Neural Networks for Analysing Music and Environmental Audio

Sigtia, Siddharth

Neural Networks for Analysing Music and Environmental Audio

Siddharth Sigtia
School of Electronic Engineering and Computer Science
Queen Mary University of London

A thesis submitted for the degree of
Doctor of Philosophy

2016

"This, I submit, is the freedom of real education, of learning how to be well-adjusted: You get to consciously decide what has meaning and what doesn't." - David Foster Wallace

Acknowledgements

I would like to begin by saying that I am grateful for the opportunity to write this thesis. The last 4 years have been undeniably tough, but this PhD provided the right set of ingredients for my academic, intellectual and personal development.

First and foremost, I would like to thank my supervisor Simon Dixon. Simon agreed to supervise me when I was still confused about my research topic. Over the following 4 years, I couldn't have asked for a more academically unconstrained environment. Simon let me read and explore whatever I was interested in, at the same time guiding me towards a practical topic of general interest. As an international student, I have had to face many financial difficulties over the last 4 years. In addition to being supportive and sympathetic, Simon recommended me for a series of academic research positions. Working as a research assistant allowed me to support myself and also provided an opportunity to pursue new research directions, all of which have found their way into this thesis in some form. His advice and comments about my work and writing have been invaluable. If it weren't for Simon's help and supervision, I would not have been able to finish this thesis. I feel fortunate and grateful to have been one of his PhD students for 4 years.

Next, I would like to thank my collaborators who assisted me with ideas and research over the last 4 years. I'm grateful for the opportunity to write a paper on evolutionary computation with Chrisantha Fernando and

Alexander Churchill. While working on the paper, I became interested in the biological and philosophical foundations of artificial intelligence, which has been a very interesting topic to learn about. I also had the opportunity to interact with and get to know both Alex and Chrisantha personally, which I've enjoyed over the last 4 years. Special thanks to Emmanouil Benetos. His expert opinion, guidance and help with many aspects of automatic music transcription research has been instrumental in some of the ideas presented in this thesis. Nicolas Boulanger-Lewandowski, who I've met only once, has been a consistent collaborator over the last 2 years. His work on conditional RNN graphs, followed by the many discussions we had over email helped me refine and improve my ideas. Mark Plumbley gave me the opportunity to work as an RA on a project on machine listening for environmental sounds. The project allowed me to explore some of my work in a different domain and helped broaden my domain of interest and inquiry, and understanding of audio signals. Sacha Krstulovic, for his expertise in speech recognition and his exacting demand for empirical rigour. John Bridle, for being an inspiring mentor during my internship at Apple Inc. I would also like to thank Adam Stark and Peter Foster for being great collaborators and sharing their expertise with me over the last 2 years. And Björn Schuller and Simon Godsill for examining this thesis, for a very interesting viva and for their helpful comments and suggestions for the final version of this thesis.

In addition to the people I directly worked with, I feel grateful to have met many inspiring people who have influenced me in myriad ways. I would like to thank Laurel, Katerina, Kathleen, Victor, Christian, Sebastian, Janis, Tian, Siying, Astrid and Bogdan for being great lab-mates and good friends. Tomack, for introducing me to many features of life in

London that I've grown to love and for being very welcoming when we first met. Dan Stowell, for very helpful discussions on machine learning and music. Matthias, for his useful advice and guidance near the end of this PhD. Mikhail, for being the best of friends over the last 4 years. I am grateful for his support and for his company through some difficult times. Beth, Matty and Lily for keeping me engaged with ideas far beyond my narrow field of academic research.

Finally, I would like to thank my parents Sanjay and Sarita and my sister, Shraddha. They were supportive even though they had no idea what this PhD was about. They were understanding and encouraging when I felt like the problems before me were intractable. I feel fortunate to have been able to live in London and relate my experiences to them and hopefully be a small window to the wider world. They have been ideal role models and I am continuously inspired by them.

Abstract

In this thesis, we consider the analysis of music and environmental audio recordings with neural networks. Recently, neural networks have been shown to be an effective family of models for speech recognition, computer vision, natural language processing and a number of other statistical modelling problems. The composite layer-wise structure of neural networks allows for flexible model design, where prior knowledge about the domain of application can be used to inform the design and architecture of the neural network models. Additionally, it has been shown that when trained on sufficient quantities of data, neural networks can be directly applied to low-level features to learn mappings to high level concepts like phonemes in speech and object classes in computer vision. In this thesis we investigate whether neural network models can be usefully applied to processing music and environmental audio.

With regards to music signal analysis, we investigate 2 different problems. The first problem, automatic music transcription, aims to identify the score or the sequence of musical notes that comprise an audio recording. We also consider the problem of automatic chord transcription, where the aim is to identify the sequence of chords in a given audio recording. For both problems, we design neural network acoustic models which are applied to low-level time-frequency features in order to detect the presence of notes or chords. Our results demonstrate that the neural network acoustic models perform similarly to state-of-the-art acoustic models, without the

need for any feature engineering. The networks are able to learn complex transformations from time-frequency features to the desired outputs, given sufficient amounts of training data. Additionally, we use recurrent neural networks to model the temporal structure of sequences of notes or chords, similar to language modelling in speech. Our results demonstrate that the combination of the acoustic and language model predictions yields improved performance over the acoustic models alone. We also observe that convolutional neural networks yield better performance compared to other neural network architectures for acoustic modelling.

For the analysis of environmental audio recordings, we consider the problem of acoustic event detection. Acoustic event detection has a similar structure to automatic music and chord transcription, where the system is required to output the correct sequence of semantic labels along with onset and offset times. We compare the performance of neural network architectures against Gaussian mixture models and support vector machines. In order to account for the fact that such systems are typically deployed on embedded devices, we compare performance as a function of the computational cost of each model. We evaluate the models on 2 large datasets of real-world recordings of baby cries and smoke alarms. Our results demonstrate that the neural networks clearly outperform the other models and they are able to do so without incurring a heavy computation cost.

Contents

List of Figures

List of Tables

List of Acronyms

ACT Automatic Chord Transcription

AED Acoustic Event Detection

AESR Automatic Environmental Sound Recognition

AMT Automatic Music Transcription

ASC Acoustic Scene Classification

ASR Automatic Speech Recognition

BPTT Backpropagation Through Time

CASA Computational Auditory Scene Analysis

CG Conjugate Gradients

ConvNet Convolutional Neural Network

CQT Constant-Q Transform

DET Detection Error Tradeoff

DFT Discrete Fourier Transform

DNN Deep Neural Network

EER Equal Error Rate

EM Expectation-Maximisation

ERB Equivalent Rectangular Bandwidth

FN False Negative

FP False Positive

GMM Gaussian Mixture Model

HF Hessian Free

HMM Hidden Markov Model

IoT Internet of Things

LSTM Long Short-Term Memory

LUT Lookup Table

MFCC Mel-Frequency Cepstral Coefficient

MIDI Musical Instrument Digital Interface

MIR Music Information Retrieval

MIREX Music Information Retrieval Exchange

MLE Maximum Likelihood Estimation

MLM Music Language Model

MLP Multi-Layer Perceptron

NADE Neural Autoregressive Distribution Estimator

NMD Non-negative Matrix Deconvolution

NMF Non-negative Matrix Factorisation

PCP Pitch Class Profile

PLCA Probabilistic Latent Component Analysis

QoS Quality of Service

RBF Radial Basis Function

RBM Restricted Boltzmann Machine

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

STFT Short-Time Fourier Transform

SVM Support Vector Machine

TP True Positive

UBM Universal Background Model

Chapter 1

Introduction

This thesis deals with the problem of using machine learning to describe the contents of audio signals. The dramatic proliferation of digital media has engendered the development of computational systems to automatically describe the contents of and retrieve information from large databases of digitally stored data like images, speech, music and videos. Here we consider the analysis of two types of audio signals: music audio and environmental audio recordings.

1.1 Aims and Motivations

In this thesis, we investigate the problem of analysing (or describing) the contents of music and environmental audio recordings with neural networks. We follow the computational perception approach, where the relationships between the inputs (audio recordings) and outputs (high-level semantic labels like chords, notes and acoustic events) are defined by mathematical (or statistical or computational) models. The model design is typically informed by prior knowledge about the application domain and the parameters of the models are estimated given many examples of inputs and outputs. Although simple in conception, the problem of imitating human perception with computational models poses several challenges. The relationship between the

inputs and the outputs do not necessarily need to be objective. For instance, identifying a well defined object category like an apple in an image is an easier problem than identifying the *mood* of an audio recording. This is due to the fact that the perceived mood depends on many subjective qualities related to the listener and is not fully encoded in the audio recording. Similarly, identifying *expressive* properties of music audio like tension, excitement and relief is a difficult task since these quantities are dependent on the listener and various cultural factors and are therefore subjective. Despite the challenges posed by the subjective nature of perception, in the last decade computational models have achieved considerable success in several domains such as computer vision, speech recognition and natural language processing. Machine learning based speech recognition and computer vision systems are now available as commercial applications and are used by millions of people every day. These recent advances and developments provide strong motivation for investigating more complex perception problems like analysing the contents of music and environmental audio recordings.

One of the motivations for building computational models for automatically analysing the content of digital data is the availability of large corpora of digital images, music, speech and other types of data. With regards to music, services like Spotify[1], Apple Music[2] and Youtube[3] provide large databases of music audio. Organisation and storage of large databases requires a large amount of human labelling and annotation to create accurate *metadata*. Specific recordings can then be retrieved or recommended based on the metadata. However, annotating millions of songs by hand can be a very expensive and time-consuming process. The collection, storage and retrieval from datasets could be greatly simplified if the process of metadata generation, which involves high-level descriptions of recordings like artist, album and genre, were to be

[1] `https://www.spotify.com/`
[2] `www.apple.com/uk/music/`
[3] `https://www.youtube.com/`

automated. These arguments also apply to large datasets of environmental sounds which are now being collected for applications in remote monitoring for health, security and surveillance.

Over the last 2 decades, research on both music audio and environmental audio has received increasing attention. Although a lot of progress has been made in both fields, a large number of studies follow a similar methodology: extract audio features from the recording followed by statistical modelling of the relationship between the audio features and the high-level descriptors. A lot of time and effort has been dedicated to discovering the right combination of acoustic features and classifiers. Given the space of all acoustic features and all classifiers, finding the appropriate combination for a specific problem by brute force is intractable. The search space is typically constrained by employing domain knowledge (from either music or environmental sound research) to design or hand-craft useful acoustic features. In this thesis, we investigate neural networks for statistical modelling. There are several motivations for applying neural networks for processing audio signals. Firstly, it has been demonstrated in many other fields that given sufficient data, neural networks can be directly applied to raw data or low-level features extracted from the data, to simultaneously learn the features and the classifier for a given task. Neural networks are compositions of simple non-linear parametric transformations. Given many examples, the parameters of the network can be estimated, consequently jointly learning the features and the classifier.

A second motivation for using neural networks is for sequential modelling. Data like audio, video and natural language (text) are inherently sequential. Neural networks offer several flexible architectures for modelling sequences. Typically, sequential modelling has been performed using state-space models like hidden Markov models. However, state space models are limited since the output spaces for some problems can be very large, for instance automatic music transcription. Estimating the parameters for a very large number of states becomes intractable with a limited number of

examples for training. Recurrent neural networks are flexible models for sequential data with a continuous state space, similar to linear dynamical systems. Therefore in addition to classification, we investigate the use of neural networks for modelling sequences of musical notes and chords.

Finally, in addition to adapting ideas from machine learning towards a specific domain, the inverse problem is of equal interest, where observations and insights from a given domain of application can benefit machine learning. For instance, convolutional neural networks were inspired by studies on the feline visual cortex. This thesis aims to discover novel ways of using neural network architectures for processing music and environmental audio. Both domains of application offer unique challenges for processing sequential data with neural network models. We hope that the methods developed and presented in this thesis can be of general interest to the machine learning community and find application in diverse domains.

Given these motivations, we consider 3 different problems in audio signal analysis. For music audio, we investigate the problem of automatic music transcription, which aims to identify the score or sequence of notes given a music recording. We also consider the related problem of chord recognition, which aims to identify the sequence of chords in a music recording. For both problems, we use neural networks for processing the audio signal and for modelling the structure in sequences of notes and chords. Finally, we investigate the problem of audio event detection for environmental sounds. In addition to comparing the performance of various neural network models for event detection, we study performance as a function of computational cost in order to determine the viability of commercial deployment of the proposed system.

1.2 Thesis Outline

The rest of the thesis is organised as follows:

- **Chapter 2** reviews the relevant literature and provides necessary background for the three problems considered in this work: automatic music transcription, automatic chord transcription and audio event detection for environmental sounds.

- **Chapter 3** presents a brief history of neural networks and lists some of the important advances in the last 2 decades. This is followed by a formal description of all the neural network architectures used in this thesis along with a discussion about numerical optimisation.

- **Chapter 4** investigates automatic music transcription with neural networks. The chapter is divided into 2 parts. The first part of the chapter presents a system for polyphonic piano music transcription. The second half of the chapter investigates the applicability of music language models on a multi-instrument polyphonic music transcription task.

- **Chapter 5** investigates automatic chord transcription with neural networks.

- **Chapter 6** investigates audio event detection for environmental sounds with neural networks.

- **Chapter 7** concludes the thesis by summarising the presented work and considering potential avenues for future research.

1.3 Associated Publications

Most of the work presented in this thesis has been previously published in journal articles or peer-reviewed conference proceedings. In this section we enumerate the publications that are related to the results and analysis presented in this thesis.

Journal Articles

[1] Sigtia, S., Stark, S., Krstulovic, S. and Plumbley, M. (2016) "Automatic Environmental Sound Recognition: Performance versus Computational Cost." IEEE/ACM Transactions on Audio, Speech, and Language Processing, 24 (11), 2096–2107.

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

Peer-Reviewed Conference Papers

[3] Sigtia, S., Boulanger-Lewandowski, N. and Dixon, S. "Audio Chord Recognition with a Hybrid Recurrent Neural Network." *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain, October 2015.

[4] Sigtia, S., Benetos, E., Boulanger-Lewandowski, N., Weyde, T., Garcez, A. and Dixon, S. "A Hybrid Recurrent Neural Network for Music Transcription." *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015.

[5] Sigtia, S., Benetos, E., Cherla, S., Weyde, T., Garcez, A. and Dixon, S. "An RNN-based Music Language Model for Improving Automatic Music Transcription." *Proceedings of the 15th International Society for Music Information Retrieval (ISMIR)*, Taipei, Taiwan, October 2014.

[6] Sigtia, S. and Dixon, S. "Improved Music Feature Learning With Deep Neural Networks." *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014.

Other Publications

[7] Foster, P., Sigtia, S., Krstulovic, S., Barker, J. and Plumbley, M. "Chime-home: A Dataset for Sound Source Recognition in a Domestic Environment." *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New York, USA, October 2015.[4]

All experiments and analysis for the publications listed above and this thesis were performed by the author. The co-authors provided advice while performing experiments and helpful comments and reviews for drafts of the above publications and this thesis. For the work presented in Sigtia et al. (2016), EB provided the output probabilities from the acoustic models by Benetos and Dixon (2012) and Vincent et al. (2010). Similarly for the results in Sigtia et al. (2014), EB provided the output probabilities from the acoustic model by Benetos et al. (2013). Note that Chapter 4 is based on [2,4,5], Chapter 5 is based on [3,6] and Chapter 6 is based on [1]. SK and the author of this thesis contributed equally to the writing of [1].

1.4 Contributions

In this section we enumerate the novel contributions from each chapter.

Chapter 4

- A hybrid RNN architecture for incorporating the predictions of an RNN music language model with the predictions of an arbitrary frame-level classifier.
- A beam search based inference procedure for the outputs of the hybrid RNN model.
- An application of ConvNets for acoustic modelling for automatic music transcription.

[4]The author assisted with the collection and annotation of the dataset presented in the publication.

- The hashed beam search algorithm, a modification to beam search to encourage diversity in explored solutions.
- A comparison of neural network acoustic models to state-of-the-art unsupervised models for piano music transcription.
- A novel method for combining the predictions of a PLCA acoustic model and an RNN music language model for multi-instrument polyphonic music transcription.

Chapter 5

- A hybrid RNN model for incorporating the predictions of an RNN chord language model with an arbitrary frame-level classifier.
- A comparison of different neural network architectures for automatic chord transcription.
- An evaluation of the proposed model on an abridged version of the MIREX automatic chord recognition dataset.
- An evaluation of the hashed beam search algorithm for automatic chord transcription.

Chapter 6

- Neural network acoustic models for acoustic event detection for environmental sounds.
- Derivations of computational cost estimates for neural network acoustic models, support vector machines and Gaussian mixture models.
- An evaluation of the proposed models on 2 large datasets of environmental sounds for industrial applications.
- A comparison of model performance as a function of computational cost.

Chapter 2

Literature Review

This chapter provides necessary background and literature review for the ideas presented in the rest of the thesis. This thesis explores machine learning approaches for analysing the *content* of audio signals. Two types of audio signals are considered: music audio and environmental audio. The problem of identifying properties or features of music (like notes, chords, rhythm, lyrics) that are encoded in the music recording is related to the field of Music Information Retrieval (MIR) (Schedl et al., 2014), while the problem of identifying the sources in a recording of environmental or ambient sounds is studied in the field of computational auditory scene analysis (CASA) (Wang and Brown, 2006).

Like the rest of the thesis, this chapter is divided into 2 parts. The first section provides a review of some basic musical concepts and the literature related to the problems considered in this thesis, automatic music transcription and automatic chord transcription. The next section reviews some of the concepts and literature related to processing audio recordings of environmental sounds.

2.1 Music Information Retrieval

In this section, we review some basic musical concepts which are used throughout the thesis. A music recording has many features like the recording formal (vinyl, cassette, CD), the lyrics (if any), the instruments used to create the piece, the particular performance of the piece and so on. The various problems studied in MIR can be broadly divided into 2 classes: problems that are related to musical properties that can be inferred from the recording (like notes, chords, beats, rhythm) or problems related to subjective qualities of music that are interpreted by the listener (user preferences, mood, social context) (Schedl et al., 2014). In this thesis, we are interested in problems related to *musical content* or properties that are encoded in the music audio recording.

2.1.1 Pitch, Interval and Scales

In this thesis, a music signal refers to an audio recording of one or more sources which may include various musical instruments and singing voices. Musical instruments can be broadly classified into 2 types: pitched instruments like guitars, pianos and cellos and unpitched instruments like drums. Pitch is a perceptual attribute that allows the ordering of sound according to a frequency-related scale (Klapuri, 2006). Although related to frequency, pitch is not an objective attribute but a perceived psycho-acoustical attribute of sound (Houtsma, 1995). A pitch is produced by the combination of a number of *harmonically related* tones. A tone is a periodic sine wave at some frequency. A pitch is denoted by a *fundamental frequency* or f_0. In addition to the fundamental frequency, the pitch constitutes a series of tones at roughly integer multiples of the fundamental frequency; $f_n = nf_0$, where $n = \{2, 3, 4, \ldots\}$. Pitch is an important attribute of sound and forms an essential component of features like melody and harmony in music. In addition to the frequency, attributes like amplitude, duration and temporal envelope all affect the perception of pitch (Houtsma, 1995).

The series of tones that constitute a pitch are also known as *partials*, where the fundamental tone is the first partial, the first overtone is the second partial and so on. Therefore for a given pitch, the energy is concentrated around the f_0 and the remaining partials.

The *chroma* and *height* are two important attributes of pitch (Shepard, 1964). In Western music, the chroma is divided into 12 *semitones* from A to G, with 5 accidentals (sharp (♯) or flat (♭)):

$$\{C, C\sharp/D\flat, D, D\sharp/E\flat, E, F, F\sharp/G\flat, G, G\sharp/A\flat, A, A\sharp/B\flat, B\}. \tag{2.1}$$

The twelfth semitone above any given note has the same chroma, but greater *height*. The height is therefore proportional to the fundamental frequency of the pitch, while the chroma cyclically repeats itself over octaves. The *interval* of 12 semitones is known as an *octave*, while the cyclic repeating property of chroma is termed *octave invariance*. This is also know as the 12-Tone Equal Temperament or 12-TET, stemming from the fact that the 12 chroma are equally spaced along the log-frequency axis within an octave. Although many different tuning systems are possible, in this thesis we focus only on the 12-TET. In 12-TET, the fundamental frequency of a pitch is defined as:

$$f = f_{\text{ref}} 2^{n/12}, n = \{\ldots, -1, 0, 1, \ldots\}, \tag{2.2}$$

where f_{ref} is a reference frequency usually set to 440 Hz.

The interval is a musical quantity that relates notes. Mathematicians as early as Pythagoras had demonstrated that strings with identical properties but different lengths produced consonant sounds if the lengths were in a particular ratio (Terhardt, 1984). Since then the physical attributes of pitches and notes are usually described in the frequency domain and intervals are defined as ratios of fundamental frequencies

(Terhardt, 1984). For instance the *octave* is defined as the interval between 2 fundamental frequencies in the ratio $2:1$. Similarly, the perfect fifth is defined by the ratio $3:2$ and the major third is defined by the ratio $5:4$ and so on. The concept of intervals is extended to *scale*. A scale is defined as an ordered set of intervals. For example the major diatonic scale is defined as $\{+2, +2, +1, +2, +2, +2, +1\}$, where each entry in the set defines an offset measured in the number of semitones from the previous note in the scale. The first pitch of the scale is called the *tonic* and the scale derives its name from the tonic. For example a major scale starting with the C note is denoted as the C-major scale. Circular rotations of a scale give rise to different *modes*. For example a circular rotation by 1 results in the Dorian mode, a rotation of 2 produces the Phrygian mode and so on. The *key* of a section of music is the defined as the scale that best fits the sequence of notes in the section.

2.1.2 Chords

A chord can be defined as *any set of three or more notes that are sounding simultaneously* (Karolyi, 1965; Benward, 2014). Like the definitions of many perceived musical quantities, the above definition is incomplete and several alternative definitions exist. This difficulty arises from the fact that there are many instances of chords in practice that deviate from the above definition. For example, it has been shown that the notes in a chord do not necessarily need to be played simultaneously (Deutsch, 1969). Humans are able to perceptually integrate notes played sequentially and often perceive them as a single harmonic object, namely the chord. Another deviation from the above definition is the fact that some chords can be played with certain notes omitted, but the listener perceives a chord by filling in the harmonic gaps (Ulrich, 1977).

Although the musical definition of a chord is difficult due to the many perceptual and subjective qualities associated with it, in the rest of the thesis we refer to the

above definition of chords, which defines the chord as a harmonic quantity formed by the combination of 3 or more notes. It is important to note that the chord is an abstract quantity that isn't necessarily encoded in the music signal, for instance chords with omitted notes. One of the most common chord types in Western music is the *triad*. A triad comprises 3 notes, a *root* or the starting point, followed by the third and fifth notes in a scale. A chord is defined by the root and relationship between the root note and other notes in the chord.

2.1.3 MIDI Representation

Representing music symbolically is a challenging problem. Music comprises elements like pitch, time and tempo that can be represented symbolically. At the same time it also contains elements like tension and expectation which are much harder to represent. Consequently, representing music in a computer is in itself a field of study (Dannenberg, 1993). The most widely used framework for computer music representation is the Musical Instrument Digital Interface (MIDI) protocol. The MIDI protocol provides 16 channels. Each note can is specified by its pitch number, onset time, offset time and amplitude. Additionally parameters such as instrument type, pitch-bend, key and tempo can also be specified.

The MIDI protocol assigns an index to each pitch as follows:

$$n = 12 \cdot \log_2 \frac{f_0}{440} + 69, \tag{2.3}$$

where 440 Hz is f_{ref} (Equation 2.2) and n is the MIDI note index. A typical piano consists of notes from A0 to C8. A0 at 27.5 Hz corresponds to MIDI pitch $n = 21$ while C8 at 4186 Hz corresponds to MIDI pitch $n = 108$. Therefore a piano spans the range of MIDI notes 21-108.

Although useful for computational analysis, the MIDI protocol is limited as it does

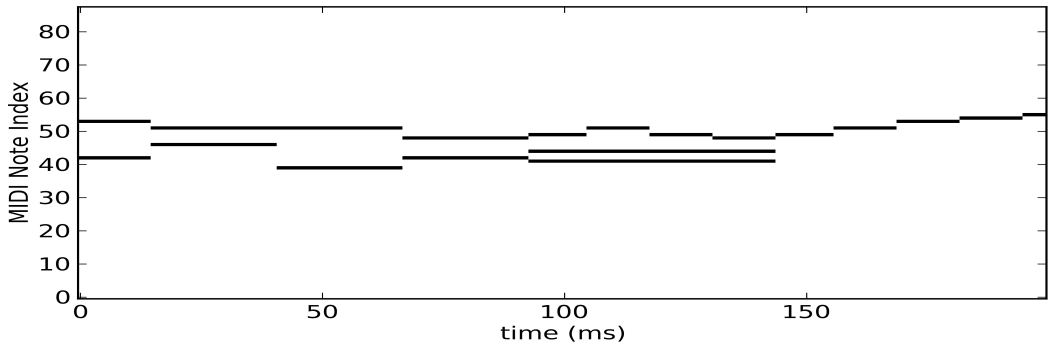


Figure 2.1: A musical piece in piano-roll notation. X-axis corresponds to time (ms), Y-axis represents pitch index

not allow for expressive musical features. Regardless, MIDI notation is extensively used in MIR. A convenient way of graphically representing MIDI is the *piano-roll* notation, where the Y-axis corresponds to pitch index and the X-axis corresponds to time (Figure 2.1).

2.1.4 Automatic Music Transcription

Automatic Music Transcription (AMT) is a fundamental problem in MIR. AMT aims to generate a symbolic, score-like transcription, given an acoustic signal (Figure 2.2). Music transcription is considered to be a difficult problem even by human experts and current music transcription systems fail to match human performance (Benetos, 2012). Polyphonic AMT is a difficult problem because concurrently sounding notes from one or more instruments cause a complex interaction and overlap of harmonics in the acoustic signal. The problem is further complicated due to the presence of more than one instrument source. Additionally, AMT systems with unconstrained polyphony have a combinatorially large output space, which further complicates the modelling problem.

The various approaches to AMT can be roughly classified into signal processing methods and statistical machine learning methods. Here we review previous work for

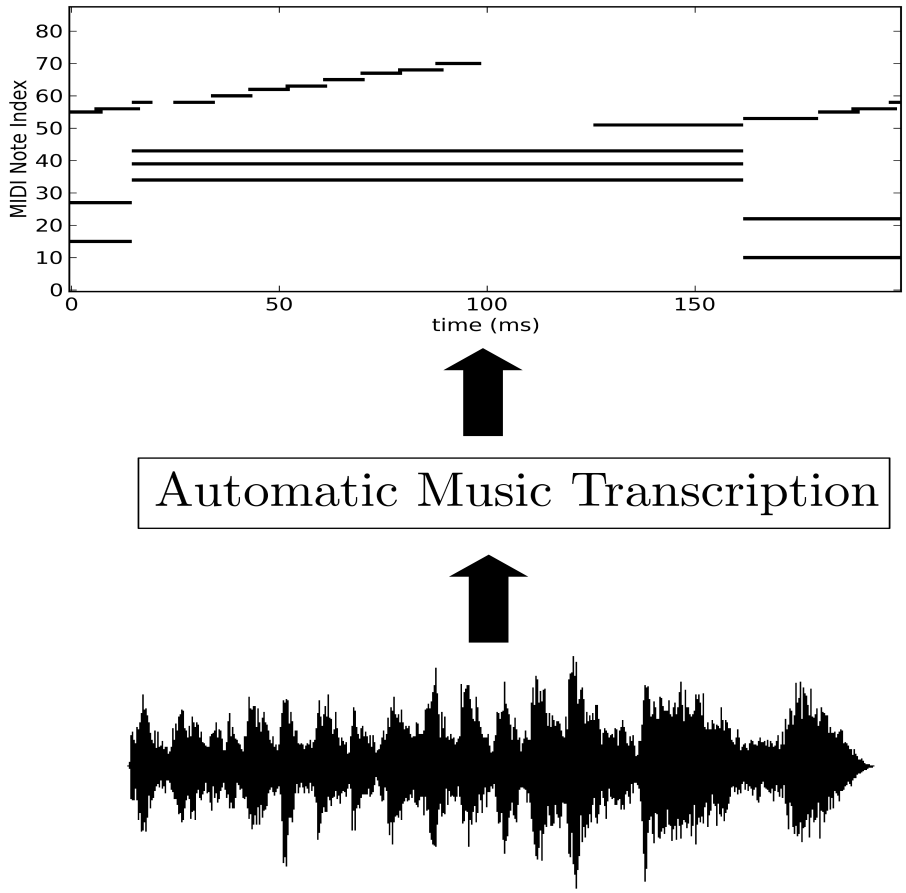


Figure 2.2: An overview of Automatic Music Transcription. The input recording is transcribed to a symbolic score-like representation.

both types of systems.

2.1.4.1 Signal Processing Methods

Signal processing methods typically utilise prior information about the distribution of energy over the set of harmonics related to each pitch in order to transcribe a given music recording. The audio is first transformed into a time-frequency representation. For each time-frequency *frame*, spectral peaks are determined. Pitch salience functions are then generated for each considered pitch and compared with the spectrum. The polyphonic pitch content of each frame is then determined iteratively or jointly for multiple pitches. The transcriptions produced by such methods are prone

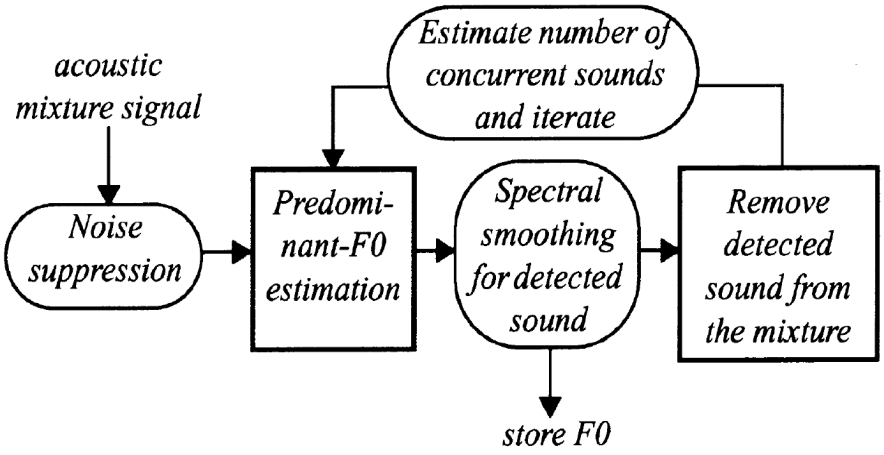


Figure 2.3: An overview of the iterative spectral subtraction method for AMT (Klapuri, 2003).

to harmonic errors. This is due to the fact that the harmonic amplitude for different pitches can vary greatly depending on the source. Additionally, pitches produced by different sources have overlapping partials which is hard to incorporate into the estimation algorithm.

Klapuri (2003) presents a signal processing based method for AMT. The proposed method iteratively estimates the fundamental frequency of the most prominent sound, subtracts the sound from the mixture and the process is repeated for the residual signal. The fundamental frequency is estimated using a band-wise pitch salience function, which accounts for the fact that the harmonic relationships between spectral components is not necessarily ideal. The spectrum of the detected sound is first smoothed in order to avoid corrupting overlapping partials and then subtracted from the mixture (Figure 2.3). A polyphony inference method is applied to stop iteration. Ryynänen and Klapuri (2005) extend the model in Klapuri (2003), by incorporating a hidden Markov model (HMM) (Rabiner, 1989). The likelihoods of different note-combinations are extracted over time. The HMM is used to impose certain musicological constraints on note transitions. The final transcription is obtained by estimating the sequence of note combinations that maximise the likelihood

of the input time-frequency representation.

Argenti et al. (2011) use a constant-Q transform (CQT) (Brown, 1991) time-frequency representation for the audio signal. A 2-D transform of the third cumulant of the signal or the *bispectrum* is computed. This is done in order to account for the non-linear interactions between partials. This is followed by computing a cross-correlation between the audio signal and a harmonic template. The highest correlation is taken as the estimate for a pitch. The pattern corresponding to the detected note is subtracted from the signal and the process is repeated.

Yeh et al. (2010) propose a rule-based system for estimating multiple pitches jointly. The system first classifies spectral peaks in the signal into sinusoids and noise. The peaks are then matched against candidate harmonic patterns. Rather than iteratively estimating each pitch separately, a combination of pitches is jointly considered by estimating overlapping partials. The polyphony of the hypotheses is gradually increased till all considered hypotheses are exhausted. Finally, a polyphony inference algorithm is used to determine the best combination of pitches.

A review of other signal processing based systems can be found in Benetos (2012).

2.1.4.2 Probabilistic Spectral Peak Modelling

An alternative to purely signal processing methods is to probabilistically model the occurrence of pitches in a music signal. Duan et al. (2010b) propose a polyphonic transcription system that probabilistically models spectral peaks. The likelihood function models the probability of detecting a peak in the spectral frame given a pitch and the complementary probability of not detecting a peak. The training data is used to learn prior probabilities of pitch occurrences. A polyphonic inference method using thresholds is used as a stopping criterion for further iterations. The method was found to be efficient at balancing harmonic and sub-harmonic errors.

Emiya et al. (2010) propose a method that models overlapping partials with a smooth autoregressive model. The background noise is modelled with a moving average filter. The resulting model generatively models the input spectrogram given simultaneous active piano notes. Badeau et al. (2009) propose an expectation-maximisation (EM) (Bishop, 2006, Chapter 9) based algorithm which performs successive single-pitch and spectral envelope estimations in order to maximise the likelihood of observing the input spectrum. Peeling and Godsill (2011) present an alternative system where the expected partial density of a given note at some frequency is assumed to be a homogeneous Poisson process. Concurrent notes are assumed to be independent, resulting in an independent Poisson process which is a superposition of the Poisson processes for each note. The resulting Poisson process is modelled with a Gaussian mixture model (GMM) (Bishop, 2006, Chapter 9) centred at the harmonics of a given note.

2.1.4.3 Full Spectrum Modelling

Another class of approaches to polyphonic music transcription aim to define a probabilistic distribution over the observed spectrograms. Typically, each note is modelled by a GMM and each spectral frame is expressed as a mixture of note GMMs. Such models have the advantage that it is very easy to incorporate note priors into the model.

Goto (2004) proposes *PreFEst*, an algorithm for predominant-F0 estimation of melody and bass lines based on MAP (maximum a posteriori) estimation. The time-frequency spectra are modelled as a weighted superposition of adapted tone models. The tone models comprise a Gaussian placed at harmonic positions along the frequency axis. MAP estimation is used to estimate the model parameters and the melody and bass lines are inferred using a multiple-agent architecture which selects the most stable trajectory. Miyamoto et al. (2007) propose a method called har-

monic temporal structure clustering (HTC) which clusters the spectral energy into musical notes. The method utilises a structured GMM with harmonic and temporal smoothness constraints. The parameters of the GMM are estimated using the EM algorithm. Yoshii and Goto (2012) propose infinite latent harmonic allocation (iLHA) based on non-parametric Bayesian methods. The model allows for an arbitrary number of sound sources and an arbitrary number of harmonic partials. Each spectral frame is modelled as a nested infinite dimensional GMM using Dirichlet processes. They propose a method to perform inference on the trained model using a modified variational Bayes algorithm.

2.1.4.4 Spectrogram Decomposition Methods

A majority of recent polyphonic transcription algorithms have been based on non-negative matrix factorisation (NMF) (Lee and Seung, 2001). Given a 2-dimensional time-frequency representation, NMF is used to decompose the spectrogram as $S \approx WH$. Essentially, the input S is decomposed into a linear combination of *basis vectors* in W and a set of *weights* in H. When applied to music analysis, S is usually the magnitude spectrogram (or any suitable time-frequency representation), W is the spectral basis matrix where each entry (row) represents a source spectrogram and H is the activity or weight matrix over time. Prior information and constrains can be incorporated into the NMF algorithm by means of regularisation terms or by directly imposing constraints on the matrix. Consequently there have been many variants of NMF applied to music transcription. Below, we review some of the important contributions.

Smaragdis and Brown (2003) propose an NMF based model for music transcription. They assume each note has a fixed spectral profile which is estimated based on the training examples, along with the temporal activity matrix for each note. Cont (2006) extend the previous approach by incorporating sparsity constraints for the

activity matrix. The motivation behind this was to express the magnitude spectrogram as the sum of the smallest possible number of bases. The model also included HMM based post-processing for note-tracking. Vincent et al. (2008) add constraints on harmonicity in the NMF model. Additionally, they impose smoothness constraints on the spectrum by expressing each basis spectrum as a sum of narrowband spectra. They use an equivalent rectangular bandwidth (ERB) scale input time-frequency representation. The harmonic constraints along with the post-processing technique were further improved in Vincent et al. (2010). Bertin et al. (2010) propose a Bayesian NMF method with harmonicity and temporal continuity constraints. The harmonicity constraints are introduced using superimposed Gaussian components while the temporal constraints are imposed using an inverse-Gamma Markov chain prior. Dessein et al. (2010) include β-divergence into the NMF objective function. Sakaue et al. (2012) present a Bayesian multi-pitch analyser called Bayesian non-negative harmonic temporal factorisation (BNHTF), where they model the harmonic and temporal structures of the wavelet spectrogram of music signals. The BNHTF integrates latent harmonic allocation (LHA) with the NMF framework for music transcription. Sakaue et al. (2013) extend the Bayesian NMF framework with the inclusion of a *characteristic prior* distribution over the latent variables. Raczynski et al. (2013) present an AMT model based on NMF, which also incorporates relationships between pitches over time. Typically, the evolution of pitches over time is modelled independently for each pitch, since jointly modelling the state space of output pitches is infeasible due to the combinatorially large space. They test their model on synthesised sounds and report that their joint model outperforms NMF based models that do not incorporate structural information related to polyphonic pitch sequences. An extension to the NMF algorithm is to express the basis spectra as 2-dimensional quantities which are *convolved* with the 2-dimensional temporal activity matrix. Smaragdis (2004) first proposed the non-negative matrix deconvolution (NMD) algorithm. Schmidt

and Mørup (2006) extend the NMD algorithm to include sparsity constraints.

2.1.4.5 Probabilistic Latent Component Analysis

Probabilistic Latent Component Analysis (PLCA) is an alternative to NMF methods. It can be regarded as a probabilistic analogue of the NMF algorithm. PLCA provides a probabilistic framework that generalises well and also allows for efficient incorporation of priors. In PLCA, the spectrogram distribution is modelled as $P(\omega)_t = \sum_z P_t(\omega|z)P_t(z)$, where z is the basis index and ω is the log-spectrogram index. The unknown distributions are estimated using the EM algorithm. The PLCA model was first introduced in Smaragdis et al. (2006). The method was extended to allow more than one template for each pitch. This was used to extend the model to be able to recognise multiple instruments (Grindlay and Ellis, 2011). Benetos and Dixon (2012) present a shift-invariant PLCA model that can transcribe polyphonic music from multiple instruments. The shift-invariant properties of the model can be used to detect frequency modulations and tuning changes, and to visualise pitch content. A computationally more efficient version of the model is presented in Benetos et al. (2013). Recently, Berg-Kirkpatrick et al. (2014) proposed an unsupervised model for piano transcription. The method is similar to PLCA, however the model includes additional random variables that model properties specific to piano sounds like the amplitude envelope of pitches. The model produced state-of-the-art results on a piano transcription task.

2.1.4.6 Sparse Coding

Approaches based on sparse coding (Olshausen and Field, 1997) are similar in motivation to NMF. Sparse coding methods aim to decompose the observed time-frequency representation into a matrix of basis vectors and another matrix for activations. However, it is assumed that the activation vector is *sparse* with respect to the number of

active sources.

Blumensath and Davies (2004) propose a sparse coding approach that used iterative re-weighted least squares to estimate the bases for polyphonic piano music. Abdallah and Plumbley (2006) present a probabilistic model that *learns* sparse linear decompositions of the input time-frequency representation. Virtanen (2004) presents a convolutive variant of a sparse coding model for sound source separation. Canadas-Quesada et al. (2008) present a method based on a harmonic matching pursuit algorithm. They employ an additional processing step where spectral smoothness constraints are imposed on the learnt bases. O'Hanlon and Plumbley (2011) propose a *structure aware* dictionary learning algorithm where prior knowledge about the number of sources in the spectrogram and the harmonic structure of the bases is incorporated into NMF and k-singular value decomposition (k-SVD) based systems. O'Hanlon et al. (2012) use group sparsity to incorporate priors into a sparse coding based AMT system, while O'Hanlon and Plumbley (2014) incorporate the concept of group sparsity in a supervised NMF based AMT system.

2.1.4.7 Classification Approaches

Recently, there has been a lot of interest in machine learning based classification approaches to AMT. The key idea is that the models aim to *classify* time-frequency frames or audio features derived from them, into the output pitches. Depending on the type of model used, the algorithms either try to learn a conditional distribution $P(y|x)$, where y are the output pitches and x is the input acoustic feature vector or they directly learn a function $f : x \rightarrow y$. Classification based approaches have the advantage that they can theoretically learn complex input-output relationships between the input features and output pitches from many pairs x, y of training examples. However, the application of such methods is limited to the availability of labelled datasets of sufficient size to be able to learn the many parameters associated

with such models.

Marolt (2004) evaluated using neural networks for note transcription and reports good performance with time-delay neural networks (TDNN). Pertusa and Iñesta (2005) also use TDNNs to classify pre-processed short-time Fourier transforms (STFTs) into notes. Poliner and Ellis (2007) use support vector machines (SVMs) (Hearst et al., 1998) along with an HMM to process the frame-level outputs of the classifiers. Guibin and Sheng (2007) also use a feed-forward neural network to classify features derived from adaptive comb filters into notes. Costantini et al. (2009) use SVMs to classify CQTs into notes. Nam et al. (2011) use a deep belief network to learn audio features which are used as inputs to an SVM for classification. Böck and Schedl (2012) use bi-directional RNNs to classify multi-channel spectrograms obtained using 2 filter-banks at different temporal resolutions. Boulanger-Lewandowski et al. (2013b) present an RNN that learns a joint distribution over the output pitches conditioned on the inputs and previous outputs from the system.

In addition to the literature reviewed here, we direct the reader to Tavares et al. (2013) for a survey on AMT approaches.

2.1.5 Automatic Chord Transcription

Automatic Chord Transcription (ACT) is one of the fundamental problems in MIR. Given a music audio recording, the aim of AMT is to produce the sequence of chord labels that represent the chords and chord changes in the given piece (Figure 2.4). Chords are considered to be a mid-level representation of the harmonic content of a musical piece. Although chords are very useful descriptors of a musical piece, manually annotating musical pieces requires considerable expert musical knowledge and is time consuming (McVicar et al., 2014).

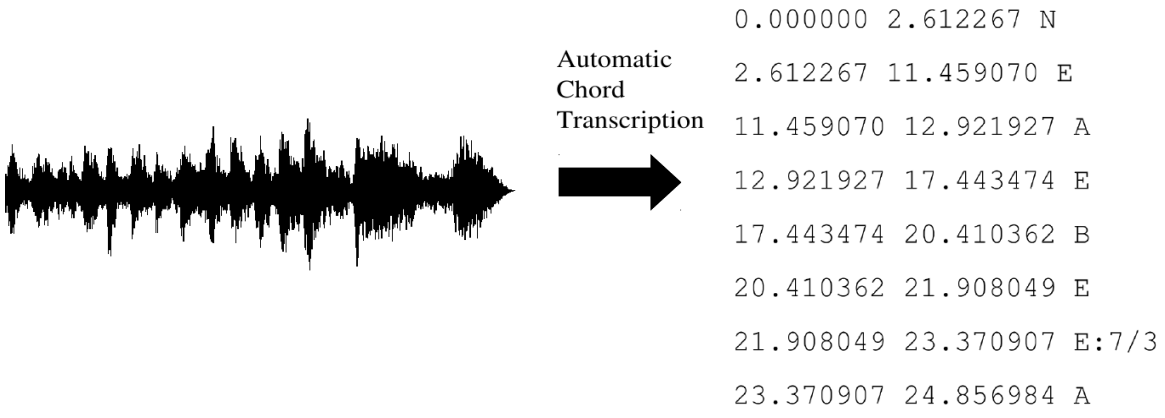


Figure 2.4: Graphical representation of the outputs produced by an ACT system (McVicar et al., 2014) for a musical excerpt. The 3 columns in the output represent the onset time, offset time and chord label, respectively.

2.1.5.1 Early Work

The first studies that considered recognising chords from audio recordings were based on polyphonic AMT (Martin, 1996; Kashino and Hagita, 1996). Fujishima (1999) is the first study that considers chord transcription as an independent task. Martin (1996) and Kashino and Hagita (1996) proposed a new *feature* called the Pitch Class Profile (PCP), which is a 12-dimensional feature that wraps spectral content in a frame of audio into one octave. The first step in computing the PCP vector involves calculating the DFT over a small window of the audio. The PCP entry for each pitch class is then computed by summing the power spectrum over the frequency bins in the DFT that are related to a given pitch class, yielding a 12-dimensional vector. Similarly to the DFT, the PCP is calculated over overlapping windows to yield a 2-D matrix of PCP vectors, called the *chromagram* (Wakefield, 1999).

The earliest systems for ACT are based on *template matching* approaches. Fujishima (1999) uses dot products between the chromagram and pre-defined binary 12-dimensional chord templates in order to estimate chords in a frame. Harte and Sandler (2005) also estimate the relative strength of different chords in an audio frame using a similar technique. Bello and Pickens (2005) also use a binary template for

each chord class. However, they model each chord as a 12-dimensional Gaussian where the means for all the notes in a chord are set to 1 and remaining means are set to 0. Additionally, they define a covariance matrix that encodes the correlations between notes appearing in a chord. Catteau et al. (2007) use a similar method based on using Gaussian distributions to model chord templates. Papadopoulos and Peeters (2008) use Gaussian chord templates and the correlation coefficient as a measure of fit between chroma vectors and chord templates. Oudre et al. (2009) demonstrate that the Kullback-Liebler divergence is a good measure for the similarity between chroma features and templates.

2.1.5.2 Background Subtraction, Harmonics and Smoothing

A musical recording can include sounds from both harmonic and percussive sources. When analysing the harmonic content of a recording, the non-harmonic parts of the mixture can be considered to be the *background* signal (Pauws, 2004). It is assumed that the recorded spectrum is a sum of the harmonic and percussive spectra. *Background subtraction* can then be applied to isolate the harmonic elements in the signal in order to recognise chords. This is known as harmonic percussive source separation (HPSS). Reed et al. (2009) and Ueda et al. (2010) demonstrate that HPSS results in improved chord transcription accuracies.

As discussed for AMT, the harmonic profiles of different instruments combine to produce a complex overlap of partials in a music signal. The presence of harmonics is an important consideration when designing features for chord transcription. Papadopoulos and Peeters (2007); Pauws (2004) and Mauch and Dixon (2010) account for harmonics in their feature extraction stages. Varewyck et al. (2008) propose a novel chroma representation that accounts for both background subtraction and the removal of harmonics.

It has been observed (Fujishima, 1999) that using chroma features alone leads

to chord estimates that fluctuated rapidly over time. Smoothing is a simple and effective technique used to overcome this issue. Harte and Sandler (2005) use a low-pass filter to temporally smooth the chromagram over time, while others have used median filtering (Khadkevich and Omologo, 2009). Bello and Pickens (2005) use the fact that chords are stable over a short duration, by averaging frames over a beat resulting in beat-synchronous chroma features.

2.1.5.3 Tuning

The quantisation of frequencies into pitches is usually performed with respect to a reference or *tuning frequency* (Equation 2.2). Typically, the tuning frequency is set for A over middle C (A4) to 440 Hz. Sheh and Ellis (2003) demonstrate that some popular music tracks are not tuned to 440 Hz, which results in unreliable estimates for chord classes. Harte and Sandler (2005) use a chromagram with 3 frequency bins per semitone and assign the sub-band with most energy as the tuning frequency. Peeters (2006) define a set of possible tuning frequencies and utilise the frequency that best represents the chromagram for each track. Mauch (2010) represents tuning as an angle defined on the interval $(-\pi, \pi)$ for each semitone.

2.1.5.4 Hidden Markov Models

The models described so far rely on efficient signal processing and simple template matching based techniques to estimate the presence of various chords in a frame of audio. However as discussed before, due to the complex mixture of sources, there is a large spectral variation present in the acoustic music signal. Additionally, these methods do not model any structural information about harmony (like the key, chord progressions) which provides essential *prior* information about the temporal structure and progression of chord sequences. A large number of ACT systems employ machine learning methods to model the variability in the acoustic features and to model the

musical and structural relationships in chords and harmonic sequences.

The most common approach to ACT uses HMMs. HMMs have been widely applied to speech recognition (Rabiner and Juang, 1993). HMMs define a joint distribution over a sequence of *observed* variables $x = \{x_0, \ldots, x_t\}$ and a set of unobserved or *hidden* variables $y = \{y_0, \ldots, y_t\}$. The joint distribution is defined as:

$$P(x, y) = P(y_0)P(x_0|y_0)\prod_{t=1}^{t} P(y_t|y_{t-1})P(x_t|y_t). \tag{2.4}$$

The above factorisation assumes:

$$P(y_t|y_0^{t-1}, x_0^{t-1}) = P(y_t|y_{t-1}), \tag{2.5}$$

$$P(x_t|y_0^{t}, x_0^{t-1}) = P(x_t|y_t). \tag{2.6}$$

Equation 2.5 is the Markov property for hidden states, which states that a state at time t is only dependent on the state at $t-1$. Equation 2.6 represents the conditional independence of the observations x_t, given the hidden state y_t. For chord recognition, the observed quantities are the acoustic feature vectors, while each hidden state corresponds to a chord label. The parameters of an HMM are learnt given a training dataset. At test time, the sequence of chords is *inferred* by estimating the most-likely sequence of chord labels given the acoustic observations. An HMM therefore defines the acoustic model in the form of the observation distribution for each chord class. Secondly, it models the probability of allowed chord transitions via the transition matrix. Sheh and Ellis (2003) first use an HMM to identify chord boundaries and predict chord labels. Bello and Pickens (2005) adapt the HMM for real-time analysis and learn the transition matrix for each song independently. Khadkevich and Omologo (2009) extend previous work by explicitly modelling chord durations.

In the HMM framework, the probability of observing an acoustic feature vector given a chord label is defined by an *observation distribution*. Typically, the observa-

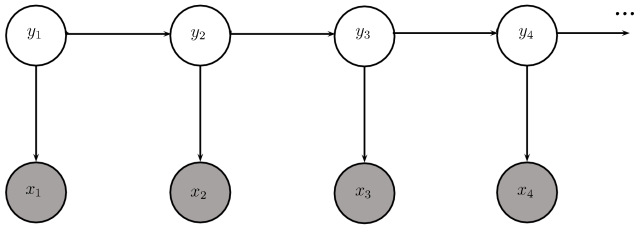


Figure 2.5: Graphical Model of an HMM. The observations x_t represent acoustic features, while the hidden variables y_t represent chord labels. The arrows denote the conditional independence assumptions from Equation 2.5 and Equation 2.6.

tion distribution is chosen to be a GMM which is parameterised by a mean vector μ and a covariance matrix Σ (Sheh and Ellis, 2003; Khadkevich and Omologo, 2009; Bello and Pickens, 2005; Cho and Bello, 2011; Reed et al., 2009; Sumi et al., 2008). The parameters of the distribution are then jointly estimated along with the remaining HMM parameters using the EM-algorithm. However, it is theoretically possible to use more complex observation distributions provided that the parameters can be estimated with EM. Burgoyne et al. (2007) use a Dirichlet distribution for the observation probabilities.

The HMM framework described above is limited since the only quantities that are modelled are the acoustic vectors and the chord labels. There have been several extensions of the HMM framework where musicological knowledge about chords are incorporated into the recognition model to improve performance. Simple HMM-based chord recognition models were extended by including another set of variables that model the *key* of the piece. The two-chain HMM jointly models the acoustic feature vector, chord label and the key for a given song. Raphael (2005) and Catteau et al. (2007) use a 2-chain HMM for chord recognition that jointly models the chord and key context in a piece and also allows for key changes within a song. The models proposed by Lee and Slaney (2008); Yoshioka et al. (2004) and Sumi et al. (2008) also incorporate key information. However, these models make the simplifying assumption

that the key for each song is fixed and therefore learn one HMM per key without allowing key changes within a song.

A few studies explicitly model information present in the *bass* frequencies of the spectrum. Ryynänen and Klapuri (2008) use a chroma representation where the first 12 dimensions correspond to a chroma vector over MIDI notes 26 to 49 (bass chroma) and 12 more dimensions that represent a chroma over MIDI notes 50-73. Other approaches model bass information more explicitly. This is done by first estimating a bass pitch class and then calculating the chord probabilities as a function of the detected bass note (Sumi et al., 2008; Yoshioka et al., 2004). Chord transcription systems have also benefited from the inclusions of beat or timing information. The likelihood of chord transitions is higher at certain metric positions, for instance the *downbeat*. Some systems use an iterative algorithm, where a downbeat estimation stage is followed by a chord estimation stage Shenoy and Wang (2005). Alternatively, Papadopoulos and Peeters (2008) jointly estimate the downbeat positions and chord sequence.

2.1.5.5 Dynamic Bayesian Networks

It was generally observed that incorporation of high-level musicological information into ACT systems helped improve performance. As discussed before, many different methods were proposed for incorporating prior knowledge into ACT systems. The use of dynamic Bayesian networks for ACT allowed many different musicological priors to be included under the same probabilistic recognition framework. Mauch and Dixon (2010) propose a dynamic Bayesian network for ACT that jointly models chords, key, metric position and bass notes along with bass and treble chromagrams (Figure 2.6). The model was shown to outperform existing models for chord recognition. Ni et al. (2012) include a chroma representation that accounts for loudness perception in the above framework and report state-of-the-art performance on a chord recognition task.

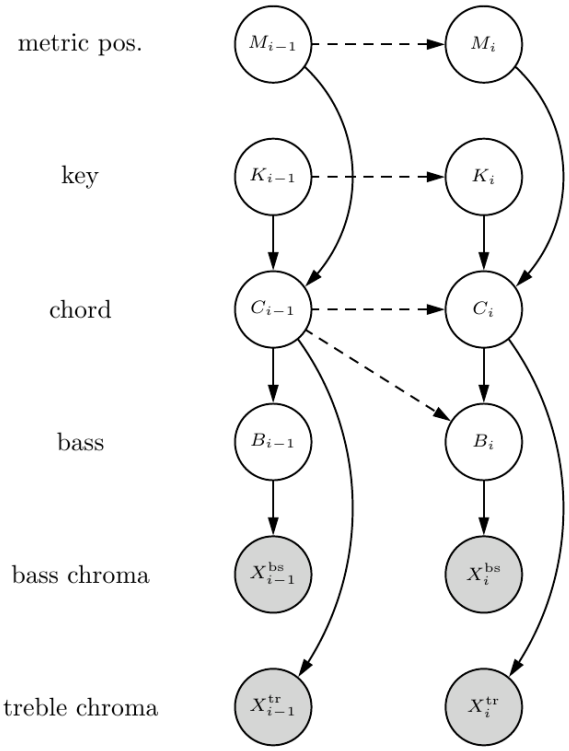


Figure 2.6: Graphical model of a dynamic Bayesian network for ACT (Mauch and Dixon, 2010). Compared to the HMM graph from Figure 2.5, the dynamic Bayesian network contains additional variables that represent bass (B_t), key (K_t) and metric position (M_t) in addition to the chord label (C_t). Additionally, the observation vector x_t is replaced by the bass (X_t^{bs}) and treble chroma (X_t^{tr}).

2.1.5.6 Classification Based Approaches

The various models discussed above have a common property. They define *generative* distributions $P(x, y)$ over the variables of interest y (the chord labels) and the observed acoustic features x. Similarly to AMT, it can be argued that since we are only interested in the chords given the feature vectors, it might be more sensible to model the conditional distribution $P(y|x)$ since the observations are fixed at test time. Estimating conditional distributions has the advantage that it relaxes the conditional independence assumptions made about observations (Equation 2.6). Furthermore, it allows the outputs y to be conditioned on feature vectors over more than one input frame. Burgoyne et al. (2007) use a conditional random field (CRF), a discriminative model, instead of an HMM for ACT. Weller et al. (2009) replace the CRF

with the SVM-struct algorithm for chord recognition. Recently, neural networks have been proposed for jointly learning the audio features and the classifier. The neural networks act on a time-frequency representation of the audio and learn to directly estimate chord sequences from the low-level acoustic inputs. Boulanger-Lewandowski et al. (2013a) propose an RNN-based system that simultaneously learns the acoustic classifier and structural information about chord sequences. Humphrey and Bello (2012) and Zhou and Lerch (2015) use a ConvNet to estimate chord labels given several frames of CQT feature inputs.

2.1.6 Deep Learning in MIR

In this section, we review some of the Deep Learning (see Chapter 3) approaches to various problems in MIR. The main motivation for applying deep learning to MIR problems is to automate the learning of audio features (Humphrey et al., 2013). In the last decade, as the popularity of machine learning approaches in MIR has grown, a large amount of time and effort has been expended in trying to determine the best combination of audio features and machine learning model. Deep learning methods provide an alternative solution to this problem by jointly learning feature transformations and the classifier from input data, given a large dataset of training examples. Although this research direction is intriguing and there have been some promising results, the applicability of such systems is limited to cases where large amounts of labelled training is available, which is a limiting factor for most MIR problems.

Lee et al. (2009) and Hamel and Eck (2010) were some of the first to apply deep neural networks to a genre recognition problem. The results demonstrate that the neural network models are able to classify frames of magnitude spectrograms directly into genre labels and are able to outperform existing machine learning approaches to the problem. Since then, there has been considerable interest in deep learning

from the MIR community. Deep learning for genre recognition is further explored in Sigtia and Dixon (2014), where a few changes to the architecture and optimisation algorithm eliminate the need for *pre-training* (Hamel and Eck, 2010) the parameters of the neural network. The idea is further extended by Dieleman and Schrauwen (2014), where neural networks are directly applied to raw audio data rather than a 2-D time-frequency representation. The results demonstrate that the network is independently able to learn frequency decomposition and also phase and translation invariant features from raw audio.

In Humphrey et al. (2012), a convolutional neural network (ConvNet) is used to learn a Tonnetz-space representation from training examples for automatic chord recognition. The results demonstrate that the learnt representation yields better accuracies than chroma features for classifying frames of audio into chord labels. Humphrey and Bello (2012) present a ConvNet based classifier that directly classifies a CQT time-frequency representation into chord labels. The ConvNet is able to yield accuracies that are competitive with state-of-the-art approaches which involve a considerable amount of domain knowledge in their design. Boulanger-Lewandowski et al. (2013a) uses an RNN to classify input features into chord labels. The RNN architecture is designed such that the predictions at any time-step are conditioned on all past predictions. This encourages temporal correlation and smoothness in the label predictions and eliminates the need for an additional post-processing step with HMMs.

Neural networks have also been applied to the problem of content-based music recommendation. Van den Oord et al. (2013) use ConvNets to predict latent factors directly from the spectrogram. Liang et al. (2015) present a method where a neural network is trained to predict semantic labels from acoustic features. The predictions are then used as a prior in a collaborative filtering model. Deep learning has been used to analyse drum patterns (Battenberg and Wessel, 2012), for learning rhythm

and melody features (Schmidt and Kim, 2013), for symbolic music modelling (Cherla et al., 2013), for singing voice detection (Lehner et al., 2015; Schlüter and Grill, 2015) and for music generation (Boulanger-Lewandowski et al., 2012). Deep learning methods have recently been applied to the problem of boundary detection (between verse and chorus for instance) in music structure analysis (Ullrich et al., 2014; Grill and Schlüter, 2015a,b). They have also been applied to onset detection (Schlüter and Böck, 2014), tempo estimation (Böck et al., 2015) and polyphonic AMT (Böck and Schedl, 2012).

2.2 Environmental Sound Recognition

The audio signal carries information from a variety of sources like human speech, music, animal sounds, street sounds, various indoor and outdoor sounds and so on. Automatic speech recognition (ASR) systems (Rabiner and Juang, 1993) are now deployed in commercial speech recognition systems that are found on computers, phones, TVs and a variety of other devices. Similarly, analysing the content of music signals has been an active area of study in MIR over the last 2 decades (Schedl et al., 2014). In addition to speech and music, audio signals contain a very variety of *non-speech, non-music* sounds which are broadly denoted as *environmental* sounds. The task described in this section belongs to the field of computation auditory scene analysis (CASA) (Wang and Brown, 2006). The problem of automatically analysing the content of environmental sounds is relatively new compared to ASR and MIR, though it has been receiving more attention in the last decade.

The ability to automatically recognise and identify environmental sounds has many applications in the field of audio indexing and retrieval. Such systems can be used to index large audio archives, be used in home security applications (e.g. detection of alarms, baby cries) and making smart sensors, to name a few applica-

tions. The problem of recognising environmental sounds has been broadly divided into 2 categories, Acoustic Scene Classification (ASC) and Acoustic Event Detection (AED). Given an audio recording, ASC aims to assign one or more semantic labels to the recording as a whole. Therefore, the entire recording is treated as an object to which a set of labels is assigned. A related problem is AED, where the system outputs semantic labels in addition to the onset and offset times for each label, similar to AMT and ACT. The type of labels that are associated with each track vary depending on the problem. For example AED systems can be used to detect different types of alarm sounds, to detect glass breaks and gunshots, to detect bird species or for various other use cases. The events can either be non-overlapping (monophonic) or overlapping (polyphonic). Due to the presence of sounds emitted by several active sources, polyphonic detection of sounds is challenging, similarly to AMT and ACT. In this thesis, we consider the problem of AED for environmental sounds.

From the literature, we observe that approaches to AED follow a pipeline similar to ASR, AMT and ACT. The audio excerpt is divided into overlapping frames from which acoustic features are extracted. A statistical or machine learning model is then used to assign class membership scores (or probabilities) to the input features. The model can be either supervised (Aucouturier et al., 2007) or unsupervised (Cauchi, 2011) depending on the type of problem, amount of data available, whether the data is labelled and other features of the given problem. AED systems usually perform a frame-level classification which assigns likelihoods of various labels to each frame. Subsequent post-processing, typically with an HMM, is used to segment the data and provide an alignment between feature frames and labels (Mesaros et al., 2010).

In the remainder of this section we review existing literature on AED for environmental sounds.

2.2.1 Acoustic Event Detection

AED systems aim to *segment* the recording into sections based on the occurrence of target labels. Typically, AED systems process short windows of audio and predict labels over chunks, rather than the entire recording. Kennedy and Ellis (2004) use SVMs with MFCC features over short windows to detect laughter in recordings. A similar approach is used to discriminate between laughter and applause by Arias et al. (2005). Stager et al. (2005) propose a low-resource method for detecting sound environments (home, office, workshop, outdoors) using k-nearest neighbours (k-NN) (Altman, 1992) applied to time-frequency features. Andersson (2004) also uses a k-NN classifier on spectral features to classify audio segments into music or speech. Ellis (2001) uses a neural network classifier to detect different types of alarm sounds. Goh et al. (2003) use k-means clustering to segment audio recordings of TV programs into segments containing commercials. Hoiem et al. (2005) present a system that uses boosted decision trees to detect events (like car horns, dogs barking) in recordings obtained from movies.

More recently, studies evaluate AED systems on much larger datasets of real-world recordings and with more sophisticated machine learning techniques. Mesaros et al. (2010) use HMMs to detect the presence of 61 classes of sounds in audio recordings. Cotton and Ellis (2011) use a convolutive NMF model to learn a time-frequency basis for 16 classes of sounds in a meeting-room setting. Mesaros et al. (2011) use MFCCs and HMMs as acoustic models to detect up to 10 events. Probabilistic latent semantic analysis is then used to model the co-occurrence of different sound events. Using this method to determine the prior probabilities of events yields improved performance over uniform priors. Heittola et al. (2013) propose a 2-step event detection process. The audio context is first modelled using GMMs. At test time, the recording is analysed to identify the audio context. This information is then used to determine a set of possible event labels associated with the context. The individual events are

finally detected using context-dependent HMMs. Amid et al. (2014) use stacked auto-encoders to learn a representation starting from standard time frequency features like MFCCs, spectral flux and spectral centroid. The learnt features are then classified using kernel based classifiers. Cakir et al. (2015) use a neural network to directly classify time-frequency features into one of 61 event classes. The problem is posed as a multi-label prediction problem and the neural network outputs are post-processed to detect the presence of multiple overlapping sound events. The results show that the neural network model outperforms a state-of-the-art NMF acoustic model on the given detection task. Mesaros et al. (2015) present an NMF method for AED. The proposed method detects overlapping events in recordings without constructing a model for each class. The method can be trained directly on audio recordings in an unsupervised setting. The method also presents methods for controlling the size of the learnt dictionary of overlapping sound events. Stowell and Clayton (2015) present a method which is an interesting deviation from typical AED systems. The proposed method decomposes the probability of a detected event into a product of the probability of onset, probability of offset and a prior on the event duration. The proposed method is compared to a standard HMM event detection system for the task of detecting 300 different types of birds. The results demonstrate that the proposed method outperforms the HMM baseline and is more accurate for estimating bird-call rates.

In addition to the literature presented above, there are 2 more studies worth mentioning. The CLEAR evaluations (Temko et al., 2006) which were part of the CHIL project (Waibel et al., 2009), compared 3 different systems for an AED and ASC task. The evaluations were performed on a database of isolated sounds and another database of recorded interactive seminars. The study presented one of the first *benchmarks* for AED/ASC systems. More recently, the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge (Giannoulis et al., 2013; Stowell

et al., 2015) performs public comparisons of state-of-the-art methods for AED and ASC. The evaluations are performed on 2 different datasets for AED and ASC tasks. The evaluations included entries from 7 different participants. The data used for the evaluations is publicly available and the challenge forms an important benchmark for future research in the area.

Chapter 3

Neural Networks

This chapter provides an overview of neural networks. A brief background and history of neural networks is followed by definitions and descriptions of the models used in the remainder of this thesis. We describe neural network architectures for acoustic modelling and recurrent architectures for modelling sequences. We discuss the optimisation framework for the network parameters and several practical heuristics that are used for efficient parameter estimation. Finally, we discuss how neural network based models can be used for density estimation and present 2 types of neural density estimators.

3.1 Background

Neural networks are a family of models that express a mathematical function as a composition of simple, non-linear functions. The overall architecture of a neural network can be regarded as a hierarchy of layers, where each *layer* of the neural network performs a simple transformation of the outputs of the previous layer. This architecture results in a network that learns a hierarchy of simple transformations of the input data, which cumulatively result in complex non-linear transformations of the inputs. The modular, layer-wise architecture is a characteristic feature of neural

networks.

The artificial neuron was first proposed by McCulloch and Pitts (1943) and was inspired by discoveries in neuroscience. The artificial neuron is a parameterised processing unit which can be used to simulate logic operations. This discovery was followed by the Perceptron algorithm (Rosenblatt, 1958), which provided a method for estimating the parameters of the artificial neuron given a set of training examples. Minsky and Papert's seminal work demonstrated the limitations of single-layer perceptrons (Minsky and Papert, 1969) which led to the proposal of multi-layer perceptrons (MLP). MLPs define an expressive family of parametric functions that were shown to be capable of approximating arbitrarily complex functions (with some error tolerance) (Hornik, 1991). Despite their attractive properties, effectively training MLPs remained an open problem. A major breakthrough in training MLPs came with the proposal of the backpropagation algorithm (Hinton, 1986; Rumelhart et al., 1988) that used the chain rule for differentiation to estimate error signals for updating the network parameters. When used to calculate error derivatives over many examples, the algorithm proved to be very slow and impractical. This posed a challenging engineering problem, especially as the number of training examples increased. In addition to the slow training time, there were also instances where the training algorithm did not converge to useful solutions at all. An efficient stochastic version of the backpropagation algorithm was proposed (LeCun et al., 1998), called stochastic gradient descent (SGD). Rather than calculating gradients over the entire training *batch*, SGD estimates the gradient over the batch by calculating gradients over a small set of samples or a *minibatch*, randomly drawn from the training set. Minibatch SGD has been shown to be very effective, especially on very large datasets. The noisy gradient estimates have also been shown to be useful for finding better parameters for the networks (LeCun et al., 1998).

Despite the many fundamental developments with respect to the neural network

architectures, training algorithms and theoretical understanding of their functioning, neural networks started falling out of favour with the machine learning/AI community in the 90s. However, in the past decade, there has been a great resurgence of interest in neural networks. The revival in interest was sparked by research that demonstrated that neural networks with many hidden layers could be trained effectively if the weights were properly initialised using unsupervised learning methods (Hinton et al., 2006). These ideas were then applied to speech recognition, which lead to dramatic improvements on both small (Mohamed et al., 2009) and large vocabulary systems (Dahl et al., 2012). Since then similar results have been obtained in object recognition (Krizhevsky et al., 2012), natural language understanding (Collobert et al., 2011b) and other areas of natural language processing (see Goldberg (2015) for a summary). The reader is referred to Schmidhuber (2015) for an extensive overview of the history and applications of neural networks.

The recent successes of neural networks can be mainly attributed to two factors. The proliferation of digital information has allowed researchers to experiment with very large datasets (Deng et al., 2009; Dahl et al., 2012; Mikolov et al., 2013). Neural networks are prone to the problem of *overfitting* (Bishop, 2006, Chapter 1), where the network performs very well on the train set but cannot generalise to new examples. The availability of large datasets allowed researchers to train networks with many parameters that were able to generalise well to new data. However, training neural networks with many parameters on large datasets can take prohibitively long. The second major contributor to the recent success of neural networks has been efficient use of modern computing infrastructure. It was observed that network training times could be drastically reduced if certain computations were performed on GPUs rather than CPUs. GPU training allowed researchers to effectively train models with many parameters, leading to advances in many domains. In addition to single GPU training, neural network training has been implemented on large CPU and GPU clusters to

train models efficiently on very large datasets (Dean et al., 2012). The hardware advances have been complemented by software libraries like Theano (Bergstra et al., 2010), Torch (Collobert et al., 2011a) and TensorFlow (Abadi et al., 2015), which allow users to easily design and train neural networks.

3.2 Neural Network Architectures

In this section we describe the neural network architectures used in this thesis. Let $X = \mathcal{R}^D$ represent the space of inputs, with $x \in \mathcal{R}^D$. Let $Y = \mathcal{R}^k$ represent the space of outputs, with $y \in \mathcal{R}^k$ and let f be some function, $f : X \to Y$. A neural network defines a function as follows:

$$f(x; \Theta) = f_{L-1} \circ f_{L-2} \circ \ldots \circ f_0(x). \tag{3.1}$$

Equation 3.1 describes a neural network with L layers. Each component function f_i represents a layer with parameters θ_i. The set of all network parameters is $\Theta = \{\theta_0, \theta_1 \ldots \theta_{L-1}\}$.

A neural network architecture is specified by the number of layers L and the type of each individual function f_i. Each layer can be chosen from one of the many possible architectures (fully connected, recurrent, convolutional, pooling) in order to solve a given problem (Szegedy et al., 2015; Sainath et al., 2015). The number of layers in a network is also known as the *depth* of the network. Typically, networks with 2 or more intermediate layers are called deep networks and the field of study related to deep networks is loosely known as *deep learning* (Bengio, 2009; LeCun et al., 2015; Schmidhuber, 2015). The layer-wise modular architectures learn a hierarchy of representations of the data. Therefore given sufficient data, neural networks with many layers can be directly applied to raw data to estimate complex functions *end-to-end*.

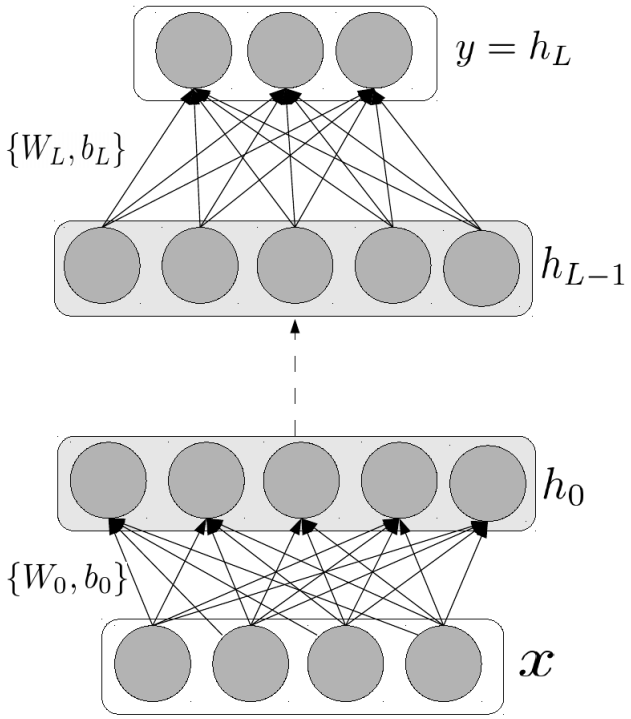


Figure 3.1: Graphical structure of a feedforward DNN. Each layer has parameters $\theta_l = \{W_l, b_l\}$. The above network contains L intermediate hidden layers and a final output layer. The dashed line represents one or more intermediate layers.

We now discuss some of the architectures in more detail.

3.2.1 Feedforward Neural Networks

The simplest type of layer is the feedforward layer. The feedforward layer performs the following transformation:

$$f_l(h_{l-1}; \theta_l) = g(W_l h_{l-1} + b_l), \tag{3.2}$$

where h_{l-1} is the output from the previous layer, W_l is a weight matrix of size $M \times N$ that transforms an M dimensional input h_{l-1} to an N dimensional vector, b_l is an N-dimensional bias vector and $\theta_l = \{W_l, b_l\}$ are the layer parameters. A feedforward layer performs an affine transformation of the input, followed by the application of an element-wise non-linearity g. A network comprising only feedforward layers is called a

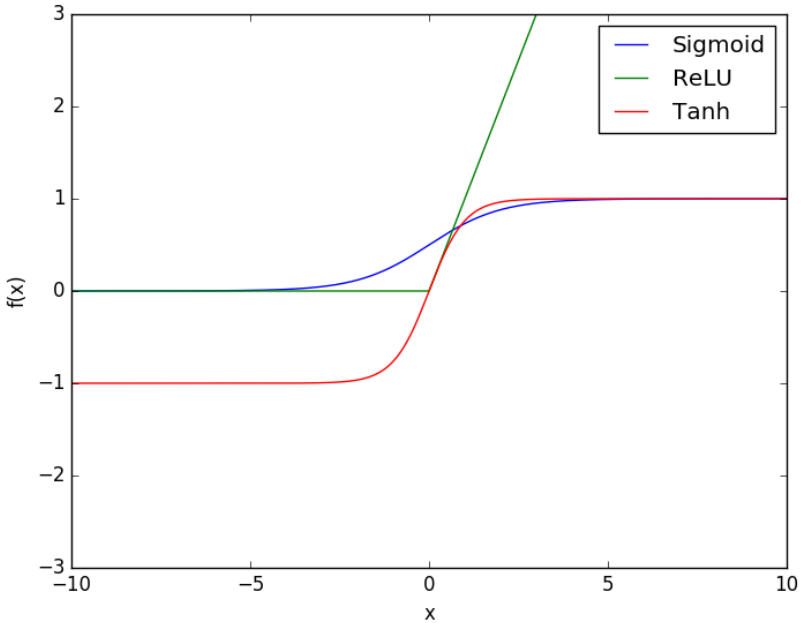


Figure 3.2: Neural network activation functions.

feedforward neural network or a deep neural network (DNN). Figure 3.1 is a graphical representation of a DNN. The non-linearity g is necessary since a composition of linear transforms is also a linear transform. The non-linearities are essential for learning complex non-linear mappings between the inputs and the outputs. Popular non-linearities found in the literature are:

- Sigmoid: $g(x) = \frac{1}{1+e^{-x}}$.
- Hyperbolic Tangent (tanh): $g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.
- Rectified Linear Unit (ReLU) (Glorot et al., 2011): $g(x) = \max(0, x)$.

From Figure 3.2 we observe that the sigmoid and tanh non-linearities saturate as $|x|$ increases. More recently, the ReLU non-linearity has become increasingly popular. Networks with saturating non-linearities need to be carefully initialised, while ReLU networks are easier to optimise since they allow gradients to flow for arbitrarily deep networks (Glorot et al., 2011). When the inputs to saturating non-linearities are very

large (or very small), the gradient with respect to the inputs becomes very small and the error signal obtained to modify the network parameters is negligible. However, for networks with ReLU activations, the gradient with respect to the inputs is always 1 provided the inputs are greater than 0, regardless of their magnitude (Figure 3.2). This ensures that there is always a path through the network along which the gradient can be propagated back, except for the unlikely case where all the activations are less than 0.

3.2.2 Convolutional Networks

A convolutional layer is obtained by replacing the dot product in Equation 3.2 by a convolution operation:

$$f_l(h_{l-1}, \theta_l) = g(h_{l-1} * W_l + b_l), \tag{3.3}$$

with the convolution operation defined as:

$$x * W = \sum_m \sum_n x[m, n] W[i - m, j - n]. \tag{3.4}$$

In practice, the input h_{l-1} is 2-dimensional and additionally comprises several channels or bands (RGB bands in an image for instance). The inputs are convolved with a set of weights or *kernels* or *filters*. The convolution layer produces a 2D *feature map* as output for each kernel. The resulting parameters are stored as a 4-dimensional tensor, where the first dimension corresponds to the number of kernels, the second dimension represents the number of input channels and the final dimensions represent the *shape* of the kernels. The bias b_l is a vector of dimensionality equal to the number of kernels and $\theta_l = \{W_l, b_l\}$ are the layer parameters.

Typically, the convolutional layers are followed by a sub-sampling or *pooling* layer. The sub-sampling is performed by computing some statistics (for e.g. max, min or

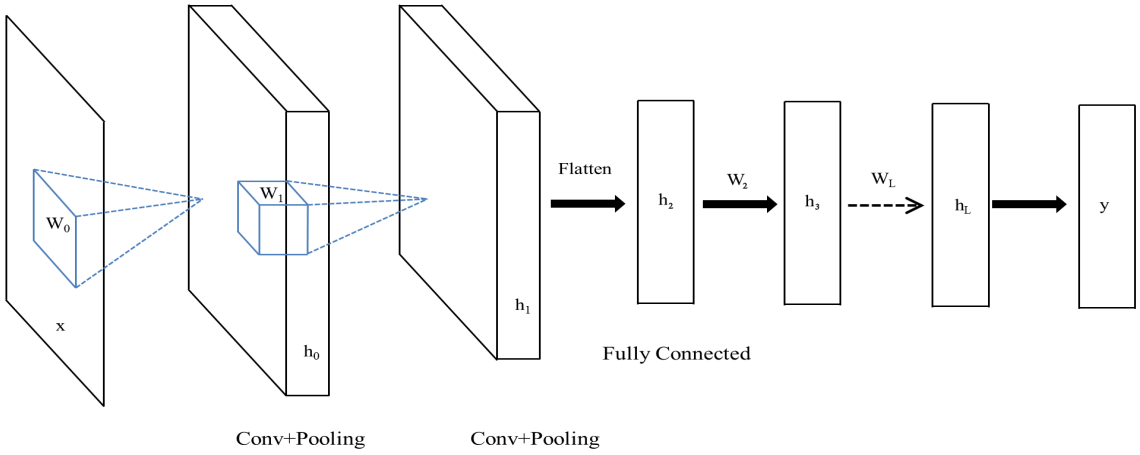


Figure 3.3: Graphical structure of a ConvNet with 2 alternating convolutional and pooling layers, followed by a series of fully connected layers. The convolutional filters produce 2-D feature maps. The feature maps are stacked together to form h_0, which is represented by the additional *depth* dimension. The filters in the next layer jointly act on receptive fields from all feature maps, which is represented by the extra depth dimension of filter w_1.

mean) over small regions of the feature map. Max pooling is the most common type of pooling used in practice (LeCun et al., 2015). The pooling layers help learn intermediate representations that are invariant to small changes in the *scale* of inputs. A typical convolutional neural net (ConvNet) consists of alternating layers of convolutional and max-pooling layers, followed by one or more fully connected layers at the top of the network. Figure 3.3 is a graphical representation of a ConvNet. More elaborate architectures can be devised by optimising the number, type and ordering of layers for a given task (Lin et al., 2013; Szegedy et al., 2015). The structure of ConvNets was inspired by studies on the feline visual cortex (Hubel and Wiesel, 1959). ConvNets were first applied to handwriting recognition (LeCun et al., 1998) and have since been used to advance the state-of-the-art in many computer vision tasks (LeCun et al., 2015). ConvNets are characterised by 3 main properties. The use of 2-dimensional filters or *local receptive fields* exploits the spatial correlations between neighbouring pixels (or neighbouring frequency bands or temporal frames for audio) and produces 2-D feature map representations that preserve the spatial

structure of the inputs. Secondly, convolving receptive fields over the input or *weight sharing* results in feature detectors or filters that are invariant to translation of objects within an image. Weight sharing also reduces the number of parameters in the network, since the same set of filters is repeated over the entire image to produce a feature map. This effectively reduces the number of model parameters and therefore the model complexity. Finally, max-pooling which is the most common pooling operation found in the literature, results in representations that are invariant to small changes in scale of the inputs and to small rotations. The combination of these 3 properties have resulted in ConvNet architectures being used extensively for vision and audio processing tasks.

3.2.3 Recurrent Networks

DNNs and ConvNets process inputs of a fixed size and are fundamentally ill-suited for processing sequences. However naturally occurring information like audio, video and natural language are all sequential. The recurrent neural network (RNN) architecture is designed specifically for processing sequences. Given a sequence of inputs $x = \{x_0, x_1 \dots x_T\}$ and outputs $y = \{y_0, y_1 \dots y_T\}$, an RNN performs the following computation:

$$h_t(h_{t-1}, x_t) = g(W^r h_{t-1} + W^f x_t + b) \tag{3.5}$$

where x_t is the input at step t, h_{t-1} is the previous state, W^r, W^f are recurrent and forward weight matrices, b is the bias and g is an element-wise non-linearity. Figure 3.4 is a graphical representation of the RNN. The sequential nature of the processing is due to the *recurrent* connections in the hidden layer. Given a sequence of inputs, the recurrent hidden layer allows the RNN to maintain a *state*. Unlike feed-forward neural networks that learn functions, an RNN can learn a sequence of computations

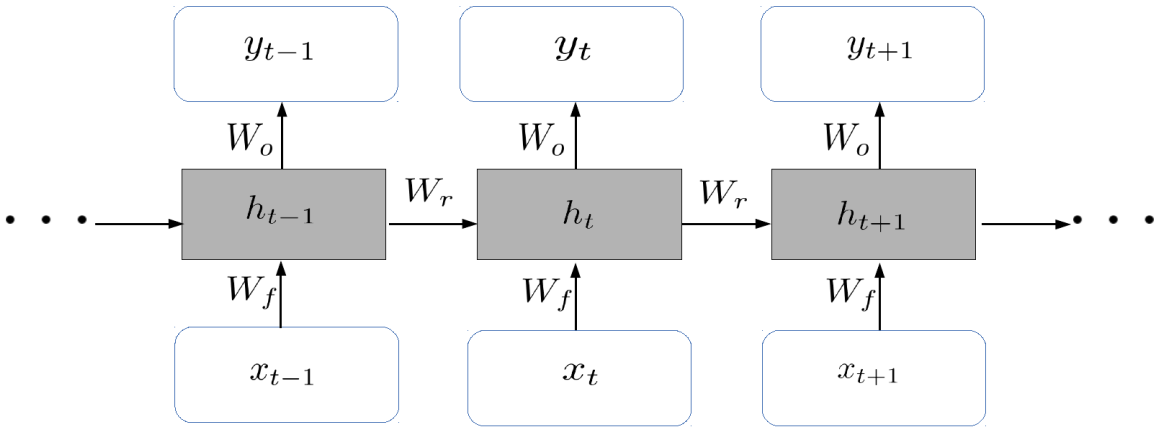


Figure 3.4: Graphical structure of an RNN for inputs $x = \{x_0, \ldots, x_T\}$ and outputs $y = \{y_0, \ldots, y_T\}$ and one hidden layer. Similar to DNNs, recurrent hidden layers can be stacked to produce *deep* RNNs. W_f, W_r, W_o represent the input, recurrent and output weight matrices, respectively.

like a program. In fact, it has been shown that RNNs can theoretically represent any non-linear dynamical system (Siegelmann, 1995). The output sequence y is generated as a function of the sequence of hidden states $h = \{h_0, h_1 \ldots h_T\}$. For the simplest case, an output y_t is generated for each input x_t as follows:

$$y_t = f(W_o h_t + b_o), \tag{3.6}$$

where f is an appropriate activation function, W_o is the output weight matrix and b_o is the bias. In addition to the simple fully connected output layer in Equation 3.6, more complicated output functions can be designed. For instance, the connectionist temporal classification (CTC) architecture allows the number of outputs to be less than or equal to the number of inputs (Graves et al., 2006) and the encoder-decoder architecture can have an arbitrary number of outputs in the output sequence (Sutskever et al., 2014). Similarly to fully connected and convolutional layers, recurrent layers can be *stacked* to produce deep RNNs (Graves, 2012a).

Recently, RNNs have been successfully applied to a wide variety of sequential modelling problems such as speech recognition (Graves et al., 2006, 2013), music

transcription (Böck and Schedl, 2012), language modelling (Mikolov et al., 2011), machine translation (Sutskever et al., 2014) and image captioning (Karpathy and Fei-Fei, 2015).

3.3 Optimisation

So far, we have seen how neural networks can be used to define a large class of parameterised functions. Given many examples of inputs and corresponding outputs, the next step is to *learn* the parameters of the neural network in order to estimate an effective function for a given task. In this section, we describe how the parameters of a neural network are estimated.

3.3.1 Objective Function

Let us a consider a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, of i.i.d. examples of input/output pairs. $\mathcal{D}$ is known as the training set. Let $\mathcal{L}$ be a loss function that computes a scalar penalty or loss when the network outputs $f(x, \Theta)$ deviate from the true outputs y. Ideally, we would like to minimise the loss function $\mathcal{L}$ over the set of *test* or unseen examples. However, in practice, the set of unseen examples or test set is not available while constructing the model. Therefore machine learning models are *trained* by minimising the loss over the set of examples in the training set (Mohri et al., 2012). The overall cost function over the training set is:

$$\mathcal{C}(\mathcal{D}|\Theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x_i, \Theta), y_i), \tag{3.7}$$

where $f(x_i, \Theta)$ represents the network outputs and y_i is the ground truth for input x_i. According to convention, we express all the optimisation problems as minimisation of an objective function with respect to some constraints. Maximisation problems are equivalent to minimising the negative objective function. Therefore, the model pa-

rameters are estimated by minimising the cost with respect to the model parameters.

$$\Theta^* = \arg\min_{\Theta} C(D|\Theta) \tag{3.8}$$

The loss function $\mathcal{L}$ can be of several forms depending on the assumptions made about the distribution of the output variables. The most common function is the squared error loss:

$$\mathcal{L}(f(x,\Theta),y) = \frac{1}{2}(f(x,\Theta)-y)^2. \tag{3.9}$$

The squared error loss function is a natural choice when modelling real-valued output variables that are assumed to be Gaussian distributed. When using the squared error loss, the network outputs $f(x,\Theta)$ represent the expectation of the conditional output distribution $\mathbb{E}_y[y|x]$ (Bishop, 2006) or the mean of the output Gaussian distribution. When neural networks are used for classification, the output variables are typically non-Gaussian. For binary classification problems, the outputs form a Bernouilli distribution where $P(y=1) = f(x,\Theta)$ and $P(y=0) = 1 - f(x,\Theta)$. The conditional distribution over the outputs is given by:

$$P(y|x,\Theta) = f(x,\Theta)^y\{1-f(x,\Theta)\}^{1-y}. \tag{3.10}$$

The model is then trained by maximising the likelihood over all training examples $x_i, y_i \in \mathcal{D}$. This is also known as maximum likelihood estimation (MLE) since the objective of training is to find a set of parameters that maximise the likelihood of the training examples. In practice, the logarithm of the loss is minimised since products of probabilities can cause floating point values to underflow very easily. By taking the log, we obtain the *cross-entropy* loss function:

$$\mathcal{L}(f(x,\Theta),y) = -y\log f(x,\Theta) - (1-y)\log(1-f(x,\Theta)) \tag{3.11}$$

Another case of interest is when the output distribution is multinomial. An output $y_k \in \{0, 1\}$ is 1 only if k corresponds to the target class, otherwise it is 0, which leads to the following cost function:

$$\mathcal{L}(f(x, \Theta), y) = -\sum_{k=1}^{K} y_k \log f(x, \Theta)_k. \tag{3.12}$$

3.3.2 Output Activation Function

From the previous section we observe that the choice of loss function depends on the type of output distribution being modelled. Similarly, the output activations of the neural networks need to be selected in order to define the desired output distribution. The output activation functions are also known as *inverse link functions* in statistics (Bishop, 2006). When modelling real-valued unbounded outputs, the output activation is the identity transform. When modelling K independent Bernouilli distributions, the output activation function is the sigmoid function (Section 3.2.1), which ensures every output $y_k \in [0, 1]$. A multinomial output distribution over K outputs, is attained using the *softmax* activation function (Bridle, 1990):

$$\mathcal{P}(y_k = 1|x) = \frac{e^{a_k}}{\sum_i e^{a_i}}, \tag{3.13}$$

where a_k is the activation of k^{th} output unit.

3.3.3 Numerical Optimisation

We are interested in the set of network parameters that minimise the chosen objective function over the training set (Equation 3.8). The cost is a function of the network parameter values and minimisation is performed in this high dimensional space. Deep neural networks can have millions or even billions of parameters (Szegedy et al., 2015). Searching by brute force through such a large space is intractable. Similarly, analyt-

ically solving $\nabla\mathcal{C}(D|\Theta) = 0$ is also intractable (Bishop, 2006, Chapter 5). Therefore, neural network parameters are estimated using numerical optimisation techniques. At any point Θ, $\nabla\mathcal{C}(D|\Theta)$ is the direction of increase for $\mathcal{C}$. Therefore moving in the direction opposite to the gradient provides the direction of *steepest descent*. Formally, given a cost function $\mathcal{C}$, a set of parameters Θ_n, the gradient $\nabla\mathcal{C}(D|\Theta)$ at Θ_n, the parameters are updated as:

$$\Theta_{n+1} \leftarrow \Theta_n - \epsilon\nabla\mathcal{C}(D|\Theta), \tag{3.14}$$

where ϵ is called the *learning rate* and is usually a small value that controls the distance moved along the negative gradient at Θ. This algorithm for numerical optimisation is called *gradient descent* (Bishop, 2006, Chapter 5). The algorithm begins with a random configuration of parameters Θ_0 and iteratively updates the parameters by calculating the gradients at each step.

In order to estimate the parameters of neural networks with gradient descent, it must be possible to efficiently compute the gradient of the cost function with respect to the model parameters. The *backpropagation* algorithm is an efficient algorithm for evaluating the derivatives of the cost function (Hinton, 1986; Rumelhart et al., 1988). The backpropagation algorithm uses the chain rule for partial derivatives to analytically derive expressions for the gradient. The gradients are first computed with respect to the parameters of the output layer. The gradients with respect to all other parameters are calculated by sequentially applying the chain rule. The only constraint for the backpropagation algorithm is that the cost function should be *differentiable* with respect to the model parameters at the points of interest. Therefore, as long as the intermediate computations are differentiable, we can define a very large family of neural network models by carefully designing the network architecture, activation functions and the loss function for a given problem. The generality of backpropagation allows for construction and training of complex neural network architectures, for

example defining cost functions like the connectionist temporal classification objective (Graves et al., 2006), adding external memory units to RNNs (Grefenstette et al., 2015) and adding an attention mechanism to RNNs (Bahdanau et al., 2014).

3.3.3.1 Stochastic Gradient Descent

The numerical optimisation algorithm described above can be used to optimise neural networks given a differentiable cost function. However the computation of the cost involves calculating the cost for *all* examples in the training set, also known as *batch training*. As mentioned before, one of the reasons for the success of neural networks has been the use of very large datasets. When using very large datasets, computing gradients for the entire dataset at every iteration makes training prohibitively slow.

In practice, neural networks are trained with the *stochastic gradient descent* (SGD) algorithm (Bottou, 2010; LeCun et al., 2012). Rather than averaging the gradients over the entire set of training examples, in SGD, gradients are computed for either a single randomly chosen sample or averaged over a small batch or *minibatch* of randomly chosen samples from the dataset. Therefore, SGD replaces the true gradient or batch gradient by an *estimate* of the true gradient, based on randomly sampling single samples or minibatches of samples from the training dataset. Small batches of data can be processed relatively quickly on GPUs and therefore SGD updates can drastically help reduce training time. SGD also efficiently tackles the redundancy in data, while the noisy gradient estimates have empirically been shown to be useful for finding good minima (LeCun et al., 2012). SGD benefits greatly from careful initialisation of the model parameters and other optimisation tricks like momentum (Sutskever et al., 2013). In Equation 3.14, the learning rate ϵ is a *hyperparameter* of the optimisation algorithm that needs to be tuned. Various studies propose different schedules for updating the learning rate. Recently, several adaptive learning rate methods like ADAGRAD (Duchi et al., 2011), ADADELTA (Zeiler, 2012) and ADAM

(Kingma and Ba, 2014) have been proposed that dynamically adapt the learning rate for each parameter based on first order gradient information.

3.3.3.2 Hessian Free Optimisation

Hessian Free (HF) optimisation is a *second* order optimisation algorithm which has successfully been used to train DNNs and RNNs (Martens, 2010; Martens and Sutskever, 2011). Given a set of parameters Θ, a local quadratic approximation to the cost is:

$$\mathcal{C}(\Theta + \delta) \approx \mathcal{C}(\Theta) + \nabla\mathcal{C}(\Theta)^T\delta + \frac{1}{2}\delta^T H\delta, \tag{3.15}$$

$$H_{i,j} = \frac{\partial^2 C}{\partial\Theta_i\partial\Theta_j}, \tag{3.16}$$

where H is called the Hessian matrix. Setting the partial derivative of Equation 3.15 with respect to δ equal to 0 gives:

$$\delta = H^{-1}\nabla\mathcal{C}(\Theta). \tag{3.17}$$

Equation 3.17 is also known as the *Newton-Raphson method* (Fletcher, 2013) and involves computing H^{-1}, which is prohibitively computationally expensive for large neural networks. This is due to the fact that in the worst case, inverting a square matrix of size N involves computations of $O(N^3)$ and for neural networks N can be of the order of millions. Unlike second order Newton's method, HF proceeds by *partially optimising* Equation 3.15 with the method of Conjugate Gradients (CG) (Fletcher and Reeves, 1964). The method of CG is used to iteratively minimise functions of the form:

$$f(x) = \frac{1}{2}x^T Ax - x^T b, \tag{3.18}$$

where $x \in \mathcal{R}^n$ and A is positive definite. Therefore at any step the HF optimisation algorithm proceeds as follows: given some set of parameters Θ_n, first calculate $\nabla\mathcal{C}(\Theta_n)$.

Then calculate a damped approximation to the Hessian matrix $H' = H(\Theta_n) + \lambda$. The damping term is introduced to ensure that the resulting matrix is positive definite. Then use CG to minimise Equation 3.18, with $x = \delta$, $A = H'$ and $b = -\nabla\mathcal{C}(\Theta_n)$. Finally update the parameter estimates $\Theta_{n+1} \leftarrow \Theta_n + \delta$.

HF has been successfully used to train RNNs to learn dependencies over long intervals. The CG optimisation step in HF works well when using large batches of data. This drastically reduces the number of training iterations used for HF, though each iteration is computationally much more expensive.

3.3.3.3 Optimising RNNs

RNNs are characterised by the recursive update of the hidden state (Equation 3.5). Consequently, the hidden state h_t is a function of all previous hidden states $\{h_0, \ldots, h_{t-1}\}$. RNNs are trained using a modified backpropagation algorithm called *back propagation through time* (BPTT) (Werbos, 1990) which accounts for the error flow due to the recursive connections. Although RNNs can theoretically learn dependencies over very long intervals, it was discovered that training them to do so in practice is quite difficult (Bengio et al., 1994).

Let us consider a sequence of inputs $x = \{x_0, \ldots, x_T\}$ and outputs $y = \{y_0, \ldots, y_T\}$. The cost for the whole sequence is $\mathcal{C} = \sum_{t=0}^{T} \mathcal{L}(f(x_t, \Theta), y_t)$, where $\mathcal{L}$ is some scalar loss. Let us consider two time-steps in the sequence, t_a, t_b with $t_a < t_b$. The gradient of the cost at t_b is as follows:

$$\frac{\partial \mathcal{C}_{t_b}}{\partial \theta} = \sum_{t \leq t_b} \frac{\partial \mathcal{C}_{t_b}}{\partial h_{t_b}} \frac{\partial h_{t_b}}{\partial h_t} \frac{\partial h_t}{\partial \theta}, \tag{3.19}$$

$$\frac{\partial h_{t_b}}{\partial h_{t_a}} = \prod_{\tau=t_a+1}^{t_b} \frac{\partial h_\tau}{\partial h_{\tau-1}}. \tag{3.20}$$

From equation 3.19 we observe that the gradient of the cost at any time t_b also includes error terms summed over all sub-sequences, which includes products of Ja-

cobian matrices $\frac{\partial h_{t_b}}{\partial h_t}$ (Equation 3.20). As the distance between t_b and t increases, the product of many Jacobian matrices either explodes if the leading eigenvalues are > 1 or vanishes if the leading eigenvalues are < 1. This implies that gradients either accumulate or vanish over long intervals and therefore learning long-term dependencies with RNNs is difficult.

3.3.4 Long Short Term Memory

The most popular approach to dealing with exploding/vanishing gradients is the use of long short-term memory (LSTM) units in an RNN (Hochreiter and Schmidhuber, 1997). In a standard RNN, the previous hidden state h_{t-1} is *multiplied* by a weight matrix before being added to the projected input. Repeated multiplication by the recurrent weight matrix over several time steps causes information from the past to be lost, making it hard to learn long-term dependencies (Section 3.3.3.3). The LSTM architecture circumvents this issue by making the update to the hidden units or cells, additive rather than multiplicative. In addition to the additive update, LSTM units employ a gating mechanism for all the inputs and output units. It is worth mentioning that these gating units are not restricted to LSTMs and have also been found useful in training very deep feedforward networks (Srivastava et al., 2015; Oord et al., 2016)

Figure 3.5 is a graphical representation of the LSTM architecture. Let x_t be the input vector at time index t, let c_t be the value of the internal cell state of the LSTM and let h_t be the outputs from the LSTM at time-step t. The input-to-hidden transformation is obtained as:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i), \tag{3.21}$$

$$\tilde{c_t} = \tanh(W_c.[h_{t-1}, x_t] + b_c), \tag{3.22}$$

where $\tilde{c_t}$ represents the same transformation as Equation 3.5. W_i, b_i are the input

gate weights and biases and W_c, b_c are the weights and biases for the inputs to the cell. The only difference between the computation of i_t and $\tilde{c}_t$ is that the outputs of the input gate are passed through a sigmoid function (as opposed to the tanh function) and have values between 0 and 1, therefore acting as *soft* gates. Next, the value of the LSTM cells is updated as follows:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f), \tag{3.23}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{3.24}$$

where f_t represent the forget gates, W_f, b_f represent the weight matrix and bias for the forget gate and $\odot$ is an element-wise multiplication. Therefore the value of the previous cell state is first passed through a forget gate before being *added* to the gated inputs to the cell. Finally the cell outputs the following:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o), \tag{3.25}$$

$$h_t = o_t \odot \tanh(c_t), \tag{3.26}$$

where o_t represents the output gate and W_o, b_o are the weights and bias for the output gate. The output of the LSTM is the value of the cell units passed through an output gate. Therefore the additive update to the cell states coupled with the use of a gating mechanism allows the LSTM to retain information over long time-periods.

Recently, RNNs have enjoyed success in many domains (Section 3.2.3) and optimising RNNs has been a problem of interest. There have been several insights that allow more effective training of RNNs. The use of gradient clipping is a simple heuristic for preventing exploding gradients (Bengio et al., 2013). Additionally, there are studies that investigate how the norm of the recurrent matrix can be constrained in order to avoid exploding/vanishing gradients (Saxe et al., 2013; Le et al., 2015). There

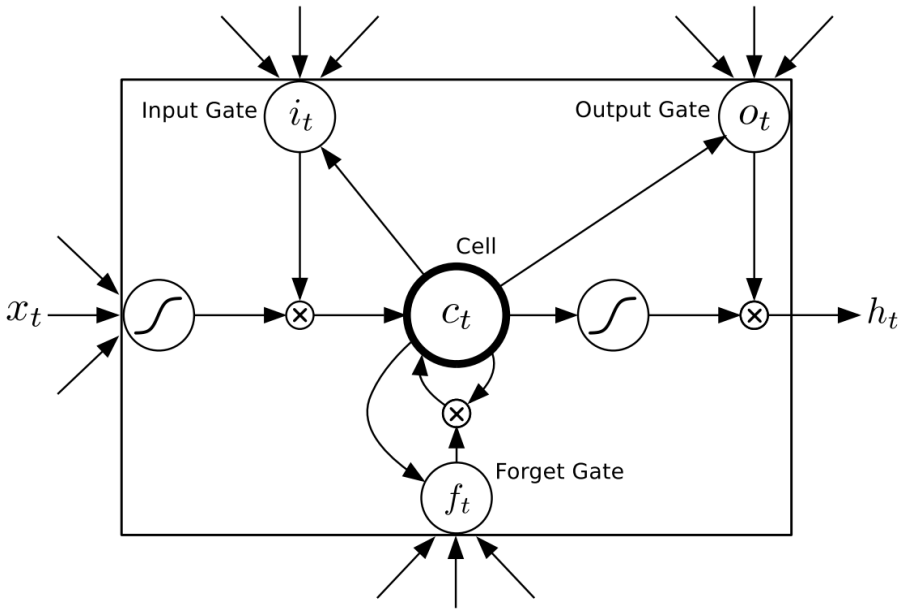


Figure 3.5: A graphical representation of the LSTM. From the figure we note that all 3 gates (input, forget, cell) receive x_t, h_{t-1} as inputs. The cell values at $t-1$ are multiplied by a forget gate and the result is *added* to the gated inputs as opposed to the standard RNN where the updates are multiplicative. Finally the cell state values are multiplied with the output gate to give the LSTM outputs h_t.

have also been some studies that explore different gating and update mechanisms to improve learning (Jozefowicz et al., 2015).

3.3.5 Regularisation

One of the most common issues with training machine learning algorithms is *overfitting*, which is when the trained model is a good predictor on the training data but performs poorly on unseen test data (Bishop, 2006, Chapter 3). When models have many parameters, which is the case with neural networks with many layers, there are many parameter configurations that might lead to good performance on the training set but *generalise* poorly to new data. Given that the test set is not available during training, there is no obvious way to overcome this difficulty. Typically, certain assumptions about the data and the model parameters are made in order to improve model generalisation. This process is known as regularisation.

Regularisation is a means to constrain the search space of model parameters. The most common form of regularisation used to train neural networks is L_2-regularisation or *weight decay*. L_2-regularisation (or *ridge* regression in statistics) is applied by introducing an additive term to the optimisation objective:

$$\tilde{\mathcal{C}}(\Theta) = \mathcal{C}(\Theta) + \frac{\lambda}{2}\|\Theta\|_2^2, \tag{3.27}$$

where λ is the weight decay constant, $\|x\|_p := \left(\sum_{i=1}^{D} |x_i|^p\right)^{1/p}$ and Θ is the set of model parameters. The L_2-regularisation term in the objective function introduces a trade-off between minimising the cost and minimising the magnitude of the model parameters. The term weight decay arises from the fact that the parameter updates have an extra $-\lambda\|\Theta\|_2$ term, which causes a decay in the parameter values over successive iterations (Section 3.3.3).

L_1-regularisation (or *lasso* regularisation in statistics) is another simple regularisation technique with the following objective function:

$$\tilde{\mathcal{C}}(\Theta) = \mathcal{C}(\Theta) + \alpha\|\Theta\|_1, \tag{3.28}$$

where α is a constant. Unlike L_2-regularisation, the L_1 constraint encourages the parameters to be *sparse* (Bishop, 2006, Chapter 3). Both L_1 and L_2-regularisation prevent overfitting by limiting the model complexity by encouraging many model parameters to be close to 0.

Recently *Dropout* (Srivastava et al., 2014) has become an indispensable part of the neural network optimisation toolkit. Dropout is a simple regularisation technique that has been shown to be effective at controlling overfitting in large neural networks. Given an input $x \in \mathcal{R}^D$ to the neural network, a binary mask $m \in \{0, 1\}^D$ is sampled from a Bernouilli distribution with probability p. The new masked input to the

network is computed as:

$$m_i \sim \text{Bernouilli}(p), \tag{3.29}$$

$$\tilde{x} = m \odot x. \tag{3.30}$$

The above operation is usually performed on all the layers of the network excluding the outputs. Dropout has proved to be a very simple and effective way to prevent overfitting in neural networks. The effectiveness of Dropout can be explained qualitatively as follows. By randomly dropping out certain activations, the network weights are forced to learn good features independently, without depending on other features in the layer. Therefore Dropout inhibits the *co-adaptation* of features (Srivastava et al., 2014). Alternatively, Dropout can also be seen as a form of model averaging. When an activation is dropped, it is equivalent to pruning parts of the network or setting certain weights to 0. Therefore with each Dropout mask that is sampled, a sub-network of the entire network is optimised. Therefore in effect, we train an *ensemble* of networks with varying architectures and shared weights. The predictions at test time represent approximate predictions from this ensemble, contributing to lower generalisation error. Recent work has cast Dropout in a probabilistic setting which allows calculation of uncertainty estimates of the predictions from neural networks (Gal and Ghahramani, 2015).

Another simple but effective regularisation technique is *early stopping* (Bishop, 2006, Chapter 5). Since the objective is to train a model that performs well on unseen data, a subset of data is randomly selected and used as a *validation set*. The examples in the validation set are not used for numerical optimisation of the model parameters (Section 3.3.3). Instead, the validation set is used to *monitor* the model performance on unseen data. Typically, the cost on the training set continues to decrease while the cost over the validation set begins to saturate or even increase (Figure 3.6). Early stopping is used to determine when to stop training and updating parameter values.

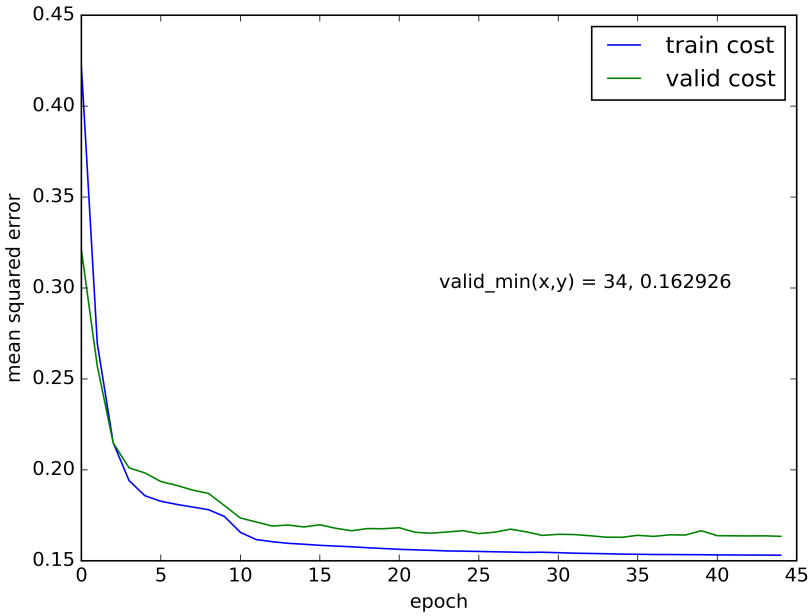


Figure 3.6: Training and validation cost as a function of the number of training epochs.

Qualitatively, saturation of cost over the validation set is seen as an indication that the model is starting to overfit the training examples and therefore stopping training when validation error is minimum is a natural choice. In our experiments, we monitor the validation cost at the end of every *epoch* of training, which is defined as one iteration over the entire training set. Training is stopped if the validation cost does not decrease after a pre-determined number of epochs.

Since overfitting is a very common occurrence in neural networks with many parameters, typically all of the above techniques are used simultaneously to try and control overfitting.

3.4 Density Estimation

So far the discussion has been restricted to the case of supervised learning, where the objective is to learn a mapping $f : X \rightarrow Y$ given many input/output pairs

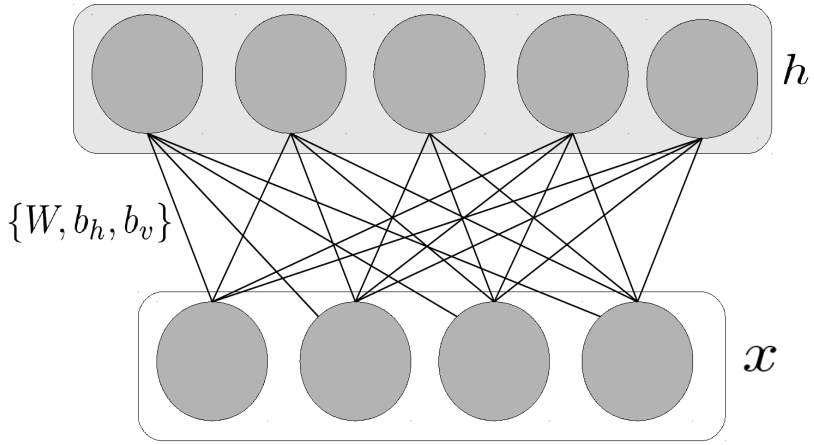


Figure 3.7: Graphical structure of the RBM. There are no connections between variables in the same layer, while every variable x_i is connected all variables h_i and vice versa. Note that the connection between variables are undirected.

as examples. Another problem of interest is the problem of estimating probability distributions over a given set of points. This is also known as *generative modelling* because the trained models can be used to generate new data by sampling from the distributions. Here we describe the restricted Boltzmann machine (RBM) and the neural autoregressive distribution estimator (NADE).

3.4.1 Restricted Boltzmann Machines

The RBM is an energy based model (LeCun et al., 2006) that estimates a joint probability distribution over a set of input variables $x \in \{0,1\}^D$ and a set of hidden or latent variables $h \in \{0,1\}^K$. Each hidden variable h_i is fully connected to all visible variables x and each visible variable x_i is fully connected to all hidden variables h, however all connections between variables in the same layer are *restricted* (Figure 3.7). The joint probability is computed as:

$$P(x,h) = \frac{\exp(-b_v^T x - b_h^T h - h^T W x)}{Z}, \tag{3.31}$$

$$Z = \sum_{x,h} \exp(-b_v^T x - b_h^T h - h^T W x), \tag{3.32}$$

where Z is the normalising constant and is known as the *partition function*. b_v, b_h are the visible and hidden biases, while W is the weight matrix. Computing the partition function exactly is intractable since it involves a summation over all possible configurations of the variables which is exponential in the dimensionality of x and h. The marginal probability of x can be expressed in terms of the free energy:

$$P(x) = \exp(-F(x))/Z, \tag{3.33}$$

$$F(x) = -b_v^T x - \sum_{i=0}^{K-1} \log(1 + e^{b_h + Wx})_i, \tag{3.34}$$

where $F(x)$ is the free energy (Bengio, 2009, Chapter 5) and K is dimensionality of the hidden vector h. RBMs are also trained using gradient based methods. The gradient of the log-likelihood involves the following terms:

$$\frac{\partial(-\log P(x))}{\partial \Theta} = \frac{\partial F(x)}{\partial \Theta} - \mathbb{E}_{\text{RBM}}[\frac{\partial F(x)}{\partial \Theta}], \tag{3.35}$$

where the second term $\mathbb{E}_{\text{RBM}}[\frac{\partial F(x)}{\partial \Theta}]$ denotes an expectation over the model distribution (Bengio, 2009, Chapter 5). Calculating the expectation analytically is intractable. The expectation of the gradient of the free energy is approximated by drawing samples from the model distribution instead. It has been found that even one sample can work well in practice. Samples are drawn using alternative Gibbs sampling steps ($h|x$ and $x|h$). This is known as the Contrastive Divergence algorithm (Hinton, 2002). The binary units of an RBM can be replaced with Gaussian visible units to estimate distributions over real-valued inputs (Welling et al., 2004).

3.4.2 Neural Autoregressive Distribution Estimator

The NADE is a distribution estimator for high dimensional binary data (Larochelle and Murray, 2011). The NADE was initially proposed as a tractable alternative

to the restricted Boltzmann machine (RBM). Consider a vector of binary variables $x \in \{0,1\}^D$. Let x_i denote the i^{th} entry in x and let $x_{<i} = \{x_0, \ldots, x_{i-1}\}$. The NADE estimates the joint distribution over high dimensional binary variables as follows:

$$P(x) = \prod_i P(x_i|x_{<i}).$$

The NADE is similar to a fully visible sigmoid belief network (Neal, 1992), since the conditional probability of x_i is a non-linear function of x_0^{i-1}. The NADE computes the conditional distributions according to:

$$h_i = \sigma(W_{:,<i}x_{<i} + b_h), \tag{3.36}$$

$$P(x_i|x_{<i}) = \sigma(V_i h_i + b_v^i), \tag{3.37}$$

where W, V are weight matrices, $W_{:,<i}$ is a sub-matrix of W that denotes the first i columns and b_h, b_v are the hidden and visible biases, respectively. The equation for the conditional distribution (Equation 3.37) is equivalent to a feed forward neural network with tied weights (Larochelle and Murray, 2011). The gradients of the likelihood function $P(x)$ with respect to the model parameters $\theta = \{W, V, b_h, b_v\}$ can be found exactly, which is not possible with RBMs (Larochelle and Murray, 2011). This property allows the NADE to be readily combined with other models and the models can be jointly trained with gradient based optimisers.

3.4.3 Distributions Over Sequences

Learning distributions over sequences is an important task. Generative sequential models find applications where prior probabilities over sequences are required, for example language modelling (Mikolov et al., 2010). RNNs can be used to define

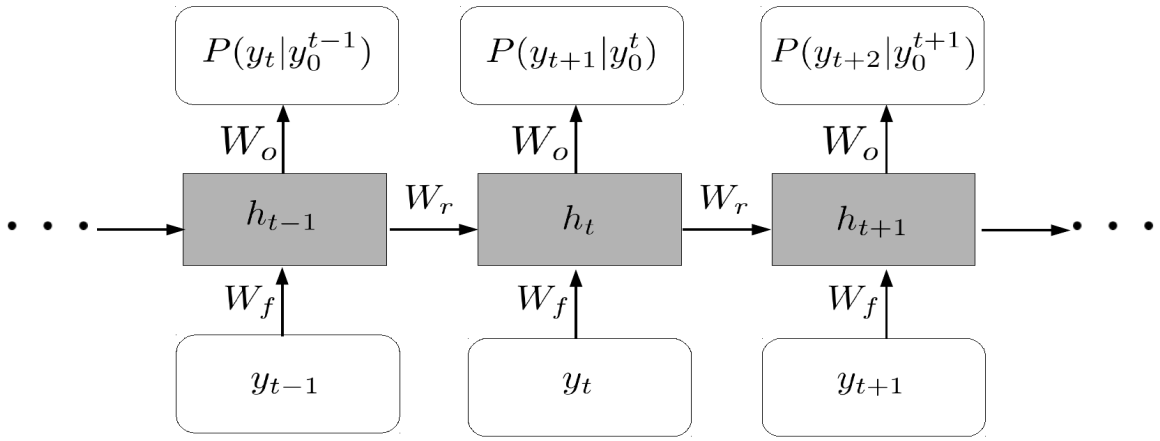


Figure 3.8: Graphical structure of the generative RNN for an input sequence $y = \{y_0, \ldots, y_T\}$. At any time t, the RNN yields a distribution over the outputs at $t+1$. W_f, W_r, W_o represent the input, recurrent and output weight matrices, respectively.

distributions over sequences. Given a sequence $y = \{y_0, \ldots, y_t\}$, the joint probability can be factorised using the product rule:

$$P(y) = P(y_0) \prod_{t>0} P(y_t|y_0^{t-1}), \tag{3.38}$$

where $y_0^t = \{y_0, \ldots, y_t\}$. An RNN that is trained to maximise the likelihood of y_t given y_0^{t-1}, yields the desired distribution over the sequence y (Figure 3.8). Alternatively, a generative RNN can be seen as an RNN model where the output at t is connected to the input at $t-1$.

In Section 3.3.2, we discussed how the output layer activation functions determine the type of output distribution. Although we discussed 3 commonly used output distributions (Gaussian, Bernouilli, multinomial), it is possible to define more complicated output distributions with neural networks. For example a GMM is used to model the output distribution in mixture density neural networks (Bishop, 1994) where the network is trained to output the means and variances of a GMM. Similarly, a recurrent mixture density network can be used to define distributions over real-valued sequences (Schuster, 1999). In general, neural networks can define more

complex families of output distributions by letting the network output the parameters of distribution estimators. As long as the log-likelihood of the distribution estimator is differentiable with respect to its parameters, the entire network can be trained with gradient descent. In Boulanger-Lewandowski et al. (2012), the authors use an RBM and a NADE to define a high-dimensional joint distribution over a sequence of binary variables.

3.5 Other Models

In addition to the neural network models described above, we also use other machine learning models for comparisons, especially in Chapter 6. Here we present the relevant background for the remaining models presented in this thesis.

3.5.1 Gaussian Mixture Models

Given a D dimensional feature vector x, a GMM (Bishop, 2006, Chapter 2) is a weighted sum of Gaussian component densities which provides an estimate of the likelihood of x being generated by the probability distribution defined by a set of parameters θ:

$$p(x|\theta) = \sum_{i=0}^{M-1} w_i \cdot g(x|\mu_i, \Sigma_i), \tag{3.39}$$

where:

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}, \tag{3.40}$$

where $\theta = (\mu_i, \Sigma_i, w_i)_{0 \leq i < M}$ is the parameter set, M is the number of Gaussian components, μ_i is the mean, Σ_i the covariance matrix and w_i is the weight for the i^{th} Gaussian component. In practice, the likelihood is computed in the log domain to avoid numerical underflow. Furthermore, the covariance matrices are constrained to

be diagonal, thus transforming Equations (3.39) and (3.40) into:

$$\log p(x|\theta) = \quad \text{logsum}_{i=0}^{M-1} \left\{ \sum_{d=0}^{D-1} \left(x_d - \mu_{i,d}\right)^2 \cdot \sigma_{i,d}^{-1} + w_i \right\} + K \tag{3.41}$$

where d is the dimension index, constant K can be neglected for classification purposes, and logsum symbolises a recursive version of function $\log(a + b) = \log a + \log(1 + e^{(\log b - \log a)})$ (Murphy, 2006).

3.5.2 Support Vector Machines

Support Vector Machines (SVMs) (Burges, 1998) are discriminative classifiers. Given a set of data points belonging to two different classes, an SVM determines the optimal separating hyperplane between the two classes of data. In the linearly separable case, this is achieved by maximising the margin between two hyperplanes that pass through a number of *support vectors*. The optimal separating hyperplane is defined by all points x that satisfy:

$$\mathrm{x} \cdot \mathrm{w} + b = 0, \tag{3.42}$$

where w is a normal vector to the hyperplane and $\frac{|b|}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin. Given that all data points x_i satisfy:

$$\begin{cases} \mathrm{x}_i \cdot \mathrm{w} + b \geq 1 & \text{for labels } y_i = +1 \\ \mathrm{x}_i \cdot \mathrm{w} + b \leq -1 & \text{for labels } y_i = -1 \end{cases} \tag{3.43}$$

It can be shown that the maximum margin is defined by minimising $\frac{\|\mathrm{w}\|^2}{2}$ (Burges, 1998). This can be solved using a Lagrangian formulation of the problem, thus producing the multipliers α_i and the decision function:

$$f(\mathrm{x}) = \mathrm{sgn}\Big(\sum_{i=0}^{N-1} y_i \alpha_i \mathrm{x} \cdot \mathrm{x}_i + b \Big), \tag{3.44}$$

where N is the number of training examples and x is a feature vector we wish to classify. In practice, most of the α_i turn out to be zero and the x_i for non-zero α_i are called the *support vectors* of the algorithm.

In the case where the data is not linearly separable, a non-linear kernel function $K(\mathrm{x}_i, \mathrm{x}_j)$ can be used to replace the dot products $\mathrm{x} \cdot \mathrm{x}_i$, which effectively projects the data into a higher dimensional space where it could potentially become linearly separable. The decision function then becomes:

$$f(\mathrm{x}) = \mathrm{sgn}\Big(\sum_{i=0}^{N-1} y_i \alpha_i K(\mathrm{x}, \mathrm{x}_i) + b \Big) \tag{3.45}$$

Commonly used kernel functions include:

- Linear: $K(\mathrm{x}, \mathrm{x}_i) = \mathrm{x} \cdot \mathrm{x}_i$
- Polynomial: $K(\mathrm{x}, \mathrm{x}_i) = (\mathrm{x} \cdot \mathrm{x}_i)^d$
- Radial basis function (RBF): $K(\mathrm{x}, \mathrm{x}_i) = \exp(-\gamma |\mathrm{x} - \mathrm{x}_i|^2)$
- Sigmoid: $K(\mathrm{x}, \mathrm{x}_i) = \tanh(\mathrm{x} \cdot \mathrm{x}_i)$

A further refinement to the SVM algorithm makes use of a *soft margin* whereby a hyperplane can still be found even if the data is non-separable (perhaps due to mislabelled examples) (Burges, 1998). The modified objective function is defined as follows:

$$\arg\min_{\mathrm{w},\xi,b} \left\{ \frac{1}{2} \|\mathrm{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{3.46}$$

$$\text{subject to: } y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i, \;\; \xi_i \geq 0,$$

where ξ_i are non-negative *slack variables*. The modified algorithm finds the best hyperplane that fits the data while minimising the error due to misclassified data points. The importance of these error terms is determined by parameter C, which can control the tendency of the algorithm to over fit or under fit the data.

3.6 Implementation

So far we have discussed the mathematical and theoretical foundations of the neural network models used in this thesis. As mentioned earlier, one of the main contributors to the recent success of neural networks has been the ability to design and train networks which are very deep, might contain millions of parameters and that can be trained on very large datasets. The flexible design and training of neural networks has been made possible by the development of open source libraries such as TensorFlow, Theano and Torch. The key idea behind all these libraries is the fact that they act as compilers for *symbolic* mathematical expressions. The libraries allow users to define computational graphs by composing symbolic representations of mathematical expressions. For example, the cost function of a particular neural network architecture can be defined as a symbolic computational graph. Once computational graphs are defined, the libraries are able to optimise various computations to compile highly efficient code for either CPUs or GPUs. Additionally, a symbolic representation of the computations allows the libraries to perform symbolic differentiation with respect to arbitrary graph inputs. When the graphs represent neural networks, this property can be used to automatically compute the gradients for a given instance of a neural network architecture. This flexible approach has allowed researchers to design and experiment with neural network architectures.

The neural network models presented in the remainder of this thesis were implemented using the Theano library for Python. Symbolic graphs were defined for different neural network architectures (DNN, RNN, CNN, NADE) and training was performed on a Tesla K40c GPU. Although writing defining neural networks from scratch using Theano is useful to gain familiarity with various aspects of the neural network architectures and indeed the dynamics of training, in the last 2 years there have been several libraries that have been built on top of Theano and TensorFlow that further simply the implementation and consequently experimentation with neu-

ral networks. Readers are referred to Lasagne[1] and TFLearn[2], which are built on top of Theano and TensorFlow respectively. These open source projects offer a large number of flexible neural network architectures, optimisers and techniques for handling low-level data.

3.7 Conclusion

In this chapter, we presented the necessary background and definitions for the neural network models used in the remaining chapters. We described feed forward, convolutional and recurrent neural network layers. We also show how neural network models can be used for density estimation. We described the gradient descent algorithm, along with a number of commonly used heuristics (like regularisation, dropout, gradient clipping) for effectively estimating the parameters of neural networks. In the following chapter, we describe how neural network models can be used for automatic music transcription of polyphonic music.

[1] `www.lasagne.com`
[2] `www.tflearn.com`

Chapter 4

Automatic Music Transcription

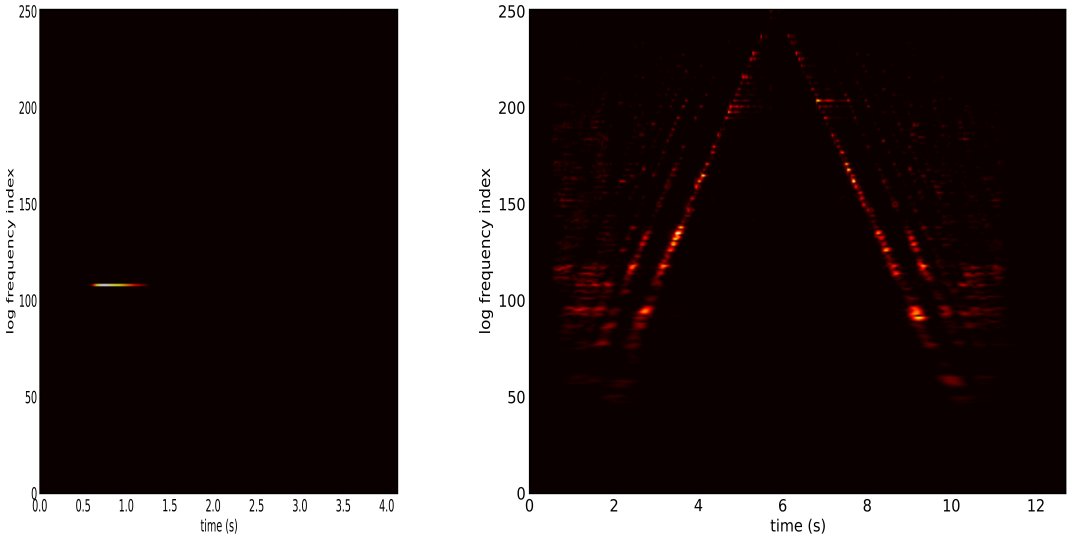
In this chapter we describe experiments which employ neural network based models for automatic music transcription. In the first section, we describe experiments with *piano music transcription*. The motivation for performing experiments with piano music is that labelled datasets of piano music are more readily available, compared to other instruments. This allows us to compare the performance of neural network transcription systems which require a lot of training data, to other methods found in the literature. In addition to processing the acoustic signal, we also investigate whether neural network *music language models* (MLMs) which define a prior probability distribution over sequences of polyphonic music, help improve transcription accuracies. Furthermore, we investigate the problem of estimating the mode of the conditional output distributions of an RNN.

In the second section, we present experiments with *multi-instrument* polyphonic music. These experiments represent a more general experimental setup, where the recording contains more than one instrument and large amounts of labelled training data are not available for each instrument. The aim of these experiments is to investigate whether MLMs can improve transcription performance, when labelled training data for the acoustic models is scarce and the MLMs are trained on a set of musical

scores that are disjoint from the training set of the acoustic model. We present a method for combining the predictions of a latent variable PLCA acoustic model with an RNN MLM and investigate system performance on a dataset of multi-instrument polyphonic music.

4.1 Polyphonic Piano Music Transcription

This section describes experiments with neural networks for transcribing piano music. We are interested in designing an *end-to-end architecture* for polyphonic AMT. It is worth noting that although it is possible to train neural network acoustic models directly on raw audio samples (Graves, 2012b; Dieleman and Schrauwen, 2014), it has been observed that such models are computationally more expensive and do not yield a major improvement in performance compared to pre-processing the waveforms into time-frequency features like the STFT or CQT. Given the fact that datasets in MIR can be orders of magnitude smaller than speech datasets, we assume that all audio waveforms in this thesis are pre-processed into some time-frequency representation. One of the motivations for using deep neural networks architectures is to allow the model to automatically *learn* the most useful transformations or features directly from a low-level time-frequency representation of the audio in order to correctly identify pitches. These experiments aim to carry out fair comparisons between neural network acoustic models and 2 state-of-the-art acoustic models, given sufficient training data. Secondly, we investigate how MLMs can be incorporated into transcription systems. Similar to language models in speech, MLMs have the potential to significantly improve transcription accuracies. Language models in speech are trained on large corpora of text. Similarly, MLMs can be trained on large corpora of musical scores without the need for any manual annotation or labelling. Despite the strong motivation for MLMs, their application to music transcription has been limited due



(a) C3 Note

(b) All notes (ascending and descending)

Figure 4.1: Constant Q Transform plots for examples in the MAPS dataset.

to the high-dimensionality of the output space of polyphonic music. In these experiments, we investigate how neural network MLMs can be incorporated into existing transcription systems. Finally, we also investigate how to better decode the high-dimensional conditional distributions obtained as outputs from an RNN. The rest of this section is organised as follows: we first describe the various neural network architectures used for acoustic modelling. We then describe the RNN MLMs. Next, we present the proposed model for combining acoustic and language model predictions. Finally we present results from experiments with the proposed model on a dataset of piano music.

4.1.1 Preprocessing

We transform the input audio to a time-frequency representation which is then used as input to the acoustic models. In the past, Sigtia et al. (2015a) used the magnitude short-time Fourier transform (STFT) as input to the acoustic models. However, here we experiment with the constant Q transform (CQT) as the input representation

(Brown, 1991). There are two motivations for this. Firstly, the CQT is fundamentally better suited as a time-frequency representation for music signals, since the frequency axis is linear in pitch (Figure 4.1b). Secondly, the log-frequency axis requires fewer frequency bins to effectively cover a given range, resulting in a more compact representation. This is practically very useful given that the human auditory range covers approximately ten octaves (from 20 Hz to 20kHz). For the CQT representation, the spectral resolution is higher for lower frequencies and decreases for higher frequencies. From a machine learning perspective, a compact representation is useful when using neural networks since it reduces the number of parameters in the model.

We downsample the audio to 16 kHz from 44.1 kHz. We then compute CQTs over 7 octaves with 36 bins per octave and a hop size of 512 samples, resulting in a 252 dimensional input vector of real values, with a frame rate of 31.25 frames per second. Additionally, we compute the mean and standard deviation of each dimension over the training set and transform each vector by subtracting the mean and dividing by the standard deviation. These pre-processed vectors are used as inputs to the acoustic model.

4.1.2 Acoustic Models

The acoustic model is used to identify the active pitches in a short frame of audio. Acoustic models can be broadly divided into 2 classes: supervised and unsupervised acoustic models. Supervised acoustic models can in theory be trained on complex mixtures of instrument sources, without having to account for each instrument separately. This is in contrast to NMF/PLCA acoustic models that require instrument specific prior knowledge for transcription. For AMT, acquiring large datasets of musical recordings with corresponding human annotated transcriptions is a difficult task (Su and Yang, 2015). Consequently, the datasets available for AMT are usually quite small and unsupervised or NMF-based acoustic models are preferred over supervised

models. In this study, we aim to compare the performance of supervised neural network acoustic models to two popular state-of-the-art acoustic models. In order to be able to make fair comparisons, we perform experiments on a dataset of piano music with sufficient data for training neural networks. In this section, we describe the various neural network architectures considered for acoustic modelling.

4.1.2.1 DNNs

Given an input frame of features x_t at any time t, the DNN with one or more hidden layers is trained to predict the probability of pitches y_t (Figure 3.1). The DNN yields a probability distribution $P(y_t|x_t)$. Each y_t is an 88-dimensional binary vector representing the keys on a piano. The input to the DNN is x_t, a frame of preprocessed CQT. A sigmoid non-linearity is applied to the activations of the output layer and each output represents $P(y_t(i) = 1|x_t)$, the probability of the i^{th} pitch being on.

4.1.2.2 RNNs

DNNs make a prediction given a frame of acoustic features as inputs. However, a single frame of features contains insufficient data since a frame is ambiguous without its context. There are many previous studies in MIR that suggest that rather than classifying a single frame of input, better prediction accuracies can be achieved by incorporating information over several frames of inputs (Bergstra et al., 2006; Boulanger-Lewandowski et al., 2013a; Sigtia and Dixon, 2014). Unlike DNNs, RNN acoustic models incorporate past inputs while making a prediction (Figure 3.4). Given a sequence of inputs $x_0^t = \{x_0, \ldots, x_t\}$, the RNN yields an output probability distribution $P(y_t|x_0^t)$. Through the recurrent hidden state (Equation 3.5), the RNN is able to model distributions that are conditioned on the entire input sequence x_0^t.

4.1.2.3 ConvNets

ConvNets contain one or more layers of convolutions, in addition to fully connected layers (Section 3.2.2, Figure 3.3). Although ConvNets were designed for processing images, they have been successfully applied to speech recognition (Abdel-Hamid et al., 2012, 2013). We use ConvNet acoustic models to estimate a distribution $P(y_t|x_{t-k}^{t+k})$, where the input to the acoustic model is a *context window* of features of size $2k+1$ and the ConvNet is trained to predict the targets corresponding to the central frame in the window. Unlike RNNs, ConvNets use a fixed size context that incorporates information from both past and *future* frames. When applied to a time-frequency input representation like the CQT, ConvNets can potentially learn features that are invariant to pitch transpositions, such as inter-harmonic spacings for music signals which are constant across log-frequency. Although this is theoretically possible with fully connected networks, we would require a very large number of training examples for the network to discover such regularities. Another important property of ConvNets is that the size of the input window can be increased without necessarily increasing the size of the convolutional filters, thereby resulting in models with fewer parameters (Section 3.2.2). Although ConvNets have been applied to some problems in MIR (Schlüter and Böck, 2014; Humphrey and Bello, 2012), they remain unexplored for AMT tasks.

4.1.3 Music Language Models

As mentioned earlier, the motivation for statistical models of sequences of music is similar to the motivation for statistical language models in speech recognition. Often the acoustic signal on its own may not contain all the information necessary to identify the correct words contained in the signal. Languages exhibit structural regularities which can be used to make better predictions. Similarly for music, the structural regularities and patterns in sequences of music can be used to improve predictions

of AMT systems. From a probabilistic perspective, when trying to estimate $P(y|x)$, language models provide a prior $P(y)$ for the likelihood of the sequence y.

Given a sequence of transcriptions in piano-roll notation $y = y_0^t$, we use the MLM to define a prior probability distribution $P(y)$. y_t is an 88-dimensional binary vector that represents the notes being played at t (one time-step of a piano-roll representation). Index 0 represents A0 (27.5 Hz) and index 87 corresponds to C8 (4186 Hz). The motivation for using MLMs is to learn harmonic rules and patterns in music and use this information to improve transcription. The harmonic content of music includes information related to musical intervals, scales and chords. In addition to the harmonic content of music, there are also temporal structural patterns like the synchronisation of pitch changes and durations of pitches or inter-onset intervals and also more higher level musical structure like verses and chorus. The MLMs presented here explicitly model the harmonic properties only, while higher-level temporal information (like beats, duration) is ignored.

Compared to language modelling for speech, modelling the structure of music is complicated by the presence of polyphony and complex long-term temporal structure. Due to polyphony, the number of possible note combinations is exponentially large, for example there are 2^{88} possible note combinations for 88 notes on a piano. Secondly, statistical modelling of the long-term repeating structure of polyphonic notes is a difficult task. Most transcription systems make the simplifying assumption that all notes are independent and model the temporal evolution of notes independently (one model per note) (e.g. Poliner and Ellis (2007)). Models of this kind are capable of learning the *durations* of notes and smoothing the predictions of the acoustic models. Although there are some studies that investigate jointly modelling the temporal and polyphonic structure of music (Raczynski et al., 2013), the problem of music language modelling has been largely ignored for AMT. In this section, we describe RNN based-architectures for MLMs.

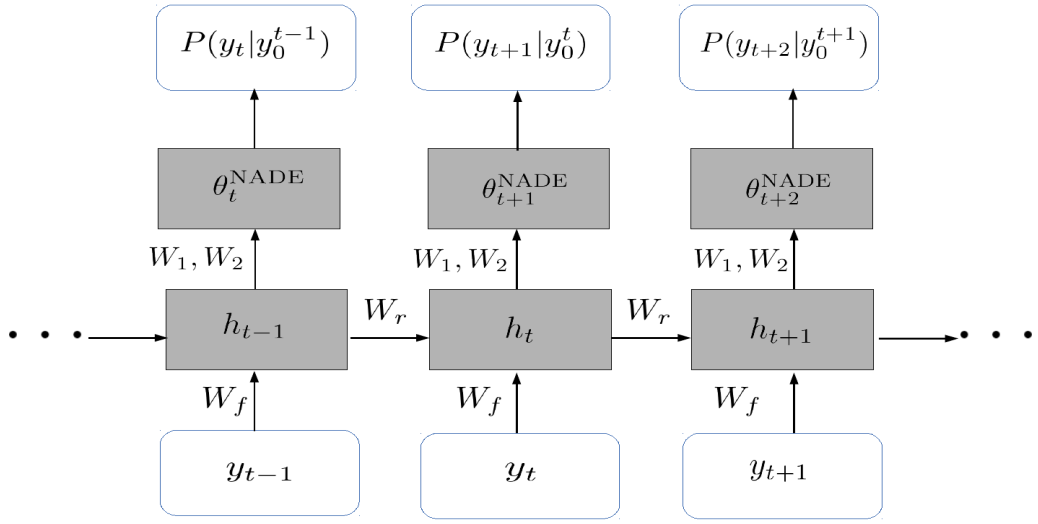


Figure 4.2: Graphical structure of the RNN-NADE for an input sequence $y = \{y_0, \ldots, y_T\}$. Compared to the generative RNN (Figure 3.8), the parameters of a NADE at time t are conditioned on the hidden state h_t and the conditional distribution $P(y_{t+1}|y_0^t)$ is obtained from the NADE for all t.

4.1.3.1 Generative RNN

In section 3.4.3 we described how RNNs can be used to define probability distributions over sequences (Equation 3.38). When the inputs to the generative RNN are formed by sequences of polyphonic music $y = y_0^t$, the model describes an MLM. Although an RNN MLM yields a distribution $P(y_t|y_0^{t-1})$, the individual pitch outputs $y_t(i)$ are conditionally independent given y_0^{t-1}, where i is the pitch index. As mentioned earlier, this is not true for polyphonic music where harmonies and chords are formed by multiple simultaneously sounding notes. Therefore, although RNN MLMs can be used to define a distribution over a sequences of symbolic music, the output distribution at each time step is limited to modelling independent pitches. It should be noted that although this is a limitation, a generative RNN is conditioned on high dimensional input sequences and is more general than using 2-state HMMs for each pitch separately.

4.1.3.2 RNN-NADE

In order to learn high dimensional, temporal distributions for the MLM, we combine the NADE and an RNN, as proposed by Boulanger-Lewandowski et al. (2012). The resulting model is shown in Figure 4.2. As mentioned earlier, the main limitation of the generative RNN is the fact that the output distribution is limited to modelling independent pitches. The NADE (Larochelle and Murray, 2011) is a distribution estimator for high-dimensional binary data. The RNN-NADE is obtained by conditioning the parameters of a NADE at every time-step on the hidden state of an RNN (Figure 4.2). The NADE can be seen as an additional *differentiable* block in the generative RNN architecture. The resulting model yields a sequence of NADEs conditioned on an RNN, that describe a distribution over sequences of polyphonic music. In order to limit the number of free parameters in the model, we only allow the NADE biases to be functions of the RNN hidden state, while the remaining parameters (W, V) are held constant over time (Section 3.4.2). We compute the NADE biases as a linear transformation of the RNN hidden state plus an added bias term:

$$b_v^t = b_v + W_1 h_t, \tag{4.1}$$

$$b_h^t = b_h + W_2 h_t, \tag{4.2}$$

where W_1 and W_2 are weight matrices from the RNN hidden state to the visible and hidden biases, while b_h, b_v are the *fixed* hidden and visible biases for the NADE (Equation 3.36). The gradients with respect to all the model parameters can be easily computed using the chain rule and the joint model is trained using the BPTT algorithm (Boulanger-Lewandowski et al., 2012).

4.1.4 Proposed Model

In this section we describe the proposed neural network model for polyphonic AMT. The model comprises an acoustic model and a music language model. In addition to DNN and RNN acoustic models, we propose the use of ConvNets for identifying pitches present in the input audio signal and compare their performance to various other acoustic models (Section 4.1.5.5). The acoustic and language models are combined using a probabilistic graphical model, yielding an end-to-end model for AMT with unconstrained polyphony. We first describe the hybrid RNN model, followed by a description of the proposed inference algorithm.

4.1.4.1 Hybrid RNN

The hybrid RNN is a graphical model that combines the predictions of any *arbitrary* frame level acoustic model, with an RNN-based language model. Let $x = x_0^T$ be a sequence of inputs and let $y = y_0^T$ be the corresponding transcriptions. The joint probability of y, x can be factorised as follows:

$$\begin{aligned} P(y, x) &= P(y_0 \dots y_T, x_0 \dots x_T) \\ &= P(y_0)P(x_0|y_0) \prod_{t=1}^{T} P(y_t|y_0^{t-1})P(x_t|y_t). \end{aligned} \tag{4.3}$$

The factorisation in Equation 4.3 makes the following independence assumptions:

$$P(y_t|y_0^{t-1}, x_0^{t-1}) = P(y_t|y_0^{t-1}) \tag{4.4}$$

$$P(x_t|y_0^{t}, x_0^{t-1}) = P(x_t|y_t) \tag{4.5}$$

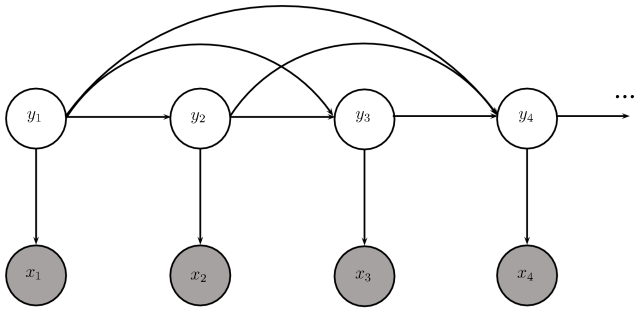


Figure 4.3: Graphical Model of the Hybrid Architecture. The variables y_t represent the output pitches, while the variables x_t represent the acoustic observations. Compared to the HMM graph (Figure 2.5) there are additional connections between each state y_t and all previous states y_τ, for $\tau < t$.

These independence assumptions are similar to the assumptions made in HMMs (Rabiner, 1989). Equation 4.4 implies that pitch content at t is dependent only on the previous pitch content y_0^{t-1} and independent of the past acoustic inputs x_0^{t-1}. Equation 4.5 implies that the acoustic observations at t are only conditioned on the active pitches at time t. Figure 4.3 is a graphical representation of the hybrid model. In equation 4.3, $P(x_t|y_t)$ is the *emission* probability of an input, given output y_t. Using Bayes's rule, the conditional distribution can be written as follows:

$$P(y|x) \propto P(y_0|x_0) \prod_{t=1}^{T} P(y_t|y_0^{t-1}) P(y_t|x_t), \tag{4.6}$$

where the marginals $P(y_t)$ and priors $P(y_0)$, $P(x_0)$ are assumed to be fixed with respect to the model parameters and are replaced by the proportionality sign.

With this reformulation of the joint distribution, we observe that the conditional distribution $P(y|x)$ is directly proportional to the product of two distributions. The prior distribution $P(y_t|y_0^{t-1})$ is obtained using a generative RNN (Section 4.1.3) and the posterior distribution over note-combinations $P(y_t|x_t)$ can be modelled using any frame based classifier. The hybrid RNN graphical model is similar to an HMM, where the state transition probabilities for the HMM $P(y_t|y_{t-1})$ have been generalised to include connections from all previous outputs, resulting in the $P(y_t|y_0^{t-1})$ terms in

Equation 4.6.

For the problem of automatic music transcription, the input time-frequency representation forms the input sequence x, while the output piano-roll sequence y denotes the transcriptions. The priors $P(y_t|y_0^{t-1})$ are obtained from the RNN-NADE MLM, while the posterior distributions $P(y_t|x_t)$ are obtained from the acoustic models. The models can then be trained by finding the derivatives of the acoustic and language model objectives with respect to the model parameters and training using gradient descent. The independent training of the acoustic and language models is a useful property since datasets available for music transcription are considerably smaller in size as compared to datasets in computer vision and speech. However large corpora of MIDI music are relatively easy to find on the internet[1] (Poliner and Ellis, 2007; Allan and Williams, 2005). Therefore in theory, the MLMs can be trained on large corpora of MIDI music, analogous to language model training in speech.

4.1.4.2 Inference

At test time, we would like to find the mode of the conditional output distribution:

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x) \tag{4.7}$$

From Equation 4.6, we observe that the priors $P(y_t|y_0^{t-1})$, tie the predictions of the acoustic model $P(y_t|x_t)$ to all the predictions made till time t. This prior term encourages coherence between predictions over time and allows musicological structure learnt by the language models to influence successive predictions. However, this more general structure leads to a more complex inference (or decoding) procedure at test time. This is due to the fact that at time t, the history y_0^{t-1} has not been optimally determined. Therefore, the optimum choice of y_t depends on *all* the past model predictions. Proceeding greedily in a chronological manner by selecting y_t that optimises

[1]for e.g `ifdo.ca/~seymour/nottingham/nottingham.html` and `http://www.musedata.org/`

$P(y_t|x_t)$ does not necessarily yield good solutions. We are interested in solutions that globally optimise $p(y|x)$. But exhaustively searching for the best sequence is intractable since the number of possible configurations of y_t is exponential in the number of output pitches (2^n for n pitches).

Beam search is a graph search algorithm that is commonly used to decode the conditional outputs of an RNN (Graves, 2012a; Boulanger-Lewandowski et al., 2013b). Beam search scales to arbitrarily long sequences and the computational cost versus accuracy trade-off can be controlled via the width of the beam. The inference algorithm is comprised of the following steps: at any time t, the algorithm maintains at most w partial solutions, where w is the beam width or the beam capacity. The solutions in the beam at t correspond to sub-sequences of length t. Next, all possible descendants of the w partial solutions in the beam are enumerated and then sorted in decreasing order of log-likelihood. From these candidate solutions, the top w solutions are retained as beam entries for further search. Beam search can be readily applied to problems where the number of candidate solutions at each step is limited, like speech recognition (Boulanger-Lewandowski et al., 2014) and audio chord estimation (Boulanger-Lewandowski et al., 2013a). However, using beam search for decoding sequences with a large output space is prohibitively inefficient.

When the space of candidate solutions is large, the algorithm can be constrained to consider only K new candidates for each partial solution in the beam, where K is known as the *branching factor*. The procedure for selecting the K candidates can be designed according to the given problem. For the hybrid architecture, from Equation 4.6 we note:

$$P(y_0^t|x_0^t) \propto P(y_0^{t-1}|x_0^{t-1})P(y_t|y_0^{t-1})P(y_t|x_t) \tag{4.8}$$

At time t, the partial solutions in the beam correspond to configurations of y_0^{t-1}. Therefore given $P(y_0^{t-1}|x_0^{t-1})$, the K configurations that maximise $P(y_t|y_0^{t-1})P(y_t|x_t)$

would be a suitable choice of candidates for y_t. However for many families of distributions, it might not be possible to enumerate y_t in decreasing order of likelihood. Boulanger-Lewandowski et al. (2013b) propose forming a pool of K candidates by drawing random samples from the conditional output distributions. However, random sampling can be inefficient and obtaining independent samples can be very expensive for many types of distributions. As an alternative, we propose to sample solutions from the posterior distribution of the acoustic model $P(y_t|x_t)$ (Sigtia et al., 2015a). There are 2 main motivations for doing this. Firstly, the outputs of the acoustic model are independent class probabilities. Therefore, it is easy to enumerate samples in decreasing order of log-likelihood (Boulanger-Lewandowski et al., 2013b). Secondly, we avoid the accumulation of errors in the RNN predictions over time (Bengio et al., 2015; Ranzato et al., 2016). The RNN models are trained to predict y_t, given the *true* outputs y_0^{t-1}. However at test time, outputs sampled from the RNN are fed back as inputs at the next time step. This discrepancy between the training and test objectives can cause prediction errors to accumulate over time.

Although generating candidates from the acoustic model yields good results, it requires the use of large beam widths. This makes the inference procedure computationally slow and unsuitable for real-time applications (Sigtia et al., 2015a). We propose using the *hashed beam search* algorithm proposed by Sigtia et al. (2015b). Beam search is fundamentally limited when decoding long temporal sequences. This is due to the fact that solutions that differ at only a few time-steps can saturate the beam. This causes the algorithm to search a very limited space of possible solutions. This issue can be solved by efficient pruning. The hashed beam search algorithm improves efficiency by pruning solutions that are *similar* to solutions with a higher likelihood. The metric that determines the similarity of sequences can be chosen in a problem dependent manner and is encoded in the form of a locality sensitive hash function (Sigtia et al., 2015b).

Algorithm 1 High Dimensional Beam Search

Find the most likely sequence y given x with a beam width w and branching factor K.

$beam \leftarrow$ new beam object
$l \leftarrow 0$
$s \leftarrow \{\}$
$m_l \leftarrow m_l$ with $y_t := \{\}$
$m_a \leftarrow m_a$ with $x := x_0$
$beam$.insert(l, s, m_a, m_l)
for $t = 1$ to T **do**
    $new\_beam \leftarrow$ new beam object
    for l, s, m_a, m_l **in** $beam$ **do**
        for $k = 1$ to K **do**
            $y' = m_a$.next_most_probable_candidates()
            $l' = \log P_l(y'|s) P_a(y'|x_t) - \log P(y')$
            $m'_l \leftarrow m_l$ with $y_t := y'$
            $m'_a \leftarrow m_a$ with $x := x_{t+1}$
            new_beam.insert$(l + l', \{s, y'\}, m_a, m_l)$
    $beam \leftarrow new\_beam$
return $beam$.pop()

Algorithm 1 describes the general beam-search algorithm. The beam object can be a priority queue (Sigtia et al., 2015a) or a hash table (Algorithm 3). The entries to the beam are tuples (l, s, m_a, m_l) containing the likelihood l of a subsequence s, followed by the acoustic model m_a and the language model m_l. The beam search comprises 2 for loops. The outer loop iterates through the input sequence, while the inner loop at each iteration evaluates K (branching factor) candidate solutions which are stored in the beam. It should be noted that according to the earlier discussion, the acoustic model computations $P(y_t|x_t)$ can be performed independently outside both loops. However, the frame-level acoustic models can be replaced by models that incorporate previous model predictions as inputs, for example the model by Boulanger-Lewandowski et al. (2013a). The acoustic models have been included within the for loop for cases where they might have to maintain a *state*. Another reason for including the acoustic models in the for loop is that at each step, the K most likely predictions are enumerated using the efficient dynamic programming

algorithm described in Algorithm 2. Algorithm 2 which is $O(K \log K + N \log N)$, allows the use of large much larger beam widths with unbounded branching factors. This fact is more clearly expressible in each iteration of the for loop in Algorithm 1.

Algorithm 2 Find Most Probable Candidates

Find the K most likely configurations of N independent Bernouilli random variables with parameters $0 < p_i < 1$.
$v_0 \leftarrow \{i : p_i \geq 1/2\}$
$l_0 \leftarrow \sum_i \log(\max(p_i, 1 - p_i))$
yield v_0, l_0
$L_i \leftarrow |\log \frac{p_i}{1-p_i}|$
sort L, store corresponding permutation R
q $\leftarrow$ min-priority queue
q.insert($L_0, \{0\}$)
while $l, v \leftarrow q$.pop() **do**
    yield $l_0 - l, v_0 \Delta R[v]^*$
    $i \leftarrow \max(v)$
    if i+i ¡ N **then**
        q.insert($l + L_{i+1}, v \cup \{i + 1\}$)
        q.insert($l + L_{i+1} - L_i, v \cup \{i + 1\}$)

$^*A \Delta B = (A \cup B) \backslash (A \cap B)$ and $R[v]$ represents the R-permutation of indices in the set v.

There are two key differences between Algorithm 1 and the algorithm by Sigtia et al. (2015a). First, the beam object in Algorithm 1 is general and does not have to be a priority queue. Secondly, for each entry in the beam we evaluate K candidate solutions. This is in contrast to the algorithm in (Sigtia et al., 2015a), where the algorithm maintains w solutions in the beam and at each iteration the locally most optimal choice is added to the previous solution. It might appear that the hashed beam search algorithm might be more expensive, since it evaluates $w * K$ candidates instead of w candidates. However, by efficiently pruning similar solutions, the algorithm yields better results for much smaller values of w, resulting in a significant increase in efficiency (Section 4.1.5.5, Figure 4.4).

Algorithm 3 describes the hash table beam object. The beam object is parameterised by the beam width w, the hash key generating function f_h and the maximum

Algorithm 3 Description of beam objects given w, f_h, k

Initialise beam object
beam.hashQ = new defaultdict of priority queues*
beam.queue = new indexed priority queue of length w**
Insert l, s, m_a, m_l **into beam**
key= $f_h(s)$
queue = beam.queue
hashQ = beam.hashQ[key]
fits_in_queue = **not** queue.full() **or** $l \geq$queue.min()
fits_in_hashQ = **not** hashQ.full() **or** $l \geq$hashQ.min()
if fits_in_queue **and** fits_in_hashQ **then**
    hashQ.insert(l, s, m_a, m_l)
    if hashQ.overfull() **then**
        item = hashQ.del_min()
        queue.remove(item)
    queue.insert(l, s, m_a, m_l)
    if queue.overfull() **then**
        item = queue.del_min()
        beam.hashQ[f_h(item.s)].remove(item)

* A priority queue of length k maintains the top k entries at all times.
** An *indexed* priority queue allows efficient random access and deletion.

number of entries per key k. Given a subsequence s, the function f_h computes the key for the subsequence, $f_h(s)$. If the current solution s has a higher log-likelihood compared to existing solutions in the beam, then the new entry s is added to the beam. Each key in the table corresponds to a priority queue of maximum capacity k. In addition to the hash table, the algorithm maintains a *global* priority queue in order to efficiently retrieve and delete (in $O(1)$) the entry with the smallest log-likelihood when necessary.

The hashed beam search algorithm offers several advantages compared to the standard beam search algorithm. The notion of similarity of solutions can be encoded in the form of hash functions. For music transcription, we choose the similarity function to be equality of the last n frames in a sequence s. $n = 1$ corresponds to a dynamic programming like decoding (similar to HMMs) where all sequences with the same final state y_t are considered to be equivalent, and the sequence with the

highest log-likelihood is retained. n = len(sequence) corresponds to regular beam search. Additionally, the hash beam search algorithm can maintain ≥ 1 solution per hash key through a process called chaining (Cormen et al., 2001, Chapter 11).

4.1.5 Evaluation

In this section we describe how the performance of the proposed model is evaluated for a polyphonic piano transcription task.

4.1.5.1 Dataset

We evaluate the proposed model on the MAPS dataset (Emiya et al., 2010). The dataset consists of audio and corresponding annotations for isolated sounds, chords and complete pieces of piano music. For our experiments, we use only the full musical pieces for training and testing the neural network acoustic models and MLMs. The dataset consists of 54 unique pieces of classical music, where each piece is played on up to 9 variations of piano and recording conditions, resulting in 270 recordings in total. 7 categories of audio are produced by software piano synthesisers, while 2 sets of recordings are obtained from a Yamaha Disklavier upright piano.

We perform 2 sets of investigations in this study. The first set of experiments investigates the effect of the RNN MLMs on the predictions of the acoustic models. For this task, we divide the entire dataset into 4 disjoint train/test splits, to ensure that the folds are music piece-independent. Specifically, for some of the musical pieces in the dataset, audio for each piece is rendered using more than one piano. Therefore while creating the splits, we ensure that the training and test data do not contain any overlapping pieces[2]. For each split, we select 80% of the data for training (216 musical pieces) and the remaining for testing (54 pieces). From each training split, we hold out 26 tracks as a validation set for selecting the hyper-parameters for the

[2]Details of splits available at: `http://www.eecs.qmul.ac.uk/~sss31/TASLP/info.html`

training algorithm (Section 4.1.5.3). All the reported results are mean values of the evaluation metrics over the 4 splits. From now on, this evaluation configuration will be named as *Configuration 1*. Table 4.1 shows the distribution of training and test data.

Although the above experimental setup is useful for investigating the effectiveness of the RNN MLMs, the training set contains examples from piano models which are used for testing. This is usually not the case in practice, where the instrument models/sources at test time are unknown and usually do not coincide with the instruments used for training. The majority of experiments with the MAPS dataset train and test models on disjoint instrument types (Benetos and Dixon, 2012; Berg-Kirkpatrick et al., 2014; O'Hanlon and Plumbley, 2014). We thus perform a second set of experiments to compare performance of the different neural network acoustic models in a more practical setting. We train the acoustic models using the 210 tracks created using synthesized pianos (180 tracks for training and 30 tracks for validation) and we test the acoustic models on the 60 audio recordings obtained from Yamaha Disklavier piano recordings (models 'ENSTDkAm' and 'ENSTDkCl' in the MAPS database). In this experiment, we do not apply the language models since the train and test sets contain overlapping musical pieces. In addition to the neural network acoustic models, we include comparisons with two popular acoustic models (Vincent et al., 2010; Benetos and Dixon, 2012) for both experiments. This instrument source-independent evaluation configuration will be named from now on as *Configuration 2*.

According to convention, we use the first 30 seconds of each recording from the test set (Vincent et al., 2010; Benetos and Dixon, 2012). The test audio is sampled at a frame rate of 100 Hz yielding $100 * 30 = 3000$ frames per test file. For 54 test files over 4 splits, we obtain a total of $648,000$ frames at test time for Configuration

Number of Frames			
Fold	Train	Validation	Test
1	1 412 539	205 667	162 000
2	1 480 979	203 421	162 000
3	1 559 251	226 363	162 000
4	1 216 892	193 273	162 000

Table 4.1: Distribution of data over the train, valid and test splits for the MAPS dataset for Configuration 1.

1[3].

4.1.5.2 Metrics

We use both frame and note based metrics to assess the performance of the proposed system (Bay et al., 2009). Frame-based evaluations are made by comparing the transcribed binary output and the MIDI ground truth frame-by-frame. For note-based evaluation, the system returns a list of notes, along with the corresponding pitches, onset and offset times. We use the F-measure, precision, recall and accuracy for both frame and note based evaluation. Formally, the frame-based metrics are defined as:

$$\mathcal{P} = \sum_{t=1}^{T} \frac{\mathrm{TP[t]}}{\mathrm{TP[t]} + \mathrm{FP[t]}},$$

$$\mathcal{R} = \sum_{t=1}^{T} \frac{\mathrm{TP[t]}}{\mathrm{TP[t]} + \mathrm{FN[t]}},$$

$$\mathcal{A} = \sum_{t=1}^{T} \frac{\mathrm{TP[t]}}{\mathrm{TP[t]} + \mathrm{FP[t]} + \mathrm{FN[t]}},$$

$$\mathcal{F} = \frac{2 * \mathcal{P} * \mathcal{R}}{\mathcal{P} + \mathcal{R}},$$

where TP[t] is the number of true positives for the event at t, FP is the number of

[3]It should be noted that carrying out statistical significance tests on a track level is an over-simplification in the context of multi-pitch detection, as argued in (Benetos, 2012).

false positives and FN is the number of false negatives. The summation over t is carried out over the entire test data. Similarly, analogous note-based metrics can be defined (Bay et al., 2009). A note event is defined to be correct if its predicted pitch is correct and its onset is within a $\pm 50ms$ range of the ground truth onset.

4.1.5.3 Network Training

In this section we describe the details of the training procedure for the various acoustic model architectures and the RNN-NADE language model. All the acoustic and language models have 88 units in the output layer, corresponding to the 88 output pitches. The outputs of the final layer of all acoustic models are transformed by a sigmoid function and yield independent pitch probabilities $P(y_t(i) = 1|x)$. The resulting cost function used for training is the binary cross-entropy function, which is used to iteratively estimate parameters that maximise the log-likelihood of the data over the entire training set.

- **DNN Acoustic Models**: For DNN training, we constrain all the hidden layers of the model to have the same number of units to simplify searching for good model architectures. We perform a grid search over the following parameters: number of layers $L \in \{1, 2, 3, 4\}$, number of hidden units per layer $H \in \{25, 50, 100, 125, 150, 200, 250\}$, hidden unit activations $act \in \{ReLU, sigmoid\}$ where ReLU is the rectified linear unit activation function (Glorot et al., 2011). We found Dropout (Srivastava et al., 2014) to be essential for improving generalisation performance. A Dropout rate of 0.3 was used for the input layer and all the hidden layers of the network. Rather than using learning rate and momentum update schedules, we use ADADELTA (Zeiler, 2012) to adapt the learning over iterations. In addition to Dropout, we use early stopping to determine when to stop training. Training was stopped if the cost over the validation set did not decrease for 20 epochs. We used mini batches of size 100 for the

SGD updates.

- **RNN Acoustic Models**: For RNN training, we constrain all the hidden layers to have the same number of units. We perform a grid search over the following parameters: $L \in \{1, 2, 3\}$, $H \in \{25, 50, 100, 150, 200, 250\}$. We fix the hidden activations of the recurrent layers to be the hyperbolic tangent function. We found that ADADELTA was not particularly well suited for training RNNs. We use an initial learning rate of 0.001 and linearly decrease it to 0 over 1000 iterations. We use a constant momentum rate of 0.9. The training sequences are further divided into sub-sequences of length 100. The SGD updates are made one sub-sequence at a time, without any mini batching. Similar to the DNNs, we use early stopping and stop training if validation cost does not decrease after 20 iterations. In order to prevent gradient explosion in the early stages of training, we use gradient clipping (Bengio et al., 2013). We clipped the gradients when the norm of the gradient was greater than 5.

- **ConvNet Acoustic Models**: The input to the ConvNet is a context window of frames and the target is the central frame in the window. The frames at the beginning and end of the audio are zero padded so that a context window can be applied to each frame[4]. Although pooling can be performed along both axes, we only perform pooling over the frequency axis. We performed a grid search over the following parameters: window size $w_s \in \{3, 5, 7, 9\}$ number of convolutional layers $L_c \in \{1, 2, 3, 4\}$, number of filters per layer $n_l \in \{10, 25, 50, 75, 100\}$, number of fully connected layers $L_{fc} \in \{1, 2, 3\}$, number of hidden units in fully connected layers $H \in \{200, 500, 1000\}$. The convolution activation functions were fixed to be the hyperbolic tangent functions, while all the fully connected

[4]Note that typically the first and last frames in a sequence are padded by repeating either the beginning or the end frames in the sequence. However since there are few very such frames during training and evaluation, padding the context windows with zeros does not affect the results and we choose to perform zero padding for convenience.

Model	Architecture
DNN	$L = 3, H = 125$
RNN	$L = 2, H = 200$
ConvNet	$w_s = 7, L_c = 2, L_{fc} = 2, w_1 = (5, 25), P_1 = (1, 3)$ $w_2 = (3, 5), P_2 = (1, 3), n_1 = n_2 = 50, h_1 = 1000, h_2 = 200$
RNN-NADE	$H_{RNN} = 200, H_{NADE} = 150$

Table 4.2: Model configurations for the best performing architectures.

layer activations were set to the sigmoid function. The pooling size is fixed to be $P = (1, 3)$ for all convolutional layers. Dropout with rate 0.5 is applied to all convolutional layers. We tried a large permutation of window shapes for the convolutional layer and the following subset of window shapes yielded good results: $w \in \{(3, 3), (3, 5), (5, 5), (3, 25), (5, 25), (3, 75), (5, 75)\}$. We observed that classification performance deteriorated sharply for longer filters along the frequency axis. 0.5 Dropout was applied to all the fully connected layers. The model parameters were trained with SGD and a batch size of 256. An initial learning rate of 0.01 was linearly decreased to 0 over 1000 iterations. A constant momentum rate 0.9 was used for all the updates. We stopped training if the validation error did not decrease after 20 iterations over the entire training set.

- **RNN-NADE MLMs**: The RNN-NADE models were trained with SGD and with sequences of length 100. We performed a grid search over the following parameters: number of recurrent units $H_{RNN} \in \{50, 100, 150, 200, 250, 300\}$ and number of hidden units for the NADE $H_{NADE} \in \{50, 100, 150, 200, 250, 300\}$. The model was trained with an initial learning rate of 0.001 which was linearly reduced to 0 over 1000 iterations. A constant momentum rate of 0.9 was applied throughout training.

We selected the model architectures by performing a grid search over the configurations described earlier in the section. The various models were trained on one train/test split and the best performing architecture was then used for all 4 splits in

our experiments (Table 4.2).

4.1.5.4 Comparative Approaches

For comparative purposes, two state-of-the-art polyphonic music transcription methods were used for experiments (Vincent et al., 2010; Benetos and Dixon, 2012). In both cases, the non-binary *pitch activation* output of the aforementioned methods was extracted, for performing an in-depth comparison with the proposed neural network models.

The multi-pitch detection method by Vincent et al. (2010) is based on non-negative matrix factorisation (NMF) and operates by decomposing an input time-frequency representation as a series of basis spectra (representing pitches) and component activations (indicating pitch activity across time). This method models each basis spectrum as a weighted sum of narrowband spectra representing a few adjacent harmonic partials, enforcing harmonicity and spectral smoothness. As input time-frequency representation, an Equivalent Rectangular Bandwidth (ERB) filterbank is used. Since the method relies on a dictionary of (hand-crafted) narrowband harmonic spectra, system parameters remain the same for the two evaluation configurations.

The multiple-instrument transcription method by Benetos and Dixon (2012) is based on shift-invariant PLCA (a convolutive and probabilistic counterpart of NMF). In this model, the input time-frequency representation is decomposed into a series of basis spectra per pitch and instrument source which are shifted across log-frequency, thus supporting tuning changes and frequency modulations. Outputs include the pitch activation distribution and the instrument source contribution per pitch. Contrary to the parametric model of Vincent et al. (2010), the basis spectra are pre-extracted from isolated musical instrument sounds. As in the proposed method, the input time-frequency representation of Benetos and Dixon (2012) is the CQT. For the investigations with MLMs (Configuration 1), the PLCA models are trained on iso-

lated sound examples from all 9 piano models from the MAPS database (in order for the experiments to be comparable with the proposed method). For the second set of experiments which investigate the generalisation capabilities of the models (Configuration 2), the PLCA acoustic model is trained on isolated sounds from the synthesised pianos and tested on recordings created using the Yamaha Disklavier piano.

4.1.5.5 Results

In this section we present results from the experiments on the MAPS dataset. As mentioned before, all results for Configuration 1 are the mean values of various metrics computed over the 4 different train/test splits. On the other hand, Configuration 2 has only 1 train/test partition. The acoustic models yield a sequence of probabilities for the individual pitches being active (posteriograms). The post-processing methods are used to transform the posteriograms to a binary piano-roll representation. The various performance metrics (both frame and note based) are then computed by comparing the outputs of the systems to the ground truth.

We consider 3 kinds of post-processing methods. The simplest post-processing method is to apply a threshold to the output pitch activities obtained from the acoustic model. We select the threshold that maximises the F-measure over the entire training set and use this threshold for testing. Pitch activities that are greater than the threshold are set to 1, while the remaining pitch activities are set to 0. The second post-processing method considered uses individual pitch HMMs for post-processing similar to Poliner and Ellis (2007). The HMM parameters (transition probabilities, pitch marginals) are obtained by counting the frequency of each event over the MIDI ground truth data. The binary pitch activities are obtained using Viterbi decoding (Rabiner, 1989), where the scaled likelihoods are used as emission probabilities. Finally, we combine the acoustic model predictions with the RNN-NADE MLMs and obtain binary transcriptions using beam search.

For all the results presented in this section, the table entry Benetos represents the PLCA acoustic model by Benetos and Dixon (2012) while the entry Vincent represents the NMF based acoustic model by Vincent et al. (2010). In Table 4.3, we present F-scores (both frame and note based) for all the acoustic models and the 3 post-processing methods using Configuration 1. From the table, we note that all the neural network models outperform the PLCA and NMF models in terms of frame-based F-measure by $3\% - 9\%$. The DNN and RNN acoustic model performances are similar, while the ConvNet acoustic model clearly outperforms all the other models. The ConvNets yield an absolute improvement of $\sim 5\%$ over the other neural network models, while outperforming the spectrogram factorisation models by $\sim 10\%$ in frame-wise F-measure. For the note-based F-measure, the RNN and ConvNet models perform better than the DNN acoustic model. This is largely due to the fact that these models include context information in their inputs, which implicitly smooths the output predictions. With regards to the ConvNet acoustic models, from Table 4.2 we note that the filters for the ConvNets are relatively long. This is in contrast to filter shapes used for processing images, which are typically small square filters for e.g. 3×3 or 5×5 (Lin et al., 2013; Szegedy et al., 2015). As mentioned before (Section 2.1.1), this is due to the fact individual pitches are composed of a fundamental frequency and a series of harmonics or overtones. Therefore, pitches aren't localised within a narrow region along the frequency axis and consequently we we observe that longer filters yield better performance compared to small square filters.

From Rows 1 and 2 of Table 4.3 we observe that the RNN-NADE MLM yields a performance increase for the PLCA and NMF acoustic models, though the improvement is small, $\sim 1\%$ F-measure. This might be due to the fact that unlike the neural network models, these models are not trained to maximise the conditional probability of output pitches given the acoustic inputs. Another contributing factor is the fact that the PLCA and NMF posteriograms represent the energy distribution

Post Processing	Thresholding		HMM		Hybrid Architecture	
Acoustic Model	Frame (%)	Note (%)	Frame (%)	Note (%)	Frame (%)	Note(%)
Benetos	64.20	65.22	64.84	66.05	65.10	66.48
Vincent	58.95	**68.50**	60.37	**68.87**	59.78	**69.00**
DNN	67.54	60.02	68.32	62.26	67.92	63.18
RNN	68.38	63.84	68.09	64.50	69.25	65.24
ConvNet	**73.57**	65.35	**73.75**	66.20	**74.45**	67.05

Table 4.3: F-measures for multiple pitch detection on the MAPS dataset, using evaluation configuration 1.

	$\mathcal{P}$		$\mathcal{R}$		$\mathcal{A}$	
Acoustic Model	Frame (%)	Note (%)	Frame (%)	Note (%)	Frame (%)	Note (%)
Benetos	59.54	73.51	69.51	60.67	48.47	49.03
Vincent	52.71	**79.93**	69.04	60.69	43.04	**52.92**
DNN	65.66	62.62	70.34	63.75	51.76	45.33
RNN	67.89	64.64	70.66	65.85	54.38	48.18
ConvNet	**72.45**	67.75	**76.56**	**66.36**	**58.87**	50.07

Table 4.4: Precision, Recall and Accuracy for multiple pitch detection on the MAPS dataset using the hybrid architecture ($w = 10, K = 4, k = 2, f_h(y_0^t) = y_t$), using evaluation configuration 1.

over pitches rather than explicit pitch probabilities, which results in many activations being greater than 1. This discrepancy in the *scale* of the acoustic and language predictions leads to an unequal weighting of predictions when used in the hybrid RNN framework. In Table 4.3 we observe that the acoustic model by Vincent et al. (2010) outperforms all other acoustic models on the note-based F-measure, while the frame based F-measure is significantly lower. This can be attributed to the use of an ERB filter-bank input representation, which offers improved temporal resolution over the CQT for lower frequencies. Recall that the reported note onset time must lie within the $\pm 50ms$ window around the ground truth onset for it to be a correct match (Section 4.1.5.2). Consequently the neural network acoustic models (especially the ConvNets) are able to achieve high frame-based scores, while the note-based scores are lower due to the temporal resolution of the input representation.

Acoustic Model	Benetos	Vincent	DNN	RNN	ConvNet
F-measure (Frame) (%)	59.31	59.60	59.91	57.67	**64.14**
F-measure (Note) (%)	54.29	**59.12**	49.43	49.20	54.89

Table 4.5: F-measures for acoustic models trained on synthesised pianos and tested on real recordings (evaluation configuration 2).

In Table 4.4, we present additional metrics (precision, recall and accuracy) for the all the acoustic models after decoding with an RNN-MLM, using Configuration 1. We observe that the NMF and PLCA models have low frame-based precision and high recall and the converse is true for the note-based metrics. For the neural network models, we observe smaller differences between the precision and recall values in both frame-based and note-based cases. Amongst all the neural network models, we observe that the ConvNet outperforms all the other models on all the metrics.

In Table 4.5, we present F-measures for experiments where the acoustic models are trained on synthesised data and tested on real data (Configuration 2). We compare the pitch activation probabilities against learnt thresholds to make a binary classification decision. From the table we note that frame based F-measure for the DNN and RNN models is similar to the PLCA model and the model by Vincent et al. (2010). We note that the ConvNet outperforms all other models on the frame-based F-measure by $\sim 5\%$. On the note based evaluations, we observe that both RNN and DNN are outperformed by all the other models. The ConvNet performance is similar to the PLCA model, while the acoustic model from Vincent et al. (2010) again has best performance on the note based metrics. We note that the performance of all models $\sim 10\%$ worse than for Configuration 1.

We now discuss details of the inference algorithm. The high dimensional hashed beam search algorithm has the following parameters: the beam width w, the branching factor K, number of entries per hash table entry k and the similarity metric f_h (Algorithm 2). We observed that a value of $K \geq 4$ produced good results. Larger

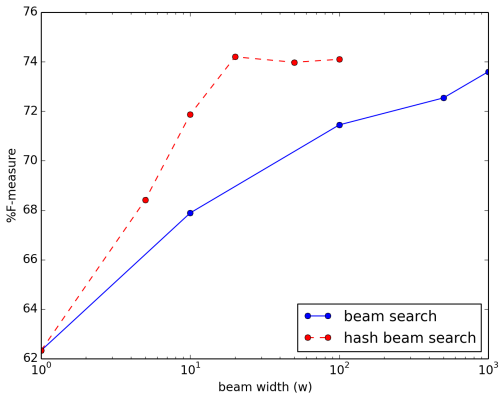
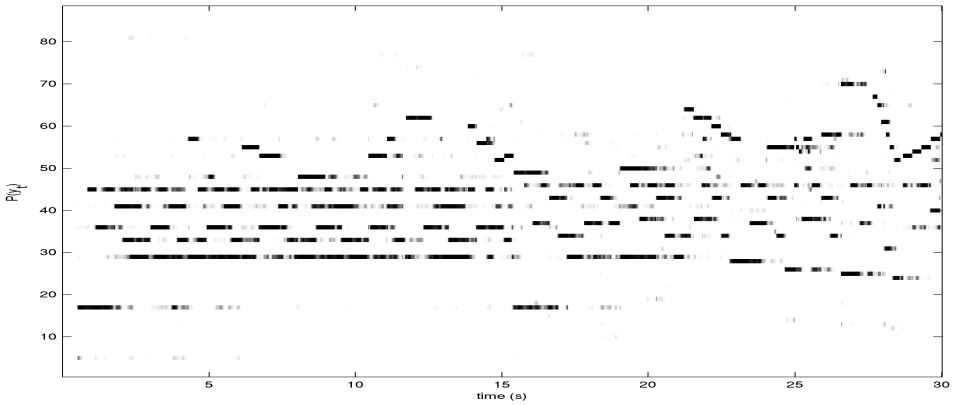


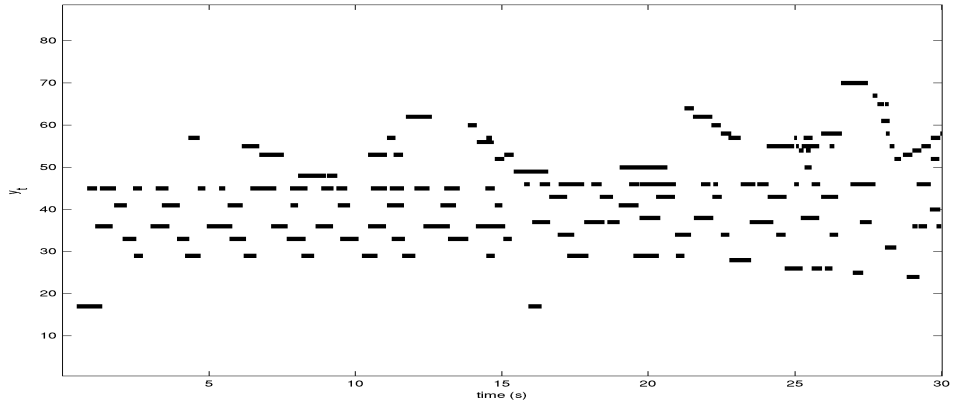
Figure 4.4: Effect of beam width (w) on F-measure. $k = 2, K = 4, f_h = y_t$

values of K do not yield a significant performance increase and result in much longer run times, therefore we set $K = 4$ for all experiments. We observed that small values of k (number of solutions per hash table entry), $1 \leq k \leq 4$ produced good results. Decoding accuracies deteriorate sharply for large values of k, as observed by Sigtia et al. (2015b). Therefore, we set the number of entries per hash key $k = 2$ for all experiments. We let the similarity metric be the last n emitted symbols, $f_h(y_0^t) = y_{t-n+1}^t$. We experimented with varying the values of n and observed that we were able to achieve good performance for small n, $1 \leq n \leq 5$. We did not observe any performance improvement for large n, therefore for all experiments we fix $f_h(y_0^t) = y_t$. Figure 4.4 is a plot showing the effect of beam width w on transcription performance. The results are average values of decoding accuracies over 4 splits. We compare performance of the hashed beam search with the high dimensional beam search by Sigtia et al. (2015a). From Figure 4.4 we observe that the hashed beam search algorithm is able to achieve performance improvement with significantly smaller beam-widths. For instance, the high dimensional beam search algorithm takes 20 hours to decode the entire test set with $w = 100$, while the hashed beam search takes 22 minutes with $w = 10$ and achieves better decoding accuracy.

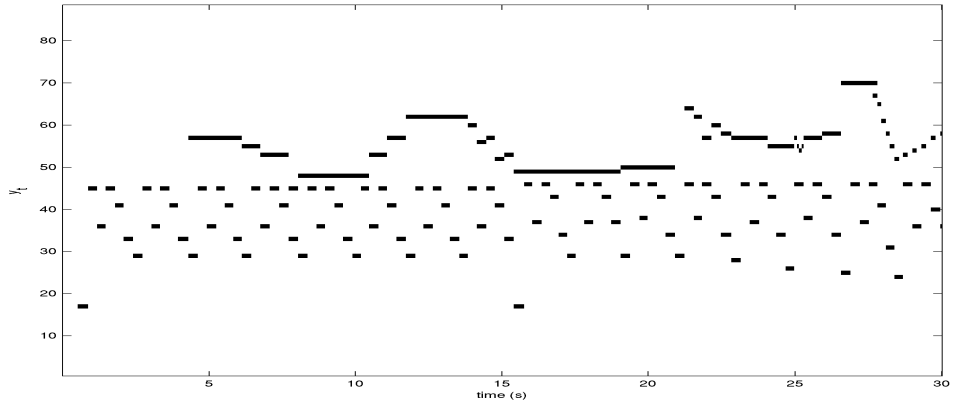
Figure 4.5 is a graphical representation of the outputs of a ConvNet acoustic



(a) ConvNet Posteriogram



(b) ConvNet Transcription



(c) Ground Truth

Figure 4.5: (a) Pitch-activation (posteriogram) matrix for the first 30 seconds of track MAPS_MUS-chpn_op27_2_AkPnStgb produced by a ConvNet acoustic model. (b) Binary piano-roll transcription obtained from posteriogram in a) after post processing with RNN MLM and beam search. (c) Corresponding ground truth piano roll representation.

model. We observe that some of the longer notes are fragmented and the offsets are estimated incorrectly. One reason for this is that the ground truth offsets don't necessarily correspond to the offset in the acoustic signal (due to effects of the sustain pedal and reverberations), implying noisy offsets in the ground truth. We also observe that the model does not make many harmonic errors in its predictions.

4.1.6 Discussion

In this set of experiments, we present a hybrid RNN model for AMT of polyphonic piano music. The model comprises a neural network acoustic model and an RNN based music language model. In addition to the DNN and RNN models we design a ConvNet architecture for AMT, which to the best of the author's knowledge has not been attempted before. Our experiments on the MAPS dataset demonstrate that the neural network acoustic models consistently outperform 2 popular acoustic models from the AMT literature on frame-based metrics, with the ConvNets clearly outperforming all other models. The neural network acoustic models are able to achieve better accuracy when applied directly to the input time-frequency representation of the audio, without the need for any feature engineering. Additionally, the neural network acoustic models have general processing architectures, unlike the models by Benetos and Dixon (2012) and Vincent et al. (2010), which are designed specifically for AMT and harmonic sounds. We note that the results also demonstrate the ability of the neural network acoustic models to generalise to new piano types. Although all acoustic models perform worse by $\sim 10\%$ F-measure, the ConvNets again outperform all other models. We also observe that the RNN MLMs improve performance on all evaluation metrics. The proposed inference algorithm with the hash beam search is able to yield good decoding accuracies with significantly shorter run times, making the model suitable for real-time applications.

We now discuss some of the limitations of the proposed model and identify direc-

tions for future work to improve the system. As discussed earlier, one of the main contributing factors to the success of deep neural networks has been the availability of very large datasets. However datasets available for AMT research are considerably smaller than datasets available in speech, computer vision and natural language processing (NLP). Therefore the applicability of deep neural networks for acoustic modelling is limited to datasets with large amounts of labelled data, which is not common in AMT (at least in non-piano music). Although the neural network acoustic models perform competitively, their performance could be further improved in many ways. Noise or deformations can be added to training examples to encourage the classifiers to be invariant to commonly encountered input transformations. Additionally, the CQT input representation can be replaced by a representation with higher temporal resolution (like the ERB or a variable-Q transform (Schörkhuber et al., 2014)), to improve performance on note based metrics.

Although we observe that the performance improvement due to MLMs is small, the abundance of musical score data and recent progress in NLP tasks with neural networks provide strong motivation for further investigations into MLMs for AMT. There are several limitations and open questions that remain. The MLMs are trained on binary vectors sampled from the MIDI ground truth. Depending on the sampling rate, most note events are repeated many times in this representation. The MLMs are trained to predict the next frame of notes, given an input sequence of binary note combinations. In cases where the same notes are repeated many times, log-likelihood can be trivially maximised by repeating previous inputs. This causes the MLM to perform a smoothing operation, rather than imposing any kind of musical structure on the outputs. A potential solution would be to perform beat-aligned language modelling for the training and the test data, rather than sampling the MIDI at some arbitrary sampling rate. Additionally, RNNs can be extended to include duration models for each of their pitch outputs, similar to second order HMMs. However,

this is a challenging problem and currently remains unexplored. It would also be interesting to encourage RNNs to learn long-term repeating patterns by interfacing RNN *controllers* with external memory units (Grefenstette et al., 2015) and also to incorporate a notion of timing or metre in the input representation for the MLMs.

The effect of tonality on the performance of the MLMs should be further investigated. The MLMs should ideally be invariant to transpositions of a musical piece to different pitches. The MIDI ground truth can be transposed to any tonality. MLMs can be trained on inputs with transposed tonalities or individual MLMs for each key can be trained. Additionally, the fully connected input layer of the RNN MLM can be substituted with a convolutive layer, with convolutions along the pitch axis to encourage the network to be invariant to pitch transpositions.

Another limitation of the proposed hybrid model is that the conditional probability in Equation 4.5 is derived by assuming that the predictions at time t are only a function of the input at t and independent of all other inputs and outputs. The violation of this assumption leads to certain factors being counted twice and therefore reduces the impact of the MLMs. The results clearly demonstrate that improvements with the MLM are maximum when the acoustic model is frame-based. The improvements are comparatively lower when combined with predictions from an RNN or ConvNet acoustic model. This is problematic since the ConvNet acoustic models yield the best performance.

Despite some of these limitations, the proposed model is able to outperform 2 state-of-the-art models on frame-based metrics and the results provide strong motivation for future research in both acoustic and music language modelling with neural networks.

4.2 Multi-Instrument Polyphonic Transcription

In the previous section, we presented a neural network based model for piano music transcription. The results demonstrate that the supervised neural network acoustic models outperform 2 state-of-the-art acoustic models from the literature. We also observed that the MLMs improve transcription results. In this section, we further investigate the effectiveness of MLMs for transcription. As previously mentioned, neural network acoustic models are not feasible for most instrument types due to scarcity of labelled training data. A lot of AMT research is therefore dedicated to developing unsupervised or NMF-based acoustic models (Vincent et al., 2010; Benetos and Dixon, 2012; Berg-Kirkpatrick et al., 2014). MLMs on the other hand can be trained on musical scores, without any need for labelling. In this section we present experiments that investigate the effectiveness of MLMs in a more realistic scenario, where the train and test data comprises more than one instrument and the acoustic model is a PLCA based model trained on multiple instruments. First, we present an alternative method for combining the predictions of a PLCA acoustic model (Benetos and Dixon, 2012) and an RNN MLM. We then investigate the performance of the proposed model on a dataset of multi-instrument polyphonic recordings.

4.2.1 Acoustic Model

For combining acoustic and music language information in an AMT context, we employ the model by Benetos and Dixon (2012), which supports the transcription of multiple-instrument polyphonic music and also supports pitch deviations and frequency modulations. The model is based on PLCA, which is a latent variable analysis method which has been used for decomposing spectrograms. For computational efficiency, we employ the fast implementation from Benetos et al. (2013), which utilises pre-extracted note templates that are also pre-shifted across log-frequency, in order

to account for frequency modulations or tuning changes. In addition, as was shown by Smaragdis and Mysore (2009), PLCA-based models can utilise priors for estimating unknown model parameters, which we use for incorporating information from the MLM into the acoustic model predictions.

The transcription model takes as input a normalised log-frequency spectrogram $x_t = \{x_{0,t}, \ldots, x_{\omega,t}, \ldots, x_{N-1,t}\}$ (where ω is the log-frequency index, t is the time index and N is the dimensionality of x_t) and approximates it as a bivariate probability distribution $P(\omega, t)$. $P(\omega, t)$ is decomposed into a series of log-frequency spectral templates per pitch, instrument, and log- frequency shifting (which indicates deviation with respect to the ideal tuning), as well as probability distributions for pitch, instrument, and tuning.

The model is formulated as:

$$P(\omega, t) = P(t) \sum_{y,f,s} P(\omega|s, y, f) P_t(f|y) P_t(s|y) P_t(y), \tag{4.9}$$

where y denotes pitch, s denotes the musical instrument source, and f denotes log-frequency shifting. $P(t)$ is the energy of the log-spectrogram, which is a known quantity. $P(\omega|s, y, f)$ denotes pre-extracted log-spectral templates per pitch y and instrument s, which are also pre-shifted across log-frequency. The pre-shifting operation is made in order to account for pitch deviations, without needing to formulate a convolutive model across log-frequency. $P_t(f|y)$ is the time-varying log-frequency shifting distribution per pitch, $P_t(s|y)$ is the time-varying source contribution per pitch, and finally, $P_t(y)$ is the pitch activation, which essentially is the resulting music transcription. As a time-frequency representation in the log-frequency domain we use the constant-Q transform (CQT) with a log-spectral resolution of 60 bins/octave (Schörkhuber and Klapuri, 2010).

The unknown model parameters ($P_t(f|y)$, $P_t(s|y)$, and $P_t(y)$) can be iteratively es-

timated using the expectation-maximisation (EM) algorithm (Dempster et al., 1977). For the *Expectation* step, the following posterior is computed:

$$P_t(s, y, f|\omega) = \frac{P(\omega|s, y, f)P_t(f|y)P_t(s|y)P_t(y)}{\sum_{y,f,s} P(\omega|s, y, f)P_t(f|y)P_t(s|y)P_t(y)}. \tag{4.10}$$

For the *Maximization* step (without using any priors) unknown model parameters are updated using the posterior computed from the Expectation step:

$$P_t(f|y) \propto \sum_{\omega,s} P_t(y, f, s|\omega)x_{\omega,t}, \tag{4.11}$$

$$P_t(s|y) \propto \sum_{\omega,f} P_t(y, f, s|\omega)x_{\omega,t}, \tag{4.12}$$

$$P_t(y) \propto \sum_{\omega,f,s} P_t(y, f, s|\omega)x_{\omega,t}. \tag{4.13}$$

We consider the sound state templates to be fixed, so no update rule for $P(\omega|s, y, f)$ is applied. Using fixed templates, 20-30 iterations using these update rules are sufficient for convergence. The output of the system is a pitch activation which is scaled by the energy of the log-spectrogram:

$$P_{PLCA}(y, t) = P(t)P_t(y). \tag{4.14}$$

After performing 5-sample median filtering for note smoothing, thresholding is performed on $P_{PLCA}(y, t)$ followed by minimum note duration pruning set to 40ms in order to convert $P_{PLCA}(y, t)$ into a binary piano-roll representation, which is the output of the transcription system, and is also used for evaluation purposes.

4.2.2 Music Language Models

We use the RNN-NADE and the generative RNN architecture as MLMs (Section 4.1.3). We choose to include the generative RNN in comparisons for the following reason. In this study, the number of examples used to train the MLMs is much larger than the experiments with piano music transcription (Section 4.2.4.1). It has been previously shown that RNNs trained with HF optimisation and large mini-batches of sequences perform well on a music prediction task (Martens, 2010; Martens and Sutskever, 2011). In preliminary experiments, we observed that RNNs trained with HF optimisation performed comparatively to RNN-NADEs trained with HF (Table 4.6).

4.2.3 Proposed Model

In this section, we describe the proposed method for combining the PLCA acoustic model with the music language model. Before the proposed system is described, it must be noted that the PLCA acoustic model and the RNN MLM are trained independently with the EM algorithm and gradient descent, respectively. The proposed model provides a means to combine the predictions from 2 independent models which are optimised for different objectives, using different datasets.

Firstly, the input music signal is transcribed using the process described in Section 4.2.1. The resulting piano-roll representation of the transcription is considered to be a sequence $y_0^T = \{y_0, y_1, y_2, \ldots, y_T\}$ that is placed as input to the MLMs (Figure 4.2). For the RNN-NADE, we compute the conditional pitch probabilities $P_t(y_i|y_{<i})$ (Equation 3.37) for all time frames t and pitch indices i, and use that as prior information for the combined model, with the prior information denoted as $P_{MLM}(y,t)$, where $P_{MLM}(y=i,t) = P_t(y_i|y_{<i})$. For the RNN, the prediction output is directly denoted as $P_{MLM}(y,t)$, since pitch probabilities are independent.

As suggested by Smaragdis and Mysore (2009), PLCA-based models use multino-

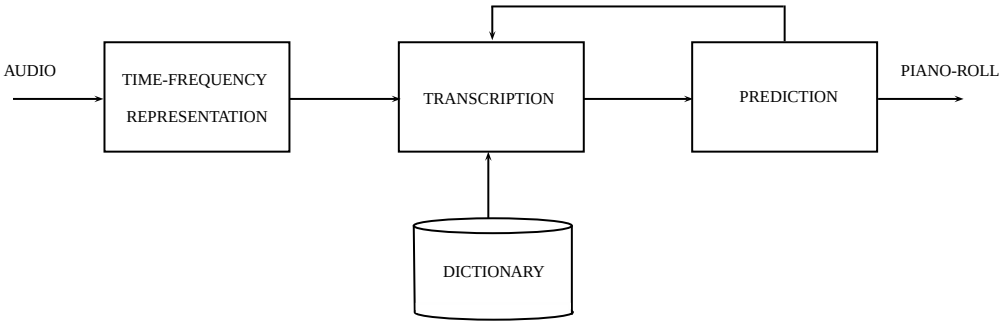


Figure 4.6: Proposed system diagram.

mial distributions; since the Dirichlet distribution is conjugate to the multinomial, a Dirichlet prior can be used to enforce structure on the pitch activation distribution $P_t(y)$ obtained from the acoustic model. Following the procedure of Smaragdis and Mysore (2009), we define the Dirichlet hyperparameter for the pitch activation as:

$$\alpha_t(y) \propto P_{PLCA}(y,t)P_{MLM}(y,t) \tag{4.15}$$

where $\alpha_t(y)$ essentially is a pitch activation probability which is filtered through a pitch indicator function computed from the symbolic prediction step (the denominator is simply for normalisation purposes and is ignored).

The recording is then re-transcribed, using as additional information the prior computed from the transcription step. The modified update for the pitch activation which replaces (4.13) is given by:

$$P_t(y) \propto \sum_{\omega,f,s} P_t(y,f,s|\omega)x_{\omega,t} + \kappa\alpha_t(y) \tag{4.16}$$

where κ is a weight parameter expressing how much the prior should be imposed; as shown by Smaragdis and Mysore (2009), the weight decreases from 1 to 0 throughout the iterations. To summarise, the initial transcription creates a symbolic prediction, which in turn improves the subsequent re-transcription of the music signal. An

overview of the complete transcription-prediction system architecture can be seen in Figure 4.6.

4.2.4 Evaluation

4.2.4.1 Dataset

For testing the transcription system, we employ the Bach10 dataset (Duan et al., 2010a), which is a freely available multi-track collection of multiple-instrument polyphonic music. It consists of ten recordings of J.S. Bach chorales, performed by violin, clarinet, saxophone, and bassoon. Pitch ground truth for each instrument is also provided. Due to the tonal and homogeneous content of the dataset (single composer, single music language), it is suitable for testing the incorporation of music language models in a multiple-instrument transcription system. For training the transcription system, pre-extracted and pre-shifted spectral templates are extracted for the instruments present in the dataset, using isolated note samples from the RWC database (Goto et al., 2003).

For training the MLMs we use the Nottingham dataset[5], a collection of 1200 folk music pieces in symbolic ABC format, which contain simple chord combinations and tunes. We trained the RNN and the RNN-NADE models using both Stochastic Gradient Descent (SGD) and HF to compare performance. The inputs to both the models are sequences of length 200 where each frame of the sequence is a binary vector of length 88 which covers the full piano note range. As before, both the RNN and the RNN-NADE are trained to maximise the likelihood of the next vector given a sequence of input vectors (Section 4.1.3).

[5] `ifdo.ca/~seymour/nottingham/nottingham.html`

Model	$\mathcal{P}$
RNN (SGD)	67.89%
RNN (HF)	69.61%
RNN-NADE (SGD)	68.89%
RNN-NADE (HF)	**70.61**%

Table 4.6: Validation results for MLMs

4.2.4.2 Metrics

We use note based precision ($\mathcal{P}$), recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) to evaluate the performance of the system (Section 4.1.5.2). As in the public evaluations on multi-pitch detection carried out through the MIREX framework (MIR), a detected note is considered correct if its pitch is the same as the ground truth pitch and its onset is within a 50ms tolerance interval of the ground-truth onset.

4.2.4.3 Results

To validate the performance of the MLMs, we calculate the prediction precision on unseen sequences of music from the Nottingham dataset of folk melodies. We utilise the same training/validation/test split as Boulanger-Lewandowski et al. (2012)[6] in order to compare results. For both the RNN and RNN-NADE models we sample 10 vectors from the conditional distribution at each time-step and calculate the expected precision against the ground truth. The reported precision is found by finding the mean over the predictions over all frames. Table 4.6 shows the results of the validation experiments. These results are of the same order as the prediction accuracies reported by Boulanger-Lewandowski et al. (2012). We found that for both the models, HF optimization gave better precision than SGD. The RNN models had a hidden layer of size 150, while the RNN-NADE models had a recurrent hidden layer of size 100 and the NADE hidden layer comprised 150 units.

Multi-pitch detection experiments are performed using the proposed system, with

[6] `http://www-etud.iro.umontreal.ca/~boulanni/icml2012`

Configuration	$F(\%)$	$Pre(\%)$	$Rec(\%)$
Configuration 1	62.02	58.51	66.12
Configuration 2 - RNN-NADE (SGD)	62.62	59.70	65.92
Configuration 3 - RNN-NADE (SGD)	64.08	61.96	66.44
Configuration 2 - RNN (SGD)	62.29	59.08	65.98
Configuration 3 - RNN (SGD)	63.85	61.14	66.90
Configuration 2 - RNN-NADE (HF)	62.20	59.14	65.68
Configuration 3 - RNN-NADE (HF)	**65.16**	**62.80**	**67.78**
Configuration 2 - RNN (HF)	62.44	59.28	66.07
Configuration 3 - RNN (HF)	62.87	60.03	66.11

Table 4.7: Note-based transcription results using various system configurations.

various configurations. A first configuration only considers the PLCA transcription system from Section 4.2.1. A second configuration takes the output of the transcription system and gives it as input to the prediction system of Section 4.2.3, where the final piano-roll is the output of the prediction step. A third configuration (presented in Section 4.2.3), re-transcribes the recording, having the prediction as a prior information for estimating the pitch activations. For the prediction system, experiments were performed using both the RNN-NADE and the RNN.

Note-based results using the various system configurations are displayed in Table 4.7. It can be seen that the best performance is achieved by the 3rd configuration when using the NADE-HF model for prediction, which surpasses the acoustic-only transcription system by more than 3%. In general, it can be seen that using the prediction system as a post-processing step (2nd configuration) always leads to an improvement over the acoustic-only model (1st configuration). A similar trend can be observed when integrating the prediction information as a prior in the transcription system (configuration 3) compared to just using the prediction system as post-processing (configuration 2); an improvement is always reported. Another observation can be made when comparing the RNN-NADE with the RNN, with the former providing a clear improvement.

Qualitatively, the MLMs are able to improve transcription performance by pro-

viding a rough estimate of which pitches are expected to appear in the recording (and which pitches are not expected to appear). The language models were trained using simple chord sequences (from the Nottingham dataset) that are representative of simple tonal music and are applicable as language models to the more complex Bach chorales.

As an example of the proposed system's performance, the spectrogram and raw output of the transcription-prediction system using the 3rd configuration is displayed for a recording from the Bach10 dataset in Figure 4.7, whereas the post-processed transcription output along with the ground truth for the same recording is shown in Figure 4.8.

For frame-based evaluation, the NADE-HF using Configuration 3 yields 74.3% F-measure. The method by Duan et al. (2010a) (where the Bach10 dataset was first introduced) yields 69.7% F-measure (with unknown polyphony). Therefore we observe an improvement $\sim 4\%$ when compared to existing methods on frame-based F-measure.

4.2.5 Discussion

In this set of experiments, we evaluate a system for automatic music transcription which incorporates prior information from a polyphonic music prediction model based on recurrent neural networks. The acoustic transcription model is based on probabilistic latent component analysis and the predictions from the MLM are incorporated in the PLCA acoustic model using Dirichlet priors. Experimental results using the multiple-instrument Bach10 dataset show that there is a clear improvement (3% in terms of note-based F-measure) due to the RNN MLM. These results also demonstrate that the MLM can be trained on symbolic music data from a different source as the acoustic data and still improve transcription performance, thus eliminating the need to acquire collections of symbolic and corresponding acoustic data (which are

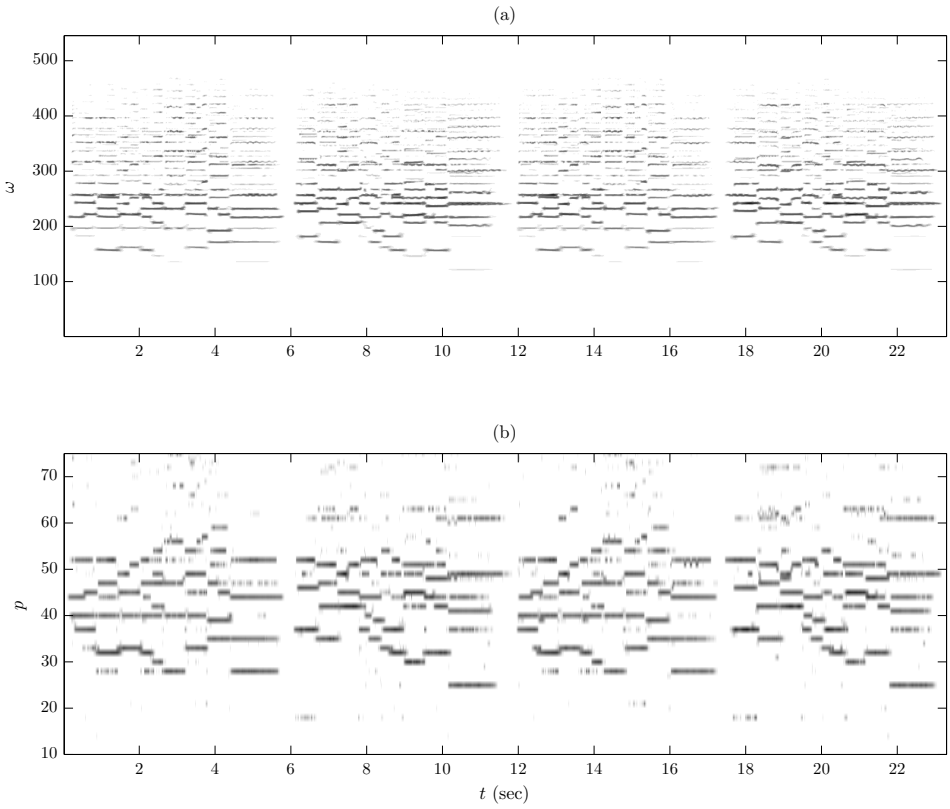


Figure 4.7: (a) The spectrogram $x_{\omega,t}$ for recording "Ach Lieben Christen" from the Bach10 dataset. (b) The pitch activation $P(y,t)$ using the transcription-prediction system using the 3rd configuration, with the NADE-HF.

scarce).

In the current system, the language models are trained on only one dataset. In the future, we would like to evaluate the proposed system using language models trained from multiple different sources to see if this helps the MLMs generalise better. The MLMs presented here have exactly the same structure as the MLMs used for piano music transcription and consequently the shortcomings discussed in Section 4.1.6 also apply here. The MLMs are trained by sampling the MIDI scores at some frame-rate and the model does not have any explicit information about timing. As argued before, a potential solution is to append timing information (like position in the bar) to the piano-roll representation of the score. Another possible solution is to use a transcription system to obtain the sequence of notes in the piece without

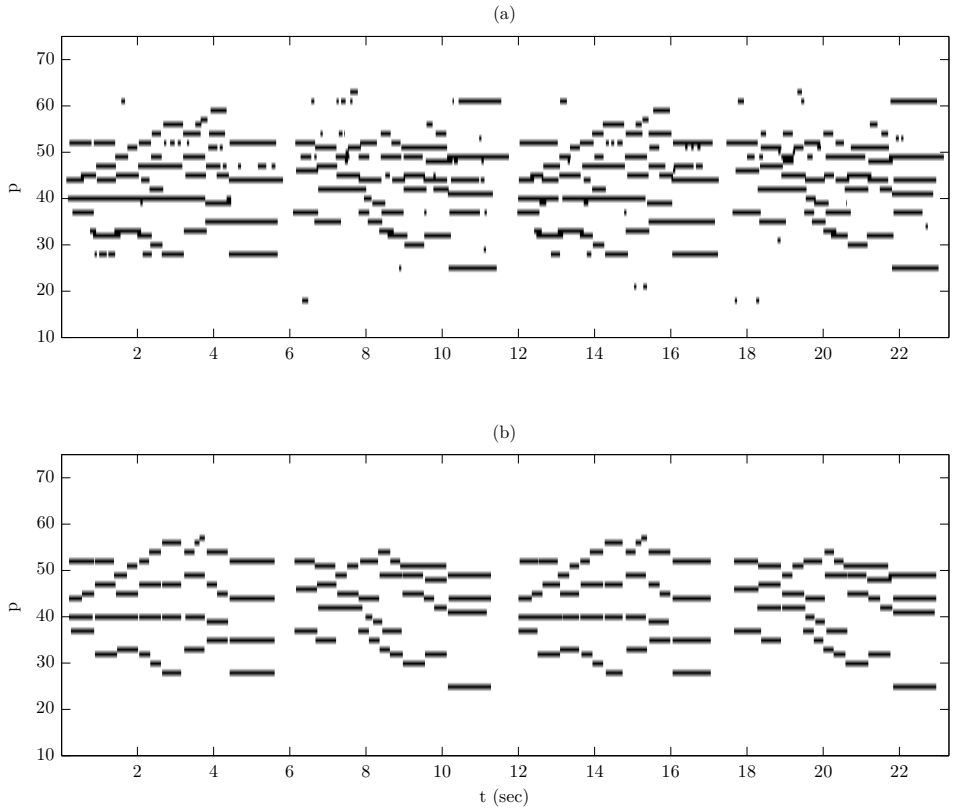


Figure 4.8: Transcription example for recording "Ach Lieben Christen" from the Bach10 dataset. (a) The post-processed output of the transcription-predicton system using the 3rd configuration, with the NADE-HF. (b) The pitch ground truth of the recording.

any alignment, for example with a method like connectionist temporal classification (CTC) (Graves et al., 2006). This is analogous to the problem of speech recognition where the output is a sequence of words or phonemes. The system can then be used to *infer* the alignments of labels with input frames. This can be further combined with a separate onset detection system to accurately estimate the note-onsets.

4.3 Conclusions

In this chapter, we presented results from 2 sets of experiments with polyphonic AMT. The experiments with piano music transcription demonstrate that neural network

acoustic models can outperform existing state-of-the-art AMT systems on frame-based metrics, given sufficient training data. The neural network acoustic models can be directly applied to classify the input time-frequency representation, eliminating the need for problem specific feature extraction. The experiments also demonstrate that music language models can help improve transcription performance. We investigate how the acoustic models and MLMs can be combined and discuss ways to improve the performance of MLMs for AMT tasks. Furthermore, we present a general modification to the beam search algorithm, which helps reduce decoding times by an order of magnitude. The proposed beam search algorithm is general and can be applied to other sequential search problems.

We further investigate MLMs by using them for multi-instrument transcription. The proposed method using Dirichlet priors helps improve F-measure by 3%, a clear improvement over previous results. These results show that MLMs and acoustic models can be trained on disjoint datasets and the MLMs can still improve transcription similar to language models in speech. This is a useful result, since training labelled training data for AMT is scarce and MLMs can be trained on musical scores and MIDI music obtained from the internet.

In the next chapter, we apply the ideas developed here to the problem of automatic chord transcription.

Chapter 5

Automatic Chord Transcription

In the previous chapter we proposed a neural network based model for AMT. In this chapter, we extend the ideas developed in Chapter 4 for automatic chord transcription. We investigate 2 aspects of ACT: the audio signal analysis and the temporal modelling of chord sequences. In addition to directly classifying the input time-frequency representation, we also present results of experiments with feature learning. This chapter is organised as follows: we first present the details of the proposed model and the inference algorithm. The next section contains details of the preprocessing, followed by a preliminary investigation of model performance. Finally, we present results with the experiments on feature learning and the inference algorithm.

5.1 Proposed Model

In this section we describe the proposed neural network model for ACT. We use the hybrid RNN architecture (Section 4.1.4.1) for transcribing chords. The model comprises an acoustic model and a *chord language model*. The motivations for using the hybrid model are twofold. Firstly, we use neural network acoustic models to estimate the probabilities of chords given input feature frames. Using neural network models allows us to jointly learn the acoustic features and the classifier from data. Neu-

ral acoustic models also allow us to experiment with different architectures (DNNs, RNNs, ConvNets). Secondly, the RNN chord language model learns the temporal structure in sequences of chord labels. RNNs generalise the first-order assumption made by HMMs and dynamic Bayesian networks and can in theory learn more general distributions over sequences of chord labels (Section 4.1.4.1).

5.1.1 Acoustic Model

In this section we describe the neural network acoustic models used in the proposed ACT system. The neural networks are used to obtain a posterior distribution $P(y_t|x_t)$ over the chord labels y_t given an acoustic observation x_t at some time t. Unlike AMT, only one chord label can be assigned to each observation and therefore the output distribution is a multinomial distribution over chord labels. The activations of the final layer are therefore passed through a *softmax* function (Section 3.3.2) as compared to a sigmoid function in Chapter 4.

5.1.1.1 Input Representation

We use the CQT as input representation for the acoustic models (Figure 5.1). The CQT has the advantage that it yields a lower dimensional representation compared to the STFT. It also yields a representation that is linear in pitch. This is a useful property that can be exploited by ConvNets to learn classifiers that are invariant to octave transpositions in pitch. We experimented with various configurations and finally determined that a CQT representation calculated over 7 octaves with 24 bins per octave yielded the best acoustic model performance. The 168-dimensional representation is further processed by subtracting the mean and normalising by the standard deviation calculated over the training set, for each dimension independently.

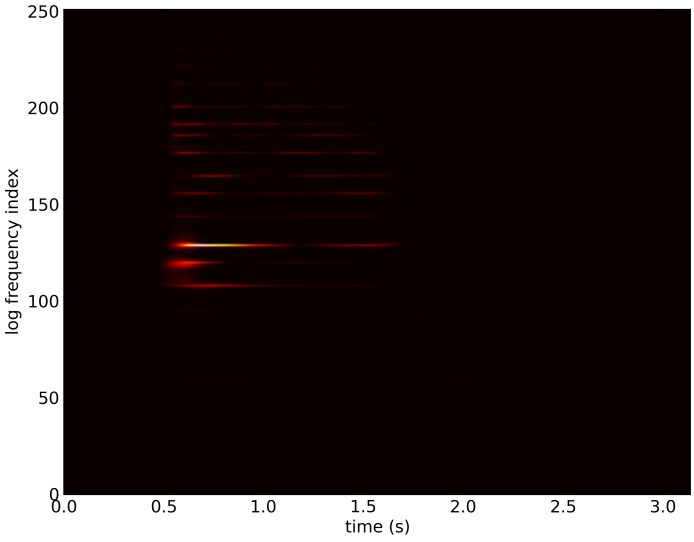


Figure 5.1: CQT representation of a C-major chord played on a piano.

5.1.1.2 Neural Network Architectures

We experiment with 3 different neural network architectures for acoustic modelling:

- **DNNs**: Feed-forward acoustic models act on individual frames of input features to yield a multinomial distribution over output chord labels. Given a frame x_t at any time t, the DNNs yield an output distribution $P(y_t|x_t)$.

- **RNNs**: We use RNN acoustic models as an alternative to DNN acoustic models. RNN acoustic models have the property that they can incorporate long-term dependencies while making predictions, due to the recursive hidden state (Equation 3.5). Since the size of the dataset available for training the acoustic models for ACT is much larger than previous experiments with AMT (Section 5.2.1), we experiment with RNNs with LSTM units for acoustic modelling (Section 3.3.3.3). In preliminary experiments, we observed that LSTM RNNs yielded better generalisation performance than RNNs with standard hidden units (Section 3.2.3). Given an input sequence x_0^t, at any time t the RNN yields a distribution $P(y_t|x_0^t)$.

- **ConvNets**: We use ConvNet acoustic models to classify 2-dimensional windows over the time-frequency representation. ConvNets have desirable properties for acoustic modelling like weight sharing and translation invariance over one or both axes depending on the architecture (Section 3.2.2). Given a 2-dimensional input x_{t-k}^{t+k+1}, ConvNets yield a posterior distribution $P(y_t|x_{t-k}^{t+k+1})$ over the chord labels.

All the acoustic models are trained with gradient descent to jointly optimise the features and the chord classifier.

5.1.1.3 Feature Learning

In addition to classifying the time-frequency input representation with neural networks, we also experiment with *feature learning* as an additional step in the acoustic modelling pipeline. Neural networks perform non-linear transformations of the inputs to each layer. By composing many such operations, complex transformations can be learnt from data. The resulting activations at each layer can be regarded as features learnt by the network.

Recently, there have been several studies in MIR that demonstrate that better classification accuracies can be achieved by using the features learnt by a neural network as inputs to a classifier (Hamel and Eck, 2010; Boulanger-Lewandowski et al., 2013a; Sigtia and Dixon, 2014). In this study, we experiment with using a DNN to learn features which are then input to a neural network classifier[1].

5.1.2 Chord Language Model

Similar to language, chord sequences exhibit complex temporal structure. We train RNN language models to learn distributions over sequences of chord labels (Section

[1]Similar experiments with AMT did not yield any improvements in performance. We suspect this is due to the considerably smaller dataset available for the piano transcription problem in Chapter 4.

3.4.3). The sequences of chord labels are obtained by sampling the ground-truth at the same rate as feature extraction from the corresponding audio. We found that the transcription accuracies were improved if we replaced the simple hidden units of an RNN with LSTM units (Section 3.3.3.3).

5.1.3 Hybrid RNN

The predictions from the acoustic model and language model are combined using the hybrid architecture (Section 4.1.4.1). The acoustic and language models are trained independently. This has the advantage that the chord language models can be trained on chord transcription data available on the internet from various sources [2], without the need for the corresponding audio.

5.1.4 Inference

At test time, we would like to estimate the mode of the distribution $P(y|x)$, where x is the input sequence and y is the output sequence of chord labels. From Section 4.1.4.2, we note that the RNN language model conditions output probabilities at any time t on *all* past predictions. Say the vocabulary of chord labels is of size N and we have a sequence of length T. Exhaustively searching for the best output sequence would be $O(N^T)$, which is intractable. Usually, beam search is used to obtain *estimates* for the output sequence y. Beam search scales linearly with the length of the sequence, which makes it an attractive choice when decoding input sequences of unknown length at test time. At any intermediate time-step t during search, beam search maintains a maximum of w partial solutions, which correspond to sequences of length t. Beam search proceeds by enumerating all possible next step predictions for the w partial solutions, sorting them in decreasing order of log-likelihood and then retaining the w top sequences of length $t + 1$ for further search. Algorithm 4 describes the beam

[2] e.g: `www.ultimate-guitar.com`

search algorithm used for estimating chord sequences.

Algorithm 4 Beam Search

Find the most likely sequence y given x with a beam width w.
$beam \leftarrow$ new beam object
$beam$.insert$(0, \{\})$
for $t = 1$ to T **do**
  $new\_beam \leftarrow$ new beam object
  for (l, s) **in** $beam$ **do**
    for y **in** C **do**
      $l' = \log P_{lm}(y|s)P_{am}(y|x_t) - \log P(y)$
      new_beam.insert$(l + l', \{s, y\})$
  $beam \leftarrow new\_beam$
return $beam$.pop()

In Algorithm 4, the beam object is a priority queue. It should be noted that Algorithm 4 and Algorithm 1 in Chapter 4 differ in the way candidate solutions at every step are generated. For ACT, the output distribution is a multinomial distribution over all possible chord labels. At any time, only one chord label can be assigned to a frame. This is in contrast to the polyphonic AMT case where multiple notes can be sounding at any time. For a piano with 88 keys, there are 2^{88} possible note combinations. The combinatorially large output space for AMT is dealt with by enumerating a fixed number of candidates at each step (Section 4.1.4.2). For ACT, the number of possible outputs at each step is limited to N, the size of the chord vocabulary. Therefore in Algorithm 4, we can enumerate all possible candidate solutions for each beam entry and then keep the top w solutions. The time complexity of Algorithm 4 is $O(NTw \log w)$, where w is number of solutions in the beam or *beam width*. The complexity of beam search increases linearly with N, T, which is desirable. The complexity is proportional to $w \log w$, where w is the beam width. Therefore increasing the beam width incurs a greater computational cost while decoding as compared to increasing the the length of the sequence or the vocabulary size.

As discussed in Section 4.1.4.2, the beam can get saturated with quasi-identical solutions when decoding long sequences. The priority queue in Algorithm 4 can be

replaced with the hashed beam object described in Chapter 4, Algorithm 3. The hashed beam search allows the algorithm to maintain *diversity* in the solutions.

5.2 Evaluation

In this section we describe the experiments to evaluate the performance of the proposed system.

5.2.1 Dataset

Unlike other approaches to chord estimation, our proposed approach aims to learn the audio features, the acoustic model and the language model from the training data. Therefore, maximum likelihood training of the acoustic and language models requires a large dataset for training. Additionally, we require the raw audio for all the examples in the dataset in order to train the acoustic model which operates on CQTs extracted from the audio.

The first dataset made available for ACT was the Beatles dataset with annotations for 180 tracks (Harte, 2010). The dataset was later expanded to include tracks by Queen and Zweieck (Mauch et al., 2009). The combined dataset contains annotations for 217 tracks[3]. The Billboard dataset[4] from McGill University provides annotations for 740 tracks (Burgoyne et al., 2011). The Billboard dataset also contains at least 197 unreleased annotations which are used as the unseen test set in the annual Music Information Retrieval Evaluation eXchange (MIREX) ACT task[5]. It should be noted that the datasets mentioned here include *only* the annotations and not the audio for any of the tracks due to copyright issues.

For our experiments we use the combined Beatles, Queen and Zweieck and the

[3] `http://isophonics.net/datasets`
[4] `http://ddmal.music.mcgill.ca/billboard`
[5] `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

Number of Frames			
Fold	Train	Validation	Test
1	963 872	327 322	449 666
2	972 726	325 710	442 442
3	966 612	327 102	447 146
4	973 924	328 818	418 118

Table 5.1: Distribution of data over the train, valid and test splits.

Billboard datasets which are used for the annual ACT task in MIREX[6]. We were able to obtain all 217 tracks from the Beatles, Queen and Zweieck dataset. Additionally, we obtained 650 out of the 740 tracks in the publicly available Billboard dataset[7]. The resulting dataset of 867 tracks was used for evaluating the proposed model. For all our experiments, we consider the major/minor chord alphabet (Harte, 2010), which is the most popular in literature (Lee and Slaney, 2006; Papadopoulos and Peeters, 2007; Humphrey and Bello, 2012; Boulanger-Lewandowski et al., 2013a; McVicar et al., 2014). We transform all given annotations to either major or minor chord labels. This yields 24 chord labels for the 12 pitch classes and 1 label for no-chord class. Since the hidden tracks for the MIREX evaluation are not publicly available, we perform 4-fold cross-validation over the 867 tracks for training and testing. All reported results are mean values of metrics over all 4 folds. The distribution of data over the 4 folds is reported in Table 5.1.

5.2.2 Metrics

We use the overlap ratio and weighted average overlap ratio for evaluation (Mauch, 2010; McVicar et al., 2014). Let there be N tracks in the test set. Let $E(y, g)$ be a function that acts on pairs of predicted labels and ground truth labels (y, g). $E(y, g)$ is 1 if the predicted label matches the ground truth label and 0 otherwise. The

[6] `http://www.music-ir.org/mirex/wiki/2015:Audio_Chord_Estimation`

[7] Details of the tracks IDs used for experiments can be found at: `http://www.eecs.qmul.ac.uk/~sss31/info.html`

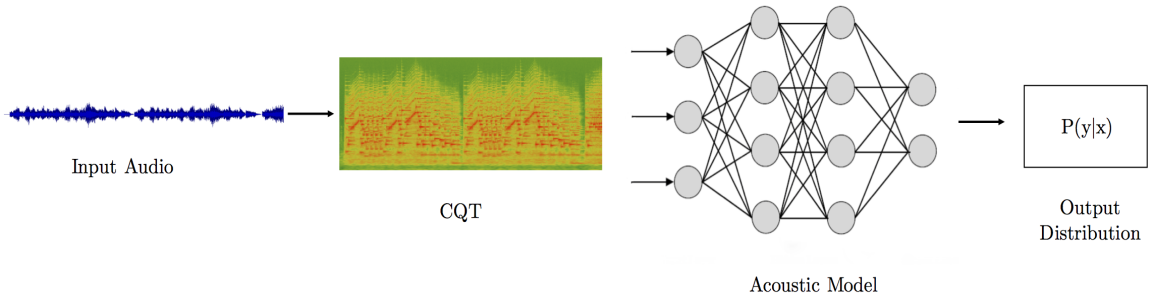


Figure 5.2: Acoustic Model Pipeline

evaluation metrics are defined below:

$$\text{Overlap Ratio (OR)} = \frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T^n} E(y_t^n, g_t^n), \tag{5.1}$$

where T^n is the length of n^{th} track in the dataset and $T = \sum_{n=1}^{N} T^n$ is the total number of frames in the test set. The overlap ratio is independent of the length of individual tracks. The weighted average overlap ratio (WAOR) is an alternative metric that accounts for individual track lengths:

$$\text{WAOR} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T^n} \frac{E(y_t^n, g_t^n)}{T^n}. \tag{5.2}$$

5.2.3 Preliminary Experiments

An overview of the acoustic model pipeline is shown in Figure 5.2. The audio is first down-sampled to 11.025 kHz. We then compute the CQT on audio frames with a hop-size of 1024 samples. The CQT is computed over 7 octaves with 24 bins per octave resulting in a 168-dimensional vector x. The vector x is input to the acoustic model which yields a probability distribution over chord labels $P(y|x)$.

5.2.3.1 Acoustic Model Training

For the purposes of training, the training data for all folds is further divided into a training set (80%) and a validation set (20%). The validation data is used to monitor

the network performance during training and to decide when to stop training. The outputs of the final layer of the acoustic model are passed through a softmax function to obtain a multinomial distribution over chord labels. All acoustic models are trained to minimise the negative log-likelihood of correct predictions over the training set.

- **DNNs**: We constrain all the hidden layers to have the same number of units to simplify searching for good network architectures. Based on previous experiments, we fix the number of layers of the network to 3 hidden layers. We perform a grid search over the following parameters: number of hidden units $n_h \in \{25, 50, 75, 100, 200, 500\}$, hidden activations $\in \{\text{ReLU}, \text{sigmoid}\}$. We used a fixed Dropout rate of 0.3 for all hidden and input layers. We use minibatch SGD with ADADELTA to estimate the model parameters. We use a batch-size of 100. Training is stopped if the validation error does not decrease after 20 epochs of training.

- **RNNs**: We train RNNs with LSTM units. We perform a grid search over the following parameters: number of hidden layers $L \in \{1, 2, 3\}$, number of LSTM units $n_h \in \{25, 50, 100, 200, 500\}$. We found that LSTM units led to better generalisation performance. We use mini-batch SGD with momentum for estimating model parameters. We average the gradients using batches of size 128. All training sequences are further divided into sub-sequences of length 200. Shorter sub-sequences are zero-padded to length 200. We use a fixed momentum rate of 0.9. We use an initial learning rate of 0.01 which is linearly reduced to 0 over 1000 iterations. Again, training is stopped if error on the validation set does not decrease after 20 epochs of training.

- **ConvNets**: We train ConvNets to correctly classify the central frame in a context window of size $2n + 1$. The input is a 2-D window of shape 7×168. Based on previous experiments with AMT, we fix the architecture as follows: the first

Model	Architecture
DNN	$L = 3, H = 100$
RNN-LSTM	$L = 2, H = 50$
ConvNet	$w_1 = (3, 25), P_1 = (1, 3), w_2 = (3, 7), P_2 = (1, 3)$ $n_1 = n_2 = 20, h_1 = 50, h_2 = 50$
RNN-LSTM-lm	$L = 2, H = 100$

Table 5.2: Model configurations for the best performing architectures.

2 layers of the network are alternating convolutional and max-pooling layers. This is followed by two fully connected layers, the output of which is then passed through a softmax function. We then perform a search over the filter sizes and the number of filters in each layer. We searched over the following filter shapes for the convolutional layers: $\{(3, 5), (3, 7), (3, 13), (3, 25), (5, 5), (5, 7), (5, 13), (5, 25)\}$. We fixed the number of filters in each convolutional layer to 20. Max-pooling over windows of size $(1, 3)$ was applied after both convolutional layers. We searched over the following number of hidden units for the fully connected layers: $\{25, 50, 100, 200\}$. A dropout of 0.4 was applied to all layers. The model parameters were estimated using mini-batch SGD and momentum. A constant momentum rate of 0.9 was used. We used a fixed learning rate of 0.001. Training was stopped if the validation error did not decrease after 20 epochs.

We performed the search over network architectures for the first fold. The best performing network architectures were used for training all other folds. The network architectures used in the experiments are summarised in Table 5.2.

5.2.3.2 Language Model Training

We train RNNs with 2 layers of LSTM units for modelling sequences of chord labels. The inputs to the language model are sequences of one-hot vectors of 25-dimensions. Each of the 25 dimensions corresponds to a chord label, with a value of 1 indicating the presence of a chord label. The RNNs are trained to minimise the negative log-

likelihood of predicting y_{t+1} given the sequence y_0^t (Section 3.4.3). We use minibatch SGD with momentum to train the RNNs. We use mini-batches of size 100 and a constant momentum rate 0.9. We use an initial learning rate of 0.001 which is linearly reduced to 0 over 1000 training epochs. Training is stopped if error on the validation set does not decrease after 20 training epochs.

5.2.3.3 HMM Comparison

Typically, the predictions from the acoustic model are noisy and are post-processed in order to enforce temporal smoothing and musicological structure. The outputs are either median filtered or the output probabilities are regarded as *observations* of an HMM model, where the hidden states correspond to chord labels (Papadopoulos and Peeters, 2007; Cho et al., 2010). At test time, Viterbi decoding is used to *infer* the most likely sequence of chord labels given the observations and the HMM parameters. The set of HMM parameters include the prior probabilities over hidden states, state transition probabilities and the parameters of the GMM acoustic model. All the HMM parameters can be learnt jointly using the Baum-Welch algorithm (Rabiner, 1989). However, studies show that similar performance can be achieved when the acoustic model is trained separately and the transition probabilities are set using prior musical knowledge (Papadopoulos and Peeters, 2007; Cho et al., 2010). We adopt the method described by Cho et al. (2010) for our experiments. We consider an HMM with 25 states, one for each chord label. We estimate the emission probabilities as $P(x_t|y_t) \propto \frac{P(y_t|x_t)}{P(y_t)}$, where the posteriors are obtained from the trained acoustic models and the marginals $P(y_t)$ are obtained by counting frequencies of chord labels over the training set. The transition probabilities for the HMM are defined as follows:

$$\log \hat{P}_{i,j} = \begin{cases} \log P_{i,j} - \log P, & \text{if } i \neq j \\ \log P_{i,j} & \text{if } i = j. \end{cases} \tag{5.3}$$

	Language Model					
	None		HMM		LSTM	
Acoustic Model	OR(%)	WAOR(%)	OR(%)	WAOR(%)	OR	WAOR(%)
DNN	57.06	56.59	61.21	60.38	62.81	62.00
RNN-LSTM	59.69	58.98	63.14	62.26	64.96	64.25
ConvNet	**61.75**	**61.10**	**63.93**	**63.33**	**65.47**	**64.78**

Table 5.3: 4-fold cross-validation results on the MIREX dataset for the major/minor prediction task.

In Equation 5.3, the parameter P is a penalty parameter that ensures that the diagonal elements have larger values than the off-diagonal elements. This is to account for the fact that due to repeating chord labels over many frames, the probability of a chord maintaining it's current state is higher than that of a chord change. In our experiments, the transition matrix is of size 25×25. We set all transition probabilities $P_{i,j}$ to a uniform value of $\frac{1}{25}$. Each row of the transition matrix is normalised so that the probabilities sum to 1. The prior probabilities for each HMM state are also uniformly set to $\frac{1}{25}$.

5.2.3.4 Results

Table 5.3 summarises the results of the preliminary experiments. The results in Column 1 are obtained directly from the outputs of the neural network acoustic models without any post-processing. The results in Column 2 are obtained by post-processing the acoustic model outputs with an HMM. We perform a grid search over the penalty parameter P (Equation 5.3) and results are reported with $P = 17$. The results in Column 3 are obtained by post-processing the acoustic model outputs with an RNN language model. The outputs are decoded using a standard priority queue beam search (Algorithm 4) with a beam width $w = 100$.

From Table 5.3 we observe that the ConvNet acoustic models outperform the DNN and RNN acoustic models. With respect to post-processing, we note that both the HMM and RNN post-processing improve system performance, with the

RNN post-processing outperforming the HMMs. Although these are encouraging results, a comparison with the best performing systems in the MIREX 2015 Audio Chord Estimation Challenge[8] reveals that the models in Table 5.3 are significantly outperformed by the best performing models at MIREX (75.58% OR).

5.2.4 Feature Learning

A very interesting property of neural networks is their ability to learn a hierarchy of representations in the intermediate or hidden layers of the network. A DNN classifier can be regarded as a sequence of trainable non-linear transformations of the input with a logistic regression classifier at the output layer, where the whole network is jointly optimised for a particular task. Rather than using the entire network, it is possible to use the activations of the intermediate layers as learnt transformations or *features* which are then used as inputs to another model for further processing (Hamel and Eck, 2010; Boulanger-Lewandowski et al., 2013a; Sigtia and Dixon, 2014).

The ability to learn features for a particular task is particularly relevant to MIR. Most MIR systems for problems like AMT and ACT follow a similar pipeline which involves feature extraction from the audio followed by classification (Humphrey, 2015). A considerable amount of time and effort has been spent in identifying the right combination of audio features and classifiers for a given task. Often these features are hand-engineered using prior domain knowledge about music and signal processing, for example chroma features and their variants for ACT. The ability to automatically *learn* good features for a task given a large dataset would greatly simplify the design of many MIR systems and allow researchers to focus attention on modelling higher level musical structure. Previously, feature learning has been applied to genre classification (Hamel and Eck, 2010; Sigtia and Dixon, 2014) and ACT (Boulanger-Lewandowski et al., 2013a). Here we investigate feature learning as a means to improve upon the

[8] `http://www.music-ir.org/mirex/wiki/2015:Audio_Chord_Estimation_Results`

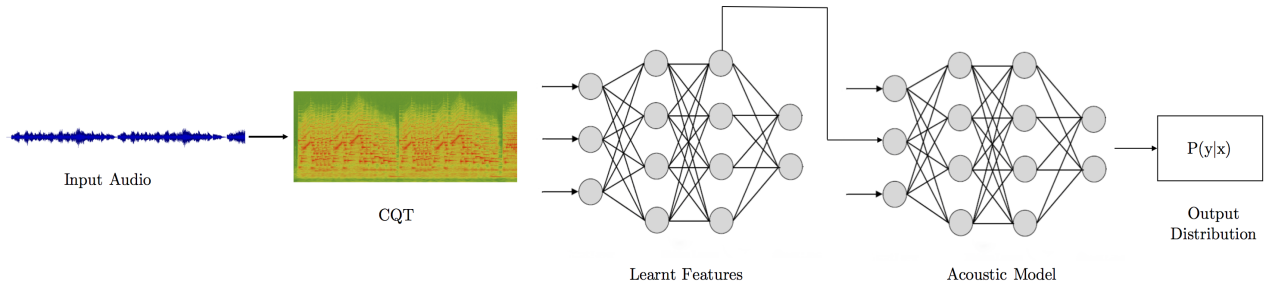


Figure 5.3: Feature Learning Pipeline

results from the previous section.

5.2.4.1 System Outline

Figure 5.3 represents the feature learning pipeline. We use the best performing DNN from the preliminary experiments for feature extraction (Table 5.2). The DNN has an input layer, 3 hidden layers with 100 unites each and an output layer. The pipeline for feature extraction is as follows: the raw audio is transformed into a time-frequency representation using the CQT and a hop-size of 1024 to obtain a sequence of 168 dimensional vectors. Individual CQT frames are then normalised by subtracting the mean and dividing by the standard deviation, calculated over the training dataset. The normalised vectors are then forward propagated through the DNN. The activations of the final hidden layer are then used as features. We observed that adding activations from other layers did not result in an improvement in performance. This results in a 100 dimensional feature vector for each audio frame.

We follow the same experimental procedure for evaluation as the previous experiments. The only difference is that the raw CQT inputs to the acoustic model are replaced with the learnt features from the DNN. We use the same 4 folds as the previous experiments and all reported results are mean values of metrics over all folds. We train DNN, RNN and ConvNet acoustic models and we perform a parameter sweet over the same parameters as described in Section 5.2.3. Additionally, we also experiment with providing context information to the DNN. There are several studies

Model	Architecture
DNN	$L = 3, H = 200$
RNN-LSTM	$L = 2, H = 100$
ConvNet	$w_1 = (5, 25), P_1 = (1, 3), w_2 = (3, 5), P_2 = (1, 3)$ $n_1 = n_2 = 20, h_1 = 100, h_2 = 50$

Table 5.4: Model configurations for the best performing architectures.

that demonstrate that providing context information to classifiers results in improved prediction accuracy (Bergstra et al., 2006; Boulanger-Lewandowski et al., 2013a; Sigtia and Dixon, 2014). Usually this is done by using a *context window* as input to the classifier, rather than individual frames. A context window of size $2k + 1$ comprises a central frame of interest, along with k frames before and after the central frame. The target label for each context window is the target label for the central frame. All the frames in a context window are appended together and the joint representation is used as input to the DNN. Rather than appending all the feature vectors together, we found that *aggregating* information over the window by calculating mean and standard deviation for each feature provided better classification accuracies. This is a form of *pooling* over the context window. Therefore we trained DNN acoustic models with mean and variance pooled features over context windows of size 7. The resulting features have 200 dimensions. Finally, rather than using the best performing acoustic model for evaluation, we retain all the trained models and average their predictions to form an *ensemble* of neural network acoustic models (Dietterich, 2000; Hinton et al., 2015). We observed an absolute improvement of $\sim 3\%$ OR for all acoustic models when averaging predictions. We use the same language models as the previous experiments. The model configurations used for evaluation are presented in Table 5.4.

	Language Model					
	None		HMM		LSTM	
Acoustic Model	OR(%)	WAOR(%)	OR(%)	WAOR(%)	OR	WAOR(%)
DNN	69.80	69.10	72.25	71.78	73.40	73.0
DNN-CW	72.90	72.50	74.68	74.35	75.53	75.07
RNN-LSTM	73.85	73.47	75.40	75.07	76.20	75.75
ConvNet	**75.24**	**74.87**	**77.13**	**76.65**	**77.95**	**77.38**

Table 5.5: 4-fold cross-validation results on the MIREX dataset for the major/minor prediction task.

5.2.4.2 Results

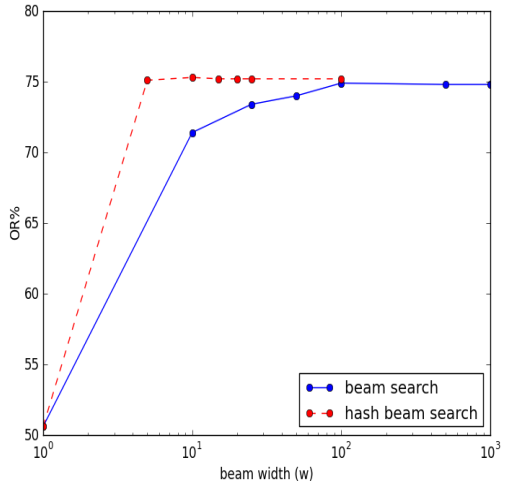
Table 5.5 summarises the results of the experiments with feature learning. Again, the results in Column 1 are obtained directly from the outputs of acoustic model. The results with HMM post-processing in Column 2 are reported with $\lambda = 19$. The results in Column 3 are obtained using the proposed hybrid RNN model. The model outputs were decoded using the hashed beam search algorithm, with $w = 10, n = 2, k = 1$ (Section 4.1.4.2).

From Table 5.5 we observe that using the learnt features as inputs to the acoustic models yields a significant improvement in overall performance for all acoustic model types. For all the three models, we observe an improvement by at least 10% for both OR and WAOR scores. From Rows 1 and 2, we note that the DNN acoustic model when provided with contextual information yields an improvement of 3% for both metrics. The RNN-LSTM acoustic model outputs both the DNN acoustic model architectures, while the ConvNet acoustic models outperforms all the other acoustic models. Again with regards to the ConvNets, from Table 5.2 we observe that longer filters yield better accuracies than small square filters. Recall that the same observations were made for AMT (Chapter 4, Table 4.2). As mentioned before, this due to the fact that the individual pitches are comprised by a fundamental frequency and a series harmonically related overtones which necessitates longer filters along the frequency axis.

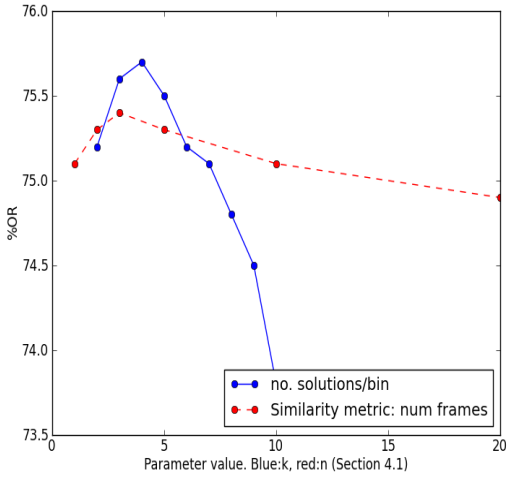
Across the columns of Table 5.5, we note that both HMM post-processing and the hybrid RNN model yield performance improvements as compared to the models without any post-processing. For all acoustic models, we note that the hybrid RNN outperforms HMM post-processing. Again, we observe that the relative improvement in performance with the hybrid RNN is most for the DNN acoustic models. The relative improvement for the RNN and ConvNet acoustic models is less due to the addition of context information to the inputs of the acoustic model, thus violating the assumptions of the hybrid RNN model (Section 4.1.4.1). Although not directly comparable, we note that the results in Table 5.5 are of the same order as the best performing entries in MIREX 2015 (75.58% OR).

To investigate the performance of the proposed hashed beam search algorithm, we plot the overlap ratio against the beam width in Figure 5.4a. We post-process the outputs of a DNN acoustic model with the RNN-LSTM language model in a hybrid setup. Figure 5.4a illustrates that the proposed algorithm can achieve marginally better decoding performance at a significant reduction in beam size. As an example, the hashed beam search yields an OR of 75.1% with a beam width of 5, while regular beam search yields 74.7% accuracy with a beam width of 1000. The time taken to run the hash beam search ($w = 5$) over the test set was 5 minutes, as compared to the regular beam algorithm ($w = 1000$) which took 17 hours to decode the test set. The algorithm's ability to yield good performance at significantly smaller beam widths indicates that it performs efficient pruning of similar paths, thus utilising the available beam width more efficiently. The run-times of the algorithm show that it can be used for real-time applications without compromising recognition accuracy.

In addition to the beam width, the hash beam search algorithm allows the user to specify the similarity metric and the number of solutions for each hash table entry. We investigate the effect of these parameters on the OR and plot the results in Figure 5.4b. We let the similarity metric be the previous n frames and observe performance



(a) %OR vs beam width ($n = 2, k = 1$)



(b) %OR vs f_h, k ($w = 25$)

Figure 5.4: Effect of varying hashed beam search parameters w, f_h, k on %OR.

as n is linearly increased for a fixed beam width of 25. From Figure 5.4b we observe that the performance is quite robust to changes in the number of past frames for small values of n. One possible explanation for the graph is that since the test data is sampled at a frame rate of 10ms, all occurrences of chords last for several frames. Therefore counting the previous n frames, effectively leads to the same metric each time. We experimented with using the previous n *unique* frames as a metric but found that the results deteriorated quite drastically as n was increased. This might reflect the limited memory of RNN language models and the issues caused due to lack of explicit duration modelling. The blue line in Figure 5.4b illustrates the effect of varying the number of solutions per hash table entry. From this graph we see that performance deteriorates significantly once the number of entries per bin crosses a certain threshold (~ 5). This is due to the fact that maintaining many solutions of the same kind saturates the beam capacity with very similar solutions, limiting the breadth of search. This can be solved by using much larger beam widths, however at the cost of increased time-complexity of the beam search algorithm (Section 5.1.4).

5.3 Discussion

We present a method for automatic chord transcription using the hybrid RNN model, with an aim to learn an end-to-end ACT system that can be applied directly to low-level features. We evaluated the model performance on the MIREX dataset for ACT. The results from preliminary experiments show that the neural network acoustic models are capable of identifying chord labels directly from the input time-frequency representation. Though we note that the performance is 10% worse than state-of-the-art ACT systems found in literature. Additionally, the results also demonstrate that the RNN chord language models are able to improve performance when combined with the acoustic models in the hybrid RNN framework, outperforming HMM based post-processing. Next, we perform experiments with feature learning. The results from these set of experiments are considerably better than the preliminary results. All the acoustic model types considered are able to perform similarly to state-of-the-art ACT systems. We note that the ConvNet acoustic models outperform the RNN-LSTM models, which in turn perform better than the DNN acoustic models. We also note that the RNN language models outperform HMM post-processing for all acoustic models. We also investigate the performance of the hashed beam search algorithm for decoding. We note that the hashed beam search is able to yield marginally better accuracies, at a significant reduction in beam size and run-times. These results demonstrate that given a sufficiently large annotated dataset, it is possible to train acoustic classifiers that learn useful features for classification from data. They also show the applicability of RNN-based language models for ACT.

The performance of the acoustic models presented here can be improved using techniques like data augmentation (Schlüter and Grill, 2015) and new optimisation strategies like batch-normalisation (Ioffe and Szegedy, 2015). The training of the RNN language models can be improved using better optimisation strategies for RNNs (Saxe et al., 2013; Le et al., 2015). Similarly to chapter 4, the RNN language models

presented here learn to model both durations and chord transitions. One way to deal with this issue would be to perform chord transcription in a beat-aligned manner, similar to beat-synchronous chromagrams (McVicar et al., 2014). Additionally, the RNNs can be used to model the chord transitions while a separate model (like an HMM) can be used to learn chord durations.

The results from this chapter follow some of the observations from Chapter 4. In both experiments we observe that the neural network acoustic models are able to learn useful mappings from CQTs to the desired output symbols (pitches or chords). We also observe that ConvNets with long filter shapes outperform DNN and RNN acoustic models on both tasks. In the following chapter, we investigate whether neural network acoustic models are also suitable for acoustic event detection for environmental sounds.

Chapter 6

Acoustic Event Detection

In the previous chapters, we presented neural networks models for automatic music and chord transcription. We observed that the neural network acoustic models are able to learn mappings from feature frames to note and chord labels given sufficient training data. In this chapter we present experiments with neural networks for an Acoustic Event Detection (AED) task for environmental sounds. The structure of the AED problem is similar to AMT and ACT, where the system outputs a sequence of semantic labels along with onset and offset times. As discussed in Section 2.2.1, the semantic labels for AED on environmental sounds depend on the particular task. In this chapter we design acoustic models for detecting baby cries and smoke alarm sounds. We compare the performance of neural network models with support vector machines and Gaussian mixture models. Additionally, since AED systems are typically deployed on embedded hardware, we derive estimates for the computational cost of each model and compare model performance as a function of the computational cost. The rest of the chapter is organised as follows: first we build the context for the experiments by describing the industrial and computational constraints for AED systems. Next, we derive computational cost estimates for each of the models considered. We then present results from the experiments and conclude the chapter with a

discussion of the results and future work.

6.1 Context

Automatic speech recognition, music classification, audio indexing and to some extent biometric voice authentication have achieved some degree of commercial success in consumer markets. A majority of these applications are typically deployed on PC platforms, cloud computing or modern smart phones. Recently, a new area of application is quickly emerging in the domain of *Internet of Things* (IoT) (Gubbi et al., 2013). In this domain, AED for environmental sounds or automatic environmental sound recognition (AESR) (Chachada and Kuo, 2014) has a significant potential to create useful applications, e.g. for security or home safety applications (Istrate et al., 2006; Vacher et al., 2010; Sitte and Willets, 2007). However in the context of IoT, algorithms and applications are subject to strict constraints imposed by the nature of embedded devices and their limited computing power. IoT devices can be broadly classified into two categories (Gubbi et al., 2013):

- Devices which perform a single function, say detecting alarms.
- Embedded devices where AED is offered as an additional service, for instance adding voice control features to a TV or sound recognition to a consumer camera. In this case, the AED algorithm must fit into the device's existing computational and cost constraints.

For both these cases, the use of high-end processors for embedded applications is infeasible since it drastically increases the cost of these devices making them commercially not viable. Typically, the following features jointly define the financial cost of a processor and the constraints imposed by embedded computing (Hennessy and Patterson, 2011):

- The *clock speed* of the processor which is related to energy consumption.
- The *instruction set* is related to chip size and manufacturing costs. In some processors, special instruction sets are included to parallelise more operations into a single clock cycle.
- The *architecture* of the processor defines the number of registers, number of cores and presence or absence of a Floating Point Unit (FPU), a Graphical Processing Unit (GPU) and/or a Digital Signal Processing (DSP) unit.
- *Onboard memory size* is an important factor related to processor cost. It affects both the computational performance, where repetitive operations can be cached to trade speed against memory, and the scalability of an algorithm, by imposing upper limits on the number of model parameters that can be stored and manipulated.

These features jointly define an upper limit on the number and type of operations that can be executed in a given amount of time. It could be argued that since most embedded devices allow internet connectivity, cloud computing can overcome the computational constraints by abstracting the computing platform and making it as powerful as necessary. However, a number of additional design considerations rule out the use of cloud computing for many AED applications. The *latency* introduced by cloud computing can be a problem for time critical security applications (Bonomi et al., 2014). The network communications in cloud computing add an additional point failure into the system which degrades the *quality of service* (QoS). Sending alerts rather than streaming audio or acoustic features is more suitable in terms of *privacy* (Medaglia and Serbanati, 2010) and *bandwidth*, both of which are critically important for consumer applications.

Due to the above considerations, IoT devices are typically devoid of an FPU, operate in Megahertz clock speed range (unlike PCs that operate in the Gigahertz

range) and do not offer onboard DSP or other specialised instruction sets. Given the additional design constraints related to privacy, latency, bandwidth and QoS, AED applications have to be deployed directly on embedded platforms with limited computational capacity. Despite these considerations, the design and evaluation of such systems is carried out with limited regard for the practical limitations of embedded platforms. Most results are obtained with floating point arithmetic on powerful computing platforms, with no constraints on the run-time and memory requirements of the final system. The experiments presented in this chapter are performed with 2 main objectives. Firstly, to compare the performance of neural networks to GMMs and SVMs on a large-scale practical AED task. Secondly, to compare their performance as a function of the computational cost in order to estimate the viability of deploying the algorithms on embedded hardware.

6.2 Computational Cost

6.2.1 Motivation

As mentioned before, one of the main aims of these experiments is to compare the performance of machine learning algorithms on an AED task *as a function of their computational cost*. While computational cost is related to computational complexity, the notions are distinct: computational cost simply counts the number of operations at a given model dimension, whereas computational complexity expresses the mathematical law according to which the computational cost scales up with the dimensions of the input feature space (Cormen et al., 2001).

We study the computational cost of different algorithms at the sound recognition or acoustic decoding stage. Unless specifically required by applications, it is uncommon for the training stage of machine learning algorithms to be implemented on embedded devices. Typically, the training stage is designed as an offline process

and is implemented on powerful scientific computing platforms equipped with clusters of FPUs and GPUs. For embedded devices, the floating point model trained offline is quantised according to the specific hardware (Smith, 1997). It is generally accepted that the quantisation error introduced by this operation does not affect acoustic modelling performance significantly (Gupta et al., 2015).

The computational cost estimates used in this study account for four types of basic operations: addition, comparison, multiplication, and lookup table retrieval (LUT). *Multiply-add* operations are commonly found in dot products, matrix multiplications and FFTs. For example, correlation operations, linear filtering and Mahalanobis distance, which are at the heart of many machine learning algorithms, rely solely on multiply-add operations. The precision of multiply-add operations when implemented using fixed point DSP is fairly straightforward to manage. *Division*, on the other hand, and in particular matrix inversion, can be more difficult to manage. With regards to matrix inversion, quantisation errors can add up to make the inversion algorithm unstable (Parhami, 2009), though in many cases it is possible to pre-compute the inversion of key parameters like the variances offline and in floating point, before applying fixed-point quantisation. *Non-linear* operations such as logarithms, exponentials, cosines and N^{th} roots are required by some machine learning algorithms. Two approaches are commonly taken to transform non-linear functions into a series of multiply-adds: either Taylor series expansion, or look-up tables (LUTs). LUTs consume more memory than Taylor series expansions, but require a lower number of instructions to achieve the desired precision. It is important to be aware of the presence of non-linearities in a particular algorithm, before assessing its cost: algorithms which rely on a majority of multiply-add operations are more desirable than algorithms relying more heavily on non-linearities.

For all the estimates presented in this chapter, it is assumed that these four types of operations have an *equal* cost. While this assumption would be true for a majority

of processors on the market, it may underestimate the cost of non-linear functions if interpolated LUTs or Taylor series are used instead of simpler direct LUT lookups. In addition to the 4 operations considered here, there are additional costs incurred with other operations (like data handling) in the processor. However, we use the simplifying assumption that the considered operations are running on the same core, thus minimising the data handling overhead.

As a case study, let us consider running the K-nearest neighbours algorithm on a Cortex-M4 processor[1], running at a clock speed of 80MHz and with 256kB of onboard memory. This is a very common hardware configuration for consumer electronic devices such as video cameras. Assuming that 20% of the processor activity is required for general system tasks, this leaves a maximum of 64 million multiplies or adds per second for sound recognition (80MHz $\times$ 80% = 64MHz). Assuming that the audio is sampled at 16kHz and audio features are extracted at, say, a window shift of 256 samples, equivalent to 62.5 frames per second, implies that the AED algorithm should use no more than 1,024,000 multiply or adds per analysis frame (64MHz / 62.5Hz = 1 024K instructions).

Assuming a K-Nearest neighbour algorithm with a 40 dimensional observation vector x, mean vector μ and a Mahalanobis distance $(x-\mu)\sigma^{-1}(x-\mu)$ with a diagonal covariance matrix σ, at run-time the algorithm involves one subtraction and two multiplications per dimension for the Mahalanobis distance calculation, plus one operation for the accumulation across each dimension: four operations in total, thus entailing a maximum of 1,024,000 instructions / 4 operations / 40 dimensions = 6 400 nearest neighbours on this platform to achieve real-time recognition.

However, assuming that each nearest neighbour is a 40-dimensional vector of 16 bits/2 Bytes values, the memory requirement to store the model would be 512kB, which is double the 256kB available on the considered platform. Again assuming

[1]Cortex-M4 Technical Reference Manual: `http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf`.

that the code size occupies 20% of the memory, leaving 80% of the memory for model storage, a maximum of $256\text{kB} \times 80\% \,/\, 40\,\text{features} \,/\, 2\text{Bytes} = 2\,560$ nearest neighbours only which could be used as the acoustic model.

Therefore while designing an AED system for embedded hardware, the limiting factor is a combination of the available processing power and onboard memory. In the case of highly non-linear algorithms, the computational cost of achieving real-time audio recognition can outweigh the model storage requirements. While only the final integration can tell if the desired computational load and precision are met, estimates can be obtained as illustrated in this case study in order to predict whether an algorithm will fit on a particular platform.

6.2.2 Cost Estimates

Most AED systems use a common 3-step pipeline (Stowell et al., 2015). The audio recording is first converted into a time-frequency representation, usually by applying the short-time Fourier transform (STFT) to overlapping windows. This is followed by feature extraction from the time-frequency representation. Typically, Mel Frequency Cepstral Coefficients (MFCCs) are used as standard features, though other spectral features have been used (Section 2.2.1). The acoustic features are then input to an acoustic model which yields a posterior probability or class membership score. This score is finally compared against a threshold to make a decision about class membership. In this section, we provide estimates for each component of the AED pipeline.

6.2.2.1 Feature extraction

Engineering the right feature space for AED is essential, since the definition of the feature space affects the separability of the classes of acoustic data. Feature extraction from audio recordings thus forms the first step in the classification pipeline, which

contributes to the overall computational cost. In this study, the acoustic models are trained on a set of input features that are typically used for audio and speech processing (Section 6.3.3). Although recent studies demonstrate that neural networks can be trained to jointly learn the features and the classifier (LeCun et al., 2015), we have found that this method is impractical for most AESR problems where the amount of labelled data for training and testing is very limited (Section 6.3.2). Additionally, by training the different algorithms on the same set of features, we are able to study the performance of the various classifiers as a function of computation cost, without having to account for the cost of extracting different features for each acoustic model[2]. It also allows a fair comparison of the discriminative properties of different classification algorithms on the same input feature space. Therefore, we factor out the computational cost of feature extraction.

6.2.2.2 Gaussian Mixture Models

Given a D dimensional feature vector x, a GMM (Bishop, 2006, Chapter 2) is a weighted sum of Gaussian component densities which provides an estimate of the likelihood of x being generated by the probability distribution defined by Equation 3.39. logsum symbolises a recursive version of function $\log(a+b) = \log a + \log(1 + e^{(\log b - \log a)})$ (Murphy, 2006), which can be computed using a LUT on non-linearity $\log(1+e^x)$. Calculating the log-likelihood of a D-dimensional vector given a GMM with M components therefore requires the following operations:

- D subtractions (i.e., additions of negated means) and $2D$ multiplications per Gaussian to calculate the argument of the exponent.
- 1 extra addition per Gaussian to apply the weight in the log domain.

[2]For example in preliminary experiments, we observed that DNNs were able to achieve the same performance with log-spectrogram inputs as speech features (Section 6.3.3). However, the GMMs and SVMs performed poorly on log-spectrogram inputs. Therefore the cost estimates additionally involved the cost of feature extraction for each acoustic model which made it harder to interpret results.

- M LUT lookups and M additions for the non-linear logsum cumulation across the Gaussian components.

This leads to the computational cost per audio frame expressed in the first row of Table 6.1.

6.2.2.3 Support Vector Machines

Given a support vector machine with λ support vectors, a new D-dimensional example can be classified using λ additions to sum the dot product results for each support vector, plus a further addition for the bias term (Equation 3.45). For each dot product, a support vector requires the computation of the kernel function, then λ multiplications to apply the α_i multipliers.

Some kernels require a number of elementary operations, whereas others require a LUT lookup. Thus, depending on the kernel, the final costs are:

- SVM Linear – $(\lambda \cdot D) + \lambda + 1$ additions and $(\lambda \cdot D) + \lambda$ multiplications.
- Polynomial – $\lambda(D + d + 2) + 1$ additions and $\lambda(D + 2)$ multiplications, where d is the degree of the polynomial.
- Radial Basis Function – $2\lambda D + \lambda + 1$ additions, $\lambda(D + 2)$ multiplications and λ exponential functions.
- Sigmoid – $\lambda(D+2)+1$ additions and $\lambda(D+2)$ multiplications and λ hyperbolic tangent functions.

The computational cost of SVMs per audio frame for these kernels is summarised in Table 6.1.

6.2.2.4 Neural Networks

We compare two types of neural network architectures, DNNs and RNNs (Section 3.2). For DNNs, we consider 2 activation functions for the hidden layers, the sigmoid

Feature	Addition	Multiplication	nonlinearity/LUT lookup
GMM	$2(M \cdot (D+1)+M)$	$2M \cdot 2D$	M
SVM - Linear	$\lambda D + \lambda + 1$	$\lambda \cdot D$	0
SVM - Polynomial	$\lambda D + \lambda + 1$	$\lambda(D+d)$	0
SVM - RBF	$2\lambda D + \lambda + 1$	$\lambda(D+2)$	λ
SVM - Sigmoid	$\lambda D + \lambda + 1$	$\lambda(D+1)$	λ
DNN - Sigmoid	$H \cdot (1+D+L+(L-1)H)+1$	$H \cdot (1+D+(L-1)H)$	$L \cdot H + 1$
DNN - ReLU	$H \cdot (1+D+L+(L-1)H)+1$	$H \cdot (1+D+(L-1)H)$	$L \cdot H + 1$
RNN - Tanh	$H \cdot (2+D+H+2(L-1)(H+1))+1$	$H \cdot (1+D+H+2(L-1)H)$	$L \cdot H + 1$

Table 6.1: Computational cost per frame of each compared model. D is the dimensionality of the feature vector, M is the number of Gaussian mixtures for a GMM. λ is the number of support vectors for a SVM, d is the degree of a polynomial kernel. For the neural networks: H is the number of hidden units in each layer and L is the number of layers.

and ReLU activations (Section 3.2.1). The sigmoid non-linearity is implemented as a LUT operation, while the ReLU activation is more suited to embedded devices as it involves only 1 comparison operation. For simplicity, every neural network architecture was constrained to have the same number H of hidden units in each of its L layers. The input dimensionality is denoted by D. The forward pass through a feed-forward neural network therefore involves the following computations:

- Multiplying a column vector of size n with a matrix of size $n \times k$ involves $n \cdot k$ additions and an equal number of multiplication operations.
- Therefore the first layer involves $(D \cdot H)$ multiplication operations and $H \cdot (D+1)$ additions, where the additional H additions are due to the bias term.
- The remaining $L-1$ layers involve the following computations: $(L-1) \cdot (H \cdot H)$ multiplications and $(L-1) \cdot (H \cdot (H+1))$ additions.
- The output layer involves H multiplications and $H+1$ additions.
- A non-linearity is applied to the outputs of each layer, leading to a total of $L \cdot H + 1$ non-linearities.
- The RNN forward pass includes an additional matrix multiplication due to the

recurrent connections. This involves $H \cdot H$ multiplications and an equal number of addition operations.

The computational cost estimates for DNNs and RNNs are summarised in Table 6.1.

6.3 Evaluation

6.3.1 Evaluation Metrics

Let us consider a binary classification problem where a positive class corresponds to the label 1 and the negative class corresponds to the label 0. Any classifier produces 2 types of errors. False negative (FN) errors are when the target class for a given example is positive but the classifier assigns 0 to the example. Similarly, a false positive (FP) error is when the target class for an example is negative but the classifier outputs 1. Complementary to these errors are true positives (TP), when the classifier correctly assigns a positive label and true negatives (TN), when the classifier correctly assigns a negative label (Japkowicz and Shah, 2011). The above quantities are typically calculated as the fraction of classifier outputs of each type divided by the total number of examples (Section 4.1.5.2). Typically, most classifiers output a real valued score which is compared to a threshold, after which a classification decision is made. Therefore, the performance of a classifier on the above metrics is a function of the threshold or *operating point*. The threshold value can be varied according to the given application. For example in some cases it might be necessary to minimise the number of false positives, which can be achieved by rejecting more examples and therefore allowing more false negatives.

Ideally, classifiers for AED applications should be compared independently of the choice of operating point. This can be achieved through the use of Detection Error Tradeoff (DET) curves (Martin et al., 1997), which plot the false positive rate against the false negative rate over a range of operating points. DET curves are equivalent to

Receiver Operating Curves (ROC) (Martin et al., 1997) plotted on a normal deviate scale, under the assumption that the FP vs FN scores are normally distributed. A curve that is closer to the origin represents a better classifier. The performance based on DET curves can be summarised into a singer figure of merit called the Equal Error Rate (EER), which is the point where the DET curve intersects the diagonal %FP = %FN. Alternatively, the EER represents the operating point where the system generates an equal proportion of false positives and false negatives.

The GMM classification system comprises 2 GMM models: the first GMM is trained to maximise the likelihood of the examples belonging to the target class $P(x|\theta_{\text{target}})$ (Bishop, 2006, Chapter 2). A second GMM known as the *universal background model* (UBM) is trained to maximise the likelihood of non-target examples $P(x|\theta_{\text{UBM}})$. At test time, the ratio between the log-likelihoods from the two models $\log P(x|\theta_{\text{target}})/P(x|\theta_{\text{UBM}})$ is the output score which is compared against a threshold. For SVMs, the argument of the sgn() function in Equation 3.45 is the output score. The sgn functions corresponds to a threshold of 0 and a margin defined by the support vectors (Equation 3.46). Choosing a threshold other than 0 corresponds to a deviation from the margin of maximum separation. According to the discussion in Section 3.3.2, the neural network outputs a probability of class membership $P(\text{target}|x)$, which is compared against a threshold for classification.

6.3.2 Datasets

For AED, the training data comprises audio recordings along with annotations or ground truth labels for each recording. Similar to AMT (Chapter 4) and ACT (Chapter 5), the ground truth labels are provided in the form of onset times and offset times for each label in a given recording. The annotations are usually performed by human listeners and therefore collecting and labelling large datasets is a time consuming and expensive process (Stowell et al., 2015). Typically, relatively small datasets are

used for evaluating AED systems. For instance the DCASE challenge (Stowell et al., 2015) and the CLEAR challenge (Temko et al., 2006) are two popular benchmarks for AED. The DCASE challenge data contains 20 examples for each of the 16 event classes, while the dataset for the CLEAR challenge contains approximately 60 examples for each of the 13 classes. These datasets are considerably smaller than the datasets available for other domains where machine learning is applied (LeCun et al., 2015).

One of the motivations for these experiments is to evaluate whether neural network acoustic models can perform better than existing approaches to AED. However as discussed in previous chapters, neural network models typically have many parameters and require large quantities of training data in order to generalise effectively. For example, we trained neural network acoustic models on the data for the DCASE challenge, but we were not able to match the GMM baseline (Stowell et al., 2015). In order to be to able to train neural network acoustic models, we perform experiments using 3 private datasets made available by Audio Analytic Ltd.[3,4] These 3 datasets are subsets of much larger data collection campaigns led by Audio Analytic Ltd. to support the development of smoke alarm and baby cry detection products. Ground truth annotations with onset and offset times obtained from human annotators are also provided. Out of the 3 datasets, 2 datasets contain recordings of baby cries and smoke alarms, while the third dataset contains a large number of recordings of background or ambient sounds. The datasets of baby cries and smoke alarm sounds provide a large number of examples for training and evaluating acoustic models. The background or world dataset on the other hand is a rich source of *impostor* examples for both training and evaluation. Any practical AED system would have to make accurate detections in the presence of thousands of ambient impostor sound sources. The background dataset provides recordings from potentially thousands of sources in

[3] `http://www.AudioAnalytic.com`

[4] The data can be made available upon setting suitable contractual agreements.

	Train		Test	
Dataset	Duration (s)	# Target Frames	Duration (s)	# Target Frames
Baby Cries	4 822	224 076	3 669	90 958
Smoke Alarms	15 271	194 142	5 043	114 753
World	5 000	312 500	4 000	255 562

Table 6.2: Distribution of train and test data for the Baby Cry, Smoke Alarms and World datasets.

order to simulate real-world conditions during training and testing.

The *Baby Cry* dataset comprises recordings obtained from two different recording conditions: one through a camcorder in a hospital environment and one through uncontrolled hand-held recorders in uncontrolled conditions (both indoors and outdoors). Each recorder is used to record a different baby. The train/test split was made evenly in order to achieve the same balance over both recording conditions (about 3/4 recordings from the hospital and the remaining from both indoor and outdoor environments) without any overlap between sources (individual babies) for training and testing. The recordings in this dataset sum up to 4 822 seconds of training data and 3 669 seconds of test data, from which 224 076 feature frames in the training set and 90 958 feature frames in the test set correspond to target baby cry sounds. The target frames correspond to frames that occur between the onset and offset for an event of interest (like baby cries) in a given recording.

The *Smoke Alarm* dataset comprises recordings of 10 smoke alarm models, recorded through 13 different channels across 3 British homes. While it might appear that smoke alarms should be relatively easy to detect, there is considerable variability due to the plethora of alarm models, room responses, ambient surroundings and different recording channel configurations which rules out simple audio fingerprinting or template matching based approaches. The training set comprises recordings from 2 homes, while the test set contains recordings from the third home. The smoke alarms were recorded using an array of consumer devices covering devices of interest for

example network cameras, in order to average out the channel effects. The dataset comprises 15 271 seconds of training data and 5 043 seconds of testing data, which result in 194 142 feature frames for training and 114 753 feature frames for evaluating the AED acoustic models.

The *World Dataset* contains about 900 recordings of approximate 10 seconds each which cover a wide variety of complex acoustic scenes recorded from numerous indoor and outdoor locations around the UK, across a wide and uncontrolled range of recording devices and potentially covering thousands of impostor sound classes. For example a 10 second recording of a train station scene covers train sounds, speech, horns, whistles and more. While it is not possible to enumerate all the sound sources, even an underestimate of 1.1 unique sources per recording results in over 1000 imposter classes for evaluation. 500 recordings are used as a source of negative examples for training the classifiers, while the remaining recordings are used as imposter examples for evaluation. The dataset consists of 5 000 seconds of data for training and $\sim$ 4 000 seconds of data for testing, which corresponds to 312 500 feature frames for training and 255 562 feature frames for testing. The datasets and their contents are summarised in Table 6.2.

6.3.3 Feature Extraction

All recordings were sampled at 16 kHz, 16 bits. All features were calculated with a window size of 512 samples and a hop size 256 samples. We extracted MFCC features which have been extensively used in speech recognition and in AED systems (Radhakrishnan et al., 2005; Clavel et al., 2005; Valenzise et al., 2007; Portelo et al., 2009) and use the first 13 MFCC coefficients. In addition to MFCCs, spectral centroid (Clavel et al., 2005), spectral flatness (Portelo et al., 2009), spectral roll-off (Valenzise et al., 2007), spectral kurtosis (Valenzise et al., 2007) and zero crossing rate (Valenzise et al., 2007) features were also computed. Concatenating all the features results in an

18-dimensional feature vector. To incorporate temporal information into the inputs, we also computed the first and second order differences of all features resulting in a 54-dimensional feature vector. It should be noted that the inputs to the RNN were the original 18-dimensional features without temporal differences, since the RNN explicitly models input context and we did not find any improvement by providing the delta features as inputs. For all experiments, the data was normalised to have zero mean and unit standard deviation for each dimension, where the mean and standard deviation were computed over the entire training set.

6.3.4 Training Methodology

6.3.4.1 Gaussian Mixture Models

GMMs were trained using the expectation maximisation algorithm (EM) (Bishop, 2006, Chapter 2). The covariance matrices were constrained to be diagonal[5]. We trained GMMs with Gaussian components $M \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$. All the Gaussians were initialised using the k-means++ algorithm (Arthur and Vassilvitskii, 2007). A single UBM was trained on the World training dataset. One GMM per target class was then trained independently of the UBM. We also performed experiments where GMMs for each class were estimated by adapting the UBM by Maximum A Posteriori (MAP) adaptation (Reynolds et al., 2000). We observed that this method yielded inferior results. We also observed worse results when no UBM was used and class membership was determined by simply thresholding the outputs of individual class GMMs. In the following sections, results are therefore only reported for the best performing method, which is the likelihood ratio between independently trained UBM and target GMMs.

[5]In preliminary experiments, we observed that a full covariance matrix did not yield any improvement in performance. Diagonal covariance matrices are also computationally cheaper.

6.3.4.2 Support Vector Machines

We compare linear, polynomial, radial basis function (RBF) and sigmoid kernel SVMs for AED. In addition to optimising the kernel parameters (d, γ) (Section 6.2.2.3), a grid search was performed over the penalty parameter C (Equation 3.46). In practice, real-world data are often non-separable, thus a soft margin has been used as necessary. SVM training is known to not scale well to very large datasets (Burges, 1998): the training algorithm has a time-complexity of $O(T^3)$, where T is the number of training examples. As a matter of fact, our preliminary attempts at using the full training datasets led to prohibitively long training times and poor generalisation performance on unseen data. Standard practice in that case is to down-sample the training set by randomly drawing T frames from the target set, and the same number of frames from the world set as negative examples for training. We performed a grid search over the number of training examples vs test accuracy. We determined $T = 2000$ to be the optimal number of examples from each class. Additionally, we also present comparative results with $T = 500$. This is done as a means to control model complexity, since the number of support vectors is automatically selected by the optimisation algorithm and controlling model complexity is a challenging problem with SVMs.

6.3.4.3 Neural Networks

All the neural network architectures contain 1 output unit with a sigmoid activation that estimates the probability of class membership of an example (Section 3.3.2). The networks were trained to maximise the likelihood of correct classification over the training set.

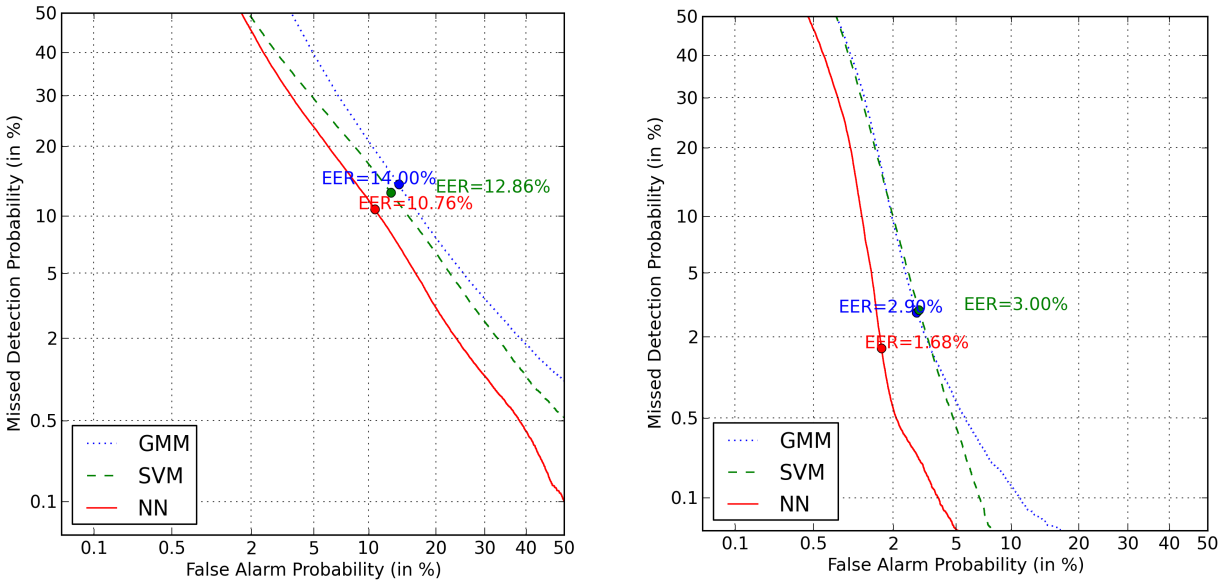
For the DNNs, a grid search was performed over the following parameters: number of hidden layers $L \in \{1, 2, 3, 4\}$, number of hidden units per layer $H \in \{10, 25, 50, 100, 150\}$, hidden activations $act \in \{sigmoid, ReLU\}$. In order to minimise parameter tuning,

we used ADADELTA to adapt the learning rate over iterations. The networks were trained using mini-batches of size 100 and training was accelerated using an NVIDIA Tesla K40c GPU. A constant dropout rate of 0.2 was used for all layers to improve results on the test data. The training was stopped if the cost on the validation set did not decrease after 20 epochs.

For the RNNs, a grid search was performed over the following parameters: number of hidden layers $L \in \{1, 2, 3\}$ and number of hidden units per layer $H \in \{10, 25, 50, 100, 150\}$. An initial learning rate of 0.001 was used and linearly decreased to 0 over 1000 iterations. A constant momentum rate of 0.9 was used for all the updates. The training was stopped if the error on the validation set did not decrease after 20 epochs. The training data was further divided into sub-sequences of length 100 and the networks were trained on these sub-sequences without any mini-batching. Gradient clipping was also used to avoid the exploding gradient problem in the early stages of RNN training. The gradient was clipped if the norm of the gradient update exceeded 10.

6.3.5 Results

In this section, model performance is analysed for the task of recognising smoke alarms and baby cries against the large number of impostor sounds from the world dataset. Tables 6.3 and 6.4 present the EER for the best performing classifiers of each type and their associated computational cost. Figure 6.1 shows the corresponding DET curves. In Figures 6.2 and 6.3 we present a more detailed comparison between the performance of various classifier types against the computational cost involved in classifying a frame of input at test time.



(a) Baby Cry Dataset

(b) Smoke Alarm Dataset

Figure 6.1: DET curves comparing frame classification performance of the best performing model of each type. Curves closer to the origin imply better performance.

Best Baby Cry classifiers	**EER**	**# Ops.**
GMM, $M = 32$	14.0	10 560
Linear SVM, $T = 2000$, $C = 1.0$, $\lambda = 655$	12.9	101 985
Feed-forward DNN, sigmoid, $L = 2$, $H = 50$	10.8	10 702

Table 6.3: Performance of the best classifiers on the Baby Cry dataset along with the optimal parameters and number of operations.

Best Smoke Alarm classifiers	**EER**	**# Ops.**
GMM, $M = 16$	2.9	5 280
Linear SVM, $T = 2000$, $C = 0.1$, $\lambda = 152$	3.0	46 655
Feed-forward DNN, sigmoid, $L = 2$, $H = 25$	1.7	4 102

Table 6.4: Performance of the best classifiers on the Smoke Alarms dataset along with the optimal parameters and number of operations.

6.3.5.1 Baby Cry Dataset

From Table 6.3 we observe that the best performing GMM has 32 components and achieves an EER of 14.0% for frame-wise classification. From Figure 6.2 (+ markers) we note that the performance of the GMMs improves till $M = 32$, after which increasing the number of components further leads to a deterioration in performance. Note

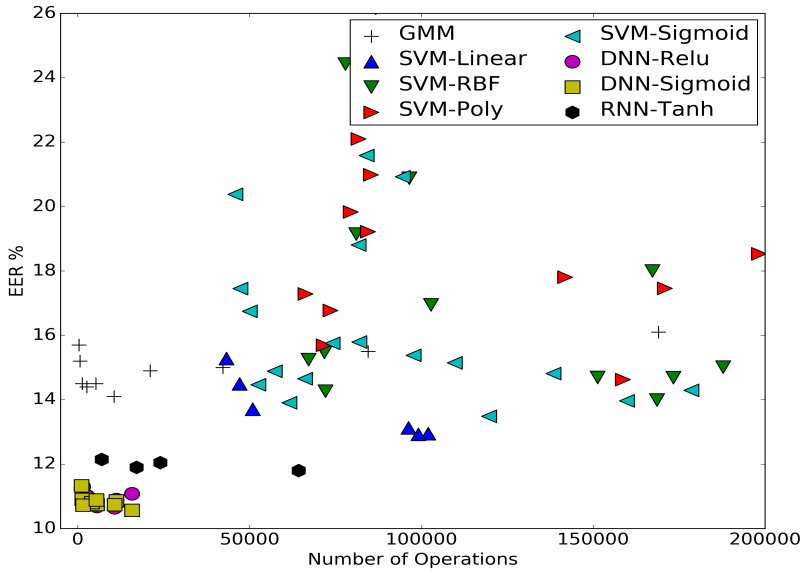


Figure 6.2: Acoustic frame classification performance (EER percentage) as a function of the number of operations per frame, for each of the tested models on the Baby Cry dataset. The number of operations and consequently the computational cost increases from left to right.

that the computational cost increases from left to right on the X-axis. Therefore for GMMs, models from left to right are in increasing order of the number of components M.

From Table 6.3, we observe that an SVM with a linear kernel is the best performing SVM classifier, with an EER of 12.9%. The SVM was trained on 2000 examples, resulting in 655 support vectors. From Figure 6.2, we note that the performance of the linear SVM (blue triangles) improves as the computational cost (number of support vectors) is increased. The number of support vectors can also be controlled by varying the parameter C (Section 6.2.2.3). For the linear SVM case, $C = 1.0$ yields the best results. We observe that SVMs with a sigmoid kernel (light blue triangles) yield similar results to the linear SVM (Figure 6.2). The best SVM with a sigmoid kernel yields an EER of 13.5%, while the second best sigmoid SVM has an EER of 13.9%. As in the linear case, we observe an improvement in test performance as the

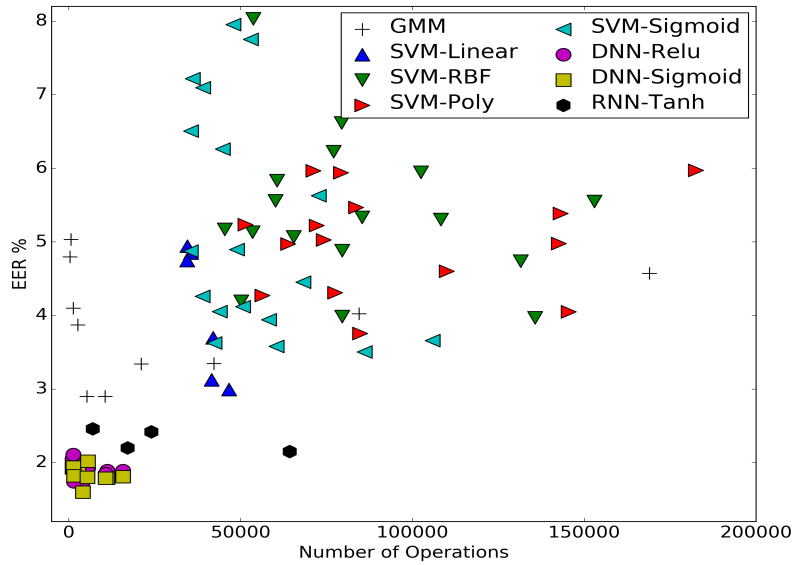


Figure 6.3: Acoustic frame classification performance (EER percentage) as a function of the number of operations per frame, for each of the tested models on the Smoke Alarm dataset. The number of operations and consequently the computational cost increases from left to right.

computational cost or the number of support vectors is increased.

From Figure 6.2, we note that SVMs with RBF and polynomial kernels (green and red triangles) are outperformed by linear and sigmoid SVMs, both in terms of %EER and computational cost. There is also no observable trend between test performance and computational cost. For the polynomial SVM (red triangles), we found a kernel with $d = 3$ yielded the best performance, while low values of gamma $\gamma \in (0.005, 0.01)$ provided the best results for the RBF kernel (Section 6.2.2.3).

The number of training examples $T = 2000$ was empirically determined to be the optimal value: adding more training examples did not yield any improvement in performance. Conversely, we tried training the same classifiers with $T = 500$ training examples, since the computational cost of SVMs is determined both by the type of kernel and the number of support vectors, the latter being controlled by varying the number of training examples and the penalty parameter C. With $T = 500$ and

$C = 0.1$, we were able to achieve a minimum EER rate of 13.7% with the linear SVM, while halving the number of operations.

From Table 6.3, the best performing neural network architecture, which achieves an EER of 10.8%, is a feed-forward DNN with sigmoid activations for the hidden units. Figure 6.1a shows that the neural network clearly outperforms all the other models. From Figure 6.2 we observe that the feed forward DNNs (circle and square markers) achieve similar test performance over a wide range of computational costs. This demonstrates that the network performance is not particularly sensitive to the specific number of hidden units in each layer. However, we did observe that networks which were deeper ($>$ 1 hidden layer) yielded better performance. From Figure 6.2, we observe that the RNN architectures (hexagonal markers) yield slightly worse performance and are computationally more expensive. An interesting observation from Figure 6.2 is that feed-forward DNNs with sigmoid activations yield similar results to networks with ReLU activations. This is a very important factor when deploying these models on embedded hardware, since a ReLU net can be implemented with only linear operations (multiplications and additions), without the need for costly Taylor series expansions or LUTs.

6.3.5.2 Smoke Alarm Dataset

From Table 6.4, we observe that the best performing GMM yields an EER of 2.9% and uses 16 mixture components. From Figure 6.3 (+ markers), we observe that the GMM performance improves till $M = 32$ components, after which the performance starts deteriorating. The best GMM with 16 components performs similarly to the best SVM model, though the computational cost of the SVM is almost 10 times more than the GMM (Table 6.4).

Similar to the results on the Baby Cry dataset, the linear and sigmoid kernel SVMs show the best performance, out of all four SVM kernel types. The best linear SVM

has an EER of 3.0%, which is very similar to the GMM score, although at a much larger computational cost. The model used 2000 training examples and $C = 1.0$. From Figure 6.3 we again observe an improvement in performance with an increase in computation cost for both the linear and sigmoid kernels (blue triangles). The best sigmoid kernel SVM scored 3.5% with 2000 training examples, while another configuration scored 3.6% with $T = 500$ and $C = 1$, at half the number of operations. Again, the polynomial and RBF kernels (red and green triangles) yield lower performance, with no observable trend in terms of performance versus computational cost (Figure 6.3).

From Table 6.4, we observe that a feed-forward sigmoid DNN yields the best performance, with an EER of 1.6%. From the DET curves (Figure 6.1b), we see that the neural network clearly outperforms the other models. From Figure 6.3 we note that the neural networks consistently perform better than the other models, over a wide range of computational costs, which correspond to different network configurations (number of layers, number of units in each layer). The ReLU networks perform similarly to the sigmoid networks, while the RNNs perform worse and are computationally more expensive. It is worth noting that the performance of all classifiers is significantly better for the smoke alarm sounds, since the smoke alarms are composed of simple tones. On the other hand, baby cries have a large variability and are therefore more difficult to detect.

6.4 Discussion

In this chapter, we compare the performance of neural network acoustic models with GMMs and SVMs on an AED task for environmental sounds. Unlike other machine learning systems, AED systems are usually deployed on embedded hardware, which imposes many computational constraints. Keeping this in mind, we compare the

performance of the models as a function of their computational cost. We evaluate the models on two tasks, detecting baby cries and detecting smoke alarms against a large number of impostor sounds. These datasets are much larger than the popular datasets found in AESR literature which enables us to train neural network acoustic models. Additionally, the large number of impostor sounds allows to investigate the performance of the proposed models in a testing scenario that is closer to practical use cases as compared to previously available data sets for AED.

Results suggest that GMMs provide a low cost baseline for classification, across both datasets. The GMM acoustic models are able to perform reasonably well at a modest computational cost. SVMs with linear and sigmoid kernels yield similar performance in terms of EER compared to GMMs, but their computational cost is overall higher. The computational cost of the SVM is determined by the number of support vectors. Unlike GMMs, SVMs are non-parametric models which do not allow the direct specification of model parameters. Though the number of support vectors can be indirectly controlled with regularisation or the number of examples used for training. Finally, our results suggest that deep neural networks clearly outperform both the GMMs and the SVMs on both datasets. The computational cost of DNNs can be controlled via the number of hidden units and the number of layers. While changes in the number of units in the hidden layers did not appear to have a large impact on performance, deeper networks appeared to perform better in all cases. Additionally, neural networks with ReLU activations achieved good performance, while being an attractive choice for deployment on embedded devices because they do not require expensive LUT lookup operations.

The results presented here are in agreement with the findings in Chapter 4 and 5. As mentioned before, the AED problem has a similar structure as AMT and ACT. We observe that neural networks are able to learn effective mappings from input feature vectors to high-level targets like note labels, chord labels and smoke alarm

and baby cry activity. Our comparisons demonstrate that given sufficient training data, neural networks are capable of outperforming state-of-the-art acoustic models in many domains. Additionally, the neural processing architectures and pipelines are general. For instance, the DNNs applied to AMT, ACT and AED are similar in their structure, though they differ in the type of input features and the specific architectures used. This is in stark contrast to other approaches which are hand-designed and specific to each task. The generality of the network architectures can be leveraged by training large networks on data for different tasks in order to try to improve the generalisation capabilities. For example, large neural networks in computer vision are often initialised by training them on a large dataset of images (Krizhevsky et al., 2012) before further tuning and adaptation for a specific task.

Chapter 7

Conclusions

In this thesis, we investigated neural networks for analysing music and environmental audio. We evaluated the performance of the models on 3 different tasks: automatic music transcription, automatic chord transcription and acoustic event detection for environmental sounds. For all three problems, we observed that the proposed models were able to perform competitively with state-of-the-art systems for each task. In this chapter, we first summarise our findings and then discuss directions for future research.

7.1 Summary

7.1.1 Automatic Music Transcription

In Chapter 4, we perform 2 sets of experiments for AMT. In the first set of experiments, we design a hybrid RNN architecture for piano music transcription. The hybrid RNN model combines the predictions of arbitrary frame-level acoustic models with the predictions of an RNN MLM. We present DNN, RNN and ConvNet architectures for acoustic modelling and compare their performance to two state-of-the-art acoustic models. Our results demonstrate that the ConvNets clearly outperform all

other models on all frame-based metrics. We also note that the MLMs consistently improve performance for all metrics, though the absolute improvement is small ($\sim 1\%$ F-measure). We then investigate whether the models are able to generalise to new piano types. From these experiments, we observe that again the ConvNets outperform all other models on frame-based metrics, though the performance of all models is worse by $\sim 10\%$ F-measure since the models are tested on unseen piano types. Finally, we propose a beam search based algorithm for inference. Our proposed hashed beam search modification to the decoding algorithm is able to drastically reduce decoding time, making the proposed method suitable for real-time applications. Overall, we note that the neural network models which are applied to CQT inputs and trained end-to-end, are able to outperform state-of-the-art models which are hand-designed for AMT. Secondly, the MLMs help improve performance on all metrics, though the improvement is small. The fact that MLMs can be trained in an unsupervised setting on corpora of musical scores provides strong encouragement for further investigation.

In the second set of experiments with AMT, we investigate the performance of MLMs in a more general setting where the test data comprises multiple instruments. Additionally the acoustic and language models are trained on separate datasets. The PLCA acoustic model is trained on isolated note samples from the RWC dataset, while the MLM is trained on folk melodies from the Nottingham dataset. We present a novel method for combining the predictions of the PLCA acoustic model with an RNN MLM using Dirichlet priors. The proposed model was evaluated on the Bach10 dataset. We observe that combining the predictions of the PLCA model and the RNN MLM using the proposed method results in a 3% absolute improvement in note-based F-measure and forms the current state-of-the-art on the Bach10 dataset. These results demonstrate that MLMs can indeed help improve performance of AMT systems and that similar to speech recognition, MLMs can be effective even when the acoustic and language models are trained on disjoint sets of data.

7.1.2 Automatic Chord Transcription

In Chapter 5, we apply the hybrid model developed for AMT to an automatic chord transcription problem. We evaluate the proposed model on a large dataset of popular music for ACT. As in Chapter 4, we present DNN, RNN and ConvNet architectures for acoustic modelling. Additionally, we use an RNN chord language model for post-processing in the hybrid RNN framework. We also compare the performance of the RNN with an HMM for post-processing. From preliminary experiments, we observe that the neural network acoustic models are able to identify chord labels from CQT inputs, though the performance of the models is worse by $\sim 10\%$ F-measure compared to state-of-the-art ACT systems. Next, due to the availability of a larger dataset for training, we investigate feature learning for ACT. We use the hidden activations of the best performing DNN from the preliminary experiments as features and train neural network acoustic models with the learned features as inputs. We observe that all acoustic models yield a performance improvement of $\sim 10\%$ F-measure when the learnt features are used as inputs. We also achieve a small improvement in acoustic model performance by averaging the predictions of all the trained acoustic models. Similar to our results for AMT, we observe that the ConvNets with long filters outperform DNN and RNN acoustic models. With respect to the chord language model, we observe that RNN language models consistently outperform the HMMs, which are the most popular post-processing method for ACT systems. We note that the proposed ACT system yields similar performance to state-of-the-art systems which are designed specifically for AMT. It should be noted that the hybrid RNN model is able to yield good performance for experiments with both AMT and ACT and is not limited to a single task.

7.1.3 Acoustic Event Detection

In Chapter 6, we investigate neural network acoustic models for acoustic event detection for environmental sounds. We compare the performance of neural network acoustic models with GMMs and SVMs. We evaluate the models on 2 separate tasks: detection of baby cry sounds and detection of smoke alarm sounds. The datasets used for evaluation are much larger than the most commonly used datasets for AED (Temko et al., 2006; Stowell et al., 2015) and more closely approximate real-world test conditions. Since AED systems are typically deployed on embedded devices, we also derive computational cost estimates for each algorithm and compare the performance of different models as a function of the computational cost.

Our results demonstrate that for both datasets, the GMMs form a low cost baseline. The GMM acoustic models perform reasonably well at a relatively low computational cost. We observe that the SVMs yield comparable performance to the GMMs, though at a much higher computational cost. We also observe that due to the non-parametric nature of SVMs, controlling computational cost is a difficult problem since the support vectors are determined at run-time. We note that for both datasets, the neural network acoustic models clearly outperform the GMMs and SVMs. We observe that deeper models perform well and that the computational cost can be easily controlled via the number of hidden units per layer and the number of layers. The DNN acoustic models yield good results without incurring a heavy computational cost. In fact the best performing DNN on both datasets has roughly the same computational cost as the best performing GMM. Additionally, we observe DNNs with both sigmoid and ReLU hidden units yield similar performance, which is a useful property since ReLUs can be implemented very efficiently on embedded hardware since they do not require LUTs or Taylor series expansions.

7.2 Future Work

In this section we discuss future work and potential research directions to extend the ideas presented in this thesis.

7.2.1 Acoustic Modelling

From Chapters 4, 5 and 6, we observe that neural network models are able to learn mappings from time-frequency inputs to high-level symbols (notes, chords, environmental events) when trained end-to-end and provided sufficient labelled training data. The general architecture of the acoustic models used for all 3 problems is similar and in some cases (AMT and AED) the models outperform hand-designed acoustic models for each problem. Although these results are encouraging there are several possibilities for improving model performance.

The first observation we make is that the neural network acoustic models perform better when more labelled training data is available. In Chapter 5, we observed that the larger dataset for training could be leveraged to learn features for ACT, which yielded an absolute improvement of 10% in F-measure. Similarly, in Chapter 6 we observed that the neural network models failed to beat a GMM baseline on the relatively small DCASE dataset (Stowell et al., 2015), but they clearly outperformed the GMMs and SVMs when trained on a large dataset of baby cries and smoke alarm sounds. Therefore, one way to improve performance for MIR and environmental sound detection is to collect larger datasets for training and evaluation. This trend has been observed in other fields such as speech recognition and computer vision, where the success of initial applications of neural networks has been followed by collection of much larger datasets for training and evaluation. The current state-of-the-art approach in both fields is to train large neural networks with many parameters on large datasets. Although admittedly, collecting and distributing music audio is complicated

by copyright issues, there is compelling evidence that both fields would benefit if evaluation and benchmarking were performed on large standardised datasets.

One way to increase the size of the training dataset is using data augmentation techniques (Krizhevsky et al., 2012). Data augmentation involves distorting the network inputs while preserving the output labels. For example for images, this can be done by making small rotations to the images or by adding noise to the pixel intensity values. Data augmentation expands the size of the training set while at the same time acting as a regulariser by adding noise to the training data. Although data augmentation is now standard practice for computer vision, it hasn't been widely adopted for MIR and environmental audio research. The main reason for this is that it is not not clear what kind of transformations can be validly applied to CQTs or any other time-frequency representation being used as inputs to the network. Though recently there has been some interest in data augmentation for MIR (Mauch and Ewert, 2013; Schlüter and Grill, 2015), we believe it is a promising direction for audio analysis research.

One important observation from all the experiments presented in this thesis is that the neural network acoustic models are general and similar architectures can be applied to different tasks. In our experiments we observed that similar DNN architectures could be trained on 3 different tasks, provided a sufficiently large training dataset was available. It could be argued that the feature engineering stage which was previously required in designing acoustic models has now been replaced by an architecture engineering stage for neural networks. However, being able to train networks end-to-end that learn the features and classifiers jointly from examples is a big generalisation compared to manually injecting domain knowledge into the acoustic features for each task. As meta-learning and Bayesian optimisation algorithms improve (Snoek et al., 2012, 2015), the task of finding the most appropriate architectures can potentially by automated.

Related to the generality of architectures, is the idea of transfer learning. In computer vision, it is now common to start with a network that is trained on a large dataset of images (Krizhevsky et al., 2012) and then fine-tune the network to a more specific task, for instance by retraining the output layer while keeping the other parameters fixed. This technique allows the network to share concepts between similar tasks. Similar ideas can be applied for processing audio signals. For example, in order to recognise chords correctly, a network should be able identify pitches. Therefore, a neural network chord recognition model can be pre-trained on a large dataset for AMT, before subsequent fine-tuning for chord recognition. It should be noted that this is only possible due to the fact that the same architectures can be applied to different tasks.

From the results in Chapter 4, we observe that the neural network acoustic models outperform all other models on frame-based metrics, but are outperformed by the model by Vincent et al. (2010) on note-based metrics. We note that for note-based metrics, a detection is made correctly if the right pitches are predicted and the onset time for the pitches is within $\pm 50ms$ of the ground truth onset. The fact that the neural network models get high frame-wise accuracy but relatively lower note-wise accuracy implies that the temporal resolution of the neural network predictions is low. This is due to the fact that inputs to the neural networks are CQTs, while the inputs to the model by Vincent et al. (2010) are ERB filter-banks. The ERB representation has higher temporal resolution and consequently the model yields better note-based performance. Both the neural networks and the model by Benetos and Dixon (2012) use CQT inputs and yield lower note-based performance. The temporal resolution of the neural network acoustic models can be improved by experimenting with alternative input representations.

All the acoustic models presented in this thesis were trained on a frame-wise basis or at a frame-level i.e. an input sequence of features was mapped to an output

sequence, with one output per frame. However, frame-level training requires the availability of ground truth labels with *alignment* or labels with onset/offset times. Collecting large annotated datasets of this form is an expensive task. Additionally, as briefly mentioned in Section 4.1.5.5, annotating offsets for many instruments can be noisy due to sustain and reverberation effects. An alternative to frame-wise training is to cast the acoustic modelling objective in the sequence-to-sequence framework. In this case the objective would be to map the sequence of input features to the sequence of output symbols, where the output symbols are not repeated per frame. For example for ACT, the output symbols would be a sequence of chord labels, e.g. $\{C_{major}, D_{major}, E_{major}\}$. This is similar to the training objective for large vocabulary speech recognition acoustic models, where the ground truth is composed of word-level transcriptions without alignments. The task of mapping a sequence of acoustic features to a shorter sequence of labels can be achieved with RNN architectures like Connectionist Temporal Classification (Graves et al., 2006) or the recently proposed sequence-to-sequence models for machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), which have been successfully applied to large vocabulary speech recognition (Bahdanau et al., 2016). The recently developed attention mechanisms can also be easily incorporated into these architectures (Bahdanau et al., 2014). In such a scenario, a separate onset detection system can be used to find the label alignments or in some cases the alignment can be inferred from the model outputs (Graves et al., 2006).

7.2.2 Music Language Models

In Chapters 4 and 5, we investigated using RNN language models for AMT and ACT, respectively. As mentioned previously, language models are a fundamental part of speech recognition systems and they have significant potential to improve systems for processing music. Similar to speech, music language models (for AMT or ACT) learn

useful priors when trying to identify notes or chords from acoustic features. Just like language, polyphonic sequences of notes or chords exhibit temporal structure. Accurate statistical models of this structure have the potential to provide priors for AMT and ACT tasks, in addition to generative models for sampling novel music and chord sequences. A practical motivation for investigating MLMs is that rather than collecting labelled training data, MLMs can be trained on music scores, guitar tabs and MIDI files. This is similar to language modelling in speech, where language models are often trained on large corpora of text data obtained from different sources, for example Wikipedia.

From Chapter 4, we observe that the MLMs trained on a corpus of folk melodies help improve the F-measure by 3% on the Bach10 dataset, giving the best reported results on the Bach10 dataset. Similarly, from Chapter 5 we observe that RNN language models for ACT consistently outperform HMM based post-processing, which is the most popular post-processing method for ACT systems. Although the performance gains on the piano transcription task were modest (1%), we believe that the remaining results provide strong motivation for further investigations with MLMs.

From Chapter 4, we note that the input representation was identified as one of the limitations of the proposed MLMs. Typically, language models are trained to maximise the log-likelihood of sequences of note or chord vectors. These sequences are formed by sampling the ground truth at the same sampling rate as the acoustic models. In this setting, there is no explicit information provided to the network about higher level temporal concepts like metrical position. The input piano-roll representation to the MLMs can be supplemented with information about position in the bar to encourage the learning of higher level temporal structure. Additionally, the proposed RNN MLMs learn durations implicitly through the repetition of frames in the training data. As argued before, when the sampling rate is high enough and there are many repetitions, log-likelihood during training can be trivially maximised by

repeating the previous frame. This happens to be different from language modelling in speech where the models are trained at a word level, therefore the duration of each symbol is immaterial. The language model simply models the transition probabilities given sequences of words. In order to decompose the problem of learning transition probabilities and durations, MLMs can be trained on sequences of notes or chords without any repetition. A separate model can be used to model the durations of note combinations or chords. Or alternatively, the RNNs can be supplemented with duration models for each symbol and both the transition probabilities and durations can be learnt jointly. The problem of repeating frames can also be addressed by supplementing the RNN with an external differentiable memory (Graves et al., 2014; Grefenstette et al., 2015). It has been shown that RNNs with external memories are able to perform copy, replacement and sorting tasks more effectively than regular RNNs. The external memory allows the RNN to write to and copy from memory as necessary, without having to compress all the information in the hidden state.

Earlier we discussed data augmentation for acoustic models. Similar ideas can be applied to MLMs to increase the size of the training dataset. For instance, the data to train MLMs can be transposed across all tonalities. This would provide the network with significantly more data and also encourage the network to learn translation invariance across pitches. An alternative method to encourage the network to learn translation invariance would be to use convolutional filters along the pitch axis, rather than the fully connected input layers to the RNN.

The ideas, methods and results presented in this thesis provide some evidence that demonstrates that neural networks can be effective for analysing the contents of music and environmental audio. However, this study merely scratches the surface of the possibilities of neural networks approaches for audio signal analysis. As in other domains, we hope that the fields of music and environmental audio processing adopt the use of neural networks for processing low-level features. This will allow

researchers to focus on modelling higher-level structural aspects of audio signals. Similarly, we hope that learning distributions (or language models) and structure over temporal sequences of notes, chords and acoustic events is explored further, since these methods have the potential to significantly advance the state-of-the-art in music and environmental audio analysis.

Bibliography

Music Information Retrieval Evaluation eXchange (MIREX). http://music-ir.org/mirexwiki/.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Samer A Abdallah and Mark D Plumbley. Unsupervised Analysis of Polyphonic Music by Sparse Coding. *Neural Networks, IEEE Transactions on*, 17(1):179–196, 2006.

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition. In *INTERSPEECH*, pages 3366–3370, 2013.

Moray Allan and Christopher KI Williams. Harmonising Chorales by Probabilistic Inference. *Advances in Neural Information Processing Systems (NIPS)*, 17:25–32, 2005.

Naomi S Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, 1992.

Ehsan Amid, Annamaria Mesaros, Kalle J Palomaki, Jorma Laaksonen, and Mikko Kurimo. Unsupervised Feature Extraction for Multimedia Event Detection and Ranking Using Audio Content. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5939–5943. IEEE, 2014.

Tobias Andersson. Audio Classification and Content Description. *Lulea University of Technology, Multimedia Technology, Ericsson Research, Corporate unit, Lulea, Sweden*, 2004.

Fabrizio Argenti, Paolo Nesi, and Gianni Pantaleo. Automatic Transcription of Polyphonic Music based on the Constant-Q Bispectral Analysis. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(6):1610–1630, 2011.

José Anibal Arias, Julien Pinquier, and Régine André-Obrecht. Evaluation of Classification Techniques for Audio Indexing. In *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*, pages 1–4. IEEE, 2005.

David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

Jean-Julien Aucouturier, Boris Defreville, and François Pachet. The Bag-of-Frames Approach to Audio Pattern Recognition: A Sufficient Model for Urban Soundscapes but not for Polyphonic Music. *The Journal of the Acoustical Society of America*, 122(2):881–891, 2007.

Roland Badeau, Valentin Emiya, and Bertrand David. Expectation-Maximization Algorithm for Multi-pitch Estimation and Separation of Overlapping Harmonic Spectra. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3073–3076, 2009.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*, 2014.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Yoshua Bengio, et al. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

Eric Battenberg and David Wessel. Analyzing Drum Patterns Using Conditional Deep Belief Networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 37–42. Citeseer, 2012.

Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

Juan Pablo Bello and Jeremy Pickens. A Robust Mid-Level Representation for Harmonic Content in Music Signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 5, pages 304–311, 2005.

Emmanouil Benetos. *Automatic Transcription of Polyphonic Music Exploiting Temporal Evolution*. PhD thesis, Queen Mary University of London, December 2012.

Emmanouil Benetos and Simon Dixon. A Shift-Invariant Latent Variable Model for Automatic Music Transcription. *Computer Music Journal*, 36(4):81–94, 2012.

Emmanouil Benetos, Srikanth Cherla, and Tillman Weyde. An Effcient Shift-Invariant Model for Polyphonic Music Transcription. In *6th International Workshop on Machine Learning and Music*, 2013.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1171–1179, 2015.

Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5 (2):157–166, 1994.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in Optimizing Recurrent Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8624–8628. IEEE, 2013.

Bruce Benward. *Music in Theory and Practice Volume 1*. McGraw-Hill Higher Education, 2014.

Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. Unsupervised Transcription of Piano Music. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1538–1546, 2014.

James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. Aggregate Features and AdaBoost for Music Classification. *Machine learning*, 65 (2-3):473–484, 2006.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU Math Expression Compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

Nancy Bertin, Roland Badeau, and Emmanuel Vincent. Enforcing Harmonicity and Smoothness in Bayesian Non-negative Matrix Factorization Applied to Polyphonic Music Transcription. *IEEE Transactions on Audio, Speech, and Language Processing.*, 18(3):538–549, 2010.

Christopher M Bishop. Mixture Density Networks. 1994.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Thomas Blumensath and Mike Davies. Unsupervised Learning of Sparse and Shift-invariant Decompositions of Polyphonic Music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*, volume 5, pages 497–500. IEEE, 2004.

Sebastian Böck and Markus Schedl. Polyphonic Piano Note Transcription with Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–124. IEEE, 2012.

Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters. In *Proceed-*

ings of the 16th International Society for Music Information Retrieval Conference (ISMIR), pages 625–631, 2015.

Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog Computing: A Platform for Internet of Things and Analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer, 2014.

Léon Bottou. Large-scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio Chord Recognition with Recurrent Neural Networks. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 335–340, 2013a.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pierre Vincent. High-Dimensional Sequence Transduction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3178–3182. IEEE, 2013b.

Nicolas Boulanger-Lewandowski, Jasha Droppo, Mike Seltzer, and Dong Yu. Phone Sequence Modeling with Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5417–5421. IEEE, 2014.

John S Bridle. Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

Judith C Brown. Calculation of a Constant-Q Spectral Transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

Christopher JC Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

John Ashley Burgoyne, Laurent Pugin, Corey Kereliuk, and Ichiro Fujinaga. A Cross-Validated Study of Modelling Strategies for Automatic Chord Recognition in Audio. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 251–254, 2007.

John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 633–638, 2011.

Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

Francisco Jesus Canadas-Quesada, Julio Jose Carabias-Orti, Raul Mata-Campos, Nicolas Ruiz-Reyes, and Pedro Vera-Candeas. Multipitch Estimation of Harmonically-Related Event-Notes by Improving Harmonic Matching Pursuit Decomposition. In *Audio Engineering Society Convention 124*. Audio Engineering Society, 2008.

Benoit Catteau, Jean-Pierre Martens, and Marc Leman. A Probabilistic Framework for Audio-based Tonal Key and Chord Recognition. In *Advances in data analysis*, pages 637–644. Springer, 2007.

Benjamin Cauchi. Non-Negative Matrix Factorisation Applied to Auditory Scenes

Classification. *Master's thesis, Master ATIAM, Université Pierre et Marie Curie*, 2011.

Sachin Chachada and C-C Jay Kuo. Environmental Sound Recognition: A Survey. *APSIPA Transactions on Signal and Information Processing*, 3:e14, 2014.

Srikanth Cherla, Tillman Weyde, Artur S d'Avila Garcez, and Marcus Pearce. A Distributed Model For Multiple-Viewpoint Melodic Prediction. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 15–20, 2013.

Taemin Cho and Juan P Bello. A Feature Smoothing Method for Chord Recognition Using Recurrence Plots. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 651–656, 2011.

Taemin Cho, Ron J Weiss, and Juan Pablo Bello. Exploring Common Variations in State of the Art Chord Recognition Systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–8, 2010.

Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events Detection for an Audio-based Surveillance System. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1306–1309. IEEE, 2005.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011a.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011b.

Arshia Cont. Realtime Audio to Score Alignment for Polyphonic Music Instruments, Using Sparse Non-negative Constraints and Hierarchical HMMs. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages 245–248. IEEE, 2006.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*, volume 2. MIT press Cambridge, 2001.

Giovanni Costantini, Renzo Perfetti, and Massimiliano Todisco. Event Based Transcription System for Polyphonic Piano Music. *Signal processing*, 89(9):1798–1811, 2009.

Courtenay V Cotton and Daniel PW Ellis. Spectral vs. Spectro-temporal Features for Acoustic Event Detection. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 69–72. IEEE, 2011.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-Dependent Pre-trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

Roger B Dannenberg. A Brief Survey of Music Representation Issues, Techniques, and Systems. 1993.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, and Quoc V Le. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1223–1231, 2012.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

Arnaud Dessein, Arshia Cont, and Guillaume Lemaitre. Real-time Polyphonic Music Transcription with Non-negative Matrix Factorization and Beta-divergence. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 489–494, 2010.

Diana Deutsch. Music Recognition. *Psychological Review*, 76(3):300, 1969.

Sander Dieleman and Benjamin Schrauwen. End-to-End Learning for Music Audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014.

Thomas G Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, pages 1–15. Springer, 2000.

Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and Non-peak Regions. *IEEE Transactions Audio, Speech, and Language Processing*, 18(8):2121–2133, November 2010a.

Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and non-peak Regions. *IEEE Transactions on Audio, Speech, and Language Processing.*, 18(8):2121–2133, 2010b.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Daniel PW Ellis. Detecting Alarm Sounds. In *Consistent & Reliable Acoustic Cues for*

Sound Analysis: One-day Workshop: Aalborg, Denmark, pages 59–62. Department of Electrical Engineering, Columbia University, 2001.

Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch Estimation of Piano Sounds using a new Probabilistic Spectral Smoothness Principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

Reeves Fletcher and Colin M Reeves. Function Minimization by Conjugate Gradients. *The computer journal*, 7(2):149–154, 1964.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2013.

Takuya Fujishima. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, volume 1999, pages 464–467, 1999.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv preprint arXiv:1506.02142*, 2015.

Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. Detection and Classification of Acoustic Scenes and Events: An IEEE AASP Challenge. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA),*, pages 1–4. IEEE, 2013.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

King-Shy Goh, Koji Miyahara, Regunathan Radhakrishnan, Ziyou Xiong, and Ajay Divakaran. Audio-Visual Event Detection Based on Mining of Semantic Audio-

Visual Labels. In *Electronic Imaging 2004*, pages 292–299. International Society for Optics and Photonics, 2003.

Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing. *arXiv preprint arXiv:1510.00726*, 2015.

Masataka Goto. A Real-time Music-scene-description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals. *Speech Communication*, 43(4):311–329, 2004.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Music Genre Database and Musical Instrument Sound Database. In *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, Baltimore, USA, October 2003.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

Alex Graves. Sequence Transduction with Recurrent Neural Networks. In *Representation Learning Workshop, ICML*, 2012a.

Alex Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer, 2012b.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 369–376. ACM, 2006.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to Transduce with Unbounded Memory. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1828–1836, 2015.

Thomas Grill and Jan Schlüter. Music Boundary Detection Using Neural Networks on Spectrograms and Self-Similarity Lag Matrices. In *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, Nice, France, 2015a.

Thomas Grill and Jan Schlüter. Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015b.

Graham Grindlay and Daniel PW Ellis. Transcribing Multi-instrument Polyphonic Music with Hierarchical Eigeninstruments. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1159–1169, 2011.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A Vision, Architectural Alements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

Zheng Guibin and Liu Sheng. Automatic Transcription Method for Polyphonic Music Based on Adaptive Comb Filter and Neural Network. In *International Conference on Mechatronics and Automation (ICMA).*, pages 2592–2597. IEEE, 2007.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1737–1746, 2015. URL `http://jmlr.org/proceedings/papers/v37/gupta15.html`.

Philippe Hamel and Douglas Eck. Learning Features from Music Audio with Deep

Belief Networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 339–344, 2010.

Christopher Harte. *Towards Automatic Extraction of Harmony Information from Music Signals*. PhD thesis, Queen Mary University of London, August 2010. @phdthesisHarte2010thesis, Address = London, UK, Author = Christopher Harte, Month = August, School = Queen Mary University of London, Title = Towards Automatic Extraction of Harmony Information from Music Signals, Year = 2010 .

Christopher Harte and Mark Sandler. Automatic Chord Ddentifcation Using a Auantised Chromagram. In *Audio Engineering Society Convention 118*. Audio Engineering Society, 2005.

Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications.*, 13(4):18–28, 1998.

Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-Dependent Sound Event Detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–13, 2013.

John L Hennessy and David A Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E Hinton. Learning Distributed Representations of Concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

Geoffrey E Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.

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

Derek Hoiem, Yan Ke, and Rahul Sukthankar. SOLAR: Sound Object Localization and Retrieval in Complex Audio Environments. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–429. IEEE, 2005.

Kurt Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural networks*, 4(2):251–257, 1991.

Adrianus JM Houtsma. *Pitch Perception*. Academic Press San Diego, London, 1995.

David H Hubel and Torsten N Wiesel. Receptive Fields of Single Neurons in the Cat's Striate Cortex. *The Journal of physiology*, 148(3):574–591, 1959.

Eric J Humphrey. *An Exploration of Deep Learning in Content-Based Music Informatics*. PhD thesis, NEW YORK UNIVERSITY, 2015.

Eric J Humphrey and Juan P Bello. Rethinking automatic chord recognition with convolutional neural networks. In *11th International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 357–362. IEEE, 2012.

Eric J Humphrey, Taemin Cho, and Juan P Bello. Learning a Robust Tonnetz-Space Transform for Automatic Chord Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456. IEEE, 2012.

Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature Learning and Deep Architectures: New Directions for Music Informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456, 2015. URL `http://jmlr.org/proceedings/papers/v37/ioffe15.html`.

Dan Istrate, Eric Castelli, Michel Vacher, Laurent Besacier, and Jean-François Serignat. Information Extraction from Sound for Medical Telemonitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):264–274, 2006.

Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350, 2015.

Otto Karolyi. *Introducing Music*. Penguin (Non-Classics), 1965.

Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, pages 3128–3137, 2015.

Kunio Kashino and Norihiro Hagita. A Music Scene Analysis System with the MRF-based Information Integration Scheme. In *Proceedings of the 13th International Conference on Pattern Recognition.*, volume 2, pages 725–729. IEEE, 1996.

Lyndon S Kennedy and Daniel PW Ellis. Laughter Detection in Meetings. In *NIST*

ICASSP Meeting Recognition Workshop, Montreal, pages 118–121. National Institute of Standards and Technology, 2004.

Maksim Khadkevich and Maurizio Omologo. Use of Hidden Markov Models and Factored Language Models for Automatic Chord Recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 561–566, 2009.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Anssi Klapuri. Introduction to Muusic Transcription. In *Signal Processing Methods for Music Transcription*, pages 3–20. Springer, 2006.

Anssi P Klapuri. Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness. *IEEE Transactions on Speech and Audio Processing*, 11 (6):804–816, 2003.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification With Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based

Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A Tutorial on Energy-Based Learning. *Predicting structured data*, 1:1–10, 2006.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521 (7553):436–444, 2015.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient Backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012.

Daniel D Lee and H Sebastian Seung. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 556–562, 2001.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

Kyogu Lee and Malcolm Slaney. Automatic Chord Recognition from Audio Using a HMM with Supervised Learning. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 133–137, 2006.

Kyogu Lee and Malcolm Slaney. Acoustic Chord Transcription and Key Extraction from Audio Using Key-Dependent HMMs Trained on Synthesized Audio. *IEEE Transactions on Audio, Speech, and Language Processing (ICASSP)*, 16(2):291–301, 2008.

Bernhard Lehner, Gerhard Widmer, and Sebastian Bock. A Low-Latency, Real-Time-Capable Singing Voice Detection Method with LSTM Recurrent Neural Networks.

In *23rd European Signal Processing Conference (EUSIPCO)*, pages 21–25. IEEE, 2015.

Dawen Liang, Minshu Zhan, and Daniel P. W. Ellis. Content-Aware Collaborative Music Recommendation Using Pre-trained Neural Networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, pages 295–301, 2015. URL `http://ismir2015.uma.es/articles/290_Paper.pdf`.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in Network. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–6, 2013.

Matija Marolt. A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music. *IEEE Transactions on Multimedia.*, 6(3):439–449, 2004.

James Martens. Deep Learning via Hessian-Free Optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 735–742, 2010.

James Martens and Ilya Sutskever. Learning Recurrent Neural Networks with Hessian-Free Optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1033–1040, 2011.

Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The DET Curve in Assessment of Detection Task Performance. Technical report, DTIC Document, 1997.

Keith D Martin. A Blackboard System for Automatic Transcription of Simple Polyphonic Music. *Massachusetts Institute of Technology Media Laboratory Perceptual Computing Section Technical Report*, 385, 1996.

Matthias Mauch. *Automatic Chord Transcription from Audio Using Computational*

Models of Musical Context. PhD thesis, School of Electronic Engineering and Computer Science Queen Mary, University of London, 2010.

Matthias Mauch and Simon Dixon. Simultaneous Estimation of Chords and Musical Context from Audio. *IEEE Transactions on Audio, Speech, and Language Processing.*, 18(6):1280–1289, 2010.

Matthias Mauch and Sebastian Ewert. The Audio Degradation Toolbox and Its Application to Robustness Evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, 2013.

Matthias Mauch, Chris Cannam, Matthew Davies, Simon Dixon, Christopher Harte, Sefki Kolozali, Dan Tidhar, and Mark Sandler. OMRAS2 Metadata Project 2009. In *Proceedings of 10th International Conference on Music Information Retrieval (ISMIR)*, 2009.

Warren S McCulloch and Walter Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

Matt McVicar, Raúl Santos-Rodríguez, Yizhao Ni, and Tijl De 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, 2014.

Carlo Maria Medaglia and Alexandru Serbanati. An Overview of Privacy and Security Issues in the Internet of Things. In *The Internet of Things*, pages 389–395. Springer, 2010.

Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen. Acoustic Event Detection in Real Life Recordings. In *Proceedings of the 18th European Signal Processing Conference (EUSIPCO)*, pages 1267–1271. IEEE, 2010.

Annamaria Mesaros, Toni Heittola, and Anssi Klapuri. Latent Semantic Analysis in Sound Event Detection. In *19th European Signal Processing Conference (EUSIPCO).*, pages 1307–1311. IEEE, 2011.

Annamaria Mesaros, Toni Heittola, Onur Dikmen, and Tuomas Virtanen. Sound Event Detection in Real Life Recordings Using Coupled Matrix Factorization of Spectral Representations and Class Activity Annotations. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 151–155. IEEE, 2015.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent Neural Network Based Language Model. In *INTERSPEECH*, volume 2, page 3, 2010.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. Extensions of Recurrent Neural Network Language Model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

Marvin Minsky and Seymour Papert. Perceptrons. 1969.

Katsuhiko Miyamoto, Hirokazu Kameoka, Haruto Takeda, Takuya Nishimoto, and Shigeki Sagayama. Probabilistic Approach to Automatic Music Transcription from Audio Signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 697–700. IEEE, 2007.

Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep Belief Networks

for Phone Recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, volume 1, page 39, 2009.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2012.

Kevin P Murphy. Naive Bayes Classifiers. University Lecture Notes, 2006.

Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A Classification-Based Polyphonic Piano Transcription Approach Using Learned Feature Representations. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 175–180, 2011.

Radford M Neal. Connectionist Learning of Belief Networks. *Artificial Intelligence*, 56(1):71–113, 1992.

Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An End-to-End Machine Learning System for Harmonic Analysis of Music. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(6):1771–1783, 2012.

Ken O'Hanlon and Mark D Plumbley. Structure-aware Dictionary Learning with Harmonic Atoms. In *19th European Signal Processing Conference (EUSIPCO)*, pages 1761–1765. IEEE, 2011.

Ken O'Hanlon and Mark D Plumbley. Polyphonic Piano Transcription Using Non-negative Matrix Factorisation with Group Sparsity. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3112–3116. IEEE, 2014.

Ken O'Hanlon, Hidehisa Nagano, and Mark D Plumbley. Structured Sparsity for Automatic Music Transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*, pages 441–444. IEEE, 2012.

Bruno A Olshausen and David J Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

Aaron Van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1747–1756, 2016.

Laurent Oudre, Yves Grenier, and Cédric Févotte. Template-based Chord Recognition: Influence of the Chord Types. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 153–158, 2009.

Hélene Papadopoulos and Geoffroy Peeters. Large-scale Study of Chord Estimation Algorithms based on Chroma Representation and HMM. In *International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 53–60. IEEE, 2007.

Hélene Papadopoulos and Geoffroy Peeters. Simultaneous Estimation of Chord Progression and Downbeats from an Audio File. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*, pages 121–124. IEEE, 2008.

Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Inc., 2009.

Steffen Pauws. Musical Key Extraction from Audio. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2004.

Paul H Peeling and Simon J Godsill. Multiple Pitch Estimation Using Non-homogeneous Poisson Processes. *IEEE Journal of Selected Topics in Signal Processing.*, 5(6):1133–1143, 2011.

Geoffroy Peeters. Chroma-based Estimation of Musical Key from Audio-Signal Analysis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 115–120, 2006.

Antonio Pertusa and José M Iñesta. Polyphonic Monotimbral Music Transcription using Dynamic Tetworks. *Pattern Recognition Letters*, 26(12):1809–1818, 2005.

Graham E Poliner and Daniel PW Ellis. A Discriminative Model for Polyphonic Piano Transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1): 154–154, 2007.

Jose Portelo, Miguel Bugalho, Isabel Trancoso, Joao Neto, Alberto Abad, and Antonio Serralheiro. Non-speech Audio Event Detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1973–1976. IEEE, 2009.

Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of Speech Recognition. 1993.

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

Stanislaw Raczynski, Emmanuel Vincent, and Shigeki Sagayama. Dynamic Bayesian Networks for Symbolic Polyphonic Pitch Modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1830–1840, 2013.

Regunathan Radhakrishnan, Ajay Divakaran, and Paris Smaragdis. Audio Analysis for Surveillance Applications. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 158–161. IEEE, 2005.

Marc'Aurelio Ranzato, Sumin Chopra, Michael Auli, and Wojciech Zaremba. Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–8, 2016.

Christopher Raphael. A Graphical Model for Recognizing Sung Melodies. In *Pro-*

ceedings of the International Society for Music Information Retrieval Conference (ISMIR), pages 658–663, 2005.

Jeremy Reed, Yushi Ueda, Sabato Marco Siniscalchi, Yuuki Uchiyama, Shigeki Sagayama, and Chin-Hui Lee. Minimum Classification Error Training to Improve Isolated Chord Recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 609–614. Citeseer, 2009.

Douglas Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital signal processing*, 10(1):19–41, 2000.

Frank Rosenblatt. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65(6):386, 1958.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning Representations by Back-propagating Errors. *Cognitive modeling*, 5(3):1, 1988.

Matti P Ryynänen and Anssi Klapuri. Polyphonic Music Transcription using Note Event Modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 319–322. IEEE, 2005.

Matti P Ryynänen and Anssi P Klapuri. Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. *Computer Music Journal*, 32(3):72–86, 2008.

Tara N Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. Convolutional, Long Short-term Memory, Fully Connected Deep Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

Daichi Sakaue, Takuma Otsuka, Katsutoshi Itoyama, and Hiroshi G Okuno. Bayesian Non-negative Harmonic-temporal Factorization and its Application to Multipitch Analysis. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 91–96, 2012.

Daichi Sakaue, Takayuki Otsuka, Katsutoshi Itoyama, and Hiroshi G Okuno. Initialization-robust Bayesian Multipitch Analyzer Based on Psychoacoustical and Musical Criteria. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 226–230. IEEE, 2013.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. *arXiv preprint arXiv:1312.6120*, 2013.

Markus Schedl, Emilia Gómez, and Julián Urbano. Music Information Retrieval: Recent Developments and Applications. *Foundations and Trends in Information Retrieval*, 8(2-3):127–261, 2014. ISSN 1554-0669. doi: 10.1561/1500000042. URL `http://dx.doi.org/10.1561/1500000042`.

Jan Schlüter and Sebastian Böck. Improved Musical Onset Detection With Convolutional Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6979–6983. IEEE, 2014.

Jan Schlüter and Thomas 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)*, pages 21–126, Malaga, Spain, 2015.

Jürgen Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015.

Erik Schmidt and Youngmoo Kim. Learning Rhythm And Melody Features With Deep Belief Networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 21–26, 2013.

Mikkel N Schmidt and Morten Mørup. Sparse Non-negative Matrix Factor 2-D Deconvolution for Automatic Transcription of Polyphonic Music. Technical report, 2006.

Chistian Schörkhuber and Anssi Klapuri. Constant-Q Transform Toolbox for Music Processing. In *7th Sound and Music Computing Conf.*, Barcelona, Spain, July 2010.

Christian Schörkhuber, Anssi Klapuri, Nicki Holighaus, and Monika Dörfler. A MATLAB Toolbox for Efficient Perfect Reconstruction Time-Frequency Transforms with Log-Frequency Resolution. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

Mike Schuster. Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 589–595, 1999.

Alexander Sheh and Daniel PW Ellis. Chord Segmentation and Recognition Using EM-trained Hidden Markov Models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 185–191, 2003.

Arun Shenoy and Ye Wang. Key, Chord and Rhythm Tracking of Popular Music Recordings. *Computer Music Journal*, 29(3):75–86, 2005.

Roger N Shepard. Circularity in Judgments of Relative Pitch. *The Journal of the Acoustical Society of America*, 36(12):2346–2353, 1964.

Hava T Siegelmann. Computation Beyond the Turing Limit. *Science*, 268(5210): 545–548, 1995.

Siddharth Sigtia and Simon Dixon. Improved Music Feature Learning with Deep Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6959–6963. IEEE, 2014.

Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S. d'Avila Garcez, and Simon Dixon. An RNN-based Music Language Model for Improving Automatic Music Transcription. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 53–58, 2014.

Siddharth Sigtia, Emmanouil Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur S. d'Avila Garcez, and Simon Dixon. A Hybrid Recurrent Neural Network for Music Transcription. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2061–2065, Brisbane, Australia, April 2015a.

Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio Chord Recognition with a Hybrid Recurrent Neural Network. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–133, 2015b.

Siddharth Sigtia, Emmanouil Benetos, and Simon 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, 2016.

Renate Sitte and Liam Willets. Non-Speech Environmental Sound Identification for Surveillance using Self-Organizing Maps. In *Proceedings of the Fourth conference on IASTED International Conference: Signal Processing, Pattern Recognition, and Applications*, pages 281–286. ACTA Press, 2007.

Paris Smaragdis. Non-negative Matrix Factor Deconvolution; Extraction of Multiple

Sound Sources from Monophonic Inputs. In *Independent Component Analysis and Blind Signal Separation*, pages 494–499. Springer, 2004.

Paris Smaragdis and Judith C Brown. Non-negative Matrix Factorization for Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 177–180. IEEE, 2003.

Paris Smaragdis and Gautham Mysore. Separation by "Humming": User-guided Sound Extraction from Monophonic Mixtures. In *IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, pages 69–72, October 2009.

Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. A Probabilistic Latent Variable Model for Acoustic Modeling. *Advances in Models for Acoustic Processing, NIPS*, 148, 2006.

Steven W Smith. The Scientist and Engineer's Guide to Digital Signal Processing. 1997.

Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. Practical bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Mathias Stager, Paul Lukowicz, Niroshan Perera, Thomas Von Büren, Gerhard Tröster, and Thad Starner. SoundButton: Design of a Low Power Wearable Audio Classification System. In *Proceedings of the Seventh IEEE International Symposium on Wearable Computers*, pages 12–17. IEEE, 2005.

Dan Stowell and David Clayton. Acoustic Event Detection for Multiple Overlapping Similar Sources. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE, 2015.

Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D Plumbley. Detection and Classification of Acoustic Scenes and Events. *IEEE Transactions on Multimedia.*, 17(10):1733–1746, 2015.

Li Su and Yi-Hsuan Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research*, 2015.

Kouhei Sumi, Katsutoshi Itoyama, Kazuyoshi Yoshii, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Automatic Chord Recognition Based on Probabilistic Integration of Chord Transition and Bass Pitch Estimation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 39–44, 2008.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with

Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

Tiago Fernandes Tavares, Jayme Garcia Arnal Barbedo, Romis Attux, and Amauri Lopes. Survey on Automatic Transcription of Music. *Journal of the Brazilian Computer Society*, 19(4):589–604, 2013.

Andrey Temko, Robert Malkin, Christian Zieger, Dušan Macho, Climent Nadeu, and Maurizio Omologo. CLEAR Evaluation of Acoustic Event Detection and Classification Systems. In *Multimodal Technologies for Perception of Humans*, pages 311–322. Springer, 2006.

Ernst Terhardt. The Concept of Musical Consonance: A Link between Music and Psychoacoustics. *Music Perception: An Interdisciplinary Journal*, 1(3):276–295, 1984.

Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based Approach for Automatic Chord Detection Using Refined Acoustic Features. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP).*, pages 5518–5521. IEEE, 2010.

Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary Detection in Music Structure Analysis using Convolutional Neural Networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014.

John Wade Ulrich. The Analysis and Synthesis of Jazz by Computer. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 865–872, 1977.

Michel Vacher, François Portet, Anthony Fleury, and Norbert Noury. Challenges in the Processing of Audio Channels for Ambient Assisted Living. In *Proceedings of the 12th IEEE International Conference on e-Health Networking Applications and Services (Healthcom), 2010*, pages 330–337. IEEE, 2010.

Giuseppe Valenzise, Luigi Gerosa, Marco Tagliasacchi, E Antonacci, and Augusto Sarti. Scream and Gunshot Detection and Localization for Audio-Surveillance Systems. In *IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 21–26. IEEE, 2007.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep Content-Based Music Recommendation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2643–2651, 2013.

Matthias Varewyck, Johan Pauwels, and Jean-Pierre Martens. A Novel Chroma Representation of Polyphonic Music Based on Multiple Pitch Tracking Techniques. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 667–670. ACM, 2008.

Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Harmonic and Inharmonic Nonnegative Matrix Factorization for Polyphonic Pitch Transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 109–112. IEEE, 2008.

Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation. *IEEE Transactions on Audio, Speech, and Language Processing.*, 18(3):528–537, 2010.

Tuomas Virtanen. Separation of Sound Sources by Convolutive Sparse Coding. In *ISCA Tutorial and Research Workshop (ITRW) on Statistical and Perceptual Audio Processing*, 2004.

Alex Waibel, Hartwig Steusloff, Rainer Stiefelhagen, and Kym Watson. *Computers in the Human Interaction Loop*. Springer, 2009.

Gregory H Wakefield. Mathematical Representation of Joint Time-Chroma Distributions. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 637–645. International Society for Optics and Photonics, 1999.

DeLiang Wang and Guy J Brown. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006.

Adrian Weller, Daniel Ellis, and Tony Jebara. Structured Prediction Models for Chord Transcription of Music Audio. In *International Conference on Machine Learning and Applications (ICMLA)*, pages 590–595. IEEE, 2009.

Max Welling, Michal Rosen-Zvi, and Geoffrey E Hinton. Exponential Family Harmoniums with an Application to Information Retrieval. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1481–1488, 2004.

Paul J Werbos. Backpropagation Through Time: What it Does and How to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Chunghsin Yeh, Axel Roebel, and Xavier Rodet. Multiple Fundamental Frequency Estimation and Polyphony Inference of Polyphonic Music Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1116–1126, 2010.

Kazuyoshi Yoshii and Masataka Goto. A Nonparametric Bayesian Multipitch Ana-

lyzer Based on Infinite Latent Harmonic Allocation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):717–730, 2012.

Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 100–105, 2004.

Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

Xinquan Zhou and Alexander Lerch. Chord Detection Using Deep Learning. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, volume 53, pages 52–58, 2015.