temporal model $P_T$, split into a language model $P_L$ and duration model $P_D$. Here, $\bar{y}_{1:K}$ represents a concrete chord sequence. For each audio frame, we move along the time-axis to the right. If the chord changes, we move diagonally to the upper right. This corresponds to the first case in Eq. 7.3. If the chord stays the same, we move only to the right. This corresponds to the second case of the equation.

only predicts whether the chord will change or not, but not which chord will follow—this is left to the language model $P_L$. This definition allows $P_D$ to consider the preceding *chord labels* $y_{1:t-1}$; in practice, we restrict the model to only depend on the preceding chord *changes*, i.e. $P_D\left(s_t \mid s_{1:t-1}\right)$. Exploring more complex models of harmonic rhythm is left for future work.

Using these definitions, the temporal model $P_T$ factorises as

$$P_T\left(y_t \mid y_{1:t-1}\right) = \begin{cases} P_L\left(\bar{y}_k \mid \bar{y}_{1:k-1}\right) P_D\left(\mathrm{c} \mid y_{1:t-1}\right) & \text{if } y_t \neq y_{t-1} \\ P_D\left(\mathrm{s} \mid y_{1:t-1}\right) & \text{else} \end{cases} . \tag{7.3}$$

The chord progression can then be interpreted as a path through a chord-time lattice as shown in Fig. 7.2.

This model cannot be decoded efficiently at test-time because each $y_t$ depends on all predecessors. We will thus use either models that restrict these connections to a finite past (such as higher-order Markov models in this chapter) or use approximate inference methods for other models (such as RNNs, used in Chapter 8).

The proposed probabilistic structure opens various possibilities for further work: we could explore better language models (e.g. by using more sophisticated smoothing techniques, or RNN-based models), or develop more intelligent duration models (e.g. taking into account the tempo and the harmonic rhythm of a song). While we focus on simple duration models and finite-context language models in this chapter, we will explore more complex models in Chapter 8.

## 7.2 Models

### 7.2.1 Acoustic Model

The acoustic model used in this chapter is a minor variation of the one introduced in Chapter 4. It is a VGG-style (Simonyan and Zisserman, 2014) fully convolutional neural network with 3 convolutional blocks: the first consists of 4 layers of 32 3×3 filters, followed by $2 \times 1$ max-pooling in frequency; the second comprises 2 layers of 64 such filters followed by the same pooling scheme; the third is a single layer of 128 12×9 filters. Each of the blocks is followed by feature-map-wise dropout with probability 0.2, and each layer is followed by batch normalisation (Ioffe and Szegedy, 2015) and an ELU activation function (Clevert et al., 2016). Finally, a linear convolution with 25 1×1 filters followed by global average pooling and a softmax produces the chord class probabilities $P_A(y_k \mid \mathbf{x}_k)$. The main differences to the model in Chapter 4 are thus the use of ELU activation functions and that dropout is performed per feature map (selected feature maps are deactivated completely).

The input to the network is a 1.5 s patch of a quarter-tone spectrogram computed using a logarithmically spaced triangular filter bank. Concretely, we process the audio at a sample rate of 44 100 Hz using the STFT with a frame size of 8192 and a hop size of 4410. Then, we apply to the magnitude of the STFT a triangular filter bank with 24 filters per octave between 65 Hz and 2 100 Hz. Finally, we take the logarithm of the resulting magnitudes to compress the input range. See Chapter 4 for a detailed description of the input processing and training schemes.

Neural networks tend to produce overconfident predictions, which leads to probability distributions with high peaks. This causes a weaker training signal because the loss function saturates, and makes the acoustic model dominate the language model at test time (Chorowski and Jaitly, 2016). Here, we investigate two approaches to mitigate these effects: using a temperature softmax in the classification layer of the network, and training using smoothed labels.

The temperature softmax replaces the regular softmax activation function at test time with

$$\sigma(\mathbf{z})_j = \frac{e^{z_j/\tau}}{\sum_{k=1}^{K} e^{z_k/\tau}}, \qquad (7.4)$$

where $\mathbf{z}$ is a real vector. High values for $\tau$ make the resulting distribution smoother. With $\tau = 1$, the function corresponds to the standard softmax. The advantage of this method is that the network does not need to be retrained.

Target smoothing, on the other hand, trains the network with with a smoothed version of the target labels. Here, we explore three ways of smoothing: *uniform*

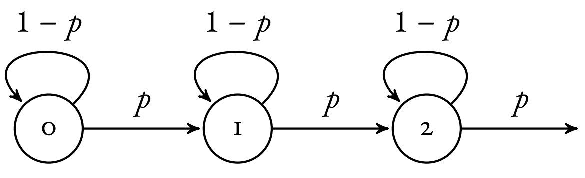


**Figure** 7.3: Markov chain modelling the duration of a chord segment. The probability of staying in one of the depicted states follows the negative binomial distribution.

*smoothing*, where a proportion of $1 - \beta$ of the correct probability is assigned uniformly to the other classes; *unigram smoothing*, where the smoothed probability is assigned according to the class distribution in the training set (Szegedy et al., 2015); and *target smearing*, where the target is smeared in time using a running mean filter. The latter is inspired by a similar approach by Ullrich et al. (2014) to counteract inaccurate segment boundary annotations.

### 7.2.2 Language Model

We designed the temporal model in Eq. 7.3 in a way that enables chord changes to be modelled explicitly via $P_L(\bar{y}_k \mid \bar{y}_{1:k-1})$ (recall that $\bar{y}_{1:K}$ is the compressed chord sequence, i.e. without repeated symbols). This formulation allows to use all past chords to predict the next. While this is a powerful and general notion, it prohibits efficient exact decoding of the sequence. We would have to rely on approximate methods to find $\hat{y}_{1:T}$. However, we can restrict the number of past chords the language model can consider, and use higher-order Markov models for exact decoding. To achieve that, we use $n$-grams for language modelling in this chapter.

$n$-gram language models are Markovian probabilistic models that assume only a fixed-length history (of length $n - 1$) to be relevant for predicting the next symbol. This fixed-length history allows the probabilities to be stored in a table, with its entries computed using maximum-likelihood estimation (MLE)—i.e., by counting occurrences in the training set.

With larger $n$, the sparsity of the probability table increases exponentially, because we only have a finite number of $n$-grams in our training set. We tackle this problem using Lidstone smoothing, and add a pseudo-count $\alpha$ to each possible $n$-gram. We determine the best value for $\alpha$ for each model using the validation set.

### 7.2.3 Duration Model

The focus of this chapter is on how to meaningfully incorporate chord language models beyond simple first-order transitions. We thus use only a simple duration

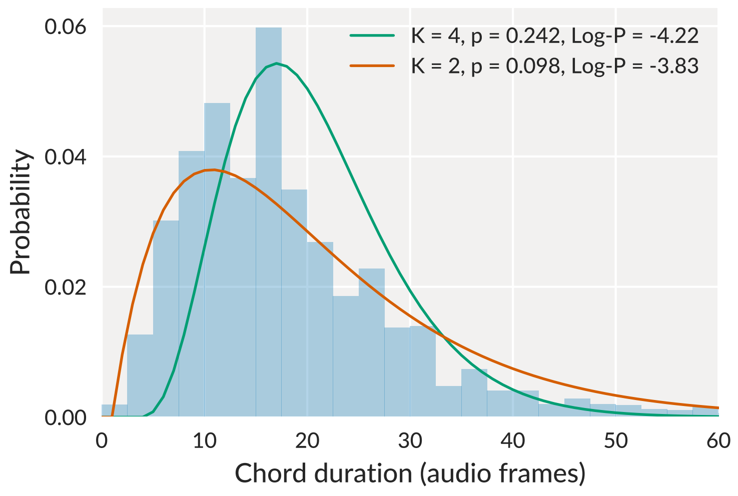


**Figure 7.4:** Histogram of chord durations with two configurations of the negative binomial distribution. The log-probability is computed on a validation fold.

model based on the negative binomial distribution, with the probability mass function

$$P(k) = \binom{k + K - 1}{K - 1} p^K (1 - p)^k,$$

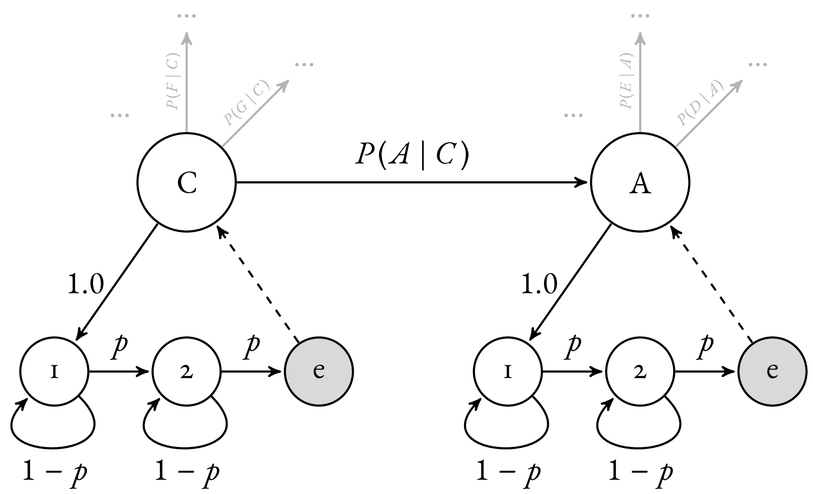
where $K$ is the number of failures, $p$ the failure probability, and $k$ the number of successes given $K$ failures. For our purposes, $k + K$ is the length of a chord in audio frames.

The main advantage of this choice is that a negative binomial distribution is easily represented using only few states in a HMM (see Fig. 7.3), while still reasonably modelling the length of chord segments (see Fig. 7.4). For simplicity, we use the same duration model for all chords. The parameters ($K$, the number of states used for modelling the duration, and $p$, the probability of moving to the next state) are estimated using MLE.

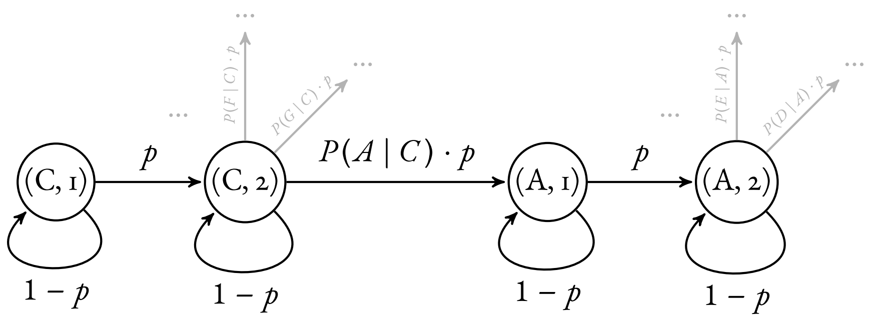
### 7.2.4 Model Integration

If we combine an $n$-gram language model with a negative binomial duration model using Eq. 7.3, the temporal model $P_T$ becomes a Hierarchical Hidden Markov Model (Fine et al., 1998) with a higher-order Markov model on the top level (the language model) and a first-order HMM at the second level (see Fig. 7.5a). We can translate the hierarchical HMM into a first-order HMM; this will allow us to use many existing and optimised HMM implementations.

To this end, we first transform the higher-order HMM on the top level into a first-order one as shown e.g. by Hadar and Messer (2009): we factor



**(a)** First-Order Hierarchical HMM.



**(b)** Flattened version of the First-Order Hierarchical HMM.

**Figure 7.5:** Exemplary Hierarchical HMM and its flattened version. We left out incoming and outgoing transitions of the chord states for clarity (except C $\rightarrow$ A and the ones indicated in grey). The model uses 2 states for duration modelling, with "e" referring to the final state on the duration level (see Fine et al. (1998) for details). Although we depict a first-order language model here, the same transformation works for higher-order models.

the dependencies beyond first-order into the HMM state, considering that self-transitions are impossible as

$$\mathcal{Y}_n = \left\{ (y_1, \dots, y_n) : y_i \in \mathcal{Y}, y_i \neq y_{i+1} \right\},$$

where $n$ is the order of the $n$-gram model. Semantically, $(y_1, \dots, y_n)$ represents chord $y_1$, having seen $y_2, \dots, y_n$ in the immediate past. This increases the number of states from $|\mathcal{Y}|$ to $|\mathcal{Y}| \cdot (|\mathcal{Y}| - 1)^{n-1}$.

We then flatten out the hierarchical HMM by combining the state spaces of both levels as $\mathcal{Y}_N \times [1..K]$, and connecting all incoming transitions of a chord state to the corresponding first duration state, and all outgoing transitions from the last duration state (where the outgoing probabilities are multiplied by $p$). Formally,

$$\mathcal{Y}_n^{(K)} = \left\{ (\mathbf{y}, k) : \mathbf{y} \in \mathcal{Y}_n, k \in [1..K] \right\} ,$$

with the transition probabilities defined as

$$\begin{aligned} P((\mathbf{y}, k) \mid (\mathbf{y}, k)) &= 1 - p, \\ P((\mathbf{y}, k+1) \mid (\mathbf{y}, k)) &= p, \\ P((\mathbf{y}, 1) \mid (\mathbf{y}', K)) &= P_L(y_1 \mid y_{2:n}) \cdot p, \end{aligned}$$

where $y_{2:n} = y'_{1:n-1}$. All other transitions have zero probability. Fig. 7.5b shows the HMM from Fig. 7.5a after the transformation.

The resulting model is similar to a higher-order duration-explicit HMM (DHMM). The main difference is that we use a compact duration model that can assign duration probabilities using few states, while standard DHMMs do not scale well if longer durations need to be modelled (their computation increases by a factor of $D^2/2$, where $D$ is the longest duration to be modelled (Rabiner, 1989)). For example, Chen et al. (2012) uses first-order DHMMs to decode beat-synchronised chord sequences, with $D = 20$. In our case, we would need a much higher $D$, since our model operates on the frame level, which would result in a prohibitively large state space. In comparison, our duration models use $K = 2$, which significantly reduces the computational burden.

## 7.3 Experiments

Our experiments aim at uncovering (i) if acoustic model overconfidence is a problem in this scenario, (ii) whether smoothing techniques can mitigate it, and (iii) whether and to which degree chord language modelling improves chord recognition results. To this end, we investigated the effect of various parameters: softmax temperature $\tau \in \{0.5, 1.0, 1.3, 2.0\}$, smoothing type (uniform, unigram, and smear), smoothing intensity $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ and smearing width $w \in \{3, 5, 10, 15\}$, and the language model order $N \in \{2, 3, 4\}$.

The experiments were carried out using 4-fold cross-validation on a compound dataset consisting of the following sub-sets: *Isophonics*[1]*:* 180 songs by the Beatles, 19 songs by Queen, and 18 songs by Zweieck, 10:21 hours of audio; *RWC Popular* (Goto et al., 2002): 100 songs in the style of American and Japanese pop music, 6:46 hours of audio; *Robbie Williams* (Di Giorgi et al., 2013): 65 songs by

[1]http://isophonics.net/datasets

Robbie Williams, 4:30 of audio; and *McGill Billboard* (Burgoyne et al., 2011): 742 songs sampled from the American billboard charts between 1958 and 1991, 44:42 hours of audio. The compound dataset thus comprises 1125 unique songs, and a total of 66:21 hours of audio.

We focus on the major/minor chord vocabulary (i.e. major and minor chords for each of the 12 semitones, plus a "no-chord" class, totalling 25 classes). The evaluation measure we are interested in is thus the weighted chord symbol recall of major and minor chords, WCSR $= {}^{t_c}/_{t_a}$, where $t_c$ is the total time the our system recognises the correct chord, and $t_a$ is the total duration of annotations of the chord types of interest.

### 7.3.1 Results and Discussion

We analyse the interactions between temperature, smoothing, and language modelling in Fig. 7.6 and Fig. 7.7. Uniform smoothing seems to perform best, while increasing the temperature in the softmax is unnecessary if smoothing is used. On the other hand, target smearing performs poorly; it is thus not a proper way to cope with uncertainty in the annotated chord boundaries.

The results indicate that in our scenario, acoustic model overconfidence is not a major issue. The reason might be that the temporal model we use in this work allows for exact decoding. If we were forced to perform approximate inference (e.g. by using a RNN-based language model), this overconfidence could cut off promising paths early. Target smoothing still exhibits a positive effect during the training of the acoustic model, and can be used to fine-balance the interaction between acoustic and temporal models.

Further, we see consistent improvement the stronger the language model is (i.e., the higher $n$ is). Although we were not able to evaluate models beyond $n = 4$ for all configurations, we ran a 5-gram model on the best configuration for $n = 4$. The results are shown in Table 7.1.

| *None* | *Dur.* | *2-gram* | *3-gram* | *4-gram* | *5-gram* |
|---|---|---|---|---|---|
| 78.51 | 79.33 | 79.59 | 79.69 | 79.81 | 79.88 |

**Table 7.1:** WCSR for the compound dataset. For these results, we use a softmax temperature of $T = 1.0$ and uniform smoothing with $\beta = 0.9$.

Although consistent, the improvement is marginal compared to the effect language models show in other domains such as speech recognition. There are two possible interpretations of this result: (i) even if modelled explicitly, chord language models contribute little to the final results, and the most important part is indeed modelling the chord duration; and (ii) the language models we used

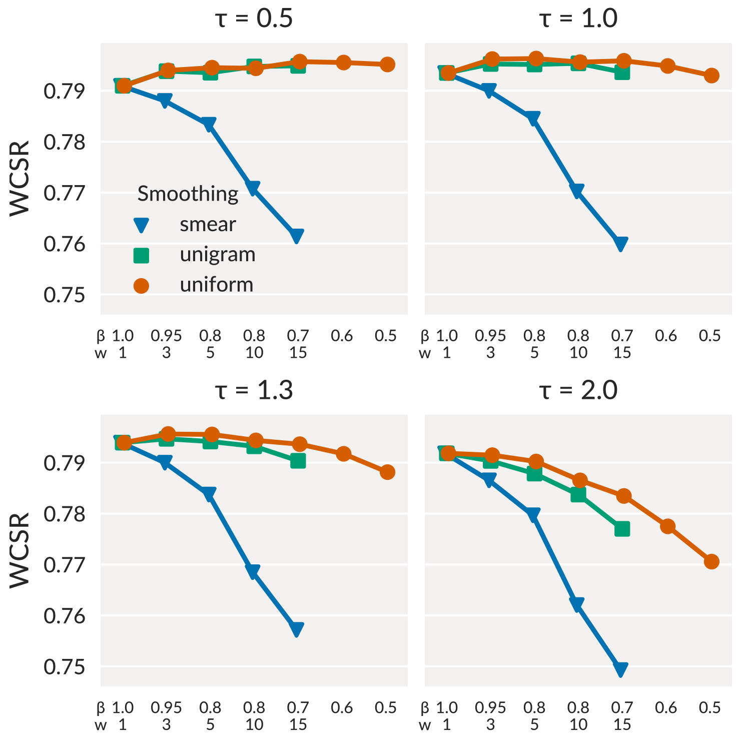


**Figure 7.6:** The effect of temperature $\tau$, smoothing type, and smoothing intensity on the WCSR. The x-axis shows the smoothing intensity: for uniform and unigram smoothing, $\beta$ indicates how much probability mass was kept at the true label during training; for target smearing, $w$ is the width of the running mean filter used for smearing the targets in time. For these results, a 2-gram language model was used, but the outcomes are similar for other language models. The key observations are the following: (i) target smearing is always detrimental; (ii) uniform smoothing works slightly better than unigram smoothing (in other domains, authors report the contrary (Chorowski and Jaitly, 2016)); and (iii) smoothing improves the results, however, excessive smoothing is harmful in combination with higher softmax temperatures (a relation we explore in greater detail in Fig. 7.7).

are simply not good enough to make a major difference. While the true reason yet remains unclear, the structure of the temporal model we propose enables the investigation of both possibilities, because it makes their contributions explicit.

Finally, our results confirm the importance of duration modelling as indicated by Chen et al. (2012). Although the duration model we use here is simplistic, it improves results considerably. However, in further informal experiments, we found that it underestimates the probability of long chord segments, which impairs results. This indicates that there is still potential for improvement in this part of our model.

## 7.4 Conclusion

In this chapter, we proposed a probabilistic structure for the temporal model of chord recognition systems. This structure disentangles a chord language model from a chord duration model. We then applied *n*-gram chord language models within this structure and evaluated various properties of the system. The key outcomes are that (i) acoustic model overconfidence plays only a minor role (but target smoothing still improves the acoustic model), (ii) chord duration modelling improves results considerably, which confirms prior studies (Chen et al., 2012; Cho and Bello, 2014), and (iii) while employing *n*-gram models also improves the results, their effect is marginal compared to other domains such as speech recognition.

Why is this the case? Static *n*-gram models might only capture global statistics of chord progressions, and these could be too general to guide and correct predictions of the acoustic model. More powerful models may be required. As shown in Chapter 6, RNN-based chord language models are able to adapt to the currently processed song, and thus might be more suited for the task at hand.

The proposed probabilistic structure thus opens various possibilities for future work. We could explore better language models, e.g. by using more sophisticated smoothing techniques, RNN-based models (see Chapter 8), or probabilistic models that take into account the key of a song (the probability of chord transitions varies depending on the key). More intelligent duration models could take into account the tempo and harmonic rhythm of a song (the rhythm in which chords change, see also Chapter 8). Using the model presented in this chapter, we could then link the improvements of each individual model to improvements in the final chord recognition score.

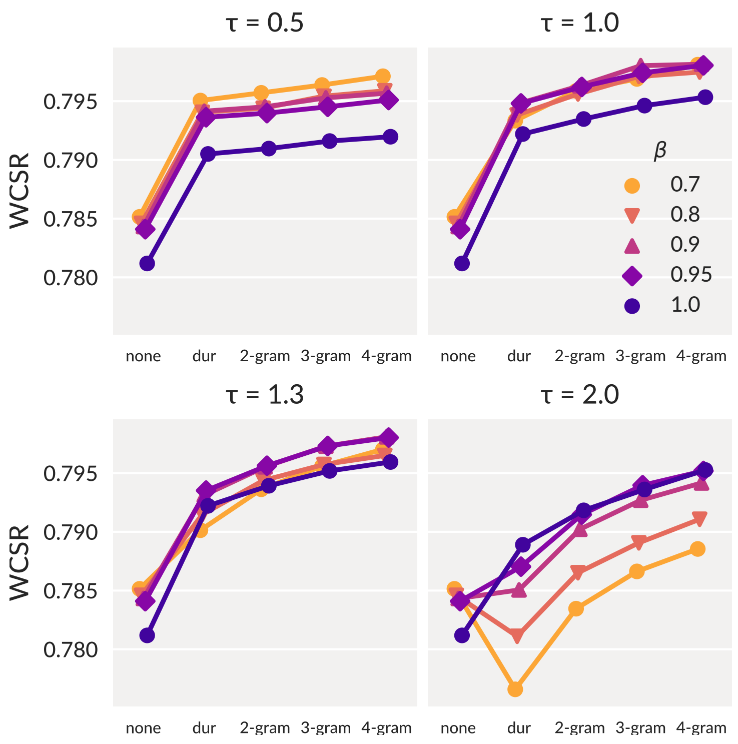


**Figure** 7.7: Interaction of temperature $\tau$, smoothing intensity $\beta$ and language model with respect to the WCSR. We show four language model configurations: *none* means using the predictions of the acoustic model directly; *dur* means using the chord duration model, but no chord language model; and *n-gram* means using the duration model with the respective language model. Here, we only show results using uniform smoothing, which turned out to be the best smoothing technique we examined in this chapter (see Fig. 7.6). We observe the following: (i) Even simple duration modelling accounts for the majority of the improvement (in accordance with (Chen et al., 2012)). (ii) Chord language models further improve the results—the stronger the language model, the bigger the improvement. (iii) Temperature and smoothing interact: at $\tau = 1$, the amount of smoothing plays only a minor role; if we lower $\tau$ (and thus make the predictions more confident), we need stronger smoothing to compensate for that; if we increase both $\tau$ and the smoothing intensity, the predictions of the acoustic model are over-ruled by the language model, which shows to be detrimental. (iv) Smoothing has an additional effect during the training of the acoustic model that cannot be achieved using post-hoc changes in softmax temperature. Unsmoothed models never achieve the best result, regardless of $\tau$.

# 8 Harmonic Language and Duration Modelling with Recurrent Neural Networks

In the previous chapter, we devised a probabilistic chord recognition framework that allows for integrating harmonic language models with acoustic models. We then evaluated finite-context language models with a simple, static duration model within this framework. Let us now go a step further, and develop more realistic chord language models and chord duration models based on RNNs. We will explore how these models affect chord recognition results, and show that the proposed integrated model out-performs existing temporal models.

This work has been previously published in Korzeniowski and Widmer (2018c).

## 8.1 Models

The chord recognition model represented by Equations 7.2 and 7.3 requires three sub-models: an acoustic model $P_A$ that predicts a chord distribution from each audio frame, a duration model $P_D$ that predicts when chords change, and a language model $P_L$ that predicts the progression of chords in the piece.

### 8.1.1 Acoustic Model

The acoustic model $P_A$ is the same convolutional neural network as in Chapter 7. As we have seen, slight target smoothing improves the results, as does using a temperature softmax (see Eq. 7.4). We use both techniques: first, we

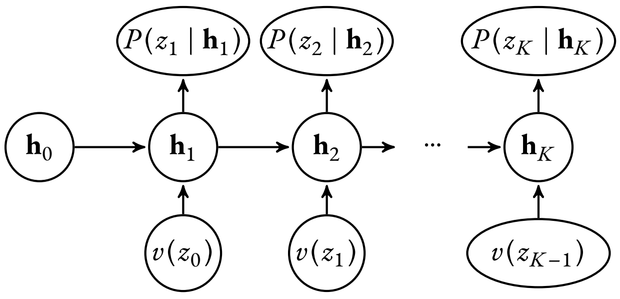


**Figure 8.1:** Sketch of a RNN used for next step prediction, where $z_k$ refers to an arbitrary categorical input, $v(\cdot)$ is a (learnable) input embedding vector, and $\mathbf{h}_k$ the hidden state at step $k$. Arrows denote matrix multiplications followed by a non-linear activation function. The input is padded with a dummy input $z_0$ in the beginning. The network then computes the probability distribution for the next symbol.

train the model using uniform smoothing (i.e. we assign 0.1 of the probability mass to other classes during training); second, during inference, we apply the temperature softmax function with $\tau = 1.3$.

### 8.1.2 Language Model

The language model $P_L$ predicts the next chord, regardless of its duration, given the chord sequence it has previously seen. As shown in Chapter 6, RNN-based models perform better than n-gram models at this task.

We follow the set-up introduced by Mikolov et al. (2010) and use a recurrent neural network for next-chord prediction. The network's task is to compute a probability distribution over all possible next chord symbols, given the chord symbols it has observed before (see Chapter 6 for details. Figure 8.1 shows an RNN in a general next-step prediction task. In our case, the inputs $z_k$ are the chord symbols given by $C\left(y_{1:T}\right)$.

We will describe in detail the network's hyper-parameters in Section 8.2, where we will also evaluate the effect the language models have on chord recognition.

### 8.1.3 Duration Model

The duration model $P_D$ predicts whether the chord will change in the next time step. This corresponds to modelling the duration of chords. Existing temporal models induce implicit duration models: for example, an HMM implies an exponential chord duration distribution (if one state is used to model a chord), or a negative binomial distribution (if multiple left-to-right states are used per chord). However, such duration models are simplistic, static, and do not adapt to the processed piece.

An explicit duration model has been explored by Chen et al. (2012), where beat-synchronised chord durations were stored as discrete distributions. Their approach is useful for beat-synchronised models, but impractical for frame-wise models—the probability tables would become too large, and data too sparse to estimate them. Since our approach avoids the potentially error-prone beat synchronisation, the approach of Chen et al. (2012) does not work in our case.

Instead, we opt to use recurrent neural networks to model chord durations. These models are able to adapt to characteristics of the processed data (as shown in Chapter 6), and have shown great potential in processing periodic signals (Böck and Schedl, 2011) (and chords do change periodically within a piece). To train an RNN-based duration model, we set up a next-step-prediction task, identical in principle to the set-up for harmonic language modelling: the network has to compute the probability of a chord change in the next time step, given the chord changes it has seen in the past. We thus simplify $P_D(s_t \mid y_{1:t-1}) \mathrel{\widehat{=}} P_D(s_t \mid s_{1:t-1})$, as mentioned earlier. Again, see Fig. 8.1 for an overview (for duration modelling, replace $z_k$ with $s_t$).

In Section 8.2, we will describe in detail the hyper-parameters of the networks we employed, and compare the properties of various settings to baseline duration models. We will also assess the impact on the duration modelling quality on the final chord recognition result.

### 8.1.4 Model Integration

Dynamic models such as RNNs have one main advantage over their static counter-parts (e.g. $n$-gram models for language modelling or HMMs for duration modelling): they consider all previous observations when predicting the next one. As a consequence, they are able to adapt to the piece that is currently processed—they assign higher probabilities to sub-sequences of chords they have seen earlier (see Chapter 6), or predict chord changes according to the harmonic rhythm of a song (see Section 8.2.3). The flip side of the coin is, however, that this property prohibits the use of dynamic programming approaches for efficient decoding. We cannot exactly *and* efficiently decode the best chord sequence given the input audio.

Hence we have to resort to approximate inference. In particular, we employ *hashed beam search* (Sigtia et al., 2015) to decode the chord sequence. General beam search restricts the search space by keeping only the $N_b$ best solutions up to the current time step. (In our case, the $N_b$ best paths through all possible chord-time lattices, see Fig. 7.2.) However, as pointed out by Sigtia et al. (2015), the beam might saturate with almost identical solutions, e.g. the same chord sequence differing only marginally in the times the chords change. Such pathological cases may impair the final estimate. To mitigate this problem, hashed beam search forces the tracked solutions to be diverse by pruning similar solutions with lower probability.

The similarity of solutions is determined by a task-specific *hash function*. For our purpose, we define the hash function of a solution to be the last $N_h$ chord symbols in the sequence, regardless of their duration; formally, the hash function $f_h\left(y_{1:t}\right) = \bar{y}_{(k-N_h):k}$. (Recall that $\bar{y}_{1:k} = C\left(y_{1:t}\right)$.) In contrast to the hash function originally proposed by Sigtia et al. (2015), which directly uses $y_{(t-N_h):t}$, our formulation ensures that sequences that differ only in timing, but not in chord sequence, are considered similar.

To summarise, we approximately decode the optimal chord transcription as defined in Eq. 7.1 using hashed beam search, which at each time step keeps the best $N_b$ solutions, and at most $N_s$ similar solutions.

## 8.2 Experiments

In our experiments, we will first evaluate harmonic language and duration models individually. Here, we will compare the proposed models to common baselines. Then, we will integrate these models into the chord recognition framework we outlined in Section 7.1, and evaluate how the individual parts interact in terms of chord recognition score.

### 8.2.1 Data

We use the following datasets in 4-fold cross-validation. *Isophonics*[1]*:* 180 songs by the Beatles, 19 songs by Queen, and 18 songs by Zweieck, 10:21 hours of audio; *RWC Popular* (Goto et al., 2002): 100 songs in the style of American and Japanese pop music, 6:46 hours of audio; *Robbie Williams* (Di Giorgi et al., 2013): 65 songs by Robbie Williams, 4:30 of audio; and *McGill Billboard* (Burgoyne et al., 2011): 742 songs sampled from the American billboard charts between 1958 and 1991, 44:42 hours of audio. The compound dataset thus comprises 1125 unique songs, and a total of 66:21 hours of audio.

Furthermore, we used the following data sets (with duplicates removed) as additional data for training the language and duration models: 173 songs from the *Rock* corpus (de Clercq and Temperley, 2011); a subset of 160 songs from the *UsPop2002*[2] for which chord annotations are available[3]; 291 songs from *Weimar Jazz*[4], with chord annotations taken from lead sheets of Jazz standards; and *Jay*

[1]http://isophonics.net/datasets

[2]https://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html

[3]https://github.com/tmc323/Chord-Annotations

[4]http://jazzomat.hfm-weimar.de/dbformat/dboverview.html

| | *GRU-512* | *GRU-32* | *4-gram* | *2-gram* |
|---|---|---|---|---|
| log-P | -1.293 | -1.576 | -1.887 | -2.393 |

**Table 8.1:** Language model results: average log-probability of the correct next chord computed by each model.

*Chou* (Deng and Kwok, 2016), a small collection of 29 Chinese pop songs.

We focus on the major/minor chord vocabulary, and following (Cho and Bello, 2014), map all chords containing a minor third to minor, and all others to major. This leaves us with 25 classes: 12 root notes × {major, minor} and the 'no-chord' class.

### 8.2.2 Language Models

The performance of neural networks depends on a good choice of hyper-parameters, such as number of layers, number of units per layer, or unit type (e.g. vanilla RNN, gated recurrent unit (GRU) (Cho et al., 2014) or long short-term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997)). The findings in Chapter 6 provide a good starting point for choosing hyper-parameter settings that work well. However, we strive to find a simpler model to reduce the computational burden at test time. To this end, we perform a grid search in a restricted search space, using the validation score of the first fold. We search over the following settings: number of layers $\in \{1, 2, 3\}$, number of units $\in \{256, 512\}$, unit type $\in \{\text{GRU}, \text{LSTM}\}$, input embedding $\in \{\text{one-hot}, \mathbb{R}^8, \mathbb{R}^{16}, \mathbb{R}^{24}\}$, learning rate $\in \{0.001, 0.005\}$, and skip connections $\in \{\text{on}, \text{off}\}$. Other hyper-parameters were fixed for all trials: we train the networks for 100 epochs using stochastic gradient descent with mini-batches of size 4, employ the Adam update rule (Kingma and Ba, 2015), and starting from epoch 50, linearly anneal the learning rate to 0.

To increase the diversity in the training data, we use two data augmentation techniques, applied each time we show a piece to the network. First, we randomly shift the key of the piece; the network can thus learn that harmonic relations are independent of the key, as in roman numeral analysis. Second, we select a sub-sequence of random length instead of the complete chord sequence; the network thus has to learn to cope with varying context sizes.

The best model turned out to be a single-layer network of 512 GRUs, with a learnable 16-dimensional input embedding and without skip connections, trained using a learning rate of 0.005. We compare this model and a smaller, but otherwise identical RNN with 32 units, to two baselines: a 2-gram model, and a 4-gram model. Both can be used for chord recognition in a higher-order

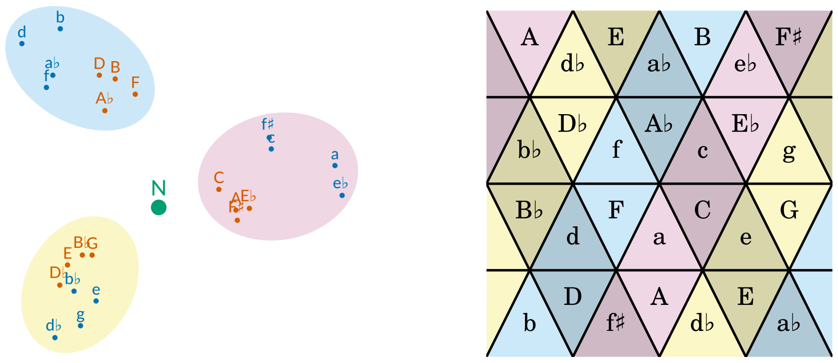


**Figure 8.2:** Chord embedding projected into 2D using PCA (left); Tonnetz of triads (right). The "no-chord" class resides in the centre of the embedding. Major chords are upper-case and orange, minor chords lower-case and blue. Clusters in the projected embedding and the corresponding positions in the Tonnetz are marked in colour. If projected into 3D (not shown here), the chord clusters split into a lower and upper half of four chords each. The chords in the lower halves are shaded in the Tonnetz representation.

HMM, as presented in Chapter 7. We train the *n*-gram models using maximum likelihood estimation with Lidstone smoothing as described in Chapter 6, using the key-shift data augmentation technique (sub-sequence cropping is futile for finite context models). As evaluation measure, we use the average log-probability of predicting the correct next chord. Table 8.1 presents the test results. The GRU models predict chord sequences with much higher probability than the baselines.

When we look into the input embedding $v(\cdot)$, which was learned by the RNN during training from a random initialisation, we observe an interesting positioning of the chord symbols (see Figure 8.2). We found that similar patterns develop for all 1-layer GRUs we tried, and these patterns are consistent for all folds we trained on. We observe (i) that chords form three clusters around the centre, in which the minor chords are farther from the centre than major chords; (ii) that the clusters group major and minor chords with the same root, and the distance between the roots are minor thirds (e.g. C, E, F, A); (iii) that clockwise movement in the circle of fifths corresponds to clockwise movement in the projected embedding; and (iv) that the way chords are grouped in the embedding corresponds to how they are connected in the Tonnetz.

At this time, we cannot provide an explanation for these automatically emerging patterns. However, they warrant a further investigation to uncover why this specific arrangement seems to benefit the predictions of the model.

|  | *GRU-256* | *GRU-16* | *Neg. Binom.* | *Exp.* |
|---|---|---|---|---|
| log-P | −2.014 | −2.868 | −3.946 | −4.003 |

**Table 8.2:** Duration model results: average log-probability of chord durations computed by each model.

### 8.2.3 Duration Models

As for the language model, we performed a grid search on the first fold to find good choices for the recurrent unit type $\in$ {vanilla RNN, GRU, LSTM}, and number of recurrent units $\in \{16, 32, 64, 128, 256\}$ for the LSTM and GRU, and $\{128, 256, 512\}$ for the vanilla RNN. We use only one recurrent layer for simplicity. We found networks of 256 GRU units to perform best; although this indicates that even bigger models might give better results, for the purposes of this study, we think that this configuration is a good balance between prediction quality and model complexity.

The models were trained for 100 epochs using the Adam update rule (Kingma and Ba, 2015) with a learning rate linearly decreasing from 0.001 to 0. The data was processed in mini-batches of 10, where the sequences were cut in excerpts of 200 time steps (20 s). We also applied gradient clipping at a value of 0.001 to ensure a smooth learning progress.

We compare the best RNN-based duration model with two baselines. The baselines are selected because both are implicit consequences of using HMMs as temporal model, as it is common in chord recognition. We assume a single parametrisation for each chord; this ostensible simplification is justified, because simple temporal models such as HMMs do not profit from chord information (Chen et al., 2012; Cho and Bello, 2014). The first baseline we consider is a *negative binomial distribution*. It can be modelled by a HMM using $n$ states per chord, connected in a left-to-right manner, with transitions of probability $p$ between the states (self-transitions thus have probability $1 - p$). The second, a special case of the first with $n = 1$, is an *exponential distribution*; this is the implicit duration distribution used by all chord recognition models that employ a simple 1-state-per-chord HMM as temporal model. Both baselines are trained using maximum likelihood estimation.

To measure the quality of a duration model, we consider the average log-probability it assigns to a chord duration. The results are shown in Table 8.2. We further added results for the simplest GRU model we tried—using only 16 recurrent units—to indicate the performance of small models of this type. We will also use this simple model when judging the effect of duration modelling on the final result in Sec. 8.2.4. Both GRU models clearly out-perform the baselines.

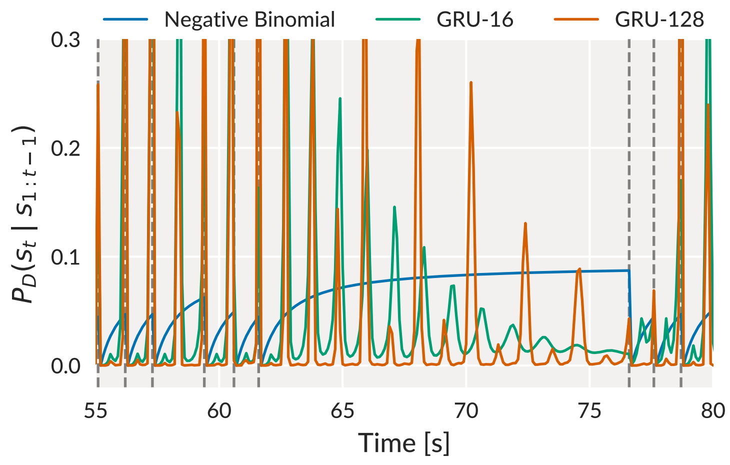


**Figure 8.3:** Probability of chord change computed by different models. Gray vertical dashed lines indicate true chord changes.

Figure 8.3 shows the reason why the GRU performs so much better than the baselines: as a dynamic model, it can adapt to the harmonic rhythm of a piece, while static models are not capable of doing so. We see that a GRU with 128 units predicts chord changes with high probability at periods of the harmonic rhythm. It also reliably remembers the period over large gaps in which the chord did not change (between seconds 61 and 76). During this time, the peaks decay differently for different multiples of the period, which indicates that the network simultaneously tracks multiple periods of varying importance. In contrast, the negative binomial distribution statically yields a higher chord change probability that rises with the number of audio frames since the last chord change. Finally, the smaller GRU model with only 16 units also manages to adapt to the harmonic rhythm; however, its predictions between the peaks are noisier, and it fails to remember the period in the time without chord changes.

### 8.2.4 Integrated Models

The individual results for the language and duration models are encouraging, but only meaningful if they translate to better chord recognition scores. This section will thus evaluate if and how the duration and language models affect the performance of a chord recognition system.

The acoustic model used in these experiments was trained for 300 epochs (with 200 parameter updates per epoch) using a mini-batch size of 512 and the Adam update rule with standard parameters. We linearly decay the learning rate to 0 in the last 100 epochs.

| *Model* | *Root* | *Maj/Min* | *Seg.* |
|---|---|---|---|
| 2-gram / neg. binom. | 81.2 | 79.5 | 80.4 |
| GRU-512 / GRU-256 | 82.1 | 80.5 | 81.4 |

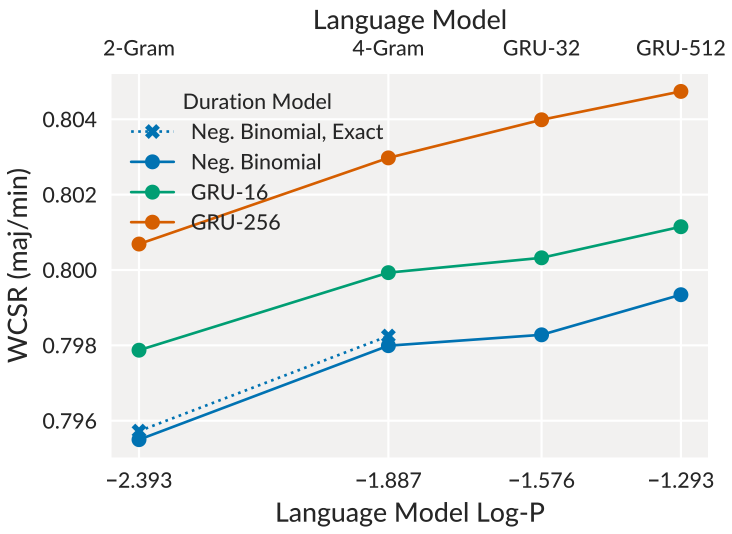
**Table 8.3:** Results of the standard model (2-gram language model with negative binomial durations) compared to the best one (GRU language and duration models).

We compare all combinations of language and duration models presented in the previous sections. For language modelling, these are the GRU-512, GRU-32, 4-gram, and 2-gram models; for duration modelling, these are the GRU-256, GRU-16, and negative binomial models. (We leave out the exponential model, because its results differ negligibly from the negative binomial one.) The models are decoded using the Hashed Beam Search algorithm, as described in Sec. 8.2.4: we use a beam width of $N_b = 25$, where we track at most $N_s = 4$ similar solutions as defined by the hash function $f_h$, where the number of chords considered is set to $N_h = 5$. These values were determined by a small number of preliminary experiments.

Additionally, we evaluate exact decoding results for the n-gram language models in combination with the negative binomial duration distribution. This will indicate how much the results suffer due to the approximate nature of beam search.

As main evaluation metric, we use the weighted chord symbol recall (WCSR) over the major/minor chord alphabet, as defined in (Pauwels and Peeters, 2013). We thus compute WCSR $= {}^{t_c}/_{t_a}$, where $t_c$ is the total duration of chord segments that have been recognised correctly, and $t_a$ is the total duration of chord segments annotated with chords from the target alphabet. We also report chord root accuracy and a measure of segmentation (see Harte (2010), Section 8.3). Table 8.3 compares the results of the standard model (the combination that implicitly emerges in simple HMM-based temporal models) to the best model found in this study. although the improvements are modest, they are consistent, as shown by a paired t-test ($p < 2.4869e - 23$ for all differences).

Figure 8.4 presents the effects of duration and language models on the WCSR. Better language and duration models directly improve chord recognition results, as the WCSR increases linearly with higher log-probability of each model. As this relationship does not seem to flatten out, further improvement of each model type can still increase the score. We also observe that the approximate beam search does not impair the result by much compared to exact decoding (compare the dotted blue line in the upper plot with the solid one).



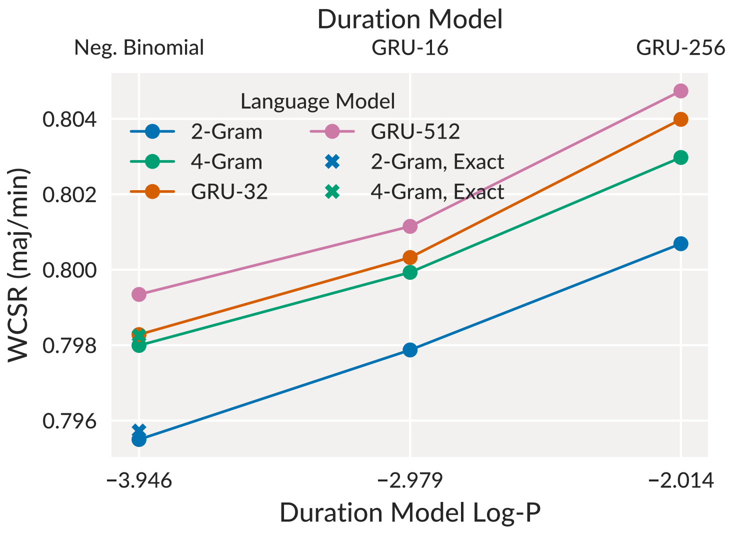


**Figure 8.4:** Effect of language and duration models on the final result. Both plots show the same results from different perspectives.

## 8.3 Conclusion

In this chapter, we developed better duration and language models based on recurrent neural networks, and employed them in the probabilistic chord recognition framework presented in Chapter 7. We illustrated why the RNN-based duration models perform better and are more meaningful than their static counterparts implicitly employed in HMMs. (For a similar investigation for chord language models, see Chapter 6.) Finally, we showed that improvements in each of these models directly influence chord recognition results.

We hope that our contribution facilitates further research in harmonic language and duration models for chord recognition. These aspects have been neglected because they did not show great potential for improving the final result (Chen et al., 2012; Cho and Bello, 2014). However, as argued in the preceding chapters, we believe that this was due to the improper assumption that temporal models applied on the time-frame level can appropriately model musical knowledge. The results presented here indicate that chord transitions modelled on the chord level, and connected to audio frames via strong duration models, indeed have the capability to improve chord recognition results.

# Part II

# Key Classification

# 9
# Key Classification: An Overview

## 9.1 The State of the Art

The musical key is the highest-level harmonic representation in Western tonal music. The key of a piece defines its harmonic centre, gives meaning to its harmonic progression, and provides the backdrop for the build-up and release of harmonic tension. It thus plays a central role in understanding the semantic content of a piece. Such understanding drives not only theoretical analyses of music, but is also relevant for modern music creators, who mix samples from various different pieces that fit well harmonically into a new composition.

However, deriving the key of a musical piece is a demanding task that only experts can perform. It is thus impractical to annotate large music collections by hand. Therefore, if we want to automatically detect tension and its release in musical audio, if we want to find similarities in harmonic structure in recordings of songs, or if we want to assist disc jockeys or creators of electronic music in finding appropriate musical samples to work with, we need reliable methods that extract the key from a musical piece.

Most key classification systems (e.g. Noland and Sandler (2007); Pauws (2004); Temperley (1999); **?**) conform to the same principle: first, they extract a time-frequency representation of the audio (such as a spectrogram or a constant-q-transform), and filter out nuisances (such as transient noise or de-tuning); then, they map this representation to pitch-class-profiles (or, chroma vectors) in order to be invariant to octaves and timbre; finally, they aggregate these features over time, and match them with template vectors for each key.

Such systems typically report a single *global key* for a piece. This limitation is reasonable for a variety of genres (like pop/rock or electronic music), but fails to cope with pieces that contain key modulations (as common in classical music). Other drawbacks of such approaches include that key templates differ for different musical genres (Faraldo et al., 2016) and favour one key mode over another (Albrecht and Shanahan, 2013). This leads to key classification systems that perform well only on the musical styles they were designed for. Although there are attempts to address these issues (Bernardes et al., 2017), ideally, we would want a model that handles different kinds of input autonomously, and does not need human intervention to e.g. balance mode probabilities.

Another line of work considers estimating key and chords simultaneously, e.g. (Di Giorgi et al., 2013; Mauch and Dixon, 2010b; Ni et al., 2012a; Pauwels and Martens, 2014). Such systems aim at exploiting the musicological relationship between key and chords in order to improve the accuracy of both. They typically estimate *local keys*, and are thus able to cope with key modulations. However, while they bear the potential to explain the harmonic content of musical audio more holistically, dedicated systems currently seem to achieve better results[1].

As with chords, deriving the key of a piece is a subjective endeavour, particularly if we want to consider key modulations. The exact position of a key change in the piece is debatable, as is the question whether the key changed in the first place. As with chord recognition, I will ignore the ramifications of this subjectivity for the sake of this thesis, and assume the given annotations to be absolute. Since the work presented in the following chapters focuses on the classification of a global key (on which experts tend to agree), I find this simplification appropriate.

## 9.2 Contributions

This part of the thesis will take the task of key classification from hand-crafted processing pipelines to end-to-end learnable models based on modern neural networks. In Chapter 10, we will design a convolutional neural network that mimics traditional key classification pipelines. We will show that such a data-driven model generalises better than previous approaches, but still performs best when trained for a specific genre. In Chapter 11, we will lift this constraint by adapting the training procedure and network architecture of the model. This will result in a key classifier that can be trained to work with multiple genres and perform better than if trained for a specific one.

[1]See results of the yearly MIREX challenges at www.music-ir.org/mirex.

# 10 Musical Key Classification using a Convolutional Neural Network

This chapter targets the classification of a *single global key* for pieces of musical audio. In contrast to previous works, we abandon hand-crafting or tuning elements in the key classification pipeline. Instead, we employ a convolutional neural network that encompasses the three stages of pre-processing, feature extraction, and classification. Although neural networks have been used for key classification, the existing approaches rely on a hand-crafted feature extraction stage: for example, Sun et al. (2009) feeds a pitch class distribution matrix into a neural network classifier; Dieleman et al. (2011) uses beat-aligned chroma and timbre features as input. Our system operates directly on the spectrogram, and it can estimate all its parameters from the data. To our knowledge, this is the first work that replaces the complete key classification pipeline with a model that can be optimised in an end-to-end manner[1].

This work has been first published in Korzeniowski and Widmer (2017a).

## 10.1 Method

Our system consists of two steps: first, we compute from the audio a logarithmically filtered log-magnitude spectrogram. This process is detailed in Sec. 10.1.1.

[1]One might argue that (Ni et al., 2012a) is also an "end-to-end" system that detects both chords and keys. However, they used feature extraction methods heavily based on expert knowledge (tuning correction, harmonic-percussive source separation, beat synchronisation, frequency-split chroma computation), and trained a dynamic Bayesian network whose structure is based on domain knowledge. Also, its key classification performance was never evaluated.

Then, we feed this time-frequency representation to the convolutional neural network, described in Sec. 10.1.2, for classification.

### 10.1.1 Input Processing

We input a spectral representation of the audio to our models. Based on our previous work on extracting harmonic information from audio as described in Chapter 3 and 4, we first compute from the audio the magnitude spectrogram $|\mathbf{S}|$ (frame size of 8 192 at 5 frames per second, where the sample rate is 44.100 kHz); then, we apply a filter bank $\mathbf{B}_{\log}^{\Delta}$ with logarithmically spaced triangular filters (24 bands per octave, from 65 Hz to 2 100 Hz), which results in a time-frequency representation in which the fundamental frequencies of notes are spaced linearly; finally, we logarithmise the magnitudes of the filtered spectrogram to compress the value range, resulting in the logarithmically filtered log-magnitude spectrogram

$$\mathbf{Q} = \log\left(1 + \mathbf{B}_{\log}^{\Delta} |\mathbf{S}|\right).$$

This representation is similar to a constant-q transform, but is much cheaper to compute. Additionally, as shown by Kelz et al. (2016), the constant-q transform does not necessarily lead to better results in tasks relying on pitch information.

### 10.1.2 Model

The proposed neural network is designed to encompass all stages of the classic key classification pipeline: a pre-processing stage of convolutional layers, a dense layer that projects the feature maps into a short representation at the time-frame level, a global averaging layer that aggregates this representation over time, and a softmax classification layer that predicts the global key of a piece. Figure 10.1 shows our model's architecture: five convolutional layers with 8 feature maps computed by $5 \times 5$ kernels, followed by a dense layer with 48 units applied frame-wise; this projection is then averaged over time and classified using a 24-way softmax layer. All layers (except the softmax layer) use the exponential-linear activation function (Clevert et al., 2016).

The convolutional layers constitute the first part of the "feature extraction" equivalent in traditional key classification systems. They are intended to process the input spectrogram, deal with detrimental factors such as noise or slight detuning, and, together with the projection layer, compute a short frame-wise description of harmonic content. This part of the network can process inputs of arbitrary lengths. Its output is aggregated in the following layers.

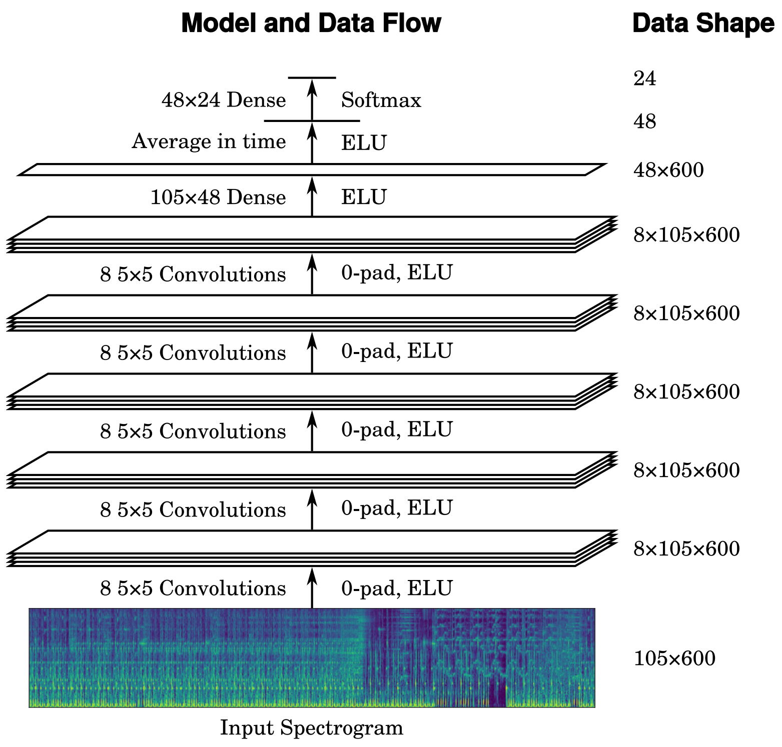


**Figure 10.1:** Neural network for musical key classification. Convolutions are 0-padded, i.e. the feature maps keep the input shape. All layers are followed by exponential-linear activations (Clevert et al., 2016), except the last, which is followed by a softmax.

Before classification, an averaging layer reduces the extracted representation to a fixed-length vector. We could employ other, more powerful methods (like recurrent layers), but we found in preliminary experiments that they fail to achieve better results.

Finally, a softmax classification layer predicts the global key for the audio. We restrict ourselves to major and minor modes only, resulting in 24 possible classes ($12 \text{ tonics} \times \{\text{major}, \text{minor}\}$) as output. This is a common restriction, since most musical pieces are in either major or minor, and as of now, there are no datasets with reliable song-level annotations of other modes.

### 10.1.3 Training

We train the model using stochastic gradient descent with momentum, back-propagating through the network the categorical cross-entropy error between true key label $y_i$ and network output $\hat{y}_i$, and apply weight decay with a factor of $10^{-4}$ for regularisation. The initial learning rate is 0.001, with a momentum factor of 0.9. If validation accuracy did not increase within 10 epochs, we halve the learning rate and continue training with the parameters that gave the best results until then. After 100 epochs, we select the model that achieved the best validation accuracy.

## 10.2 Experiments

Our experiments aim at (i) comparing the proposed system to reference systems, and (ii) examining the effect that the type of training data (in our case, the musical genre) has on the results. Template-based algorithms require specialised key templates for genres like electronic dance music in order to perform well (Faraldo et al., 2016). We want to see if, and how strongly, our system is affected by this. In the following, we discuss the data, the evaluation metrics, the reference systems, and the different set-ups of our system that we used in the experiments.

### 10.2.1 Data

We use three datasets in the course of our experiments: the GiantSteps key dataset (Knees et al., 2015), the GiantSteps-MTG key dataset, and a subset of the McGill Billboard dataset (Burgoyne et al., 2011).

The *GiantSteps Key Dataset*[2] comprises 604 two-minute audio previews from www.beatport.com, with key ground truth for each excerpt. It consists of

[2]https://github.com/GiantSteps/giantsteps-key-dataset

various sub-genres of electronic music. We use this dataset for testing purposes only, and will refer to it as *GS*.

The *GiantSteps MTG Key dataset*[3], collected by Ángel Faraldo from the Music Technology Group at Universitat Pompeu Fabra, comprises 1486 two-minute audio previews from the same source. These excerpts are distinct from the ones in the GS dataset. From this set, we only use excerpts labelled with a single key and a high confidence for training (1077 pieces). We will refer to this dataset as $GS^{MTG}$.

The *McGill Billboard Dataset*[4] comprises 742 unique songs sampled from the American Billboard charts between 1958 and 1991, and thus consists of mostly pop and rock music. Unfortunately, only the tonic, and not the mode (major or minor), is annotated for each piece. We therefore estimate the mode using the tonic and chord annotations, following a simple procedure for each piece: 1. select all chords whose root is the tonic; 2. if more than 90% of these chords are major, the key mode is assumed to be major, and vice-versa for minor; 3. else, discard the song, because we cannot confidently estimate the mode. Similarly, we discard songs with multiple annotated tonics. This leaves us with 625 songs with key annotations. We then divide the set into subsets of 62.5% for training, 12.5% for validation, and 25% for testing. The exact division and key ground truths are available online[5]. We will refer to this dataset as $BB^{TV}$ and $BB^{TE}$ for the train/validation and test sub-sets, respectively.

### 10.2.2 Data Augmentation

The datasets provide only few training data (1077 in $\mathrm{GS}^{\mathrm{MTG}}$, 391 in $\mathrm{BB}^{\mathrm{TV}}$), compared to datasets used in computer vision (e.g. 40000 in CIFAR-10 or 60000 in MNIST). The generalisation capability of deep neural networks, however, depends on a large number of training samples. We thus have to rely on data augmentation to increase the number of training examples artificially.

Several augmentation techniques for audio input have been explored (Schlüter and Grill, 2015), with pitch shifting being particularly popular in harmony-related tasks, as used by Cho (2014); Humphrey and Bello (2012), and for training the acoustic model for chord recognition in Chapter 4. Since pitch shifting is an expensive time-domain operation, most works manipulate the time-frequency representation to emulate it. In this work, however, we found that using a time-domain pitch shifting algorithm[6] directly on the audio gave better results in terms of classification accuracy. We therefore shift each training song in the range

[3]https://github.com/GiantSteps/giantsteps-mtg-key-dataset

[4]http://ddmal.music.mcgill.ca/research/billboard

[5]http://www.cp.jku.at/people/korzeniowski/bb.zip

[6]We used the SoX software available at http://sox.sourceforge.net/.

of -4 to +7 semitones (and adjust the target key accordingly), which increases the amount of training data by a factor of 12.

### 10.2.3 Metrics

Evaluating key classification results requires a more detailed quantitative analysis than computing accuracy scores. In particular, although we consider the task to be a simple 24-way classification problem when designing the system, some classes are semantically closer to each other than others. For example, the key of A-minor is called the "relative minor" to the key of C-major, as they share all pitch classes, and differ only in the tonic. Therefore, it is reasonable to consider some errors to be more severe than others.

The MIREX evaluation campaign[7] developed an evaluation strategy and introduced a single weighted measure that reflects the above considerations. Following their guidelines, key predictions can fall into the following categories:

**Correct:** if the tonic and the mode (major/minor) of prediction and target correspond.

**Fifth:** if the tonic of the prediction is the fifth of the target (or vice versa), and modes correspond.

**Relative Minor/Major:** if modes differ and either (a) the predicted mode is minor and the predicted tonic is 3 semitones below the target, or (b) the predicted mode is major and the predicted tonic is 3 semitones above the target.

**Parallel Minor/Major:** if modes differ but the predicted tonic matches the target.

**Other:** Prediction errors not caught by any category, i.e. the most severe errors.

We first compute the ratio of predictions that fall into each category. We then calculate the MIREX *weighted* score as $w = r_c + 0.5 \cdot r_f + 0.3 \cdot r_r + 0.2 \cdot r_p$, where $r_c$, $r_f$, $r_r$, and $r_p$ are the ratios of the correct, fifth, relative minor/major, and parallel minor/major, respectively. These ratios reveal more about the capability of the algorithms than accuracy (i.e., the "correct" ratio) alone. They allow us to see the kind of mistakes the system makes, and at the same time, assign a single number for comparing its performance with others.

The ratios of the individual error categories cannot be compared in isolation, but only in context with the other ratios. The only numbers that can be compared individually are the weighted score (because it aggregates all error types),

[7]http://www.music-ir.org/mirex

| *Test Set* | *Method* | *Train Set* | *Weighted* | *Correct* | *Fifth* | *Relative* | *Parallel* | *Other* |
|---|---|---|---|---|---|---|---|---|
| GS | $\text{CK}^1$ | $\text{GS}^{\text{MTG}}$ | **74.3** | **67.9** | 6.8 | 7.1 | 4.3 | **13.9** |
| | $\text{CK}^2$ | $\text{BB}^{\text{TV}}$ | 57.3 | 47.0 | 6.5 | 12.6 | 16.6 | 17.4 |
| | $\text{CK}^3$ | $\text{GS}^{\text{MTG}}$, $\text{BB}^{\text{TV}}$ | 69.2 | 61.9 | 6.8 | 8.6 | 6.3 | 16.4 |
| | $\text{EDM}^{\text{A}}$ | | 65.6 | 57.8 | 7.3 | 6.6 | 10.8 | 17.6 |
| | $\text{EDM}^{\text{M}}$ | | 70.1 | 63.7 | 8.6 | 2.7 | 6.5 | 18.5 |
| | $\text{EDM}^{\text{T}}$ | | 44.6 | 33.6 | 8.8 | 15.4 | 9.9 | 32.3 |
| | QM | | 50.4 | 39.6 | 11.9 | 13.2 | 4.3 | 31.0 |
| $\text{BB}^{\text{TE}}$ | $\text{CK}^1$ | $\text{GS}^{\text{MTG}}$ | 72.8 | 62.5 | 7.6 | 13.2 | 12.5 | **4.2** |
| | $\text{CK}^2$ | $\text{BB}^{\text{TV}}$ | **83.9** | **77.1** | 9.0 | 4.9 | 4.2 | 4.9 |
| | $\text{CK}^3$ | $\text{GS}^{\text{MTG}}$, $\text{BB}^{\text{TV}}$ | 79.7 | 70.8 | 9.7 | 9.0 | 6.3 | **4.2** |
| | $\text{EDM}^{\text{A}}$ | | 78.7 | 70.8 | 11.8 | 2.8 | 5.6 | 9.0 |
| | $\text{EDM}^{\text{M}}$ | | 28.9 | 14.6 | 2.1 | 16.0 | 42.4 | 25.0 |
| | $\text{EDM}^{\text{T}}$ | | 75.4 | 66.7 | 12.5 | 6.3 | 2.8 | 11.8 |
| | QM | | 60.9 | 52.1 | 11.8 | 4.2 | 8.3 | 23.6 |

**Table 10.1:** Results of various training configurations of our proposed model and of reference systems. Boldface indicates best results. GS and $\text{GS}^{\text{MTG}}$ refer to the GiantSteps datasets (electronic music), $\text{BB}^*$ to various subsets of the Billboard dataset (pop/rock music). $\text{CK}^*$ denote the proposed model trained on different data sets, $\text{EDM}^*$ and QM denote reference systems by (Cannam et al., 2016; Faraldo et al., 2016).

percentage of correct classifications (because it corresponds to classification accuracy), and the "other" error ratio (because it tells us how often a system predicts unrelated keys). That is why, in Table 10.1, we will highlight the best results only for these categories.

### 10.2.4 Setups and Reference Systems

We train our method in three configurations: $\text{CK}^1$, trained on $\text{GS}^{\text{MTG}}$; $\text{CK}^2$, trained on $\text{BB}^{\text{TV}}$; and $\text{CK}^3$, trained on both $\text{GS}^{\text{MTG}}$ and $\text{BB}^{\text{TV}}$. We evaluate each of the trained models on GS and on $\text{BB}^{\text{TE}}$. This way, we observe the system's performance when trained on the *same* genres as it is tested on, when trained on a *different* genre (the cross-genre setup), and when trained on multiple genres (to see if it can learn a unified model for different genres).

We compare our method against the Queen Mary Key Detector (QM) (Cannam et al., 2016) and three variations of the method presented by Faraldo et al. (2016) ($\text{EDM}^{\text{A}}$, $\text{EDM}^{\text{M}}$, $\text{EDM}^{\text{T}}$). For both systems, open source implementations are available[8]. QM consists of a hand-crafted pre-processing stage and correlates the obtained chromagrams with key profiles based on Bach's Well Tempered Clavier. The three EDM systems also use a hand-crafted pre-processing

[8]http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html
https://github.com/angelfaraldo/edmkey

stage, but use different key profiles for classification: EDM$^{A}$ uses key profiles automatically derived from a set of electronic music; EDM$^{M}$ uses hand-tuned profiles based on the automatically derived ones (effectively disabling the prediction of major keys); EDM$^{T}$ uses profiles based on European classical music. Of all submissions to MIREX, EDM$^{M}$ and QM achieved the best results on the electronic and classical music datasets used for evaluation, respectively. We thus consider both to be state of the art.

### 10.2.5 Results

Table 10.1 shows the evaluation results of all training configurations of our proposed model, and of the reference systems. We determine the statistical significance of the results using a Wilcoxon signed rank test, with the error types representing the ranks. If trained on the correct genre, our model clearly outperforms the reference systems: 74.3 vs. 70.1 ($\alpha = 0.001$) for the GiantSteps dataset, and 83.9 vs. 78.7 ($\alpha = 0.014$) for the Billboard dataset.

Examining the cross-genre setups, we observe a significant drop in key classification accuracy: tested on GS (electronic music), a model trained on BB$^{TV}$ (pop/rock) achieves a weighted score of only 57.3, compared to 74.3 when trained on electronic music from GS$^{MTG}$. However, we also see that the number of severe mistakes (category "other") that our system commits in this setup is not higher than those of the reference systems: the model only predicts a completely unrelated key 17.4% of the time—similar to the reference systems specialised on this genre (17.6% and 18.5% for EDM$^{A}$ and EDM$^{M}$, respectively); vice-versa, when trained on GS$^{MTG}$ and evaluated on BB$^{TE}$, it achieves the lowest rate of severe mistakes (4.2%).

The most common error occurring in these cross-genre setups is predicting the wrong mode (resulting in parallel minor/major) and predicting the relative minor/major key. This suggests that while the model is still able recognise some fundamental concepts of tonality, finer characteristics vary too much between pieces of different genres.

We wanted to see if the proposed model can be trained to provide a good unified key estimator for multiple genres by combining training data. The resulting system CK$^{3}$ does not reach the performance of the specialised ones (69.2 vs 74.3 on GS, 79.7 vs. 83.9 on BB$^{TE}$); however, on GS, it performs as well as EDM$^{M}$, which is tuned manually to give good results on electronic music datasets (69.2 vs. 70.1, $\alpha = 0.94$). It still performs better than EDM$^{A}$, which is also trained on electronic music, but without manual post-training adaptations (69.2 vs. 65.5, $\alpha = 0.02$).

The numbers presented for the EDM$^{*}$ systems for the GiantSteps dataset differ from the ones originally reported in (Faraldo et al., 2016). This is mainly because we applied a stricter criterion for the "fifth" category: we require the predicted mode to match the target mode, while (Faraldo et al., 2016) ignores the mode for this category. Also, according to personal correspondence with the author, changes in the library used in the original implementation worsened the results relative to the original ones.

## 10.3 Conclusion and Future Work

We have presented a global key classification system based on a convolutional neural network. Compared to previous work, this model can be automatically trained end-to-end, without the need of expert knowledge in feature design or specific pre-processing steps such as tuning correction or spectral whitening.

We have shown experimentally that the model performs state-of-the-art on datasets of electronic music and pop/rock music. Additionally, we are planning to evaluate the proposed model on more genres, e.g. classical music.

Another feature of the proposed model is its ability to adapt to multiple types of music without changing the model itself; it just needs to be re-trained with a training set extended to the type of music of interest. While it still showed good performance in such a scenario, it did not reach the level of its specialised counterparts.

A clear limitation of the proposed method is that it only estimates a global key for a complete piece. While this is adequate for certain types of music, other types (e.g. classical music) involve key modulations that our method currently cannot capture. A possible easy fix could be to apply our model using a sliding window over the spectrogram. Extending the proposed method in such a way is left to future work.

Finally, we have to keep in mind that even with data augmentation, we are still working with small datasets. Although we increase the number of training samples by a factor of 12 using pitch-shifting, this is not equivalent to having available 12 times as many musical pieces: the musical content of the artificial data points is still the same as in the seed data point, just in a different key. We expect the system's performance to improve once more training data is available.

# 11 Genre-Agnostic Key Classification

The model developed in Chapter 10 generalised better across musical genres than hand-crafted approaches, but it still achieved the best results when tuned specifically for a musical style. In this chapter, we present modifications to the model structure and its training procedure that enable the model to learn a key classifier that is agnostic to genre. Not only does it perform better than the model from Chapter 10 on all genres the latter is optimised for; it does so not despite, but *because* it is trained on various musical styles, instead of a specific one (see Section 11.2.4).

This work has been previously published in Korzeniowski and Widmer (2018b).

## 11.1 Method

We build upon the same audio processing pipeline used in Chapter 10, and input to the network a log-magnitude log-frequency quarter-tone spectrogram (5 frames per second, frame size 8 192, sample rate 44 100 Hz). We limit the frequency range to the harmonically most relevant 65 Hz to 2 100 Hz, based on the findings in Chapter 3.

The network structure proposed in Chapter 10 was modelled after typical processing pipelines used for key classification. It features five convolutional layers of $5 \times 5$ kernels for spectrogram processing, followed by a dense projection into a frame-wise embedding space, which is then averaged over time and classified using a softmax layer. All layers except the last use the exponential-linear activation function (Clevert et al., 2016) (ELU). The architecture, which we name *KeyNet*, is summarised in Table 11.1a.

During training, the model is shown the complete spectrogram of a piece. Its weights are then adapted using stochastic gradient descent to minimise the categorical cross-entropy between the predicted key distribution and the ground truth. We will refer to the KeyNet architecture, when trained using full spectrograms, as *KeyNet/F*.

### 11.1.1 Adaptations of the Training Procedure

The outlined training scheme has two drawbacks. First, the computation of a single update is expensive; the network has to process the full spectrogram (e.g. 600×105 values for a two-minute piece), and keep intermediate results for back-propagating the error. Training is thus slow and requires much memory. Second, it keeps the variety of the data lower than necessary, as the network sees the same spectrograms at every epoch.

To circumvent these drawbacks, we show the network only short snippets instead of the whole piece at training time. These snippets should be as short as possible to reduce computation time, but have to be long enough to contain the relevant information to determine the key of a piece. From our datasets, we found 20 s to be sufficient (with the exception of classical music, which we need to treat differently, due to the possibility of extended periods of modulation—see Sec. 11.2.1 below). Each time the network is presented a song, we cut a random 20 s snippet from the spectrogram. The network thus sees a different variation of each song every epoch.

During testing, the network processes the whole piece. This gives better results than when using only a snippet. Since we do not have to store intermediate results and process each piece many times as in training, memory space and run time are not an issue. We will refer to KeyNet models trained using spectrogram snippets as *KeyNet/S*.

We expect this modification to have the following effects. (i) Back-propagation will be faster and require less memory, because the network sees shorter snippets; we can thus train faster, and process larger models. (ii) The network will be less prone to over-fitting, since it almost never sees the same training input; we expect the model to generalise better. (iii) The network will be forced to find evidence for a key in each excerpt of the training pieces, instead of relying on parts where the key is more obvious; by asking more of the model, we expect it to pick up more subtle relationships between the audio and its key.

### 11.1.2 Adaptations of the Model Structure

The KeyNet architecture uses a dense layer to project the processed spectrogram into a key embedding space. In its original formulation, which uses an embed-

| *Layer Type* | *FMaps* | *Params* |
|---|---|---|
| Input | | |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Dense-ELU | | $2 \cdot N_f$ |
| Pool-Time Avg. | | |
| Dense-Softmax | | 24 |

**(a)** KeyNet Architecture

| *Layer Type* | *FMaps* | *Params* |
|---|---|---|
| Input | | |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $2N_f$ | $3 \times 3$ |
| Conv-ELU | $2N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $4N_f$ | $3 \times 3$ |
| Conv-ELU | $4N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $8N_f$ | $3 \times 3$ |
| Conv-ELU | $8N_f$ | $3 \times 3$ |
| Conv-ELU | 24 | $1 \times 1$ |
| Pool-Global Avg. | | |
| Softmax | | |

**(b)** AllConv Architecture

**Table 11.1:** Neural Network architectures. $N_f$ is parameter that controls the model complexity. Horizontal lines denote dropout layers (Srivastava et al., 2014). Here, dropout is applied on complete feature maps, not individual units. Each convolution is followed by batch normalisation (Ioffe and Szegedy, 2015). *FMaps* indicates the number of feature maps, while *Params* the parameters of the layer (kernel size, pool size, or number of units).

ding space with 48 dimensions and 8 feature maps in the convolutional layers, this projection accounts for 65 % of the network's parameters. Dense layers are also more prone to over-fitting than convolutional layers.

We thus propose to use a network architecture that does away with dense layers, and relies on convolutions and pooling only. At the same time, we move away from modelling the network based on traditional key classification methods—recall that the components of KeyNet were designed to correspond to components in typical key classification pipelines—and instead use a general network architecture for classification, based on the all-convolutional net (Springenberg et al., 2015). The new architecture is summarised in Table 11.1b, and will be referred to as *AllConv*. We will train this architecture only with the snippet method.

We expect this change to improve results and generalisation because (i) convolutional layers over-fit less than dense layers; (ii) given the same number of parameters, deeper networks are more expressive than shallower ones (Eldan and Shamir, 2016; Liang and Srikant, 2017); (iii) comparable architectures have shown to perform well in other audio-related tasks such as chord recognition (see Chapter 4) or audio scene classification (Eghbal-Zadeh et al., 2016).

## 11.2 Experiments

We first evaluate how the proposed modifications affect the key classification performance in Section 11.2.3. Then, we analyse how the amount and genre of training data influence results in Section 11.2.4.

### 11.2.1 Data

Since we are interested in how well the models generalise across different genres, we use datasets that encompass three distinct musical styles. As in the previous chapter, we apply pitch shifting in the range of -4 to +7 semitones to increase the amount of training data.

**Electronic Dance Music:** Here, we use songs from the GiantSteps MTG Key dataset[1], collected by Ángel Faraldo. It comprises 1486 distinct two-minute audio previews from www.beatport.com, with key ground truth for each excerpt. We only use excerpts labelled with a single key and a high confidence (1077 pieces), and split them into 80 % training and 20 % validation. For testing, we use the GiantSteps Key Dataset[2]. It comprises 604 two-minute audio previews from the same source (but distinct from the training set).

**Pop/Rock Music:** For this genre, we use the McGill Billboard dataset (Burgoyne et al., 2011)[3]. It consists of 742 unique songs sampled from the American Billboard charts between 1958 and 1991. We split these songs into subsets of 62.5% for training, 12.5% for validation, and 25% for testing. We determine the global key for each song using the procedure described in Chapter 10, which leaves us with 625 songs with key annotations in total. The exact division and key ground truths are available online[4].

[1]https://github.com/GiantSteps/giantsteps-mtg-key-dataset
[2]https://github.com/GiantSteps/giantsteps-key-dataset
[3]http://ddmal.music.mcgill.ca/research/billboard
[4]http://www.cp.jku.at/people/korzeniowski/bb.zip

**Classical Music:** To cover this genre, we collected 1504 (mostly piano) pieces from our internal database for which we could derive the key from the piece's title. Classical pieces often modulate their key, but usually start in the key denoted in the title. We thus only use the first 30 s of each recording. Tracking key modulations is left for future work. We then select 81 % for training, 9 % for validation, and 10 % for testing.

### 11.2.2 Metrics

We adopt the standard evaluation score for Key Classification as defined in the MIREX evaluation campaign[5]. It goes beyond simple accuracy, as it considers harmonic similarities between key classes. A prediction can fall into one of the following categories:

**Correct:** if the tonic and the mode (major/minor) of prediction and target correspond.

**Fifth:** if the tonic of the prediction is the fifth of the target (or vice versa), and modes correspond.

**Relative Minor/Major:** if modes differ and either a) the predicted mode is minor and the predicted tonic is 3 semitones below the target, or b) the predicted mode is major and the predicted tonic is 3 semitones above the target.

**Parallel Minor/Major:** if modes differ but the predicted tonic matches the target.

**Other:** Prediction errors not caught by any category, i.e. the most severe errors.

Then, a weighted score can be computed as $w = r_c + 0.5 \cdot r_f + 0.3 \cdot r_r + 0.2 \cdot r_p$, where $r_c$, $r_f$, $r_r$, and $r_p$ are the ratios of the correct, fifth, relative minor/major, and parallel minor/major, respectively. We will use this weighted score for our comparisons.

### 11.2.3 Evaluation of the Adaptations

To evaluate the effect of our proposed adaptations, we train the three setups (KeyNet/F, KeyNet/S, AllConv) with the combined data of all datasets. We will consider *validation* results in the first sets of experiments, and show results on the testing sets only for our analyses and final evaluations. This way, we ensure that the final results are unbiased.

[5]http://www.music-ir.org/mirex

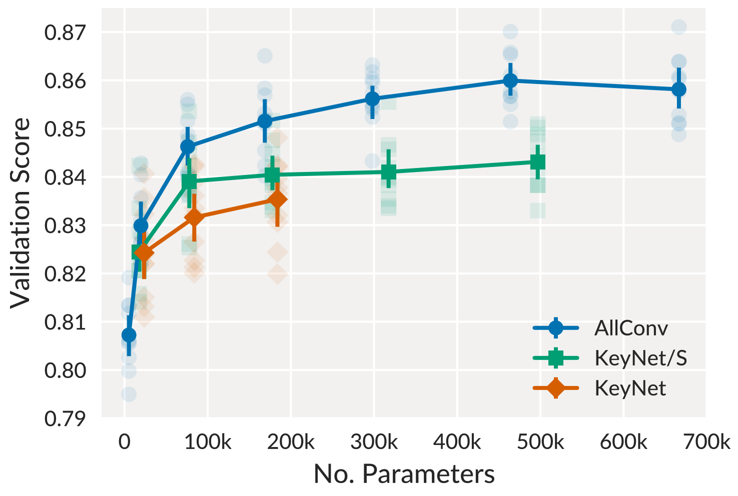


**Figure 11.1:** Average validation score over 10 runs for the different model setups. Whiskers represent 95 % confidence intervals computed by bootstrapping. Transparent dots show results of the individual runs. We see that given similar network sizes, the AllConv model performs best. Also, using snippet training (KeyNet/S) improves results compared to full spectrogram training (KeyNet/F), and enables training larger networks.

The capacity of a neural network depends not only on the architecture, but also its size (i.e., the number of parameters). For a fair comparison, we evaluate each architecture with varying network sizes. For the AllConv architecture, we select the number of feature maps $N_f \in \{2, 4, 8, 12, 16, 20, 24\}$. For the KeyNet architecture, the network size depends on the number of feature maps in the convolutional layers and the size of the embedding space. For practical reasons, we set the size of the embedding space to be $2N_f$, and select $N_f \in \{8, 16, 24, 32, 40\}$. Note that if we train on full spectrograms (KeyNet/F), we could not train networks with $N_f > 24$ due to memory constraints. For each model, we tried dropout probabilities of $p \in \{0.0, 0.1, 0.2\}$.

Figure 11.1 presents the results of the three model configurations. For each model and model capacity, we select the best dropout probability based on the validation results. The experiments show that both adaptations are beneficial. Training with snippets instead of full spectrograms gives better results at smaller network capacities and enables training of larger networks. The AllConv architecture achieves even better results, regardless of its size.

We can quantify two reasons for this, which are consequences of the expected benefits of the adaptations: *better generalisation* through increased data variety and the absence of dense layers, and *better expressivity* through deeper architectures and by training the network on a more difficult task. For the first, better generalisation, we compare the average ratio of validation accuracy to training accuracy for each of the models (higher indicates less over-fitting): 0.948, 0.969, and 0.982 for KeyNet/F, KeyNet/S, and AllConv, respectively. For the second, *model expressiveness*, we compare the model's capability to fit the training data in terms of accuracy: 0.837, 0.858, and 0.907 for KeyNet/F, KeyNet/S, and AllConv, respectively. Stronger models that generalise better achieve better results.

### 11.2.4 Influence of Training Data

We then want to see how the amount and genre of the datasets used for training affects results. To this end, we select the hyper-parameter settings for AllConv and KeyNet/S that achieved the best average results in the previous experiment: $N_f = 20, p = 0.1$ for AllConv, $N_f = 40, p = 0.1$ for KeyNet/S. Additionally, we consider smaller models of each type, i.e. $N_f = 8$ for AllConv and $N_f = 16$ for KeyNet/S, both without dropout. Under these settings, both architectures have a comparable number of parameters. We train these models using all possible 1, 2, and 3-combinations of the datasets, and evaluate them on all data. The results are shown in Fig. 11.2.

The main observations are: (i) increasing model capacity is more beneficial to the AllConv model than KeyNet/S, regardless of dataset; (ii) adding capacity to the AllConv model enables it to better deal with diverse data—the biggest gains of additional parameters are achieved if the model is trained on a combined dataset (pink line)—while this is not always the case for KeyNet/S (see the Billboard results, where it seems that adding classical music to the training set impairs the performance of this model); (iii) given enough capacity in the AllConv model, training using the complete data performs better than (or almost equal to) fitting a specific genre, while the opposite is the case for KeyNet/S, where specialised models outperform the general ones. We thus argue that the AllConv model not only copes better with diverse training data, but that it leverages the diversity in the training data to perform as well as it does.

## 11.3 Evaluation

Motivated by the results above, the remainder of our analysis focuses on the AllConv model. To thoroughly investigate its performance and compare it to the state of the art, we evaluate it on the following unseen datasets:

**KeyFinder[6]:** 1 000 songs from a variety of popular music genres. Unfortunately, we have only the audio for 998 of the songs available.

[6]http://www.ibrahimshaath.co.uk/keyfinder/

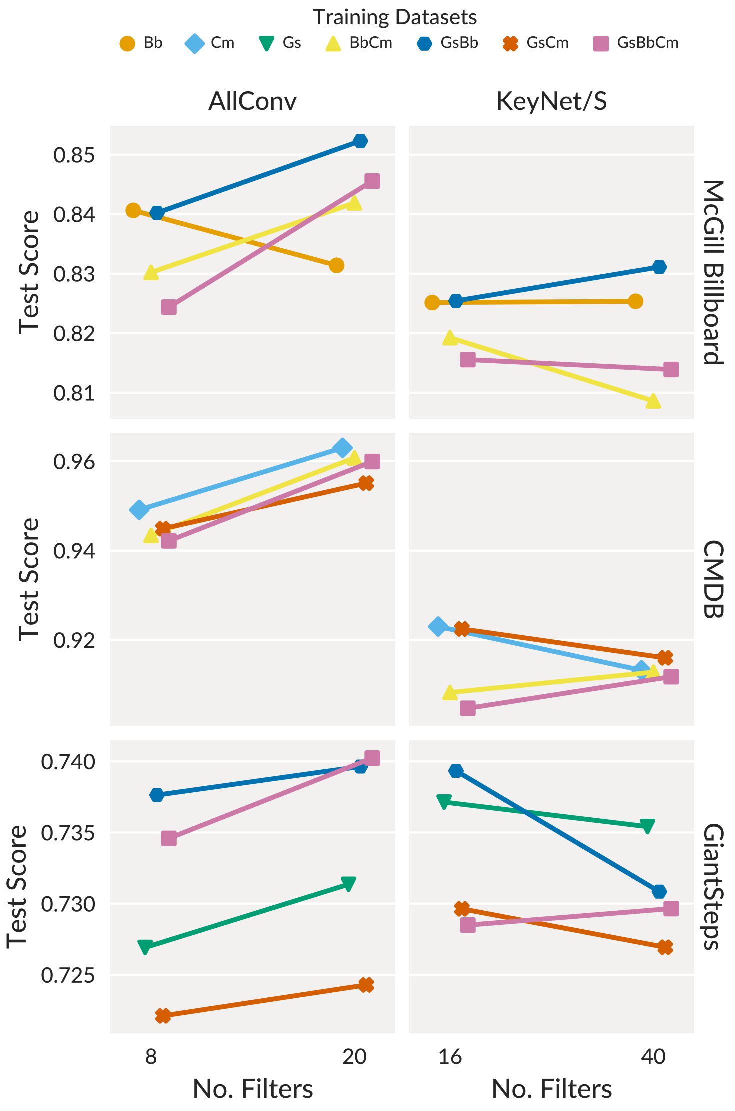


**Figure 11.2:** Average test scores over 10 runs for each architecture (columns), split by dataset (rows). The smaller models are on the left of each column. Colours indicate the training data used: *Bb* stands for the Billboard dataset, *Cm* for the classical music dataset, and *Gs* for the GiantSteps dataset. Each row shows the results of runs where the training set also contained the training data of the respective test set genre (e.g. in the first row, we only see runs where McGill Billboard data was included in training).

| *Dataset* | *Model* | *Weighted* | *Correct* | *Fifth* | *Relative* | *Parallel* | *Other* |
|---|---|---|---|---|---|---|---|
| GiantSteps | AllConv | **74.6** | **67.9** | 7.0 | 8.1 | 4.1 | **12.9** |
| | CK[1] | 74.3 | **67.9** | 6.8 | 7.1 | 4.3 | 13.9 |
| Billboard | AllConv | **85.1** | **79.9** | 5.6 | 4.2 | 6.2 | **4.2** |
| | CK[2] | 83.9 | 77.1 | 9.0 | 4.9 | 4.2 | 4.9 |
| Classical | AllConv | 96.6 | 95.2 | 1.4 | 1.4 | 1.4 | 0.7 |
| | - | - | - | - | - | - | - |
| KeyFinder | AllConv | **76.1** | **70.0** | 5.7 | 7.4 | 4.7 | **12.1** |
| | bgate | 72.4 | 65.0 | 8.6 | 6.5 | 5.4 | 14.4 |
| Isophonics | AllConv | **82.5** | **76.3** | 7.6 | 5.4 | 3.7 | **7.1** |
| | BD1 | 75.1 | 66.0 | 13.6 | 5.1 | 3.9 | 9.2 |
| R. Williams | AllConv | **81.2** | **72.4** | 10.8 | 10.3 | 1.3 | **5.2** |
| | HS1 | 77.1 | 68.8 | 10.1 | 9.0 | 3.2 | 9.0 |
| Rock | AllConv | 74.3 | 69.3 | 6.5 | 1.7 | 6.0 | 16.5 |
| | - | - | - | - | - | - | - |

**Table 11.2:** Evaluation results. Best results are in boldface. CK[1] and CK[2] refer to the models presented in Chapter 10, bgate to the one by Faraldo et al. (2017), BD1 to the one by Bernardes et al. (2017), and HS1 to the one by Schreiber (2017).

**Isophonics[7]:** 180 songs by The Beatles, 19 songs by Queen, and 18 songs by Zweieck. Since these songs contain key modulations, we split them into single key segments and retain only segments annotated as major or minor keys, as was done for the 2017 MIREX evaluation campaign[8].

**Robbie Williams (Di Giorgi et al., 2013):** 65 songs by Robbie Williams, which we also split into single key segments as outlined above.

**Rock[9] (de Clercq and Temperley, 2011):** 200 songs taken from Rolling Stone's "500 Greatest Songs of All Time" list. As with the McGill Billboard dataset, only the tonics are annotated. We first split the songs according to the annotated tonics, and then follow a similar procedure as described in Chapter 10: if more than 80 % of the tonic chords are in either major or minor, the mode is set accordingly; if there are no tonic chords in a segment, we consider dominant chords in the same way.

We select the best AllConv model based on the validation score over the compound data of Electronic, Pop/Rock and Classical music. On average,

[7]http://isophonics.net/datasets

[8]http://www.music-ir.org/mirex/wiki/2017:Audio_Key_Detection_Results

[9]http://rockcorpus.midside.com/

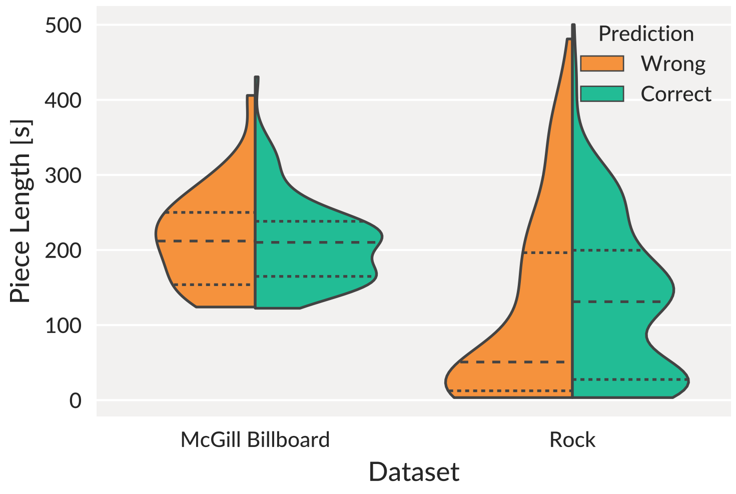


**Figure 11.3:** Distributions of the length of correctly and incorrectly classified excerpts depending on the dataset they come from. Densities are estimated using kernel density estimation. Horizontal lines with long dashes indicate the median, those with short dashes the quartiles. The densities are normalised, i.e. they do not indicate how many instances were classified correctly (or incorrectly), but only the distribution of except lengths within each group.

models with $N_f = 20$ and dropout probability of $0.1$ performed best. However, the best single model used $N_f = 24$ (see Fig. 11.1), and was consequently chosen as final model.

In Table 11.2, we compare this model to other models proposed in the academic literature. For each dataset, we show the results of the best competing system, if available. For the GiantSteps and Billboard datasets, the best competing systems were variants of the model we explored in Chapter 10. For the pre-segmented Isophonics and Robbie Williams datasets, we use the results available on the MIREX 2017 website[8]. For the KeyFinder dataset, we report the best results achieved using the open-source reference implementation[10] of the algorithms from (Faraldo et al., 2017).

As we can see, the proposed model *performs best for all datasets* for which comparisons were possible. Keep in mind that the systems we compare to are often specifically tuned for a genre (CK[1], CK[2], HS1, bgate) or set up to favour certain key modes prevalent in a dataset (BD1), while we use the same, general model for all datasets. For example, CK[1] performs badly on the Billboard dataset ($w = 72.8$), BD1 on the GiantSteps ($w = 59.6$), and HS1 on the Isophonics dataset ($w = 64.1$). In this light, it is remarkable that the proposed model consistently out-performs the others.

However, the results also point us to a deficiency of the model. Recall that for some datasets (e.g. Rock), we split the files according to key annotations,

[10]https://github.com/angelfaraldo/edmkey

and process each excerpt individually. If we compare the results on the Rock dataset with those on the Billboard dataset, we see a large discrepancy, although both sets comprise similar musical styles. As Fig. 11.3 demonstrates, the duration of a classified excerpt plays a major role here: for the Billboard set, the median length of excerpts classified correctly matches the one of incorrect classifications; for the Rock set, however, the median lengths differ greatly: 131 s vs. 51 s, for correctly and incorrectly classified excerpts, respectively. The distribution of excerpt lengths that are classified correctly is thus very different from the one of incorrectly classified excerpts in the Rock set. The shorter an excerpt, the more likely it is classified incorrectly.

This is not surprising per se. Determining the key of a piece requires a certain amount of musical context. However, it shows that in order to move beyond global key classification, and towards recognising key modulations, it will not suffice to detect key boundaries and apply known methods within these boundaries. To recognise key modulations, classifying short excerpts individually will reach a glass ceiling. Instead, we will need models that consider the hierarchical harmonic coherence of the whole piece.

## 11.4 Conclusion

We have presented a genre-agnostic key classification model based on the system developed in Chapter 10, with improvements of the training procedure and network structure. These improvements enable faster training, better generalisation, and training larger and thus more powerful models. These models can leverage diverse training data instead of being impaired by it. The resulting key classifier generalises well over datasets of different musical styles, and out-performs systems that are specialised for specific genres.

# 12 Conclusion

In the first part of this thesis, we considered chord recognition. We first developed powerful acoustic models based on deep neural networks, and processed their predictions with a conditional random field—a simple first-order model that primarily smoothed the acoustic model's predictions. We then investigated how to create data-driven temporal models that go beyond smoothing. We learned that such models need to be employed at a different hierarchical level than it is common: instead of audio-frames, they need to operate on sequences of chord symbols. We further explored two types of such models (finite-context *n*-grams and recurrent neural networks), and compared their capability to predict chord sequences. We found, perhaps to little surprise, that the RNN models perform better, as they learn to adapt to the processed piece by remembering sub-sequences of chords. We then developed a probabilistic model that integrates these chord-level language models with the predictions of an acoustic model, by using a model of chord duration. Finally, we evaluated how improvements in language and duration modelling affect the final chord recognition performance.

In the time I conducted these studies, research has further developed the capability of acoustic models to deal with extended chord vocabularies (Cho, 2014; Deng and Kwok, 2016; Humphrey and Bello, 2015; McFee and Bello, 2017; Wu and Li, 2018). This development can be considered orthogonal to my research, as it focuses on a different facet of the problem. However, at some point, these two lines of research need to be combined into a full chord recognition system. This directly leads to a number of open challenges regarding chord language models. How should they deal with a large vocabulary of symbols that are (i) related to each other in a hierarchical way (e.g. E:min and E:min9), and treating them independently neglects important information, (ii) are extremely unevenly distributed, with rare chords forming a long tail, while major chords alone represent up to half of the data (Burgoyne et al., 2011), and (iii) annotator disagreement increases with annotation granularity, i.e. annotators disagree more about chords that are rare (de Clercq and Temperley, 2011; Humphrey and Bello, 2015)?

These points outlined above necessitate the development of hierarchical approaches for modelling and evaluating chord language models, but also full chord recognition systems, as already delineated by Humphrey (2015) (Section 4.5). Sharing information between related chords was also proven to be useful for acoustic modelling (McFee and Bello, 2017). Whether hand-modelled or learned from data (Vendrov et al., 2015), chord language models will need a notion of chord hierarchy to meaningfully operate with extended chord vocabularies.

In the second part of this thesis, we considered key classification. Here, we first developed a convolutional neural network whose structure was inspired by traditional key classification algorithms. Although it performed better than state-of-the-art methods, its accuracy dropped when it was trained using genre-diverse data. To overcome this problem, we used a training procedure that increased data diversity, and applied a VGG-style convolutional neural network for classification. Using this structure and training procedure, we made the network learn a genre-agnostic model that out-performs state-of-the-art genre-specific models on a variety of data sets.

Research on key classification has further focused on tweaking pre-processing pipelines and key profiles. These methods still require the usage of key profiles specialised to genres (Faraldo et al., 2017), or setting genre-specific parameters by hand (Bernardes et al., 2017). The models we developed solve this problem if training data is available for a specific genre, but their performance still suffers when confronted with out-of-training genres. However, we do not know whether this is a problem of the algorithm, or if key classification indeed does require genre-specific information that cannot be deduced from understanding other genres.

The main limitation of this approach is that it can only extract the global key of a piece, and is ignorant of key modulations. Although the presented methods can be used to detect keys of short segments (using a pre-computed segmentation or a sliding window), we saw in Chapter 11 that this will likely reach a glass ceiling: classification accuracy dropped for short audio excerpts. We thus concluded that future systems need to consider the hierarchical harmonic structure of a piece to properly track key modulations.

Taking this thought further, we may conjecture that only a model that considers harmony holistically will be able to match human performance for any of these tasks. We may argue that by dividing the concept of musical harmony into isolated sub-tasks, we indeed hide information that is hard to recover. Therefore, we might need a model that considers many hierarchical aspects of harmony at the same time to better understand each one of them. Deep neural networks are a natural choice for such problems. They can easily be trained for multiple objectives, while learning both shared and task-specific representations of the input at the same time. However, we can expect that modelling tonal harmony as a whole in a single neural network will require us to develop novel network architectures. We might not be able to rely on standard models to solve this task. This opens up new challenges, but only with challenges come opportunities for original research.

# Bibliography

J. Albrecht and D. Shanahan. The Use of Large Corpora to Train a New Type of Key-Finding Algorithm: An Improved Treatment of the Minor Mode. *Music Perception: An Interdisciplinary Journal*, 31(1):59–67, Sept. 2013.

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug. 2013.

G. Bernardes, M. E. P. Davies, and C. Guedes. Automatic Musical Key Estimation with Adaptive Mode Bias. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, Mar. 2017.

S. Böck and M. Schedl. Enhanced Beat Tracking With Context-Aware Neural Networks. In *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*, Paris, France, Sept. 2011.

S. Böck, F. Krebs, and G. Widmer. A Multi-model Approach to Beat Tracking Considering Heterogeneous Music Styles. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, Oct. 2014.

S. Böck, F. Krebs, and G. Widmer. Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.

S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. Madmom: A new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia (ACMMM)*, Amsterdam, Netherlands, Oct. 2016.

N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Audio Chord Recognition With Recurrent Neural Networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, Nov. 2013.

J. A. Burgoyne, L. Pugin, C. Kereluik, and I. Fujinaga. A Cross-validated Study of Modelling Strategies for Automatic Chord Recognition. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, Sept. 2007.

J. A. Burgoyne, J. Wild, and I. Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, Oct. 2011.

C. Cannam, M. Mauch, M. E. Davies, S. Dixon, C. Landone, K. Noland, M. Levy, M. Zanoni, D. Stowell, and L. A. Figueira. MIREX 2016 Entry: Vamp Plugins from the Centre for Digital Music. Technical report, MIREX, 2016.

R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia. Chord Recognition Using Duration-Explicit Hidden Markov Models. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, Oct. 2012.

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, Doha, Qatar, Oct. 2014.

T. Cho. *Improved Techniques for Automatic Chord Recognition from Music Audio Signals*. Dissertation, New York University, New York, 2014.

T. Cho and J. P. Bello. On the Relative Importance of Individual Components of Chord Recognition Systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):477–492, Feb. 2014.

T. Cho, R. J. Weiss, and J. P. Bello. Exploring Common Variations in State Of The Art Chord Recognition Systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, July 2010.

J. Chorowski and N. Jaitly. Towards Better Decoding and Language Model Integration in Sequence to Sequence Models. *arXiv:1612.02695*, Dec. 2016.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *International Conference on Learning Representations (ICLR), arXiv:1511.07289*, San Juan, Puerto Rico, Feb. 2016.

T. de Clercq and D. Temperley. A Corpus Analysis of Rock Harmony. *Popular Music*, 30(01):47–70, Jan. 2011.

J. Deng and Y.-K. Kwok. Automatic Chord Estimation on Seventhsbass Chord Vocabulary Using Deep Neural Network. In *2016 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016.

B. Di Giorgi, M. Zanoni, A. Sarti, and S. Tubaro. Automatic Chord Recognition Based on the Probabilistic Modeling of Diatonic Modal Harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, Sept. 2013.

S. Dieleman, P. Brakel, and B. Schrauwen. Audio-based Music Classification with a Pretrained Convolutional Network. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, Oct. 2011.

S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degrave. Lasagne: First Release, Aug. 2015.

T. Do and T. Arti. Neural Conditional Random Fields. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Chia Laguna, Italy, May 2010.

H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer. CP-JKU Submissions for DCASE-2016: A Hybrid Approach Using Binaural I-Vectors and Deep Convolutional Neural Networks. Technical report, DCASE2016 Challenge, Sept. 2016.

R. Eldan and O. Shamir. The Power of Depth for Feedforward Neural Networks. In *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, New York, USA, June 2016.

D. P. W. Ellis, A. Berenzweig, and B. Whitman. The "uspop2002" Pop Music Data Set. https://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html, 2003.

A. Faraldo, E. Gómez, S. Jordà, and P. Herrera. Key Estimation in Electronic Dance Music. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information*

*Retrieval*, number 9626 in Lecture Notes in Computer Science, pages 335–347. Springer International Publishing, Mar. 2016.

A. Faraldo, S. Jordà, and P. Herrera. A Multi-Profile Method for Key Estimation in EDM. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017.

S. Fine, Y. Singer, and N. Tishby. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1):41–62, July 1998.

T. Fujishima. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, Oct. 1999.

X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, USA, Apr. 2011.

M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, Oct. 2002.

A. Graves. Sequence Transduction with Recurrent Neural Networks. In *Proceedings of the 29th International Conference on Machine Learning, Workshop on Representation Learning (ICML)*, Edinburgh, Scotland, June 2012.

U. Hadar and H. Messer. High-order Hidden Markov Models - Estimation and Implementation. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing (SSP)*, Cardiff, United Kingdom, Aug. 2009.

C. Harte. *Towards Automatic Extraction of Harmony Information from Music Signals*. Dissertation, Queen Mary University of London, London, United Kingdom, Aug. 2010.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

E. J. Humphrey. *An Exploration of Deep Learning in Content-Based Music Informatics*. Dissertation, New York University, New York, 2015.

E. J. Humphrey and J. P. Bello. Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In *Proceedings of the 11th International*

*Conference on Machine Learning and Applications (ICMLA)*, Boca Raton, USA, Dec. 2012.

E. J. Humphrey and J. P. Bello. Four Timely Insights on Automatic Chord Estimation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.

E. J. Humphrey, T. Cho, and J. P. Bello. Learning a Robust Tonnetz-Space Transform for Automatic Chord Recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

O. İzmirli and R. B. Dannenberg. Understanding Features and Distance Functions for Music Sequence Alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, Netherlands, Aug. 2010.

R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the Potential of Simple Framewise Approaches to Piano Transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, Aug. 2016.

M. Khadkevich and M. Omologo. Time-frequency Reassigned Features for Automatic Chord Recognition. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011.

D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR), arXiv:1412.6980*, San Diego, USA, May 2015.

P. Knees, A. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff. Two Data Sets for Tempo Estimation and Key Detection in Electronic Dance Music Annotated from User Corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.

F. Korzeniowski and G. Widmer. Feature Learning for Chord Recognition: The Deep Chroma Extractor. In *Proceedings of the 17th International Society* *for Music Information Retrieval Conference (ISMIR)*, New York, USA, Aug. 2016a.

F. Korzeniowski and G. Widmer. A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, Sept. 2016b.

F. Korzeniowski and G. Widmer. End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, Aug. 2017a.

F. Korzeniowski and G. Widmer. On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017b.

F. Korzeniowski and G. Widmer. Automatic Chord Recognition with Higher-Order Harmonic Language Modelling. In *2018 26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, Sept. 2018a.

F. Korzeniowski and G. Widmer. Genre-Agnostic Key Classification With Convolutional Neural Networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Sept. 2018b.

F. Korzeniowski and G. Widmer. Improved Chord Recognition by Combining Duration and Harmonic Language Models. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Sept. 2018c.

F. Korzeniowski, D. R. W. Sears, and G. Widmer. A Large-Scale Study of Language Models for Chord Prediction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, Apr. 2018.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, Williamstown, USA, June 2001.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, Apr. 2018.

S. Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? In *International Conference on Learning Representations (ICLR), arXiv:1610.04161*, Toulon, France, Apr. 2017.

M. Lin, Q. Chen, and S. Yan. Network in Network. In *International Conference on Learning Representations (ICLR), arXiv:1312.4400*, Banff, Canada, Apr. 2014.

L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals. Segmental Recurrent Neural Networks for End-to-End Speech Recognition. In *Interspeech 2016*, San Francisco, USA, Sept. 2016.

M. Mauch and S. Dixon. Approximate Note Transcription for the Improved Identification of Difficult Chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, Netherlands, Aug. 2010a.

M. Mauch and S. Dixon. Simultaneous Estimation of Chords and Musical Context From Audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289, Aug. 2010b.

M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 Metadata Project 2009. In *Late Breaking Session of the 10th International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, Oct. 2009.

B. McFee and J. P. Bello. Structured Training for Large-Vocabulary Chord Recognition. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, Oct. 2017.

B. McFee and D. Ellis. Analyzing Song Structure with Spectral Clustering. In *Proceedings of 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, Oct. 2014.

M. McVicar, R. Santos-Rodriguez, Y. Ni, and T. D. Bie. Automatic Chord Estimation from Audio: A Review of the State of the Art. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):556–575, Feb. 2014.

T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent Neural Network Based Language Model. In *Interspeech 2010*, Chiba, Japan, Sept. 2010.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*, Jan. 2013.

M. Müller, S. Ewert, and S. Kreuzer. Making Chroma Features More Robust to Rimbre Changes. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr. 2009.

Y. Ni, M. McVicar, R. Santos-Rodriguez, and T. De Bie. An End-to-End Machine Learning System for Harmonic Analysis of Music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, Aug. 2012a.

Y. Ni, M. McVicar, R. Santos-Rodriguez, and T. De Bie. Using Hyper-genre Training to Explore Genre Information for Automatic Chord Estimation. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012b.

K. Noland and M. Sandler. Signal Processing Parameters for Tonality Estimation. In *Audio Engineering Society Convention 122*. Audio Engineering Society, May 2007.

N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama. Separation of a Monaural Audio Signal into Harmonic/Percussive Components by Complementary Diffusion on Spectrogram. In *2008 16th European Signal Processing Conference (EUSIPCO)*, Lausanne, France, Aug. 2008.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to Construct Deep Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR), arXiv:1312.6026*, Banff, Canada, Apr. 2014.

J. Pauwels and J.-P. Martens. Combining Musicological Knowledge About Chords and Keys in a Simultaneous Chord and Local Key Estimation System. *Journal of New Music Research*, 43(3):318–330, July 2014.

J. Pauwels and G. Peeters. Evaluating Automatically Estimated Chord Sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

S. Pauws. Musical Key Extraction From Audio. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, Spain, Oct. 2004.

J. Peng, L. Bo, and J. Xu. Conditional Neural Fields. In *Advances in Neural Information Processing Systems 22 (NIPS)*, Vancouver, Canada, Dec. 2009.

M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, editors. *Inside the Jazzomat: New Perspectives for Jazz Research*. Schott, Mainz, 1. auflage edition, Nov. 2017.

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.

C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. Mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proceedings of the 15th International Conference on Music Information Retrieval (ISMIR)*, Taipei, Taiwan, Oct. 2014.

S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist Probability Estimators in HMM Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174, Jan. 1994.

J. Schlüter. *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*. Dissertation, Johannes Kepler University, Linz, Austria, July 2017.

J. Schlüter and T. Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.

H. Schreiber. MIREX 2017: CNN-Based Automatic Musical Key Detection Submissions HS1/HS2/HS3. Technical report, MIREX, 2017.

A. Sheh and D. P. W. Ellis. Chord Segmentation and Recognition using EM-Trained Hidden Markov Models. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR)*, Washington, USA, Oct. 2003.

S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon. Audio Chord Recognition With A Hybrid Recurrent Neural Network. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.

S. Sigtia, E. Benetos, and S. Dixon. An End-to-End Neural Network for Polyphonic Piano Music Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, May 2016.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, Sept. 2014.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. In *International Conference on Learning Representations (ICLR), Workshop Track, arXiv:1412.6806*, May 2015.

A. Srinivasamurthy, A. Holzapfel, A. T. Cemgil, and X. Serra. A Generalized Bayesian Model for Tracking Long Metrical Cycles in Acoustic Music Signals. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016.

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*, 15(1):1929–1958, June 2014.

J. Sun, H. Li, and L. Lei. Key Detection Through Pitch Class Distribution Model and ANN. In *2009 16th International Conference on Digital Signal Processing (ICDSP)*, Santorini, Greece, July 2009.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567*, Dec. 2015.

D. Temperley. What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception: An Interdisciplinary Journal*, 17 (1):65–100, Oct. 1999.

Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama. HMM-based Approach for Automatic Chord Detection Using Refined Acoustic Features. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Dallas, USA, Mar. 2010.

K. Ullrich, J. Schlüter, and T. Grill. Boundary Detection in Music Structure Analysis Using Convolutional Neural Networks. In *15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, Oct. 2014.

I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-Embeddings of Images and Language. *arXiv:1511.06361*, Nov. 2015.

G. Widmer. Getting Closer to the Essence of Music: The Con Espressione Manifesto. *ACM Transactions on Intelligent Systems and Technology*, 8(2): 19:1–19:13, Oct. 2016.

Y. Wu and W. Li. Music Chord Recognition Based on Midi-Trained Deep Feature and BLSTM-CRF Hybrid Decoding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, Apr. 2018.