Neural Probabilistic Models for Melody Prediction, Sequence Labelling and Classification

Srikanth Cherla

Department of Computer Science

City University London

This report is submitted for the purpose of

Doctor of Philosophy

City University London June 27, 2016

Contents

List of Figures

List of Tables

I would like to dedicate this thesis to my loving parents, whose unwavering support and faith in me have seen me through the hardest of times. This work would not have been possible otherwise.

Acknowledgements

I would, first and foremost, like to extend my sincerest gratitude to my supervisors Tillman Weyde and Artur Garcez, who gave me the opportunity to work on a topic of my choice and offered me advice and support whenever needed. I would like to extend a special thanks to Son Tran and Emmanouil Benetos, from whom I have learned a lot and without whose collaboration and regular feedback on my work, this dissertation would not be anywhere as good as it is. And I am also grateful to Daniel Wolff, Reinier de Valk and Siddharth Sigtia for the many interesting and enriching discussions on music and machine learning during the PhD. I would like to thank my examiners José Manuel Iñesta Quereda and Eduardo Alonso for their time and interest in my work, and their feedback which greatly helped improve the quality of the dissertation. I would like to thank Gregory Slabaugh and Emmanouil Benetos for reading and approving my MPhil-PhD transfer report. Darrell Conklin, Marcus Pearce and Tom Collins offered me very insightful advice at various stages of this work and I thank them for it.

It has been a pleasure to work and interact with some excellent people at City — senior tutor for research Stephanie Wilson, my colleagues Luca Pino, Andy Lambert, Muhammad Asad, Rishi Tank. I am also very appreciative of the staff at City University London for ensuring smooth sailing when it came to various forms of support — Mark Firman for being a good friend and an excellent department administrator, Gill Smith and Nathalie Chatelain for their help on many occasions, Chris Marshall and Shafiq Mundiya for their prompt response to any technical support requests, and Rapidity for printing all my posters.

I am grateful to my uncle Sita Rama Kumar, his wife Pavani and their daughter Trishala for so graciously welcoming me to live with them during the last months of the PhD. Thanks to Nina for her love and companionship, and to my friends Katerina, Jan, Maria, Carlos, Juanjo, Margarita, Dimitris, Mikhail, Georgios, Victor, and Mary for reminding me to take the much needed breaks from work.

I'm grateful for the financial support offered to me by City University London through a three-year studentship and travel grants in the course of the program.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures. I grant powers of discretion to the City University London librarian to allow the thesis to be copied in whole or in part without further reference to myself (the author). This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Srikanth Cherla
June 27, 2016

Abstract

Data-driven sequence models have long played a role in the analysis and generation of musical information. Such models are of interest in computational musicology, computer-aided music composition, and tools for music education among other applications. This dissertation begins with an experiment to model sequences of musical pitch in melodies with a class of purely data-driven predictive models collectively known as Connectionist models. It was demonstrated that a set of six such models could perform on par with, or better than state-of-the-art n-gram models previously evaluated in an identical setting. A new model known as the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM), was introduced in the process and found to outperform the rest of the models. A generalisation of this modelling task was also explored, and involved extending the set of musical features used as input by the models while still predicting pitch as before. The improvement in predictive performance which resulted from adding these new input features is encouraging for future work in this direction.

Based on the above success of the RTDRBM, its application was extended to a non-musical sequence labelling task, namely Optical Character Recognition. This extension involved a modification to the model's original prediction algorithm as a result of relaxing an assumption specific to the melody modelling task. The generalised model was evaluated on a benchmark dataset and compared against a set of 8 baseline models where it faired better than all of them. Furthermore, a theoretical extension to an existing model which was also employed in the above pitch prediction task - the Discriminative Restricted Boltzmann Machine (DRBM) - was proposed. This led to three new variants of the DRBM (which originally contained Logistic Sigmoid hidden layer activations), with Hyperbolic Tangent, Binomial and Rectified Linear hidden layer activations respectively. The first two of these have been evaluated here on the benchmark MNIST dataset and shown to perform on par with the original DRBM.

Chapter 1

Introduction

This chapter begins with an overview of the main task that is to be addressed in this dissertation — that of modelling sequential patterns in monophonic music. This task is of interest in several applications, many of which will be mentioned here to explain the relevance of this work, its goals, approach and results. The main objectives and research questions will be stated, and the extent to which these have been accomplished will be discussed. This will be followed by a list of original contributions proposed in this dissertation, published work, software developed in the process, and an outline of the rest of the document.

1.1 Problem Overview

Predictions can be made about anything that changes with time. Various natural phenomena exhibit perceivable patterns in time that allow one to make predictions about their evolution. For example, we are able to make in some cases, quite reliable predictions about the weather, the stock market, etc. either intuitively or by measurement. Music is one such phenomenon that is inherently temporal in nature and thus a candidate for the same predictive mechanisms that we apply to other time-varying phenomena. The interest in modelling the temporal properties of music using computers, both for analysis and synthesis, dates back to the 1950s with its roots in disciplines like language modelling and signal processing (Hiller and Isaacson, 1957). *Computer Music*, as it is commonly known, and the techniques employed in it have also evolved ever since. This section introduces the goals, methodology and motivation for music modelling by considering three key questions (1) What to predict? (2) How to predict? (3) Why predict?. The answers to these questions are presented with notable examples from previous work.

1.1.1 What to Predict?

Music, as considered in this dissertation, is a time-varying auditory phenomenon which exhibits structure and organisation. For example, a regular rhythmic pattern of "tick" sounds is musical although it does not have any variation in pitch. Likewise, a series of notes of equal-length (in time) and varying pitch is musical even though it does not have any rhythmic variation. In richer examples of music, structured variations across several such properties come together at any point in time to result in a range of musical forms that together make up a piece of music. These variations occur across different frequencies and at different temporal resolutions. They are perceived as variations in melody (pitch, scale-degree, etc.), harmony (chord type, chord voicing, etc.), rhythm (time-signature, metrical position, etc.) and timbre (zero-crossing rate, harmonicity, etc.). In this dissertation, we limit our attention to music in the form of melodies.

A system for modelling music is one that is able to take advantage of the said structural regularities and make *reasonable* predictions regarding the evolution in time of any musical property such as those mentioned earlier. The simplest case, involving the prediction of a single property is considered in (Pearce and Wiggins, 2004). There, the task is to predict a probability distribution over the possible values of pitch of the next note, given a sequence of pitches of notes that precede it. This was accomplished using n-gram models of varying context lengths (Manning and Schütze, 1999). A system which deals with a slightly more complex case is presented in (Cherla et al., 2013) (this author's MSc dissertation), where both note pitches and the corresponding note durations are taken together as a *(pitch, duration)* tuple to train a variable order Markov model (Begleiter et al., 2004) on sequences of such tuples. Given a recent history of tuples, the model makes predictions about the pitch and duration of the next note. A very similar approach was adopted for generating variations of drum loops in (Marchini and Purwins, 2010). There, the musical properties to be predicted are related to timbre and rhythm.

In a different prediction scenario which goes beyond just melodies, Toiviainen (1995) uses an auto-associator neural network for generating jazz solos of the style of Charlie Parker using the *target-note technique*. While the predicted musical properties here are purely melodic in nature (pitch, note duration, phrasing and dynamics), both melodic as well as harmonic information is used to make predictions. A different goal is considered in (Allan and Williams, 2004), where a Hidden Markov Model (Rabiner, 1989) is employed for harmonizing Bach chorales. Given the sequence of notes corresponding to the melody to be harmonised, the model suggests possible harmonisations with up to three voices at each time-step.

Thus, one can see how music is rich in internal structure and contains a multitude of properties that can be predicted and themselves be used for making accurate predictions about other properties. In the work presented in this dissertation, the task of interest is that of predicting musical pitch. **This dissertation focuses on models that predict the pitch of the next note in a melody given either just the pitch, or other melodic features of notes that precede it.**

1.1.2 How to Predict?

Once the musical properties of interest have been determined, the next choice is that of the prediction model. The idea is to view music as a programmable process and to have computers predict according to the process. One has to specify the class of the model, its parameters (which can often be *learned* from data), a way in which to input the musical properties of interest to the model and what it is expected to predict. These choices are influenced by the nature of the music modelling task of interest, and in turn influence the quality of predictions made.

Probably the most popular example where this is done is David Cope's system called *Experiments in Musical Intelligence* (EMI). EMI generates new music by analysing the score structure of a MIDI sequence in terms of recurring patterns (a signature), creating a database of the meaningful segments, and learning the style of a composer, given a certain number of pieces (Cope, 1996). Another example is evolutionary music - the audio counterpart of evolutionary art, which is based on the fundamental idea of a genetic algorithm. Here one starts with some initial music data (a piece, melody, or loop in audio or symbolic representation) which is initialised either randomly or based on human input. Then through the repeated application of computational steps analogous to biological selection, recombination and mutation, the aim is to produce more musical data. *GenJam* (Biles, 1994) is one such system developed for composing Jazz solos.

A dictionary-based prediction for automatic composition is discussed in (Assayag et al., 1999). Two dictionaries, namely, the *Motif Dictionary* and the *Continuation Dictionary* are used to represent and continue a given melody. A generation algorithm is used for continuation of a (so far) predicted sequence. A context variable is maintained which determines the maximum previous sequence to consider while making the prediction. The prediction is based on whether the context matches any of the motifs in the motif dictionary. The continuation dictionary gives the probabilities of various continuations and is used to choose the next symbol. Probabilistic context-free grammars are proposed for generating jazz solos in (Keller and Morrison, 2007). The process is divided into two stages — rhythm generation and melody generation on top of the generated rhythm. A grammar is first used to generate a coherent skeletal rhythmic sequence, which is then filled with notes. In order to connect pitch classes to underlying chord progressions, terminal symbols of the context-free grammar are interpreted as chord tones, colour tones, approach tones, and so on which are common in jazz parlance. Thus, a pitch is generated according to the chord with which it co-occurs.

Markov models have been very popular in research owing to the highly intuitive (and indeed simple) manner in which they model sequences as the ratio of the number of instances of a symbol following a given context and the number of instances of the context itself, given the sequence data. They are based on the simplifying assumption, known as the Markov assumption, that in a sequence of symbols where the probability of occurrence of the symbol at a certain time-step depends on those that have occurred before it, the influence of only those belonging to its immediate past (a "context") needs to be considered and not that of the entire sequence until the beginning. This assumption is introduced in greater detail in Section 3.5. It greatly simplifies the number of model parameters to be determined. Markov models happen to be one of the first class of models to be employed in modelling musical sequences in the early 1950s by Olson (1967) and were subsequently popularized as a model for music composition by Hiller & Isaacson (1979) in the fourth movement (also known as *"Experiment Four"*) of their *Illiac Suite*. Ever since these early applications, there have been several others using this method in a variety of ways (Ames, 1989). However, the number of parameters to be determined for Markov models increases exponentially with the length of the context in the model. A solution to this problem is proposed in *The Continuator* (Pachet, 2003), which operates mainly on short melodic phrases. Sequences of symbols interpreted from MIDI input are parsed using an incremental parsing algorithm to train a variable-order Markov model (Ron et al., 1996) that maintains various possible sequences of symbols and their probabilities of occurrence. The system progressively learns new phrases from a musician to eventually develop a more accurate representation of her/his style. A variant of the basic Markov model is the Hidden Markov Model (HMM). In the work of Paiement (2008), a total of three interdependent HMMs are used to break down the process of music generation into sub-tasks. The first one models the underlying rhythm of a MIDI melody, the second the intervallic variations given the rhythm, and the third predicts pitches that satisfy constraints imposed by an input chord progression and the intervals predicted by the second HMM.

This dissertation explores Connectionist models (Rumelhart et al., 1986) for music prediction. These have been successfully applied in the past to learning harmonic style (Bellgard and Tsang, 1994; Hörnel and Menzel, 1998), Jazz solo generation (Toiviainen, 1995), rhythm analysis (Weyde and Dalinghaus, 2003; Battenberg and Wessel, 2012), music composition (Todd, 1989; Mozer, 1991), and polyphonic music generation (Boulanger-Lewandowski et al., 2012). Among these the relevant approaches will be reviewed in greater detail in Chapter 2.

1.1.3 Why Predict?

Meyer (1956) hypothesised in his influential theory on musical expectation that prediction is at the very heart of music cognition, and involves the listener generating expectations about the evolution of a given piece of music which are either satisfied, delayed or thwarted as determined by the composer and the performer. This results in the range of emotions and notions of musical style that is available to us. Other theoretical expositions building upon and refining this idea have also been proposed over the years (Narmour, 1992; Huron, 2006). Musical expectation has been studied using any of three different means (1) behavioural findings (2) computational modelling, and (3) neuroscientific evidence (Rohrmeier and Koelsch, 2012). Predictive models of music cognition, which belong to the second category, allow the verification of theoretical accounts of expectation and their consequences in novel contexts through a concrete realisation of the underlying theory (Wiggins et al., 2011). Studies in music cognition research require the generation of specific types of stimuli in order to verify a certain hypothesis regarding music cognition (Omigie et al., 2013; Egermann et al., 2013). The more comprehensive these stimuli, the greater the confidence one can achieve in the results of an experiment. In such situations, generative models which can learn structure in music to produce such stimuli accurately on demand serve as valuable tools. They have the potential to provide a link between theoretical, neural and behavioural accounts of music prediction (Rohrmeier and Koelsch, 2012). Another potential application of predictive models of music is inspired by the role of language models in Speech Recognition (Rabiner, 1989). Automatic Music Transcription (AMT) (Klapuri and Davy, 2007) involves transcribing music given as an music audio signal into an equivalent digital representation of its musical score, much as Speech Recognition involves transcribing a given audio signal of a spoken sentence into a sequence of words. Here a predictive model of music, also known as a *music language model*, that captures sequential structure in collections of musical scores is used to augment the predictions of an acoustic model (that relies purely on the instantaneous musical signal) with those of its own to determine the musical notes occurring at each time-instant. This has been successfully demonstrated in the context of music with the help of Hidden Markov Models (Ryynänen and Klapuri, 2008) and more recently, recurrent Connectionist models (Boulanger-Lewandowski et al., 2012; Sigtia et al., 2014).

Models for predicting music can be employed in generating ambient music in shopping malls, restaurants, museums and other private/public venues (Moo; Amb; Hol). The aim is to create a pleasant and welcoming environment for visitors to such places in the hope of improving their overall aesthetic experience there. A very similar idea is also applicable to context-sensitive music generation in computer games (Land and McConnell, 1994; Casella and Paiva, 2001; Brown, 2012). Given parameters that define various instantaneous situational variables during gameplay, predictive models of music can be triggered to generate appropriate background music that can make the game more engaging and immersive for a player.

Algorithmic composition tools that employ predictive models of music can also facilitate a shift of focus of the composer from fine details to the bigger picture. It is sufficient if there exists an idea in the mind of a composer about what she/he wishes the composition to convey and how it should sound, in a broad sense without attention to the finer details, as long as there is a suitable means (an interface) for her/him to communicate this to a machine. A system that is able to generate musical ideas in the form of a score can be of use in compositional assistance for amateur musicians and students of music. Rapid compositional prototyping tools such as *RapidComposer* (Mus) already allow for such possibilities with computer programs for tasks like automatic phrase, rhythm and chord generation. Other systems allow the creation of royalty-free music on demand for video content creators with the aid of predictive models of music (Juk).

Another possible, and little explored application area for predictive models of music is in medicine. Music therapy uses music for achieving certain therapeutic objectives in order to meet an individual's physical, emotional, mental, social and cognitive needs. It aims to develop potentials and/or restore functions of the individual which will improve her/his quality of life through prevention, rehabilitation or treatment (Darnley-Smith and Patey, 2003). The use of *intelligent systems* in music therapy has received very limited attention in the past (Erkkilä et al., 2004). This author is of the opinion that computational methods for music analysis and generation have great potential to assist music therapy practitioners and empower individuals undergoing rehabilitation.

1.2 Objectives

While the various tasks in music modelling are indeed diverse just as the range of techniques available to address these tasks as described above, the work presented in this dissertation commenced with the goal to extend that which originated in the framework for music analysis and generation known as *Multiple Viewpoints for Music Prediction* (Conklin and Witten, 1995). The key characteristics of this approach are (1) an event-based representation of music, (2) the use of statistical modelling techniques proven to be successful in natural language processing and text compression to model musical structure, and (3) combining multiple models to incorporate information from several musical features to predict a chosen feature. This work was originally introduced in (Conklin, 1990) and further extended in (Pearce, 2005; Whorley et al., 2013). As much as there exist provisions within the multiple viewpoints approach to deal with polyphonic music, the focus in the present work is limited only to monophonic music modelling.

In the original work on multiple viewpoints and that which followed, Markov models were exclusively employed for music modelling using this framework. While this is a reasonable choice, Markov models are often faced with a problem related to data sparsity known as the *curse of dimensionality* (Bengio et al., 2003). This refers to the exponential rise in the number of model parameters to be estimated with the length of the modelled sequences as a result of treating each probability distribution that can be predicted by the model as one of its parameters. Models belonging to the Connectionist class (such as neural networks) bypass this problem, as they do not require enumerating all state transition probabilities and rely on a set of abstract parameters from which these probabilities can be derived.

The aforementioned lure of connectionist models motivated their adoption into language modelling during the past decade (Bengio et al., 2003; Mnih and Hinton, 2008; Collobert et al., 2011; Socher et al., 2011; Mikolov and Zweig, 2012), and the result of this was a demonstration of their superiority over the prevailing state-of-the-art n-gram models in word prediction tasks. It was thus decided at the outset of the present work to study the efficacy of connectionist models in mirroring this success in the domain of monophonic music modelling as defined within the framework of *Multiple Viewpoints*. It was also suggested in (Conklin and Cleary, 1988; Whorley et al., 2013) that neural networks might be suitable alternatives to n-gram models for music modelling with multiple viewpoints but very little research in this direction has ensued (Cox, 2010). This led to the three research questions listed below.

1.2.1 Research Questions

1. Given a previous study that evaluated several n-gram and variable order Markov models at a pitch prediction task (Pearce and Wiggins, 2004) (*cf.* Section 3.5) on a corpus of monophonic music, how do common connectionist architectures compare with them? This question is explored in Chapters 4 and 5.

2. How does combining multiple connectionist models each of which relies on different sets of melodic features as input compare with a single model that incorporates all these features, on the same pitch prediction task. This question is explored in Chapter 6.

3. Can the above questions lead to novel contributions that go beyond the said application domain into the area of machine learning, which the modelling approach in this dissertation is based on? The answer to this question is in Chapters 5 and 7.

1.3 Original Contributions

The following are the key contributions of the work presented in this dissertation:

1. Demonstration of the efficacy of a set of non-recurrent and recurrent connectionist models in outperforming n-gram models on the task of monophonic music modelling and prediction, following an approach inspired by language modelling.

2. Proposal of a new machine learning model known as the Recurrent Temporal Discriminative Restricted Boltzmann Machine and its application to both musical and non-musical sequence learning tasks.

3. Extension of the theory underlying an existing model, known as the Discriminative Restricted Boltzmann Machine, and experiments to evaluate the practical benefits of these extensions.

At the outset, the goal of this dissertation was to demonstrate the efficacy of connectionist models at modelling sequences of pitch in monophonic music in the context of an information theoretic approach that was previously exemplified through the use of *n*-gram models in (Conklin and Witten, 1995; Pearce, 2005; Whorley et al., 2013). During the first phase of the work summarised in this thesis (Chapters 4 and 5), a comparative evaluation of six different connectionist models was carried out, namely the Feed-forward Neural Network (FNN), the Restricted Boltzmann Machine (RBM), the discriminative RBM (DRBM), the Recurrent Neural Network (RNN), the Recurrent Temporal Restricted Boltzmann machine (RTRBM) and the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM). It was demonstrated through this evaluation that these connectionist architectures can perform on par with, or better than the best of *n*-gram models previously evaluated on a corpus of folk and chorale melodies (Pearce and Wiggins, 2004).

The last of the models listed above — the RTDRBM, is a novel contribution of the present work. While the RTDRBM and RTRBM have the same structure, the two differ in how the values of their respective parameters are determined. As it shall be explained in Chapter 5, the RTRBM is what is known as a *generative* model, and the RTDRBM is its *discriminative* counterpart. In practice, the choice of either is determined by the application and the amount of data available to estimate their parameters (Ng and Jordan, 2001; Bishop and Lasserre, 2007). While the former was originally introduced to generate sequences, the intended purpose of the latter is to label sequences. The difference between the RTRBM and the RTDRBM is also akin to that between the RBM and DRBM which are also structurally identical, but have different modelling goals. The RTDRBM was shown to outperform all other models evaluated in the above mentioned pitch prediction task (*cf.* Chapter 5), and also a set of baseline models evaluated in (Nguyen and Guo, 2007) on the benchmark OCR dataset (Kassel, 1995; Taskar et al., 2004).

Another novel contribution of the present work is in relation to a previously proposed classification model, namely the Discriminative Restricted Boltzmann Machine (DRBM). This was based on the observation that while the performance of a variety of hidden layer activations (Logistic Sigmoid, Hyperbolic Tangent, etc.) has been explored with other connectionist models and their validity theoretically or empirically verified, the same had not been done with the DRBM which relied only on Logistic Sigmoid activations. On further investigation into the possibility of making the same extensions to the DRBM, it could be proved in theory that Logistic Sigmoid, Hyperbolic Tangent, Binomial and Rectified Linear hidden layer activations in the DRBM are all special cases of the same general form and can be derived in a very similar manner using this underlying form. The first two out of these three theoretical extensions were also experimentally evaluated, and it was found that DRBMs containing either Hyperbolic Tangent or Binomial hidden layers performed on par with that containing a Logistic Sigmoid hidden layer.

1.4 Publications

Below is a list of research papers published during the past three years in conferences. Parts of this dissertation are based on some of these:

1. Cherla S., Tran S.N., Weyde T. and d'Avila Garcez A., *"Hybrid Long- and Short-Term Models of Folk Melodies"*. In: Proc. International Society for Music Information Retrieval Conference, 2015.

2. Cherla S., Tran S.N., d'Avila Garcez A. and Weyde T., *"Discriminative Learning and Inference in the Recurrent Temporal RBM for Melody Modelling"*. In: Proc. International Joint Conference on Neural Networks, 2015.

3. Tidhar D., Benetos E., Wolff D., Dumon E., Cherla S. and Weyde T., *"Incremental Dataset Definitions for Large Scale Musicological Research"*. In: Proc. Digital Libraries for Musicology Workshop, 2014.

4. Cherla S., Weyde T. and d'Avila Garcez A., *"Multiple Viewpoint Melodic Prediction with Fixed-Context Neural Networks"*. In: Proc. International Society for Music Information Retrieval Conference, 2014.

5. Sigtia S., Cherla S., Benetos E., Dixon S., Weyde T., and d'Avila Garcez A., *"RNN-based Music Language Models for Improving Automatic Music Transcription"*. In: Proc. International Society for Music Information Retrieval, 2014.

6. Cherla S., Weyde T. and d'Avila Garcez A., *"MIR in Music Education Wiki"*. In: DMRN+8: Digital Music Research Network One-day Workshop, 2013.

7. Cherla S., Weyde T., d'Avila Garcez A. and Pearce M., *"A Distributed Model for Multiple Viewpoint Melodic Prediction"*. In: Proc. International Society for Music Information Retrieval Conference, pp. 15-21, 2013. ***(Best Student Paper Prize)***

8. Cherla S., d'Avila Garcez A. and Weyde T., *"A Neural Probabilistic Model for Predicting Melodic Sequences"*. In: Proc. 6th International Workshop on Machine Learning and Music, 2013.

9. Benetos E., Cherla S. and Weyde T., *"An Efficient Shift-Invariant Model for Polyphonic Music Transcription"*. In: Proc. 6th International Workshop on Machine Learning and Music, 2013.

1.5 Software

The following code, related to the work presented in this dissertation is available on Bitbucket (https://bitbucket.org):

1. **Connectionist Melody Models:** A library for training and evaluating connectionist models for modelling sequences in musical melodies through prediction implemented in Python (Chapters 4 and 5).
   Link: https://bitbucket.org/freakanth/connectionist-melody-models.git

2. **Recurrent Temporal Discriminative RBM:** An implementation in Python of the Recurrent Temporal Discriminative Restricted Boltzmann Machine introduced in this dissertation (Chapters 5 and 7).
   Link: https://bitbucket.org/freakanth/drbm-generalisation.git

3. **Discriminative RBM:** An implementation in Python of the extensions proposed in this dissertation to the Discriminative Restricted Boltzmann Machine (Chapter 7).
   Link: https://bitbucket.org/freakanth/drbm-generalisation.git

1.6 Dissertation Outline

The next chapter presents a review of literature on music modelling relevant to the present work. It contains an introduction to music in the symbolic form, previous work in information theoretic music modelling and connectionist models for learning structure in music. This is followed by an overview of various concepts and definitions in Chapter 3, which will help the reader better understand the description of the connectionist melody models described in Chapters 4 and 5. Chapter 6 presents the techniques and results of combining the models presented in the preceding chapters in an attempt to improve prediction performance. Chapter 7 extends a novel connectionist architecture (the RTDRBM) first introduced in Chapter 5 for melody prediction, and carries out an evaluation of this model on a more general sequence labelling tasks. This chapter also presents theoretical extensions proposed in the present work to one of the models employed for melody prediction in Chapter 4, and experiments to evaluate the significance of these proposals. This leads to the conclusions of the dissertation in Chapter 8 with comments on directions for future work.

Chapter 2

Related Work

The inspiration for the work reported in this dissertation, at the outset, came from an existing body of literature on information theoretic methods for modelling musical structure where the interest is in establishing a connection between mathematical quantities that measure the amount of information and redundancy in musical structure with elements of musical style, mood and the cognition of music (Meyer, 1957; Youngblood, 1958; Conklin and Witten, 1995; Huron, 2006). The final goal of such work is generally two-fold — the analysis of existing pieces of music and also the synthesis of new music by example. Based on the observation that the use of connectionist models for information theoretic music modelling has received little attention in recent years, with the majority turning to n-gram and variable order Markov models for addressing this task, the present work commenced with a comparative study between the predictive performance of n-gram models and their connectionist counterparts for sequences of musical pitch. This prediction task is analogous to the better known word prediction task in language modelling (Brown et al., 1992), and the evaluated *melody models* are inspired by literature on connectionist language models (Schwenk and Gauvain, 2002). An added motivation for exploring connectionist models in the present work stems from the increased interest in connectionist research during the past decade, associated with the *Deep Learning* movement (Lecun et al., 2015), which has led to new and interesting insights and architectures implementing these insights for addressing various tasks, not excluding music and language modelling. The review of literature presented in this chapter begins with an overview of music in the symbolic form, which the data used in the present work is comprised of. This is followed by a review of the information theoretic roots of the music modelling task considered here, and of the previous musical applications of connectionist theory and architectures which have also been employed here.

2.1 Music in the Symbolic Form

The symbolic form is one of the many ways in which music may be represented. In the field of Music Information Retrieval, it serves to complement the musical information contained in the audio form. The preference of one over the other is determined mainly by the end goal as each is more suitable than the other for extracting certain kinds of information. The audio form is mainly intended for acoustic analysis of the musical signal, while the symbolic form is oriented towards musicological analysis of the musical score. The symbolic form is closer to the composer and serves as a means for her/him to express musical ideas. And the audio form is what is heard by the listener. In between the two is the performer who interprets what the composer has written down in the score and creates music which is recorded as a signal and perceived by the listener. An audio recording of a performance thus contains all the information about a musical work that is carried by a symbolic score, and additionally, new information corresponding to the performer's personal interpretation according to the degrees of freedom left by the symbolic score which, more often than not, involves remaining faithful to dimensions like rhythm, melody, and harmony. Analysis of audio data begins with the musical signal, which is transformed into a more informative numerical representation such as the spectrogram before any analysis is carried out on it. On the other hand, the analysis of symbolic data begins with an explicit knowledge of basic musicological information (contained in the corresponding musical score) which can be used as is or other information derived from it. There exist special file-types and corresponding algorithms to parse these file-types for both audio and symbolic music data. While the gap between applications relying on one or the other type of data representation is indeed diminishing, some typical applications where predominantly audio data is used, are music classification (Scaringella et al., 2006; Fu et al., 2011) and audio cover song identification (Serra et al., 2010), whereas in the case of symbolic data typical applications are algorithmic composition (Nierhaus, 2009) and predictive modelling of music (Rohrmeier and Koelsch, 2012). There also exist applications in between, such as automatic music transcription (Klapuri and Davy, 2007) which deal with both symbolic and audio data. This dissertation deals purely with music data in the symbolic form.

2.1.1 The Musical Score

Music in the symbolic form essentially refers to the content of a musical score. The musical score is a representation by means of which the composer communicates to the musicians, which musical gestures have to be performed. It is a structured organization of symbols which describe acoustic events and the gestures needed for their production, and is the form for representing a musical work as a composition. The score is thus a symbolic description of a musical work that allows musicians to produce a correct performance, and serves as the ideal version of a musical work. The central role played by this symbolic representation, which is the main tool used by musicologists in their research, is to provide ready access to musical parameters that are easily and directly represented in symbolic notation. These parameters can range from global ones such as key- and time-signature, tempo, repetitions, etc. to local ones such as musical pitches and durations of notes, their intensities, etc (Orio, 2006a). From local parameters it is possible to extract information about the melody, harmony, and the rhythm. Information about sound intensity, or loudness, is usually vague and expressed in a subjective scale from very soft (***ppp***) to very loud (***fff***) sound. The symbolic score carries almost no information about certain other relevant musical dimensions, in particular timbre, orchestration and arrangement, articulation, room acoustics, and spatialization of sound sources, which all play a fundamental role in the experience of music listening and are related to music as a performing art. Figure 2.1 illustrates an example musical score of the four-part harmonisation of a hymn.

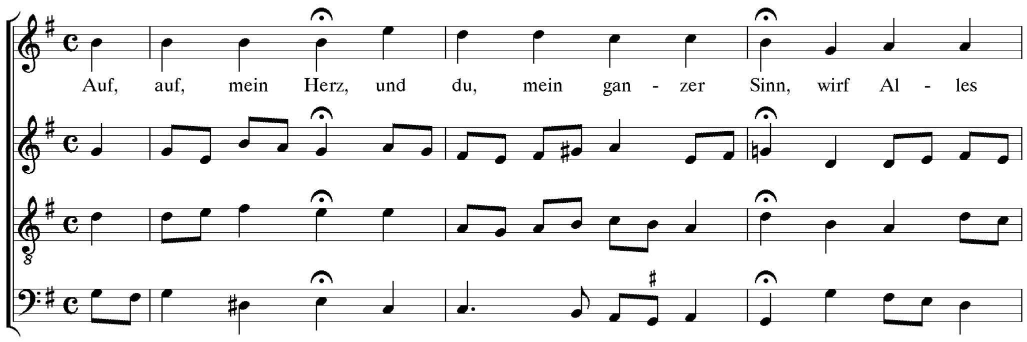

Auf, auf, mein Herz, und du, mein gan - zer Sinn, wirf Al - les

Fig. 2.1 The musical score of the first four measures from the four-part harmonised hymn “*Auf, auf, mein Herz, und du mein ganzer Sinn*” written by Sigmund von Birken, and harmonised by Johann Sebastian Bach.

2.1.2 Digital Representations of the Musical Score

The motivations for digitising musical scores, which were originally available as hand-written or printed text, is to facilitate wider access through (online) digital libraries, compatibility with computer programs for music analysis and synthesis, and large scale dissemination of musical information (Orio, 2006b). Digital representations of musical notation are of value to students learning music using computer software (Dittmar et al., 2012), musicologists interested in the computational analysis of music (Huron, 2002), and music composers of the Western tradition (Mak). They are also of use to composers dealing with the automatic generation of music with the aid of computers, as seen in the prolific area of Algorithmic Composition (Nierhaus, 2009). The fact that various musicological features are explicitly represented in these formats (unlike audio where they must be obtained through analysis of the signal), makes them suitable for this purpose.

Examples of well-known formats for digitally representing musical scores are ABC (Walshaw), which is mainly used to code simple monophonic scores of traditional music, GUIDO and MuseData (Hewlett, 1997), Lilypond (Nienhuys and Nieuwenhuizen, 2003) which is an open source project, and Kern (Huron, 2002). An interesting effort in music representation, which partially builds upon MuseData, is MusicXML (Good, 2001) that exploits the powerful features of XML as a portable format for information exchange. MusicXML is supported both by commercial software and by open source projects. The Music Encoding Initiative (MEI) (Roland, 2002) is a parallel open source project that also adopts XML as the schema of choice in order to represent musical documents. It aims to mirror the success of the Text Encoding Initiative (TEI) whose goal was to create a comprehensive yet extensible standard for the encoding and transmission of textual documents in electronic form. Other popular symbolic data formats which are more oriented towards synthesis, rather than analysis, are MIDI and Guitar Pro (Bellini et al., 2005). Using the appropriate parser for a file-type one can retrieve this information for playback or computational analysis from file encoded in the above formats.

2.1.3 The `**kern` format and `music21` Python Library

The data used in the present work is represented in `**kern` files. The `**kern` representation format is designed to encode purely syntactic information contained in a musical score for analysis purposes. It allows encoding of pitch (*e.g.,* concert pitch, accidentals, clefs, key signatures, harmonics, glissandi, etc.), duration (*e.g.,*, canonic musical duration, rests, augmentation dots, grace notes, time signature and tempo), articulation (*e.g.,* fermata, trills, accents), timbre (*e.g.,* instrument and instrument class), and many other structural components of a score (*e.g.,* phrase markings, bar lines, repetitions, etc.) (Huron, 1997; Pearce, 2005). Figure 2.2 shows the `**kern` file corresponding to the score in Figure 2.1.

```
!!!COM: Bach, Johann Sebastian
!!!CDT: 1685/02/21/-1750/07/28/
!!!OTL@@DE: Auf, auf, mein Herz, und du mein ganzer Sinn
!!!AGN: chorale
**kern  **kern  **kern  **kern
*ICvox  *ICvox  *ICvox  *ICvox
*Ibass  *Itenor *Ialto  *Isoprn
*I"Bass *I"Tenor    *I"Alto *I"Soprano
*clefF4 *clefGv2    *clefG2 *clefG2
*k[f#]  *k[f#]  *k[f#]  *k[f#]
*G: *G: *G: *G:
*M4/4   *M4/4   *M4/4   *M4/4
*met(c) *met(c) *met(c) *met(c)
*MM100  *MM100  *MM100  *MM100
8GL 4d  4g  4b
8F#J    .   .   .
=1  =1  =1  =1
4G  8dL 8gL 4b
.   8eJ 8eJ .
4D# 4f# 8bL 4b
.   .   8aJ .
4E; 4e; 4g; 4b;
4C  4e  8aL 4ee
.   .   8gJ .
=2  =2  =2  =2
4.C 8AL 8f#L    4ddnX
.   8G  8e  .
.   8A  8f# 4dd
8BB 8BJ 8g#J    .
8AAL    8cL 4a  4cc
8GGJ    8BJ .   .
4AA 4A  8eL 4cc
.   .   8f#J    .
=3  =3  =3  =3
4GG;    4d; 4gnX;   4b;
4G  4B  4d  4g
8F#L    4A  8dL 4a
8EJ .   8eJ .
4D  8dL 8f#L    4a
.   8cJ 8eJ .
==  ==  ==  ==
!!!YOR1: 371 vierstimmige Choralges&auml;nge von Johann Sebastian Bach,
!!!EED:  Craig Stuart Sapp
!!!EEV:  2009/05/22
```

Fig. 2.2 The Kern file corresponding to the score in Figure 2.1 ("*Auf, auf, mein Herz, und du mein ganzer Sinn*" by Johann Sebastian Bach).

The musical information contained in these `**kern` files is extracted using the `music21` Python library (Cuthbert and Ariza, 2010). As stated by its creators, `music21` is an "object-oriented toolkit for analysing, searching, and transforming music in symbolic (score-based) forms". This library provides users with a common interface for parsing a variety of symbolic music file-types. There exist functions in the library to directly extract several of the above mentioned musicological features, while others (*cf.* Section 3.1) can be extracted with additional scripts written using this library.

2.2 Music and Information Theory

The foundations of modern information theory were laid in the seminal work of Shannon (1948). The key idea that led to it influencing a range of disciplines, not excluding music, was concerned with the measurement of information content and uncertainty through a quantity known as *entropy*. This quantity can be defined in the context of a *communication system* consisting of a source that encodes and transmits information as a time-varying signal across a channel to a receiver which then decodes the signal. In the literature pertaining to music and information theory reviewed below, of interest is what is known as a discrete noiseless system which assumes that (1) the signal is a sequence of discrete symbols, each chosen from a finite set (such as characters in the English language, words in a dictionary, or tones in the chromatic scale), (2) that the channel does not contribute noise that in any way modifies the signals, and (3) the signal is generated by a probabilistic Markov process at the source such that the probability of a certain symbol occurring at each location in the sequence depends on those that have occurred prior to it. Given these assumptions, the entropy at a certain location in the sequence quantifies the uncertainty regarding the value of the symbol that could occur at that location. Entropy is highest when one has no information regarding the next symbol, i.e. all symbols are equally likely to occur with the same probability. On the contrary, it is lowest when it can be determined exactly which symbol will occur i.e., that symbol will occur with a probability of 1. The value of entropy typically lies between 0 and an upper limit which is determined by, and increases with, the number of possible symbols.

Shannon demonstrated these ideas in the context of sequences of characters of the English language where, by increasing the number of immediately preceding symbols which are considered while speculating about the next symbol, one can reduce the entropy or equivalently make more informed predictions about the next symbol (Shannon, 1951). One can analogously replace the said characters of the English alphabet with musical symbols corresponding to pitches in the chromatic scale, durations of notes, or combinations of these and other musical dimensions thus opening up various analytical possibilities of music with the help of mathematics and computers. This is perhaps what inspired several researchers to undertake the information theoretic analysis of music. Early interdisciplinary work in music and information theory (Meyer, 1957; Youngblood, 1958; Cohen, 1962) attempted to quantify various musical phenomena such as style, genre and emotion with the help of information theoretic quantities such as entropy and mutual information. These techniques evolved through the years both in terms of the questions they intended to answer, as well as the techniques employed in order to answer these questions more precisely. In this section, we review some notable early information theoretic approaches for music analysis followed by a review of the Multiple Viewpoints approach and two of its derivatives that are relevant to the present work.

2.2.1 Early Work

One of the first scholars to discuss the potential importance of information theory in the study of music was Meyer (1957), best known for his exposition on the theory which relates musical expectation with emotion and meaning in music (Meyer, 1956). In a philosophical discourse on the *Meaning in Music and Information Theory*, he developed an analogy wherein the process of performance, propagation and perception of music can be viewed as a communication system. According to Meyer, a piece of music contains information that is propagated between a source (the performer) and a receiver (the listener) through a channel (the acoustic space through which music propagates). The composer can be seen as the probabilistic process that generates the sequence of musical symbols. Musical style, in this context, is regarded as a sequence of probabilities belonging not just to the music, but to the habit responses of the composer, performer and the practiced listener as well. Expectations are elicited in the listener by music of a certain style only when it deviates from the "norms" that contain the former's habit responses, Otherwise the music tends to go unnoticed, and the expectations are said to be *latent*. Meaning is conveyed through these expectations when they are satisfied, delayed or thwarted at the will of the composer. It is this that creates an uncertainty in the listener that entropy can be associated with. Meyer claimed that music communicates what is known as *embodied meaning* through this uncertainty. Two different sources of musical noise are discussed in the exposition - the first is physical noise which relates to sounds that might mask the music itself from being heard by the listener, while the second is cultural noise which refers to the disparities between the habit responses required by the musical style and those which a given listener actually possesses.

Another early theoretical account on the subject was provided by Cohen (1962), whose view of both the composer and the listener as probability systems is very similar to that of Meyer (1957) but is expressed with greater emphasis on the influence of culture on the creation and assimilation of music. In this respect, it also highlights a distinction akin to Meyer's *embodied* and *designative* meanings of music. This account, having been published a few years following (Meyer, 1957), presents a review of analysis-synthesis work in music and information theory carried out in this period in order to illustrate the theoretical ideas. It also highlights the limiting assumptions of the information theoretic approach for music analysis and synthesis.

An abstract, more mathematical and intuitive interpretation of information theory in the context of music was presented in (Coons and Kraehenbuehl, 1958). Their conclusion regarding the purpose of structural organisation in music echoes that of Meyer, that music is to contain patterns that will first attract a listener's attention by presenting patterns which are informative (or contain some degree of surprise) and thus eliciting expectations regarding what is to ensue, and then reward this attention by satisfying these expectations with patterns of lower information content (or lesser degree of surprise). This variety is to be achieved while still maintaining a unity across a composition, to measure which they propose two information theoretic quantities — *index of articulateness* and *index of hierarchy*. The former is the average change in information over the course of a musical pattern. It measures how neatly the conditions of unity and variety have been arranged so that the force of neither is dulled (Coons and Kraehenbuehl, 1958). The latter is given by the average reduction of information by a pattern from beginning to end. It also assigns a higher score to patterns where the reduction is sudden as opposed to gradual. As stated by the authors, the index of hierarchy is a measure of how successfully a variety of events has been arranged to leave an impression of unity. Their claims are accompanied by numerical interpretations and simple examples wherever applicable.

In a continuation of the above work (Kraehenbuehl and Coons, 1959), information theoretic justifications for various commonly adopted compositional practices are provided. Different sequences of three or less symbols ($AAAB$, $AABA$, $ABAC$, etc.) are rank-ordered according to their two indices of hierarchy and articulatedness. Assuming that these symbol sequences reflect song structures and set out to explain how their positions in the rank-ordering might reflect their historic use in composition, and also suggest in which contexts other less occurring sequences might be suitable. For instance, the ordering justifies the very frequent use of the $AABA$ song structure in popular music, and allows them to speculate that structures such as $ABAC$ and $ABBA$ are open-ended and better suited at the beginning of long movements. The two indices also allow the authors to justify the preference of structural patterns of four rather than three by composers. It was also observed that a reduction in information is commonly associated with a sense of beginning in music, while a gain in information contributes to a sense of continuing. And finally, this work also offers an information theoretic explanation regarding the "brightness" of the major triad and the "darkness" of the minor triad.

In an early attempt to quantify musical style, Youngblood (1958) focused on the quantity known as Relative Entropy in order to analyse melodies from Gregorian chants and selected musical works of three composers of the Romantic era, namely Robert Schumann, Felix Mendelssohn and Franz Schubert. The analysis is limited to zeroth- and first-order probability distributions of the twelve tones of the chromatic scale. The average values of relative entropy, which is simply the ratio of the measured entropy and the maximum possible entropy given a certain number of symbols, across a composer's music is interpreted as a measure of the degree of restraint practiced by a composer in using all possible tones and tone pairs in a given composition. This has been referred to as *redundancy* here and in other work. It was noted here through a comparative analysis of average relative entropies, how in the music of the three composers one can observe the differences in their respective use of chromaticism. In comparison to the music of the three Romantic composers, it was found that tones other than the tonic and the dominant were more evenly distributed in the Gregorian chants. The author also suggested that the analysis of harmony and inclusion of other informative musical dimensions such as rhythm, position in phrase and modulation might lead to better insights into the similarities and differences between the styles of the three composers.

Hiller and his collaborators carried out further information theoretic experiments in this direction. The earliest of these (Hiller and Bean, 1966), examined four sonatas each composed by one of four stylistically distinct composers, namely Wolfgang Amadeus Mozart, Ludwig van Beethoven, Alban Berg and Paul Hindemith. The goal of this early research was to determine the ability of information theoretic measures to reflect the stylistic differences that exist in the musics of these composers. The analysis was carried out on first order pitch distributions of both the set of 12 notes of the chromatic scale as well as the of 21 notes which distinguish between multiple interpretations of the same note (maintaining a distinction between notes such as Eb and D# which differ only in their function). Their results showed theat there was a progressive increase in information content, and decrease in redundancy from the Mozart example, to those of Beethoven, Hindemith and Berg which was observed in both the 12- and 21-note sets. The study also highlighted other aspects of stylistic choice made by each composer within their corresponding sonata, such as variation in the values of entropy between successive sections, choice of notes, amounts of chromaticism, etc.

A similar analysis of Anton Webern's symphony (Op. 21) was carried out by Hiller and Fuller (1967) with the aim to demonstrate that a music theoretic analysis of the symphony complements an information theoretic one. It extended the work in (Hiller and Bean, 1966) by computing second order probability distributions as well in addition to zeroth and first order distributions, over musical pitch, intervallic relationships between pitches, rhythm and a combination of pitch and rhythm. Entropies and redundancies computed over different sections of the piece were able to highlight differences in structural complexity of the chosen musical features. This study also examined the effects of sample size and the size of the alphabet (number of symbols) on the reliability of the estimated distribution, which essentially concluded that a small sample size and a large alphabet size lead to potentially unreliable estimates of entropies and redundancies that might potentially lead to errors in the analsysis.

A limitation of analysing music only in the symbolic form is that it ignores several aspects of musical expression whose gradations are often continuous in nature and cannot be expressed purely in terms of a finite alphabet as in the case of, say, musical pitch in the Western sense. This was first noted in (Knopoff and Hutchinson, 1981) where, in order to illustrate this point, an information theoretic analysis of dissonance in chorales harmonised by J.S.Bach was carried out. This study was extended in (Knopoff and Hutchinson, 1983) to the analysis of musical style along the lines of the example set in (Youngblood, 1958), while also highlighting the role of sample size (duration of the piece whose style is under question) in judging the reliability of the observations.

Finally, in an interesting experiment of information theory in measuring the influence of musical training in improving musical originality (Coffman, 1992), 34 seventh grade students were asked to create a composition on a MIDI keyboard, first prior to attending a nine-week long music course and then after it. An information theoretic analysis of these pre- and post-instruction compositions revealed that the latter exhibited significantly higher pitch entropy, indicating less predictability in pitch choices and greater freedom of choice, despite a reduction in the average length of the compositions. A more extensive review of other work in music and information theory is available in (Pearce, 2007).

2.2.2 Multiple Viewpoint Systems and Related Work

The work carried out in this dissertation is most closely related to the work initiated by (Conklin and Witten, 1995), entitled *Multiple Viewpoints for Music Prediction*. It is a framework proposed for the analysis (and synthesis) of musical data that addresses two main shortcomings of many of its precursors by (1) computing prediction probabilities that are dynamically adapted during prediction, and (2) incorporating information from multiple musical features that can be extracted from a given piece of music. It employs an event-based representation of music extracted from symbolic music data where each event corresponds to the occurrence of a musical note. A given piece of music is decomposed into parallel streams of features, known as *types*. Each *type* is either a directly observable musical dimension such as *pitch* and *note duration*, or an abstract one derived from them such as *inter-onset interval* or *pitch contour*. A *viewopint* is essentially a function that maps sequences of one type onto those of another. *Types* that can be mapped onto each other via viewpoints are considered to be potentially useful sources of information about those related to them. Thus the predictions about a certain *type* of interest can be obtained by combining the predictions made by models for sequences of other *types* related to it using ensemble methods (Tax et al., 2000) after mapping their respective predictions to those of the *type* of interest using the appropriate viewpoints.

There are two kinds of models that carry out the above combination whose predictions are, in turn, combined — a long-term model (LTM), and a short-term model (STM). The LTM is a model whose parameters are estimated offline from a dataset of melodies. It represents more global stylistic characteristics analogous to those acquired by a listener over a longer time-span. The parameters of the STM are estimated on-the-fly while making predictions on data, without anything done beforehand. The STM highlights the importance of context-specific information available in music while it is being processed by a listener, in the generation of expectations. Predictions (in the form of probability distributions) made by each model about a certain musical event in a sequence are combined using ensemble methods, and this has been shown to improve the quality of predictions over individual models in the past (Conklin and Witten, 1995; Pearce, 2005). The significance of this framework is in its extension of previous work in language modelling to music with an information theoretic backing which facilitates an objective evaluation of models for music prediction. The idea of combining corpus-based long-term and context-sensitive short-term predictions from different models was originally a feature of cache-based language models (Kuhn and De Mori, 1990).

The multiple viewpoints framework was adopted in (Pearce, 2005) where it was used to implement a cognitively plausible system for music prediction that was able to generate human-like melodic expectations. This work also introduced a set of new musicological features (*types*) in addition to those originally introduced in (Conklin and Witten, 1995) that were found to be effective in making more accurate predictions. In order to choose the optimal subset of these features for prediction, this work employed an iterative feature selection algorithm. In addition to this, a comprehensive evaluation which tested a wide range of n-gram and variable order Markov models was also carried out on a corpus of 8 datasets in order to establish a benchmark against which future prediction models, such as those presented here, may be compared. And more recently, the framework was extended further in (Whorley et al., 2013). The chief contribution of this work was in addressing the little explored area of musical harmony with the Multiple Viewpoints framework. These are in the form of an empirical analysis of the time-complexity of an existing approach for this purpose (Conklin, 2002), the adoption of approaches from previous systems for automatic harmonisation (Hild et al., 1992; Allan and Williams, 2004) into the Multiple Viewpoints framework, and the introduction of a novel method which allowed the use of different viewpoints in different voice-parts which was previously not possible. This work also modified the *type* selection algorithm used in (Pearce, 2005) to work more efficiently given a larger set of types to choose from. This work also proposed a musicological solution to the problem of dealing with the curse of dimensionality that results from taking the Cartesian product of pitches from multiple voice-parts in order to represent harmony.

2.3 Music and Connectionism

The many contributions made during the past three decades to computer-assisted analysis and generation of music with the aid of Connectionist architectures can be seen to have occured in two waves, in parallel with developments in Connectionist research itself. During the first wave, the founding principles of Connectionism were introduced (Rumelhart et al., 1986) through the idea of Parallel Distributed Processing according to which mental phenomena occur as a result of simultaneous interactions between simple elementary processing units, as opposed to the then prevailing notion of Sequential Symbolic Processing which explained the same phenomena in terms of sequential interactions between complex goal-specific units. Its significance is largely theoretical, with a few experimental and empirical results to support the feasibility of the theory. Following several years of reduced interest, the second wave further strengthened the claims made by its precursor through a series of successful high-impact real-world applications. This was owing to both the proposal of newer theories, and the availability of greater computational power and vast amounts of data that enabled the demonstration of the efficacy of these theories nearly two decades on (Bengio, 2009; LeCun et al., 2012). The innovations that came about as a result of these two phases trickled down to several application domains (Krizhevsky et al., 2012; Hinton et al., 2012; Collobert et al., 2011) of which music is one (Todd and Loy, 1991; Griffith and Todd, 1999; Humphrey et al., 2012). This section reviews notable contributions among the many that demonstrated the application of connectionism to symbolic music modelling during these two waves in order to present a historical perspective together with an overview of the techniques employed.

2.3.1 The First Wave

The first set of notable approaches which apply Connectionism to the analysis and generation of symbolic music were proposed in the years following the publication of the influential text on *Parallel Distributed Processing* (Rumelhart et al., 1986). While the breadth of contributions to the field during this period is indeed vast, we present a brief historical perspective on only those architectures and algorithms that are of relevance to the present work, and refer the reader to (Rumelhart et al., 1986; Medler, 1998) for more in-depth and comprehensive reviews. Many of the inventions and algorithms proposed during this period persisted through the decades that followed and significantly impacted research in Artificial Intelligence, and the now-thriving field of Machine Learning. These were the years that saw the maturation of the previously proposed Perceptron (Rosenblatt, 1958) into the Multi-Layer Perceptron (also known as the Feed-forward Neural Network) and the invention of the Backpropagation algorithm (Rumelhart et al., 1988) which offered a simple and efficient means to train this model on data, thus leading to a surge in its popularity. The architecture of the Feed-forward Neural Network (FNN) was further adapted to deal with sequential data into the Recurrent Neural Network (RNN) (Elman, 1990; Jordan, 1986), and likewise, the Backpropagation algorithm extended into the Backpropagation Through Time (BPTT) (Werbos, 1990) to train this new architecture. Other algorithms were also proposed around the same time to carry out real-time learning in the RNN architecture (Williams and Zipser, 1989). Another significant innovation from this period, which is relevant here is the Boltzmann Machine family of models (Smolensky, 1986; Hinton et al., 1984), which consists of undirected graphical models that learn joint probability distributions of sets of visible and latent variables through a process of minimisation of an energy function associated with configurations of these variables. Probabilistic inference can be carried out in these models to determine conditional distributions, typically of interest in various prediction tasks.

Contributions to Connectionist theory and Artificial Intelligence, such as the above, generated interest in their adoption into several application domains that foresaw their potential benefits. This included the computer-assisted analysis and synthesis of music. One of the first systems for this purpose, known as *HARMONET* (Hild et al., 1992), was designed for harmonising chorales in the style of J.S.Bach. It consists of a symbolic (rule-based) component together with a recurrent neural network, and generates four part harmonisations of a given chorale melody. The role of the neural network is to generate human-like harmonisations within the rules dictated by music theory, which when taken literally tend to result in "aesthetically offensive musical output". HARMONET divides the harmonisation task into three subtasks. In the first, a harmonic skeleton of the chorale melody is generated for every quarter note of the given melody (which essentially involves determining the *bass* voice of the chorale) using a recurrent neural network. The network takes as inputs harmonies generated at previous time-steps, and also the local context and global position (with respect to the beginning of the melody) of the note at the current time-step to generate a harmony for it. A novel representation for the pitch of each musical note was introduced at this stage which encodes the harmonic functions that contain the note, thus introducing hand-crafted musicological information as input to the network. This is followed by the generation of the *alto* and *tenor* voices taking into account the given *soprano* voice in the melody, and the *bass* voice generated in the previous step. Finally, ornamenting eighth notes are added to the result at each chord by another network which takes into account the local harmonic context. The system was evaluated by an audience of music professionals who judged the quality of the harmonisations. By treating each of the possible harmonizations of the first network above as classes and changing its output units to *softmax* (Specht, 1990), the system can be used for predicting harmonic expectation over time.

The work initiated in the context of HARMONET was later extended to create MELONET (Feulner and Hörnel, 1994) - a system comprised of a hierarchy of neural networks operating at different time-scales which models melodies as sequences of harmony-based motifs and varies one of the chorale voices generated by HARMONET. It uses, what are known as delayed-update neurons in a recurrent network which, by integrating their inputs over a certain time-span reflect long-term information about the melody input. It works hand-in-hand with HARMONET to generate the said variations. In a subsequent publication, a committee of such neural networks, each of which has learned a specific harmonisation style, was used to recognise different styles of harmonisation according to how *expected* it is to each network (Hörnel and Menzel, 1998).

Chorale harmonization has also been the focus in (Bellgard and Tsang, 1994) where, in contrast to HARMONET, the approach relies solely on a connectionist model — the Boltzmann Machine (BM) (Hinton et al., 1984; Smolensky, 1986). Four-part writing in practice is regarded here as being the result of a unique set of choices made by the composer between various competing harmonization techniques, which is not clearly defined in practice and is thus essentially an imprecise and noisy process. This is where the stochastic nature of the model employed is highlighted as an advantage over the deterministic nature of models from previous work i.e., HARMONET, CHORAL (Ebcioğlu, 1988). The task is viewed as one of pattern completion (or gap-filling) where a given chorale melody is only the partial specification of a complete piece of information which is the harmonized chorale. A BM learns local harmonic constraints through a series of overlapping time-windows extracted at each time-step in the chorale. Harmonization is achieved in an identical fashion, but with the learned model slid (in time) along a given chorale melody. Its visible units are comprised of a mixture of multinomial and binomial units which represent three octaves of musical pitch, musical rest and phrase-control variables. The energy-function associated with a Boltzmann machine to assess the quality of learning in the model is also used to assess the quality of the harmonies generated by the model. Sliding the BM in time along a temporal input gives, what is referred to by the authors as the Effective Boltzmann Machine (EBM).

As music is inherently temporal in nature, recurrent neural networks (RNNs) are a natural choice for modelling musical structure. In one of the first applications of neural networks to music (Todd, 1989), a special case of the RNN known as the Jordan network (Jordan, 1986) was made to memorize and interpolate between melodies of different styles. The network consists of an input, hidden and an output layer. A part of the input layer consist of a set of *plan units* that indicate the style of the melody being learned or produced. The rest is a set of *context units* which maintain the memory of the sequence produced so far by combining the effects of the most recently predicted output in the sequence (which is fed back as input) and an exponentially decreasing sum of all of the network's previous inputs. The input layer is fully connected to the hidden layer which is in turn fully connected to the output layer. The network models sequences of pitches and durations, and uses a fixed size time-window of notes in its input context units and predicts the same number of notes as those in the input time-window for the next time-step which are fed back to its input layer. All melodies are transposed into the key of C, and a binary one-hot representation (a vector containing all 0s and a single 1 corresponding to a particular value) is used for pitch. A time-slice representation is used for duration where the length of a note is given by the number of consecutive evenly spaced time-slices (of eighth-note duration), with additional information about its onset. The purpose of the network is to memorize melodies that it has come across, associating each melody with a plan so that it can also interpolate between melodies when plans are interpolated, and change melodies dynamically as well when plans are changed.

An often cited work in connectionist music composition is that of Mozer (1991), where an RNN named CONCERT is empoyed for learning structure in melodies to generate novel variations on them. In contrast to the above described approach in (Todd, 1989), this network uses an Elman RNN (Elman, 1990) and also contains a learning stage (absent in the other) where the backpropagation through time (BPTT) algorithm (Werbos, 1990) is applied to tune the weights of the network to the prediction task. The task is to predict the next note, given the previous one and the state of its hidden layer in the most recent time-step which accounts for the notes further back in time that are not dealt with explicitly. The shortcomings of the network's architecture in dealing with long-term memory and global structure of a musical piece are addressed by taking into account the notes in the melody at multiple time-resolutions, and also employing an additional parameter that enabled controlling its sensitivity to recent versus not so recent notes in a melody. With the generation of aesthetically pleasing melodies being the focus of the network, the task-unaware one-hot representation of notes in it is abandoned (or retained only for the sake of interpreting results) in favour of a perceptually motivated one, based on earlier empirical observations by Shepard (1982). The model was evaluated by having it extend a C major diatonic scale, learn the structure of diatonic scales, learn random walk sequences of pitches, learn specific kinds of phrase patterns and generating new melodies in the style of J. S. Bach.

A different approach inspired by the *Target-note Technique* in Bebop jazz is explored by Toiviainen (1995), wherein given a typical jazz chord progression an auto-associator network emulates the creativity of an improviser. The melodies generated by the model rely on the starting notes at any given point in time, together with the current chord to determine the possible melodic patterns, and the next chord in the progression to determine the possible target notes to follow. Several constraints that reflect the typical practices in jazz improvisation, such as the relationship between the musical pitch of a note in the melody and the root of the current chord, typical chord-types occurring in jazz progressions, typical syncopation in improvised melodies, etc. influenced the design choices for the architecture of the network. The network relies on the Hebbian learning rule for updating its connections while learning from data. A moving time-window approach was adopted for representing time, where each window corresponded to one half-measure. Thus in each step of its operation during the generative process, the network generated a melody of length equal to a half-measure, which was fed back into it in order to generate the next one, and so on. The fact that such a network learns to generate music from examples in a dataset, much like a typical jazz musician who improvises based on the repertoire that she/he has paid attention to over time is what motivates this approach. The author concludes that “the melodies produced by the network resemble those of a beginning improviser”, based on a qualitative assessment of its generations learned from excerpts of solos played by the trumpet player Clifford Brown, over chord changes in George Gershwin’s *“I’ve Got Rhythm”*.

The above list of connectionist systems for the analysis and synthesis of symbolic music consists of notable contributions among those that laid the foundations for future work on the subject. It is, by no means exhaustive, and there exist several others that explore other musical phenomena with connectionist architectures considered beyond the scope of this review. We point the inquisitive reader to (Todd and Loy, 1991; Griffith and Todd, 1999) for a comprehensive summary of work carried out in the field during, what we here refer to as, the first wave of connectionism.

2.3.2 The Second Wave

The second wave of interest in neural networks and connectionism, which has prevailed for nearly a decade (with hardly any signs of subsiding) at the time of writing of this dissertation, can be said to have come about towards the end of what is generally known as the *AI Winter* (Hendler, 2008). Its success has been attributed to the culmination of three key factors — theoretical and empirical advances in connectionist research, the presence of very powerful hardware in modern computers, and the availability of vast amounts of data. This wave brought with it several new innovations in connectionist architectures and algorithms which also fueled a revival in the study and application of older ones brought about by its precursor. The theoretically known, but often practically infeasible concept of a deep neural network (a feed-forward neural network with more than one hidden layer) was made into reality during this period with the introduction of new methods for pre-training these networks layer-by-layer in an unsupervised fashion before training on a certain task in a supervised manner (Bengio et al., 2007; Hinton et al., 2012). The Restricted Boltzmann Machine (RBM), a generative unsupervised model which was, in part responsible for this turnaround, was extended in many different ways to serve as a supervised learning model and a classifier (Salakhutdinov et al., 2007; Larochelle and Bengio, 2008), a sequence learning model (Sutskever and Hinton, 2007; Sutskever et al., 2009; Taylor et al., 2007) and generalised to handle different types of data (Welling et al., 2004). The RBM, in turn, soared in popularity thanks to the Contrastive Divergence algorithm (Hinton, 2002; Tieleman, 2008) which made it possible to train this model more efficiently than was previously possible. Likewise, the limitation of recurrent neural networks in modelling very long-term memory was also addressed to increase their effectiveness as sequence models (Martens and Sutskever, 2011). A previously proposed architecture to address the same issue of long-term memory — the Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) was also re-visited and is now even more widely used as a sequence model, with proposals of other models inspired by it (Chung et al., 2014). Another architecture that underwent a breakthrough is the Convolutional Neural Network which is now the de facto standard for object recognition and related image recognition and classification tasks (Krizhevsky et al., 2012). All these advances had a significant impact on three application areas — Natural Language Processing, Speech Processing and Computer Vision (Lecun et al., 2015), the very tasks in which the failure of Artificial Intelligence to perform well in the past was an important reason for the loss in faith in the field, i.e. the AI Winter.

This revival of interest in connectionist research inspired a body of work that deals with a diverse set of musical tasks using symbolic music. One such application was in modelling melodies by capturing short melodic motifs in them using a Time Convolutional RBM (TC-RBM) (Spiliopoulou and Storkey, 2011). In contrast to other RBM-based sequence models (Sutskever et al., 2009; Taylor and Hinton, 2009), the TC-RBM does not make use of any recurrent connections and relies on the idea of convolution through time over fixed-length subsequences within a window centered at each time-step (Lee et al., 2009). Furthermore, a weight-sharing mechanism which features in this model helps it achieve translation invariance along time, which is desirable as motifs can occur anywhere in a musical piece. The approach models both the pitch and duration of notes, and uses an implicit representation of time by discretising it in eighth-note intervals. A two-fold evaluation of this model was carried out with the model on the Nottingham Folk Music Database[1]. A qualitative evaluation involved the analysis of the latent distributed representations learned by the TC-RBM when presented with musical data in its visible layer, which were found to convey information about the scale, octave and chords. In a quantitative evaluation, the model was made to predict the next k time-steps given a fixed-length context. The prediction log-likelihood was computed approximately by sampling from the model,and the Kullback-Leibler divergence was used to determine the closeness of the model's predictions to the empirical distribution.

As a continuation of a previously proposed probabilistic grammar based approach for generating Jazz solos known as the *Impro-visor* (Keller and Morrison, 2007), a Deep Belief Network (Hinton et al., 2006) (DBN, a probabilistic generative model made up of a stack of the aforementioned Restricted Boltzmann Machines) was experimented with for the same purpose (Bickerman et al., 2010). As modelling entire melodies, or solos requires dealing with long-term dependencies that are not feasible with a non-recurrent model such as the DBN, only 4-bar jazz *licks* (short, coherent melodies) are modelled at each time-step. As in some of the approaches outlined above, a sliding-window is used to model temporal information, with a window-size of one measure (4 beats) of the piece of music, and a step-size of 1 beat. The visible (input) layer of the DBN simultaneously modelled the joint distribution of the chromatic pitch-class, duration and onset, and octave of the melody note, and the chord underlying the melody, thus allowing the model to associate chords with various melodic features which is a key factor to consider in jazz music. The model was trained generatively using the Contrastive Divergence algorithm (Hinton, 2002; Tieleman, 2008) on a large corpus of 4-bar jazz licks. With the DBN being a stochastic generative model, novel jazz licks could be sampled one beat at a time from it in generative mode. While it could be demonstrated that

[1]www.chezfred.org.uk/freds/music/tunes/Indexreels.htm

the model does indeed generate the desired licks, the authors conclude in favour of their previous grammatical approach to lick generation over the DBN stating the subjective quality of the generated licks and the large training time of the DBNs to support this choice.

The approaches described above use non-recurrent models which have largely been superceded, when it comes to the modelling of sequential data, by recurrent models that are a more natural fit for temporal data. In an attempt towards style-independent polyphonic music generation in (Boulanger-Lewandowski et al., 2012), an RNN-RBM is made to model sequential information directly from the piano-roll notation (Orio, 2006b). The reason for dealing with this notation is to avoid making any kind of prior assumptions regarding the nature of the modelling task that would simplify it, thus leaving much for the model to determine by itself. The RNN-RBM is a stochastic model and can be understood as a sequence of RBMs, which at each time-step of the sequence are conditioned by the hidden layer of an RNN. Thus in addition to the RNN modelling sequential information, the RBM models correlations between variables (MIDI note values) that occur simultaneously at each time-step. The latter is often ignored in standard RNNs, and can be viewed as an advantage of this model given sufficient data since it also entails the need for a greater number of model parameters. The model is targeted at the task of automatic music transcription and is thus required to model time in seconds in contrast to other symbolic music modelling approaches that represent time relative to the musical score, thus requiring an additional step of alignment between the audio and symbolic formats. Time, in this model, is represented in terms of consecutive slices of the quantised musical signal. It is trained using the mini-batch gradient descent and the Backpropagation Through Time algorithms. It was found that this model outperforms others addressing the same task. This work has also inspired other very close extensions with the same goal, that claim improved performance (Goel et al., 2014; Lyu et al., 2015).

A previous approach by Eck and Schmidhuber (2002) for modelling Blues music with a Long Short-Term Memory (LSTM) RNN can be said to have influenced the above described one (Boulanger-Lewandowski et al., 2012) in its choice to not incorporate any prior musicological information in order to simplify the modelling task. As mentioned earlier, the LSTM is an enhanced version of the basic RNN and has been shown to be able to successfully model longer temporal dependencies than the latter. Here, once again, successive slices of the musical signal are treated as time-steps. A quantisation step-size of 8 notes per measure was used, and thus the 12-bar blues musical segments used for training the model were each

96 time-steps in length. The first experiment carried out with this model involved having it learn and generate a musical chord structure, from which the authors conclude that this is a fairly straightforward task for the model, and also expected given its previous success in tasks involving counting. In the second experiment both melody and chords are learned, leading to a conclusion that the LSTM is indeed able to generate a blues melody constrained by the learned chord structure that sounds better than a random walk across the pentatonic scale and are faithful to the examples in the training set. The evaluation in this case is left to the listener who is encouraged to visit a webpage[2] containing the pieces of music generated by the network.

A more recent study with the LSTM (Franklin, 2006) carried out further experiments with this model on jazz-related tasks. Here, various note representations were studied in order to incorporate musical knowledge into the network. This can be contrasted with the approach adopted in (Eck and Schmidhuber, 2002; Boulanger-Lewandowski et al., 2012) that avoids making any music theoretic assumptions. A pitch representation based on major and minor thirds known as the *circle-of-thirds* representation, and a duration representation known as the *modular-duration representation* which extends that proposed in (Mozer, 1991) were used to train the dual pitch/duration LSTMs. Two experiments were carried out. The first focused on short musical tasks, and only sequences of musical pitch were considered. These included outputting in sequence the four chord tones given a dominant seventh chord as input, determining whether or not a given sequence of notes are ordered chromatically, and reproducing a specific 32 note melody of the form *AABA* given only the first note as input. A single network was used for all these tasks. In the second experiment, which focused on long musical tasks, the objective was to learn the melody of the song *Afro Blue* composed by the jazz percussionist Mongo Santamaria. Two separate networks are used to learn musical pitch sequences and note duration sequences respectively. The study concludes in favour of the LSTM and a detailed qualitative analysis of the results with respect to the authors' expectations.

Lambert et al. (2015) trained a two-layered RNN on the Mazurka dataset (MAZ)[3], an audio dataset of expressively performed piano music. The first layer for the system is a Gradient Frequency Neural Network (GFNN) (Large et al., 2010), which uses nonlinear oscillators to model metre perception of a periodic signal. The second layer contains LSTM units which model the output of the GFNN and pre-

[2]`http://people.idsia.ch/~juergen/blues/` (last accessed: 20th October, 2015)
[3]http://www.mazurka.org.uk/

dict rhythmic onset as a time-series activation function. This work builds on previous experiments involving a symbolic data in which the authors find that the LSTM performs time-series modeling significantly better when GFNNs are used exclusively (Lambert et al., 2014a,b). Their GFNN-LSTM model was able to predict rhythmic onsets with an f-measure of 71.4%.

As stated before, there seems to be very little work focusing on connectionist models for information theoretic music modelling. One such attempt is presented in (Cox, 2010), where the relationship between entropy and meaning in music inspired by (Meyer, 1956, 1957) is explored with the help of Recurrent Neural Networks that estimate instantaneous entropy for music with multiple parts in the analysis of a string quartet piece composed by Joseph Haydn. The model considered here contains two components - a long-term model (LTM), and a short-term model (STM) (Conklin and Witten, 1995). The parameters of each model are learned through exposure to appropriate data. The LTM models global stylistic characteristics acquired by a listener over a longer time-span. The STM models context-specific information, available in a melody while it is being processed by the listener, in the generation of expectations. Predictions made by each model are combined using ensemble methods, and this has been shown previously to improve the quality of predictions over individual models in the past (Conklin and Witten, 1995; Pearce, 2005). The work demonstrates that the entropies as predicted by the model are sensitive to the effects of cadences, resolutions, textural change, and interruptions in music.

2.4 Motivating this Dissertation

The work presented here began with an interest in the above described Multiple Viewpoints approach to music modelling (Conklin and Witten, 1995; Pearce, 2005; Whorley et al., 2013), with the aim of improving the performance in a melody prediction task addressed in them, by leveraging the potential benefits of connectionist architectures. With n-gram models dominating much of the previous work on music modelling in this context (*cf.* Section 2.2.2), this has presented the opportunity for their comparison with other modelling techniques, of which those inspired by Connectionism are viable alternatives. To the author's knowledge, a connectionist approach to information theoretic music modelling as considered here has received very limited attention (Cox, 2010) so far, and no quantitative evaluation directly comparable with previous work that uses n-gram models exists.

The first observation from the above review is that much of the work employing connectionist models for modelling musical structure largely focuses on qualitative assessment of the musical output (Todd, 1989; Mozer, 1991; Bellgard and Tsang, 1994; Franklin, 2006).The rest is focused solely on subjective quality of the music generated by a model (Eck and Schmidhuber, 2002; Bickerman et al., 2010; Toiviainen, 1995; Hild et al., 1992). Moreover, it is only during a little over the past decade that efforts towards estabilishing a quantitative standard for models of music prediction (in addition to qualitative assessments) have come about (Pearce and Wiggins, 2001, 2004; Potter et al., 2007). These are akin to already existing evaluation standards in language modelling that help compare different approaches in the domain through standard corpora (Paul and Baker, 1992; Marcus et al., 1993; Miller, 1995) and evaluation measures (Chen et al., 1998). In (Pearce and Wiggins, 2004), a comprehensive evaluation of n-gram models for musical pitch was carried out on a corpus containing 8 datasets of different sizes and melodic complexities. Such standards are indeed important, and the present work compares the models proposed in it to the said n-grams wherever appropriate in an attempt to contribute to the benchmark. Application-independent evaluation in the case of polyphonic music generation has also been adopted in more recent work with connectionist models (Boulanger-Lewandowski et al., 2012; Goel et al., 2014; Lyu et al., 2015).

As highlighted by the above review, music and language modelling are but two of the many areas of research that have been impacted by the information theoretic approach. Given the parallels between music and language here both in terms of the modelling techniques as well as the means of evaluation, there is further reason to believe that a connectionist approach for modelling musical sequences would lead to some degree of success much like in the case of recent work in language modelling. Neural Language Models (NLMs), which have risen in popularity over the past decade primarily address one key issue n-gram models are faced with. This is known as the *curse of dimensionality*, and interferes with effective maximum-likelihood estimation of the parameters in n-gram models as the length of the modelled sequences increases. Specifically, if one models sequences of length l that are composed of different temporal arrangements of $|\chi|$ discrete symbols, the number of n-gram model parameters to estimate is $|\chi|^l$. It is not desirable that the number of parameters increases exponentially with l. Models which employ distributed architectures such as neural networks tend to bypass this problem, as they do not require enumerating all state transition probabilities, but rather the weights of the network encode only those dependencies necessary to minimize prediction error. Moreover, one often does not encounter every possible n-gram of words in the corpus and must often back-off to a lower-order (less than n) model. The proposed solution to these drawbacks is demonstrated in two of the earliest NLMs proposed (Schwenk and Gauvain, 2002; Bengio et al., 2003), which employ feed-forward neural networks for predicting probability distributions over the possible vocabulary. Such models learn a reusable real-valued vector representation corresponding to each word, which reflects information regarding the co-occurrence of words in the corpus, is learned implicitly in the model. This idea is directly inspired by previous work in data-driven distributed representation learning (Hinton, 1986), and is a common feature in subsequent NLMs as well (Morin and Bengio, 2005; Mnih and Hinton, 2008; Mikolov et al., 2010; Collobert et al., 2011). Reviewing work in neural language modelling made clear the various choices available and ideas that could and have been adopted here, and influenced the choice of representation (localist vs distributed) for musical features, network architecture (recurrent vs non-recurrent) in order to represent sequences, and helped develop an understanding of how these architectures can be trained on data (various optimisation algorithms available for learning the model parameters, and the hyperparameters involved in the learning process). For example, in the present work, the one-hot encoding (*cf.* Section 3.1.2) is employed for representing the values of various musical features given their discrete nature. These vectors are concatenated into longer vectors when multiple features are to be introduced as input to a model. Both recurrent and non-recurrent architectures for modelling musical sequences have been explored. And among the various optimisation algorithms available for training models, mini-batch gradient descent has been used.

Chapter 3

Preliminaries

This chapter presents an overview of concepts needed to understand the chapters that follow. The first, and major part of this dissertation (Chapters 4, 5 and 6) focuses on monophonic music prediction using connectionist models. The first question to address in this regard is how to translate the contents of a digitised musical score into a representation that can then be made use of by the models. For this, we turn to the *Multiple Viewpoints* representation which offers a means to take into account the rich structural information available in music. Once the sequences of Multiple Viewpoint features have been extracted for a given piece of music, they are converted into sequences of what are known as *one-hot* encoded vectors. These vectors are presented as inputs to a connectionist prediction model. The parameters of this type of prediction model are *learned* purely through exposure to data, with the help of an iterative gradient-based learning algorithm whose each iteration involves computing the gradient of the model's prediction error on the given data with respect to its parameters and updating them according to this gradient, with the goal to minimise the error. We use the information theoretic measure known as cross entropy to evaluate the predictive performance of the model both during this learning stage and also while making predictions on unseen musical sequences. Evaluation is carried out on a corpus containing a collection of folk and chorale melodies which serves as a means to compare the models here to those proposed in previous work. The above mentioned procedure for learning parameters of connectionist models from data is also relavant to the second part of the dissertation (Chapter 7) which extends and evaluates a novel connectionist architecture for sequence labelling introduced in Chapter 5 and an also an existing energy-based connectionist model for classification (Larochelle and Bengio, 2008).

3.1 Representation

In order to input the information available in the musical score to the prediction models, we adopt the *Multiple Viewpoints* representation of music introduced by Conklin et al. (1990; 1995). The underlying motivation for this framework was to extend the application of statistical modelling techniques originally applied to text and language data (Bell et al., 1989; Manning and Schütze, 1999) to the music domain, based on the observation that musical events have a rich internal structure and can be expressed in terms of directly observable or derived musical features. The framework of Multiple Viewpoint Systems (MVS) for music prediction was proposed in order to efficiently handle this rich internal structure of music by exploiting information contained in these different musical feature sequences, while at the same time limiting the dimensionality of the models using these features. The description that follows is limited only to the application of this framework to monophonic music, and the reader is referred to (Conklin, 2002; Whorley et al., 2013) for its extensions to harmony and polyphony. Much of the introduction provided in this section is based on (Conklin, 1990; Conklin and Witten, 1995; Pearce, 2005).

The Multiple Viewpoints framework views a melody as a sequence of *musical events*, where an event corresponds to the occurrence of a note in a musical score. A melody is thus represented as a sequence of events $\left(s^{(1)},\ldots,s^{(T)}\right)$, where each event $s^{(t)}$ is associated with a note at location (or time-step) t in the sequence. The shorthand representation that will be used for such a sequence between, and including, time-steps t_1 and t_2 is $s^{(t_1:t_2)}$. Each event s is composed of a finite set of basic attributes, each associated with a *type* τ. Each type may assume a value drawn from a finite alphabet with a domain $[\tau]$, denoting the set of all syntactically valid elements of that type. A semantic domain $[[\tau]]$ corresponding to each type denotes the set of possible meanings for the elements of its syntactic domain $[\tau]$, and a function $[\![\cdot]\!]_\tau : [\tau] \rightarrow [[\tau]]$ serves as a mapping between the two domains. The Cartesian product of the domains of n basic types $\tau_1, \tau_2, \ldots, \tau_n$ is referred to as the *event space*, ξ:

$$\xi = [\tau_1] \times [\tau_2] \times \ldots \times [\tau_n]$$

Thus, an event s is an instantiation of the attributes $\tau_1, \tau_2, \ldots \tau_n$, or an n-tuple in the event space. And the event space ξ denotes the set of all representable events, and its cardinality is denoted by $|\xi|$. The set of all sequences that can be constructed from the elements of ξ is denoted by ξ^*. Since the domain of each individual type is finite, so is the cardinality of the event space. Henceforth, a reference to the name of a type is written in typewriter font in order to distinguish it from other text.

A *viewpoint* modelling a type τ is a partial function, $\Psi_\tau : \xi^* \rightharpoonup [\tau]$, which maps sequences of events onto elements of type τ[1]. Each viewpoint is also associated with a *type set* $\langle\tau\rangle \subseteq \{\tau_1, \ldots, \tau_n\}$, stating which basic types the viewpoint is derived from and is, therefore, capable of predicting (Conklin, 1990).

A *multiple-viewpoint system* (MVS) is a set of models, each of which is trained on subsequences of one *type*, whose individual predictions are combined in some way to influence the prediction of the next event in a given sequence. It allows one to incorporate useful information from sequences of any arbitrary type in the context (an *input* type τ_{in}) derived from a corresponding sequence of events to influence the predicted probability distribution over a certain type corresponding to the next event in the sequence (a *target* type τ_{tgt}). This is realised through the introduction of multiple models, each of which relies on a different source of information (its respective input type) to make a prediction about the target type. The accuracy of the prediction depends on how informative the input type is of the target type. It has been assumed in previous work on MVS that the input type belongs to the type set of the target type. It is possible to combine the information provided by different input types for possibly better predictive performance on the target type.

In previous work implementing this idea through n-gram models (Conklin and Witten, 1995; Pearce, 2005) it is first required to make a prediction about the input type itself and then map this prediction to the domain of the target type. Put differently, these approaches first model the distribution $P\left(\tau^{(t)}_{\texttt{in}} | \tau^{(1:t-1)}_{\texttt{in}}\right)$, and then rely on a deterministic mapping between the input and target types to derive $P\left(\tau^{(t)}_{\texttt{tgt}} | \tau^{(t)}_{\texttt{in}}\right)$.

3.1.1 Relevant Viewpoints and Viewpoint Categories

Previous work (Conklin and Witten, 1995) has defined different categories of viewpoints, some of which are relevant to the present work and will be described here. The first is the *basic viewpoint*, and models *basic types* which are directly observable, or given in a musical score. Among others, this category includes chromatic pitch `pitch`, note duration `dur`, key-signature `keysig` and onset time of a note

[1]There exists a distinction in the understanding of viewpoints in the usage by (Conklin and Witten, 1995) and (Pearce, 2005). Here, we adhere to the latter where viewpoints are considered to be a purely representational formalism.

Symbol	Interpretation	Example
τ	A typed attribute	`pitch`
$[\tau]$	Syntactic domain of τ	$\{60,\ldots,72\}$
$\langle\tau\rangle$	Type set of τ	{`pitch`}
$[\![\tau]\!]$	Semantic domain of τ	$\{C_4, C\#_4, \ldots, B_4, C_5\}$
$[\![\cdot]\!]_\tau : [\tau] \to [\![\tau]\!]$	Semantic Interpretation of $[\tau]$	$[[60]]_{\texttt{pitch}} = C_4$
$\Psi_\tau : \xi^* \rightharpoonup [\tau]$	see text	see text

Table 3.1 Sets and functions associated with types (Pearce, 2005).

`onset`. A *derived viewpoint* models a *derived type* which does not feature in the event space, but can be derived from any of the basic types or other derived types. A derived type at a certain time-step may be undefined, in which case the derived viewpoint is also undefined. For example, the derived type for musical interval (`int`) is not defined for the first event, as it is computed as the difference in the MIDI values of two consecutive values of `pitch`. In such a case, it is denoted by $\perp$, as given below:

$$\Psi_{\texttt{int}}\left(s^{(1:t)}\right) = \begin{cases} \perp & \text{if } t = 1, \\ \Psi_{\texttt{pitch}}\left(s^{(1:t)}\right) - \Psi_{\texttt{pitch}}\left(s^{(1:t-1)}\right) & \text{otherwise}. \end{cases}$$

The motivation for constructing a derived viewpoint is to capture the rich variety of relational and descriptive terms in a musical language (Conklin, 1990). The final viewpoint which is relevant here is a *linked viewpoint*, which models a *product type*. A product type combines the information contained in multiple types into a single type. It is constructed by taking the Cartesian product over two or more types, in other words "linking" them. A product type τ, denoted as $\tau = \tau_1 \otimes \ldots \otimes \tau_n$ and its associated linked viewpoint $\Psi_\tau\left(s^{(1:t)}\right)$ have the following properties:

$$[\tau] = [\tau_1] \times \ldots \times [\tau_n]$$

$$\langle\tau\rangle = \bigcup_{k=1}^{n} \langle\tau_k\rangle$$

$$[\![\tau]\!] = [\![\tau_1]\!] \; and \; \ldots \; and \; [\![\tau_n]\!]$$

$$\Psi_\tau\left(s^{(1:t)}\right) = \begin{cases} \perp & \text{if } \Psi_{\tau_i}\left(s^{(1:t)}\right) \text{ is undefined for any } i \in \{1,\ldots,n\}, \\ \Psi_{\tau_1}\left(s^{(1:t)}\right), \ldots \Psi_{\tau_n}\left(s^{(1:t)}\right) & \text{otherwise}. \end{cases}$$

The various basic and derived types relevant to the present work are listed in Table 3.2. Of these, we directly deal with `pitch` in Chapters 4 and 5 when introduc-

τ	$[\![\cdot]\!]_\tau$	$[\tau]$	$\langle\tau\rangle$
keysig	key-signature	$\{-7,-6,\ldots,6,7\}$	{keysig}
mode	mode	$\{0,9\}$	{mode}
pitch	chromatic pitch	$\mathbb{Z}^*$	{pitch}
onset	event onset time	$\mathbb{Z}^*$	{onset}
int	pitch interval	$\mathbb{Z}$	{pitch}
referent	referent or tonic	$\{0,\ldots,11\}$	{keysig}
intfref	int from tonic	$\mathbb{Z}$	{pitch}
ioi	inter-onset interval	$\mathbb{Z}^+$	{onset}

Table 3.2 The various basic and derived types relevant to the present work, with their categorisation, syntactic and semantic domains, and type sets.

ing the connectionist models for melody, and additionally `intfref` and `ioi` while demonstrating the combination of multiple input types in these melody models in Chapter 6.

The first of these types — chromatic pitch — is represented by the type `pitch`, and the values assumed by this type conform to the MIDI standard, i.e. $[\![60]\!]_{\texttt{pitch}} = \mathrm{C}_4$ or middle C. In the datasets as a whole, $[\texttt{pitch}] = \{47, 48, \ldots, 90, 91\}$ which means that $[\![\texttt{pitch}]\!]$ ranges from B_2 to G_6. Key-signatures are represented by the type `keysig` which can assume any value from the set $\{-7,-6,\ldots,6,7\}$ and represents the key-signature in terms of the number of sharps or flats, as follows:

$$\texttt{keysig} = \begin{cases} sharps & \text{if } sharps > 0 \\ -flats & \text{if } flats > 0 \\ 0 & \text{otherwise}. \end{cases}$$

In the datasets used in the present work, $[\texttt{keysig}] = \{-5,-4,\ldots,3,4\}$ where, for example, $[\![-3]\!]_{\texttt{keysig}}$ = 3 flats, $[\![5]\!]_{\texttt{keysig}}$ = 5 sharps and $[\![0]\!]_{\texttt{keysig}}$ = no sharps or flats. The `referent` type represents the referent or tonic at a given moment in a melody. It is derived from the basic type `keysig` and uses the type `mode` (whose value is 0 if major and 9 if minor) to disambiguate relative major and minor keys. It is given by

$$\Psi_{\texttt{referent}}\left(s^{(1:t)}\right) = \Psi_{\texttt{mode}}\left(s^{(1:t)}\right) + \begin{cases} \left(\Psi_{\texttt{keysig}}\left(s^{(1:t)}\right) \times 7\right) \mod 12 & \text{if } \Psi_{\texttt{keysig}}\left(s^{(1:t)}\right) > 0 \\ \left(\Psi_{\texttt{keysig}}\left(s^{(1:t)}\right) \times -5\right) \mod 12 & \text{if } \Psi_{\texttt{keysig}}\left(s^{(1:t)}\right) < 0 \\ 0 & otherwise. \end{cases}$$

The `referent` type allows us to take into consideration the effects of tonality, through the type `intfref` which is derived from it. The domain of the latter is $[\texttt{intfref}] = \{0, 1, \ldots, 11\}$, where, for example $[\![0]\!]_{\texttt{intfref}} = \text{tonic}$, $[\![4]\!]_{\texttt{intfref}} = \text{mediant}$ and so on. This viewpoint is motivated by the hypothesis that melodic structure is influenced by regularities in pitch defined in relation to the tonic.

The onset time of a musical event (a note) is represented by the type `onset`. The domain of this type $[\texttt{onset}]$ is $\mathbb{Z}^*$, the set of non-negative integers. One may define the granularity of the time representation by setting the *time-base* during pre-processing to any appropriate positive integer. The time-base corresponds to the number of time-units in a semibreve thereby limiting the granularity of the time representation to a minimum unit inter-onset interval that may be represented. The present work assumes a time-base of 96 for pre-processing, as in (Pearce, 2005). With this time-base, for example, a note duration of 24 time-units corresponds to a crotchet. Following (Conklin and Witten, 1995), the first time point of any composition is zero corresponding to the beginning of the first bar whether complete or incomplete. The first event in a composition may have a non-zero onset due to an opening anacrusis. The derived type `ioi` is informative of the rhythmic evolution of a piece, and represents the inter-onset interval between an event and its predecessor. The viewpoint for the `ioi` type is defined as follows:

$$\Psi_{\texttt{ioi}}\left(s^{(1:t)}\right) = \begin{cases} \perp & \text{if } t = 1 \\ \Psi_{\texttt{onset}}\left(s^{(1:t)}\right) - \Psi_{\texttt{onset}}\left(s^{(1:t-1)}\right) & \text{otherwise} \end{cases}$$

3.1.2 One-Hot Encoding of Viewpoint Features

As described above, each type assumes a value drawn from a finite alphabet with domain $[\tau]$. In order to represent this information as features to a connectionist model, one relies on the binary *one-hot* encoding which converts categorical variables into feature vectors. The binary one-hot encoding of an instance of the type with a domain $[\tau]$ is a binary vector of dimensionality $|[\tau]|$, each of whose elements corresponds to a value in $[\tau]$ with the element corresponding to the value of that particular instance set to 1 (and the rest being 0). Its use is common in machine learning, including the connectionist models of the present work. Prediction in such models involves computing one or more linear projections of input features through weight-matrices in order to map these to the desired outputs (in this case, classes). The weights are such that more important features dimensions are given a higher weight, and thus greater emphasis than those that aren't as important. The one-hot encoding assigns each category to a dimension of the feature vector thus allowing the model to associate such a preference for certain categories through the dimensions corresponding to them. This would not be possible when using a scalar feature with integer values corresponding to the categories, which might indeed seem like a simpler and more intuitive way of representing the categorical information. As a simple example, consider a hypothetical type $\tau_{\texttt{hyp}}$, with $[\tau_{\texttt{hyp}}] = \{3,4,5\}$ such that $|[\tau_{\texttt{hyp}}]| = 3$. One possible set of one-hot encodings of the values 3, 4 and 5 is $[1,0,0]$, $[0,1,0]$ and $[0,0,1]$ respectively. The ordering of the vectors with respect to the elements of $[\tau_{\texttt{hyp}}]$ is not of any particular significance.

3.2 Corpus

Evaluation in Chapters 4, 5 and 6 which deal with melody modelling was carried out on a corpus containing 8 datasets of different sizes and complexities, which serves as a means for comparing the melody models introduced here with n-gram melody models from previous work (Pearce and Wiggins, 2004). The corpus contains datasets of folk melodies of 7 different traditions, and chorale melodies. The data in this corpus can be obtained in the `**kern` format (Huron, 1997) from the *Centre for Computer Assisted Research in Humanities* at Stanford University, California[2] and the *Music Cognition Laboratory* at Ohio State University[3].

The datasets comprising the corpus are as follows (a summary is available in Table 3.3). Six of these are from the Essen Folk Song Collection (EFSC) (Schaffrath and Huron, 1995) and consist of folk melodies, mostly from Europe and China, with a few fron other parts of the world. The present work uses a subset of the EFSC containing 119 Yugoslavian folk melodies, 91 Alsatian folk melodies, 93 Swiss folk melodies, 104 Austrian folk melodies, 213 German folk melodies (dataset `kinder`), and 237 Chinese folk melodies. The number of predictable musical events in these ranges between 2,691 for the smallest one (Yugoslavian folk songs) and 11,056 for the largest (Chinese folk melodies). Another subset of the corpus consists of 152 folk songs and ballads from Nova Scotia, Canada (Creighton, 1966). And the rest are 185 chorale melodies harmonized by J. S. Bach (BWV 253 to BWV 438) (Riemenschneider, 1941).

[2]Website: `http://www.ccarh.org` (last accessed on July 2, 2015).
[3]Website: `http://kern.humdrum.net` (last accessed on July 2, 2015)

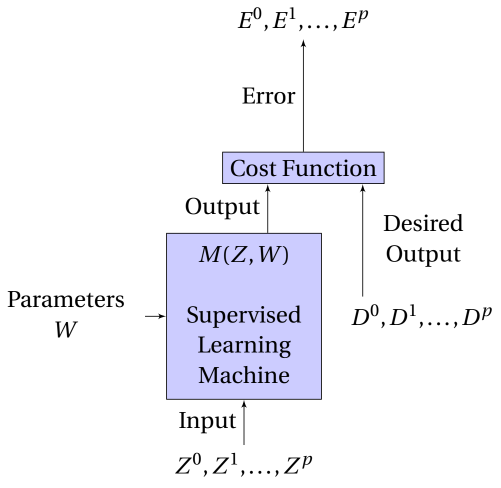
ID	Dataset	Melodies	No. events	Pitches
0	Yugoslavian folk songs	119	2691	25
1	Alsatian folk songs	91	4496	32
2	Swiss folk songs	93	4586	34
3	Austrian folk songs	104	5306	35
4	German folk songs	213	8393	27
5	Canadian folk songs	152	8553	25
6	Chorale melodies	185	9227	21
7	Chinese folk songs	237	11056	41

Table 3.3 The folk and chorale melody datasets used in the present work with their respective total number of musical events and number of prediction categories.

3.3 Gradient-based Machine Learning for Prediction

The prediction models that will be introduced in the Chapters 4 and 5 belong to the class of *Machine Learning* models. Machine learning is a sub-field of Computer Science which deals with developing predictive models that are not programmed to explicitly address a particular task, but can instead *learn* how to do this by exploiting the underlying statistical or mathematical regularities in the task through exposure to large amounts of data representative of it. These are abstract mathematical models characterised by a set of parameters which can be tuned as the models are presented with more and more data, and thus often fully rely on the available data in order to make predictions. For example, in the present case this involves learning to predict a probability distribution over the possible values of the pitch of the next note in a melody given a dataset of melodies from which sequential regularities can be extracted. There exist three distinct types of machine learning, namely supervised learning, unsupervised learning and reinforcement learning. Here we deal with models belonging only to the former two. Out of the six different models that will be encountered here, two - the restricted Boltzmann machine and the recurrent temporal restricted Boltzmann machine belong to the class of unsupervised learning models while the rest learn in a supervised manner.

Of the different classes of machine learning models, the connectionist models considered here belong to one in which the process of tuning (or *learning*) model parameters is iterative and involves updating the values of these parameters in each iteration according to the gradient of the error between the model's predictions and the desired predictions as observed in the data. This *gradient-*



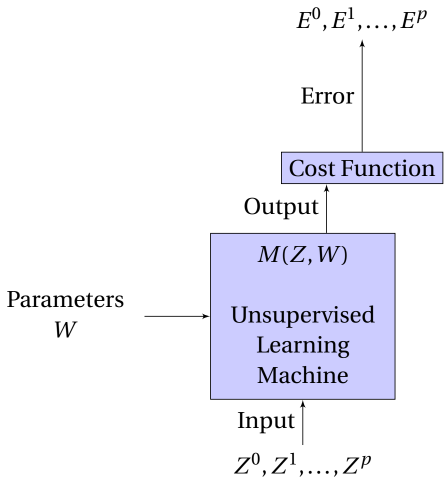


Fig. 3.1 Block-diagrams of supervised (left) and unsupervised (right) machine learning models (LeCun et al., 2012), with the key distinction between the two being the knowledge of the *Desired Output* in the former but not in the latter.

based learning involves three steps (1) estimating the error of the model's output (or prediction) with respect to the desired output using a *cost function* (the error measure) (2) computing the gradients of the model's parameters with respect to the cost function, and (3) optimising the model parameters guided by the magnitude and direction of this gradient (Bottou, 2004; LeCun et al., 2012).

3.3.1 Error Estimation

A machine learning model (Figure 3.1) computes a function $M(Z^p, W)$ where Z^p is the p^{th} pattern representative of a real-world prediction task, and W represents the set of adjustable parameters of the model which are to be tuned to optimal values for the task. In the case of supervised learning, a cost function $E^p = C(D^p, M(Z^p, W))$, measures the discrepancy between D^p, the desired output for pattern Z^p, and the model's output. In the present work, since each of the models predicts a discrete probability distribution over a set of possible output values, we use the *negative log-likelihood* cost function.

$$\begin{aligned} E^p &= -\log P(M(Z^p, W) = D^p) \\ E &= \frac{1}{\mathscr{P}} \sum_{p=1}^{\mathscr{P}} E^p \end{aligned} \tag{3.1}$$

where $P(M(Z^p, W) = D^p)$ gives the probability assigned by the model to the desired output category for the p^{th} pattern as dictated by the task. Intuitively, this cost function maximizes the agreement of the model with the observed data, or in other words makes the data the most probable given the model. On the other hand, in unsupervised learning the cost function is independent of any desired outputs and can be written as $E^p = C(M(Z^p, W))$. In each unsupervised model considered here, generally speaking, its output $M(Z^p, W)$ can be re-interpreted as the probability assigned by it to the p^{th} pattern Z^p, and thus once again the same *negative log-likelihood* cost function can be used to learn its parameters.

$$\begin{aligned} E^p &= -\log P(M(Z^p, W)) \\ E &= \frac{1}{\mathscr{P}} \sum_{p=1}^{\mathscr{P}} E^p \end{aligned} \tag{3.2}$$

As Chapters 4 and 5 shall illustrate, it is possible to treat a supervised learning task as an unsupervised one by interpreting the inputs and their corresponding outputs in the supervised learning task, together as inputs in an equivalent unsupervised learning task. It opens up the possibility of using unsupervised learning algorithms to learn the parameters of models which can then be used to address a supervised learning task. In the present work, this is demonstrated with two models, namely the Restricted Boltzmann Machine (RBM) and the Recurrent Temporal Restricted Boltzmann Machine (RTRBM).

An important assumption in both the above learning scenarios is that all the patterns are *independent and identically distributed*, according to which each pattern (which is a random variable by itself) has the same probability distribution as the others and all are mutually independent. This essentially means that the sequential order of occurrence of the patterns, i.e. the order in which they are presented to the model, is irrelevant for the model to make predictions about them. In probabilistic terms, the two properties

$$\begin{aligned} P(Z^q | Z^p) &= P(Z^q) \\ P(Z^p, Z^q) &= P(Z^p) P(Z^q | Z^p) \\ &= P(Z^p) P(Z^q) \end{aligned}$$

hold for any two arbitrary patterns Z^1 and Z^2 from the data, and thus facilitate the optimisation in (3.1) and (3.2) (Bishop, 2006).

The quantity E in (3.1, 3.2) is the *expected risk*, which is the (theoretical) average of errors E^p over data observed in the real-world. The goal for a machine learning model is to minimise what is known as *empirical risk*, denoted by E_{train}, which is the average of errors E^p over a finite amount of data called the *training set*. This data, in the case of supervised learning, is a set of input/output pairs $\{(Z^1, D^1), \dots, (Z^p, D^p)\}$, or simply a set of input patterns $\{Z^1, \dots, Z^p\}$ in the case of unsupervised learning. The training set is assumed to be a large enough representative sample of the real-world data, thus making E_{train} a reasonable approximation of E (Vapnik and Kotz, 1982). The learning problem is thus reduced to finding the values of the model parameters W that minimise $E_{train}(W)$. In practice, one is more interested in the predictive performance of the model in the field which is measured by data different from the training set, known as the *test set*. This gives an estimate of the model's ability to *generalise*, that is, to predict the correct outputs for inputs it has not previously seen. Given this setup, the minimisation variable W is to be iteratively adapted as a response to observing events Z^p occurring in the training set.

During the iterative learning process, the empirical risk E_{train} progressively decreases over iterations. However, this decrease might not always have a desirable effect on the model's error on the test set E_{test}, which typically reduces up to a certain iteration and worsens beyond it. This behaviour is known as *overfitting* and is a consequence of the optmisation process when it results in a model which gives greater importance to correctly predicting the specific examples presented to it during training (memorising the training set) over examples from the test set belonging to the same data distribution and which it has not encountered previously (generalisation over the test set). As the test set is generally unavailable until the model is deployed in a real-world scenario at which stage its training is expected to have been completed, one relies on a proxy of the test set to check for overfitting. This is known as the *validation set*. The primary reason for using the validation set is to test for overfitting and terminate the learning process when it occurs. It is in no way used to tune the model parameters.

3.3.2 Gradient Computation

The value of the error E_{train} computed over the training set is useful in taking the model, initially unaware of the task, one step closer to addressing it with every iteration of the learning process. The gradient $\frac{\partial E_{train}}{\partial W}$ of the cost function of a model with respect to the model parameters determines the mangitude and sign of the update to be applied to each parameter in order to achieve this. The expression for this gradient depends on the choice of cost function. In some cases it is possible to compute the exact gradient of the function analytically, while in others an approximation is required when the computation is intractable. In the present work, we consider six different connectionist architectures for prediction. The first three of these are non-recurrent models - the Feed-forward Neural Network (FNN), the Restricted Boltzmann Machine (RBM), and the Discriminative Restricted Boltzmann Machine (DRBM). The gradient of the cost function can be exactly computed in the case of the FNN (Rumelhart et al., 1988) and the DRBM (Larochelle and Bengio, 2008). In the case of the RBM, since computing the exact gradient is intractable, it is approximated efficiently using the Contrastive Divergence (CD) algorithm (Hinton, 2002; Tieleman, 2008). The remaining three models can be viewed as the recurrent counterparts of the first three. These are the Recurrent Neural Network (RNN), the Recurrent Temporal Restricted Boltzmann Machine (RTRBM), and the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM). In these recurrent models, the same gradient computation algorithm as in their non-recurrent counterparts is extended using the Backpropagation Through Time (BPTT) algorithm. The algorithms mentioned here will be described in greater detail in Chapters 4 and 5.

3.3.3 Parameter Optimisation

At each iteration, given the model's outputs and their corresponding desired values, the value of each model parameter is updated with the magnitude and sign of the gradient of the cost function E_{train} with respect to it. While there exist several ways of updating the model parameters, the simplest update-rule (employed in the present work), known as *gradient descent*, is given by

$$W_t = W_{t-1} - \eta_t \frac{\partial E}{\partial W} \tag{3.3}$$

where η_t is known as the learning rate, and its purpose is to scale the magnitude of the update to facilitate arriving at the global minimum of the cost function. In the simplest case, η_t is a scalar constant which remains constant across different values of t. Alternatively, its value can be adapted accordingly as learning progresses. In the present work, we employ an adaptive scalar η_t. There exist two variants of gradient descent, which differ in how often one executes (3.3) during training. These are *Batch Gradient Descent* and *Stochastic Gradient Descent*.

Batch Gradient Descent: In the batch variant of the gradient descent update, successive estimates W_t of the set of optimal parameters are computed using the formula

$$W_t = W_{t-1} - \eta_t \frac{1}{P} \sum_{p=1}^{P} \frac{\partial E^p}{\partial W} .$$

That is, each iteration involves computing the average of the gradients of the cost function $\frac{\partial E}{\partial W}$ over the entire training set. When the learning rate η_t is small enough, the algorithm converges towards a local minimum of the empirical risk E_{train}.

Stochastic Gradient Descent: In stochastic (or online) gradient descent, the update given by (3.4) is carried out after computing the prediction error on each example Z^p. Each iteration of stochastic gradient descent involves choosing an example Z^t at random at time t, and updating the parameters W_t according to the formula

$$W_t = W_{t-1} - \eta_t \frac{\partial E^t}{\partial W} . \tag{3.4}$$

The stocastic gradient descent update relies on the hope that random noise introduced by this procedure will not perturbate the average behaviour of the algorithm.

These two variants of gradient descent have contrasting properties. Stochastic gradient descent is usually faster (in the number of iterations) than batch gradient descent particularly on large, redundant datasets as it avoids, to an extent, re-computing gradients on samples that repeat in the training data. This computation is more likely to happen in batch gradient descent as the gradient for each update is computed over the entire dataset. Stochastic gradient descent often results in better solutions because of the noise in the updates. Nonlinear networks such as those employed here have multiple local minima of differing depths, one of which is to be detected during training. Batch gradient descent will discover the minimum of whatever basin the weights are initially placed in. In stochastic learning, the noise present in the updates (as a result of the gradient being computed over a single example) can result in the weights jumping into the basin of another, possibly deeper, local minimum. However, the same noise also prevents full convergence to the minimum, which stalls out due to weight fluctuations. In order to deal with such fluctuations, one can either decrease the learning rate or have adaptive batch size. Another method to remove noise is to use “mini-batches”, whose size is somewhere between a single example and the entire dataset. This approach often allows one to leverage the advantages of both batch and stochastic learning.

3.4 Model Selection

In addition to the parameters of a given model whose optimal values are arrived at through the learning procedure, there exist others determining whose optimal values goes beyond this procedure. These are known as *hyperparameters*. Hyperparameters can be understood as parameters that set the context in which to carry out learning, and are often manually specified by a user working with the learning models. Each model class (neural network, support vector machine, hidden Markov model, etc.), and likewise, optimisation algorithm (gradient descent, conjugate gradients, etc.) has its own distinct set of hyperparameters which are rooted in the theory that underlies it. There is usually a preference for classes of models and optimisation algorithms with fewer hyperparameters unless the contrary offers some valuable advantage, as this reduces the extent of manual involvement on the part of the user. Each instantiation of a given class of models with a particular combination of its hyperparameter is considered to be a different model. For example, given the model class $M(h^1, h^2, h^3)$ with the set of scalar hyperparameters $\{h^1, h^2, h^3\}$, its instantiations $M(h^1 = a, h^2 = b, h^3 = c)$ and $M(h^1 = a, h^2 = c, h^3 = b)$ (for some arbitrary scalar values a, b and c) are considered to be two different models. And it is in relation to these individual models that the above learning procedure was described. Given a model class and optimisation algorithm, the process of determining the best instantiation of this class is known as *model selection*. In the present work, we employ the simplest available approach for model selection, known as *grid search*. This is essentially an iterative, brute-force approach where, in each iteration, one specifies the range of values of each hyperparameters to be examined essentially specifying an $\mathscr{H}$-dimensional grid over which to search for the best model. In each iteration, one hopes to find a global optimum (in terms of model performance) with respect to the hyperparameters within the chosen range of values and repeats the process until this is the case or further iterations are found not to be feasible. While this approach is fairly straightforward to understand and implement, it is also very inefficient as it suffers from *the curse of dimensionality*, wherein an increase in the number of hyperparameters, or the range of values of each hyperparameter considered results in an exponential increase in the number grid points to evaluate the models over. This becomes an issue particularly when training and evaluating each model takes a significant amount of time, and the search range is large. There indeed do exist other recent proposals for more efficient model selection including Bayesian Optimisation (Bergstra et al., 2011; Hutter et al., 2011; Snoek et al., 2012), Random Search (Bergstra and Bengio, 2012) and Gradient-based Optimisation (Chapelle et al., 2002) but these have not been considered here.

Below are listed various model selection hyperparameters that one will encounter in Chapters 4, 5, 6 and 7. Further details can be found in most standard machine learning texts (Bishop, 2006; Bengio, 2012; Hinton, 2012; Murphy, 2012).

1. **Number of Hidden Units:** This is the model hyperparameter which determines the size of the hidden layer, and thus its capacity and the number of model parameters it is characterised by. A smaller number of hidden units offers a computational advantage over the opposite case. A model with a larger hidden layer, as much as it is capable of generalising over more complex patterns present in the data, is also more prone to overfitting than a smaller one.

2. **Hidden Layer Activations:** This is the model hyperparameter corresponding to the (typically non-linear) function applied to the input to each hidden unit of the connectionist models. Its main purpose is to serve as a *quashing function* to bring the input to the hidden unit into a specified range, and to induce non-linearity into an otherwise linear transformation. Some of the non-linearities are faster to compute than others, but typically, whether or not one type of activation or another is good for a certain task is determined through experimentation.

3. **Learning Rate:** Learning rate is an optimisation hyperparameter which scales the magnitude of the model parameter update in each training iteration. A small value results in smaller updates, and thus longer training time. Conversely, a large value results in faster learning, however, there is a greater tendency for the optimiser to overlook the optimal solution in the error surface by skipping over it. It is common practice to use an adaptive learning rate, which is progressively decreased depending on heuristics extracted from the learning process (Senior et al., 2013).

4. **Weight Decay:** Updates during the learning process might often result in large parameter values in comparison with the input features which minimises the influence of the features, thus leading to overfitting. Weight decay is a form of regularisation that penalises large values of parameters, thus keeping them in check.

5. **Momentum:** The momentum method is used to speed up the learning process when the error surface is such that the gradients computed at succes-

sive iterations tend to result in an oscillating path towards the minimum. A higher learning rate will only further amplify these oscillations and is thus not helpful. The momentum method speeds up learning by dampening the oscillations (Bengio, 2012; Sutskever et al., 2013) through an additional term included in the parameter update in each iteration. The scaling coefficient for this term is the momentum hyperparameter, which is an optimisation hyperparameter. Where applicable in the present work, the momentum method proposed by Polyak (1964) has been used.

6. **Early-Stopping:** Early-stopping (Prechelt, 1998) is a regularisation method which prevents overfitting as well as reduces training time. This is based on the observation that, while the prediction error of a model on the training set progressively decreases with every training iteration, the error on unseen data tends to first decrease and then progressively increase when overfitting occurs. The validation set together with an appropriate heuristic can be used to detect when this happens, and terminate the training process such that the best model until that iteration is used for prediction on the test data. The hyperparameters related to early-stopping depend on the choice of the heuristic, which will be explained in detail in later chapters where it is used.

3.4.1 Cross-Validation

In order to carry out model selection as described above, the data is split into training, validation and test sets. It is often the case, in practice, that one is only given the training and test sets. In such a situation, it is common practice to set aside a small portion of the training data as validation data. The validation set is used to determine at what stage the learning process is to be terminated, and the performance of the model thus obtained on the test set is then reported. However, this procedure carried out only once could result in an over-estimation or conversely and under-estimation of the true performance of the model on the dataset especially when the dataset is small. This is also known as a biased estimate. One way of addressing this issue is through the use of k-fold cross-validation (Kohavi et al., 1995; Dietterich, 1998; Bishop, 2006) in which the data is divided into k disjoint subsets of approximately equal size. At each point in the grid, the model is trained (and validated) k times, each time leaving out a different subset to be used for testing and an average of k performance estimates thus obtained is then reported. In the present work $k = 10$, and it is also ensured that where applicable the folds are identical to those used in previous work in order to facilitate fair comparison. The value of $k = 10$ serves as a compromise between the bias associated with low values of k and the high variance associated with high values of k (Kohavi et al., 1995). This was also regarded as necessary by (Pearce and Wiggins, 2004) who also uses the aforementioned melody corpus in Section 3.2, given the small sizes of the datasets contained in it. Moreover, the use of cross-validation would facilitate the analysis of results for statistical significance wherever it is considered necessary using statistical tests such as the paired t test (Dietterich, 1998). The choice of the t test is justified here by its previous use in (Pearce and Wiggins, 2004) whose data splits, evaluation metric and methodology have been adopted here as well for the sake of comparison.

3.5 Prediction Task

The present work focuses on the task of predicting a probability distribution over the possible values of the pitch of a musical event $s^{(t)}$ at time-step t, given the context of musical events $s^{(1:t-1)}$ leading up to it. In the simplest case, the context is made up of the type sequence $\tau^{(1:t-1)}_{\texttt{pitch}}$. Here we present a formalism for this task, which is the focus of Chapters 4 and 5. Chapters 6 and 7 use progressively more general versions of this task, which will be explained in detail there.

The simplest case of the prediction task has strong parallels with previous work in language modelling (Manning and Schütze, 1999). Thus, the analogy to natural language is used here to explain it. In statistical language modelling, the goal is to build a model that can estimate the joint probability distribution of subsequences of words occurring in a language L. A statistical language model (SLM) can be represented by the conditional probability of the next word $w^{(T)}$ given all the previous ones $[w^{(1)}, \ldots, w^{(T-1)}]$ (written $w^{(1:T-1)}$), as

$$P\left(w^{(1:T)}\right) = P\left(w^{(1)}\right) \prod_{t=2}^{T} P\left(w^{(t)} | w^{(1:t-1)}\right) .$$

The present work treats notes in a monophonic melody analogous to words in the above language example. This is inspired by the work in (Conklin and Witten, 1995) where a similar analogy was made between sequences of characters in the English language and notes in music. Here we use an event-based representation of music, wherein the occurrence of each note is treated as a *musical event*. Much in the same way as an SLM, the prediction models here model the conditional distribution $P\left(\tau^{(t)}_{\texttt{pitch}}|\tau^{(1:t-1)}_{\texttt{pitch}}\right)$ given a sequence $s^{(1:T)}$ of musical events from a musical language $L_{\texttt{pitch}}$, such that $\tau^{(t)}_{\texttt{pitch}} \in [\tau_{\texttt{pitch}}]$. To simplify the notation, $\tau^{(t)}_{\texttt{pitch}}$ is written as $p^{(t)}$, and likewise $\tau^{(1:t)}_{\texttt{pitch}}$ as $p^{(1:t)}$. Thus, we have

$$P\left(p^{(1:T)}\right) = P\left(p^{(1)}\right) \prod_{t=2}^{T} P\left(p^{(t)}|p^{(1:t-1)}\right) . \tag{3.5}$$

A simplifying assumption commonly made in this regard, is that only a fixed-number of most recent time-steps influence the prediction about the next event. This is known as the Markov assumption, given by $P(p^{(t)}|p^{(1:t-1)}) \sim P(p^{(t)}|p^{(t-n+1:t-1)})$. For each prediction, context information is obtained from the $(n-1)$ events $s^{(t-n+1:t-1)}$ immediately preceding $s^{(t)}$, thus, reducing (3.5) to

$$P\left(p^{(1:T)}\right) = P\left(p^{(1)}\right) \prod_{t=2}^{T} P\left(p^{(t)}|p^{(t-n+1:t-1)}\right) . \tag{3.6}$$

In language modelling, models that rely on the Markov assumption to model n-length sequences of words are known as n-gram models, where n is the *order* of the model. There are two reasons for this assumption. Firstly, it makes each event together with its corresponding context i.i.d in relation to other events and their respective contexts, thus making it possible to carry out learning on a dataset where the features corresponding to the context can be viewed as inputs to the learning model, and the events to be predicted as the target outputs of the model. Secondly, the amount of data required to directly determine the posterior probabilities of an event at time-step t given its entire history rises exponentially with t. This is, to a great extent, alleviated by the Markov assumption. Specifically, if one models a sequence of length n of type τ that is composed of different temporal arrangements of $|[\tau]|$ discrete symbols, the number of n-gram model parameters to estimate is $|[\tau]|^n$. However, it is to be noted that for the same reason as that mentioned above, higher values of n might also not be desirable. In Chapter 4, we will see how the use of non-recurrent connectionist models with hidden states offers a computational advantage under this assumption, and in the case of the recurrent models of Chapter 5 offers the possibility to altogether do without it. Finally, one would also observe in (3.6) that certain events at the beginning of a sequence would lack a valid context, i.e. $\{s^{(t)} \mid 1 \leq t \leq (n-1)\}$. This issue may be dealt with in a variety of ways depending on the model being used for the purpose. For the connectionist models used in the present work, these are described in Sections 4.4.

3.6 Evaluation

A measure of the success of a predictive model of melody should take into account the possibility that there can be more than one *"correct"* continuation of the melody at each time-step owing to the subjectivity involved in the creation of music, where the choice made by a composer carries some degree of flexibility with it. When the predictive model is purely data-driven, this subjectivity is reflected in the training set in the form of example melodies. This set can be said to have been drawn from a discrete probability distribution over the various symbols that are contained in it. The true nature of this distribution is unknown and the predictive model attempts to learn it using the data available in the training set. Given such a model which predicts a probability distribution over a discrete alphabet at every time-step, its learning objective may be viewed as one of minimizing the distance between its predicted distribution and that of the training set. How well this is achieved is evaluated on a test set, as described above in Section 3.3.

Previous work in language modelling has turned to information theory in order to answer this question of objectively evaluating the success of models that address a very similar word-prediction task (Brown et al., 1992). A measure known as Perplexity, has been widely used in evaluating predictive models of language (Chen et al., 1998). Perplexity measures how well a probabilistic model predicts a sample (such as the test set). It may be used to compare models, where a low perplexity indicates the model is good at predicting the sample. It is derived from the information theoretic quantity known as relative entropy which is a measure of the difference between two probability distributions (Joyce, 2011). Here we use a close variant of Perplexity known as Cross Entropy (H_c), which represents the mean divergence between the entropy calculated from the predicted distribution and that of the correct prediction label (and can be interpreted as the distance between these two distributions) for every sample in some given data. It is obtained by simply taking the logarithm of perplexity. It can be computed over all the events belonging to different sequences in the test set $\mathscr{D}_{test}$, as

$$H_c(P_{mod}, \mathscr{D}_{test}) = \frac{-\sum_{s\in\mathscr{D}_{test}} \sum_{t=1}^{T_s} \log_2 P_{mod}(s^{(t)}|s^{(1:t-1)})}{\sum_{s\in\mathscr{D}_{test}} T_s} \tag{3.7}$$

where P_{mod} is the probability assigned by the model to the pitch of the musical event $s^{(t)}$ in the melody $s \in \mathscr{D}_{test}$ given its preceding context, and T_s is the length of s. Since cross entropy measures the degree of average uncertainty of a model when predicting a sequence of events in a corpus, it can be used to compare the performance of different models on the corpus (Brown et al., 1992; Pearce, 2005). This choice of evaluation measure allows comparison with previous work.

3.7 Summary

This chapter described in detail, the background and terminology necessary to understand much of what follows in Chapters 4, 5, 6 and 7 of this dissertation. We first described the representation schema for the monophonic melodies. We adopt the Multiple Viewpoints representation of music, in which a melody is viewed as a sequence of musical events such that the occurrence of each note is an event in time (to be referred to henceforth also as a *time-step*). At each time-step a variety of features of musicological significance can be extracted and used for making predictions about the same, or other related features in the future time-steps. Here, we first provided the mathematical formulation of the prediction task, which involves a model predicting a probability distribution over the different possible values of musical pitch occurring in a dataset. The models are fully data-driven machine learning models and learn regularities in sequential structure directly from data. Their parameters are determined using gradient-based learning methods. In the present work, we employ a monophonic music corpora consisting of 8 datasets of varying sizes and complexities for comparison with previous work. The models are evaluated using the information theoretic measure known as cross entropy.

Chapter 4

Non-recurrent Melody Models

The first task undertaken in this dissertation is a study of various connectionist architectures for monophonic music modelling, and a comparison of their predictive performance to previously proposed n-gram based melody models under the same modelling assumptions (Pearce and Wiggins, 2004). This serves to evaluate the efficacy of the said connectionist models with respect to previous work, and also to determine whether the results here mirror those observed in recent work demonstrating the predictive superiority of connectionist language models over their n-gram counterparts (Bengio et al., 2003; Mnih and Hinton, 2008; Collobert et al., 2011; Mikolov et al., 2010). In this first of two chapters on *connectionist melody models*, we describe a set of non-recurrent connectionist architectures that address the prediction task introduced in Section 3.5. In these models, time is represented by considering the sequence of events occurring within the bounds of a fixed-size time-window (a *context*) immediately preceding a particular event in a sequence, as a single feature vector that serves as the input. They are "non-recurrent" as there exists no means by which the state of the model from the past can be re-introduced as input to it at any given point in time. We consider two classes of models in this chapter, namely Feed-forward Neural Networks (Rumelhart et al., 1988) and restricted Boltzmann machines (Smolensky, 1986), the latter of which have been trained both in an unsupervised (Hinton, 2002; Tieleman, 2008) and a supervised (Larochelle and Bengio, 2008) manner. The predictive performance of these models is compared to that of state-of-the-art n-gram models on the chorale and folk melody corpus described in Section 3.2. The key results of these experiments are that (1) in its best case, each class of connectionist models outperforms n-gram models, and (2) the predictive performance of the connectionist models progressively improves upto context lengths longer than in the case of n-gram models.

4.1 Representing Sequences

The Markov assumption (*cf.* Section 3.5) holds when modelling sequences with non-recurrent connectionist models, which means that only a fixed number of events immediately preceding the one to be predicted are given as inputs to the models. The feature vectors corresponding to individual events are concatenated (while preserving their serial order) into a single feature vector, which serves as input to the prediction model (Figure 4.1). There may, or may not be an overlap between events in consecutive time-windows. Some recent work in connectionist music modelling has relied on this representation of time (Bickerman et al., 2010; Spiliopoulou and Storkey, 2011). These are presented to the model in its input layer, and the prediction is made in the output layer. In the remainder of this chapter and the one that follows, a "time-step" refers to the occurrence of a musical event. Also, we adhere to the same simplified notation for musical pitch used in Section 3.5, i.e., that $\tau^{(t)}_{\texttt{pitch}}$ is written as $p^{(t)}$, and likewise $\tau^{(1:t)}_{\texttt{pitch}}$ as $p^{(1:t)}$. To predict the pitch of the musical event $p^{(t)}$ at time-step t, non-recurrent connectionist models consider the pitches of the sequence of n ($n \in \mathbb{Z}^+$) immediately preceding musical events $p^{(t-n+1:t-1)}$. This is achieved by concatenating the feature vectors corresponding to each of these events along their lengths, as illustrated in Figure 4.1. In this figure, the value of musical pitch $p^{(t)}$ of an event $s^{(t)}$ at time t is represented as a one-hot encoded vector $\mathbf{p}^{(t)} \in \{0,1\}^{|[\tau_{pitch}]|}$ (*cf.* Section 3.1.2), where $|[\tau_{pitch}]|$ is the size of the syntactic domain of the type `pitch`. The result of this concatenation is the final input feature vector $\mathbf{x}^{(t)} \in \{0,1\}^{n|[\tau_{pitch}]|}$.

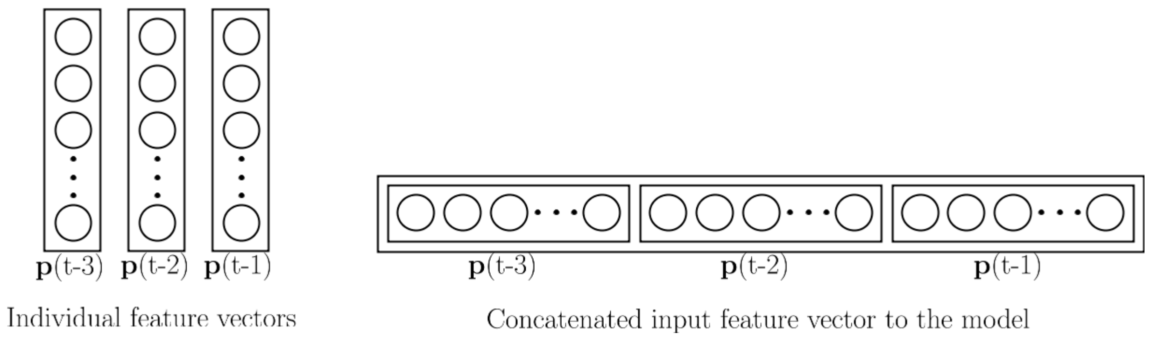

p(t-3)
p(t-2)
p(t-1)
Individual feature vectors
p(t-3)
p(t-2)
p(t-1)
Concatenated input feature vector to the model

Fig. 4.1 Illustration of the feature representation used as input with non-recurrent connectionist architectures for sequences. Here, feature vectors corresponding to three time-steps $\mathbf{p}^{(t-3)}$, $\mathbf{p}^{(t-2)}$ and $\mathbf{p}^{(t)}$ are concatenated into a single vector which serves as the input to the model which predicts the event at time-step t.

4.2 Feed-forward Neural Network

Feed-forward neural networks (FNNs) (Hornik et al., 1989)learn a functional mapping between an input vector $\mathbf{x}$ and a scalar output y (or alternatively, a vector output $\mathbf{y}$). One may understand an FNN as a layered architecture in which a given input $\mathbf{x}$ in the bottom layer propagates up through the layers, undergoing an affine projection and (often) a non-linear transformation at each layer to produce an output $\mathbf{y}$. Its parameters are optimized according to a criterion which minimizes some form of error between the model's output $\mathbf{y}$ and the expected output $\mathbf{y}^*$.

In its simplest form, an FNN (Figure 4.2) consists of an input layer $\mathbf{x} \in \mathbb{R}^{n_i}$, a hidden layer $\mathbf{h} \in \mathbb{R}^{n_h}$, and an output layer $\mathbf{y} \in \mathbb{R}^{n_o}$. Each individual element of a layer is referred to as a "unit" i.e. input *unit,* hidden *unit* and output *unit.* A given input vector $\mathbf{x}$ is mapped onto the hidden layer by first multiplying it with an $n_i \times n_h$ weight-matrix $W_{ih} \in \mathbb{R}^{n_i \times n_h}$ and applying an element-wise non-linear function f_h to the result $\mathbf{u}_h$ of this product:

$$\begin{aligned} \mathbf{u}_h &= \mathbf{b}_h + W_{ih}^\top \mathbf{x} \\ \mathbf{h} &= f_h(\mathbf{u}_h) \ . \end{aligned}$$

where $\mathbf{b}_h \in \mathbb{R}^{n_h}$ is the hidden layer bias vector. Likewise, the hidden layer activation vector $\mathbf{h}$ undergoes a similar transformation to produce the outputs.

$$\begin{aligned} \mathbf{u}_o &= \mathbf{b}_o + W_{ho}^\top \mathbf{h} \\ \mathbf{y} &= f_o(\mathbf{u}_o) \end{aligned}$$

where f_o is a non-linear function applied at the output layer, which depends on the nature of the machine learning task of interest. The term $\mathbf{b}_o \in \mathbb{R}^{n_o}$ is the output layer bias, and the projection happens via the weight-matrix $W_{ho} \in \mathbb{R}^{n_h \times n_o}$. Thus, for a given input $\mathbf{x}$, the output $\mathbf{y}$ is calculated as

$$\mathbf{y} = f_o\left(\mathbf{b}_o + W_{ho}^\top \cdot f_h\left(\mathbf{b}_h + W_{ih}^\top \mathbf{x}\right)\right) \ .$$

The input-to-hidden weights W_{ih}, hidden-to-output weights W_{ho}, together with the hidden and output layer biases $\mathbf{b}_h$ and $\mathbf{b}_o$ respectively make up the parameters of the model.

In the present work, the feature representation described above in Section 4.1 allows the encoding of temporal information in the input (first) layer of the FNN which was originally a non-temporal model. This encoding is inspired by Time-

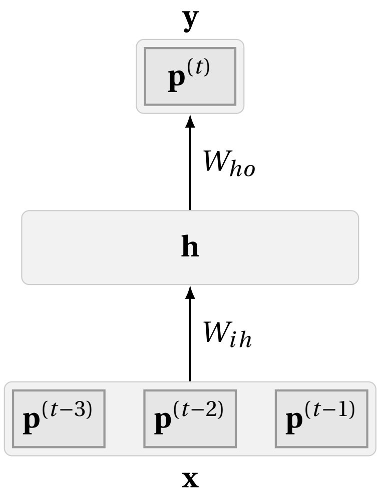


Fig. 4.2 The architecture of a feed-forward neural network melody model with a single hidden layer. It models sequences of 4 musical events so as to predict the fourth event in the output layer on top, given the first three in the input layer at the bottom.

Delay Neural Networks (Waibel et al., 1989) which were first applied to speech recognition before seeing application elsewhere in other domains. We employ networks with only a single non-linear hidden layer between the input and output layers, as described above (*cf.* Figure 4.2). It is possible for more than one hidden layer to exist, in which case the model is said to be a *deep* feed-forward neural network. This is beyond the scope of the present work, and the reader is referred to (Bengio, 2009) for further details. Several options exist with regards to choosing the hidden layer non-linearities. While exploring these, one is faced with a trade-off in terms of learning speed, modelling capability, and the availability of sufficient data before arriving at a suitable solution through a combination of intuition and experimentation. Here we explore three alternatives, namely Logistic Sigmoid, Hyperbolic Tangent and Rectifier units. The functions corresponding to these unit types are listed in Table 4.1. The choice of the function f_o applied at the output layer of the network depends on the nature of the task at hand. In our case, the task of predicting the next musical pitch may be viewed as a C-class classification task where each class corresponds to a value of musical pitch. For this classification task, the network will have C output units where the value of each unit y_c is obtained by applying the *softmax* function σ_{max}, to the activation of that unit u_{oc} (Bridle, 1990) as given by

$$y_c = \sigma_{max}(u_{oc}) = \frac{\exp(u_{oc})}{\sum_{c' \in C} \exp(u_{oc'})} . \tag{4.1}$$

Name	Function $f_h(z)$
Logistic Sigmoid	$\frac{1}{1+\mathrm{e}^{-z}}$
Hyperbolic Tangent	$\frac{\mathrm{e}^{z}-\mathrm{e}^{-z}}{\mathrm{e}^{z}+\mathrm{e}^{-z}}$
Rectifier	$max(0,z)$

Table 4.1 Different types of activation $f_h(z)$ for the hidden layer of the feed-forward neural network (also applicable to the recurrent neural networks of Section 5.1), together with their defining functions.

Each output of the network can thus be interpreted as a conditional probability $P(y_c|\mathbf{x})$.

4.2.1 Learning

The feed-forward neural network (FNN) can be learned using the Backpropagation algorithm (Rumelhart et al., 1988). This algorithm applies the chain rule of differentiation to propagate the error between the target output and the output of the model backwards into the network, and use these derivatives to appropriately update the model parameters. Given the cost function E_p for each training example pair $(\mathbf{x}^p, \mathbf{y}^p)$ at the output layer, the derivatives of the cost with respect to the parameters of the output layer of the network are given by

$$\frac{\partial E^p}{\partial W_{ho}} = \frac{\partial E^p}{\partial \mathbf{u}_o}\frac{\partial \mathbf{u}_o}{\partial W_{ho}}$$
$$\frac{\partial E^p}{\partial \mathbf{b}_o} = \frac{\partial E^p}{\partial \mathbf{u}_o}\frac{\partial \mathbf{u}_o}{\partial \mathbf{b}_o}$$

where,

$$\frac{\partial E^p}{\partial \mathbf{u}_o} = \frac{\partial E^p}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{u}_o}.$$

And likewise, those of the input layer parameters are given by

$$\frac{\partial E^p}{\partial W_{ih}} = \frac{\partial E^p}{\partial \mathbf{u}_h}\frac{\partial \mathbf{u}_h}{\partial W_{ih}}$$
$$\frac{\partial E^p}{\partial \mathbf{b}_h} = \frac{\partial E^p}{\partial \mathbf{u}_h}\frac{\partial \mathbf{u}_h}{\partial \mathbf{W}_h}$$

where,

$$\frac{\partial E^p}{\partial \mathbf{u}_h} = \frac{\partial E^p}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{u}_h}$$
$$\frac{\partial E^p}{\partial \mathbf{h}} = \frac{\partial E^p}{\partial \mathbf{u}_o} \frac{\partial \mathbf{u}_o}{\partial \mathbf{h}}.$$

These gradients are used to update the values of the corresponding network parameters. For the FNN used here, the cost function is negative log-likelihood, given by

$$E(\mathbf{y},\mathbf{y}^*) = -\sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}_{train}} \mathbf{y}\log(\mathbf{y}^*) \tag{4.2}$$

where $\mathbf{y}$ is the output distribution predicted by the network, $\mathbf{y}^*$ the true distribution (a one-hot encoding of the class-label) and $\mathbf{x}$ their corresponding input vector from the training data $\mathcal{D}_{train}$. Using this cost function and the above expressions of the gradients, the optimal parameters of the model corresponding to a given dataset can be obtained using the iterative gradient-based optimisation technique described in Section 3.3.

4.3 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) (Smolensky, 1986) is an undirected bipartite graphical model. It contains a set of visible units $\mathbf{v} \in \mathbb{R}^{n_v}$ and a set of hidden units $\mathbf{h} \in \mathbb{R}^{n_h}$ which make up its visible and hidden layers respectively. The two layers are fully inter-connected but there exist no connections between any two hidden units, or any two visible units. Additionally, the units of each layer are connected to a bias unit whose value is always 1. The edge between the i^{th} visible node and the j^{th} hidden node is associated with a weight w_{ij}. All these weights are together represented as a weight matrix $W \in \mathbb{R}^{n_v \times n_h}$. The weights of connections between visible units and the bias unit are contained in a visible bias vector $\mathbf{b} \in \mathbb{R}^{n_v}$. Likewise, for the hidden units there is a hidden bias vector $\mathbf{c} \in \mathbb{R}^{n_h}$. The RBM is fully characterized by the parameters W, $\mathbf{b}$ and $\mathbf{c}$. Its bipartite structure is illustrated in Figure 4.3.

The RBM is a special case of the Boltzmann Machine – an energy-based model (LeCun et al., 2006) which gives the joint probability of every possible pair of visible and hidden vectors via an energy function E, according to the equation

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\mathrm{e}^{-E(\mathbf{v},\mathbf{h})}$$

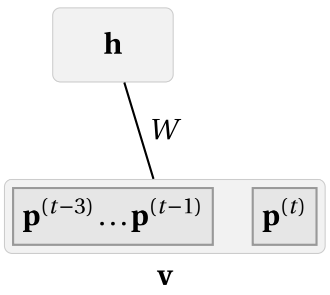


Fig. 4.3 The architecture of a Restricted Boltzmann Machine. It models a sequence of 4 musical events so as to predict the fourth event given the first three. The visible layer of the model contains both the inputs to the model and its predictions.

where the “partition function”, Z, is given by summing over all possible pairs of visible and hidden vectors

$$Z = \sum_{\mathbf{v},\mathbf{h}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})} \, .$$

and ensures that $P(\mathbf{v}, \mathbf{h})$ is a probability. The joint probability assigned by the model to a visible vector $\mathbf{v}$, is given by summing (marginalising) over all possible hidden vectors:

$$P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})}$$

In the case of the RBM, the energy function E is given by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W}\mathbf{h} \, .$$

In its original form, the RBM models Bernoulli distributions in its visible and hidden units. When this is the case, the activation probabilities of the units in the hidden layer given the visible layer (and vice versa) are $P(\mathbf{h} = \mathbf{1}|\mathbf{v}) = \sigma(\mathbf{c} + W^\top \mathbf{v})$ and $P(\mathbf{v} = \mathbf{1}|\mathbf{h}) = \sigma(\mathbf{b} + W\mathbf{h})$ respectively, where $\sigma(\mathbf{x})$ is the logistic sigmoid function $\sigma(\mathbf{x}) = (1 + e^{-\mathbf{x}})^{-1}$ applied element-wise to the vector $\mathbf{x}$. This, however, is not binding and its visible layer can be generalized to model other distributions such as Multinomial and Gaussian (Welling et al., 2004; Dahl et al., 2012).

4.3.1 Learning

In order to use the RBM as a prediction model for sequences, a feature vector corresponding to a fixed-length sequence of events is presented as input to the model in its visible layer. The process of obtaining this feature vector is as described in Section 4.1. There are two ways in which the the model can be made to predict the probability distribution of interest $P(p^{(t)}|p^{(t-n+1:t-1)})$. The first involves learning the parameters of the model by optimising a cost function which maximises the negative log-likelihood of the joint probability distribution of the sequences $P(p^{(t-n+1:t)})$, illustrated in Figure 4.3 for $n = 4$. This is the unsupervised approach and is also known as *generative learning*. With this approach, one can infer the conditional probability distribution of interest $P(p^{(t)}|p^{(t-n+1:t-1)})$ from the above joint distribution. The second approach involves directly optimising the conditional distribution of interest, without having to first compute the joint probability distribution. This is the supervised approach also known as *discriminative learning*, and the model that results is known as the Discriminative RBM (DRBM). We describe these two approaches to learning the RBM's parameters below.

Generative Learning

Learning in energy-based models can be carried out in a *generative* fashion, by updating the weights and biases in order to minimize the overall energy of the system with respect to the training data. This amounts to maximizing the log-likelihood function L over the training data $\mathcal{V}$ (containing N examples), which is given by

$$L = \frac{1}{N} \sum_{n=1}^{N} \log P(\mathbf{v}_n)$$

where $P(\mathbf{v})$ is the joint probability distribution given by

$$P(\mathbf{v}) = \frac{e^{-E_{free}(\mathbf{v})}}{Z},$$

with $Z = \sum_{\mathbf{v}} e^{-E_{free}(\mathbf{v})}$, and

$$E_{free}(\mathbf{v}) = -\log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} .$$

The probability that the RBM assigns to a vector $\mathbf{v}_n$ belonging to the training data can be raised by adjusting the weights and biases to lower the energy associated with that vector and to raise the energy associated with others not in the training data. Learning can be carried out using gradient-based optimisation, for which the gradient of the log-likelihood function with respect to the RBM's parameters θ needs to be calculated first. This is given by

$$\frac{\partial L}{\partial \theta} = -\left\langle \frac{\partial E_{free}}{\partial \theta} \right\rangle_0 + \left\langle \frac{\partial E_{free}}{\partial \theta} \right\rangle_\infty$$

where $\langle\cdot\rangle_0$ denotes the average with respect to the data distribution, and $\langle\cdot\rangle_\infty$ that with respect to the model distribution. The former is readily computed using the training data $\mathcal{V}$, but the latter involves the normalisation constant Z, which very often cannot be computed efficiently as it involves a sum over an exponential number of terms.To avoid the difficulty in computing the above gradient, an efficiently computable and also effective approximation of the gradient was proposed in the Contrastive Divergence method (Hinton, 2002; Tieleman, 2008). This method exploits the bipartite structure of the RBM, and has been widely used to train the RBM. The present work employs this algorithm for learning the parameters of the RBM generatively.

Discriminative Learning

The generatively trained RBM described above models the joint probability $P(\mathbf{v})$ of the set of visible units $\mathbf{v}$. However, one is often interested in a conditional distribution of the form $P(y|\mathbf{x})$. It has been demonstrated in (Salakhutdinov et al., 2007; Larochelle and Bengio, 2008) how discriminative learning and inference can be carried out in the RBM, thus making it feasible to use it as a classifier on its own. This is done by assuming one subset of its visible units to be inputs $\mathbf{x}$, and the remaining a set of multinomial units $\mathbf{y}$ representing the class-conditional probabilities $P(y|\mathbf{x})$. This is illustrated in Figure 4.4 for a sequence of length 4 ($n = 4$). The weight matrix W can be interpreted as two matrices $R \in \mathbb{R}^{n_i \times n_h}$ and $U \in \mathbb{R}^{n_c \times n_h}$, where n_i is the input dimensionality and n_c is the number of classes and $n_v = n_i + n_c$. Likewise, the visible bias vector $\mathbf{b} \in \mathbb{R}^{n_v}$ is also split into a set of two bias vectors — a vector $\mathbf{a} \in \mathbb{R}^{n_i}$ and a second vector $\mathbf{d} \in \mathbb{R}^{n_c}$, as shown in Figure 4.4.

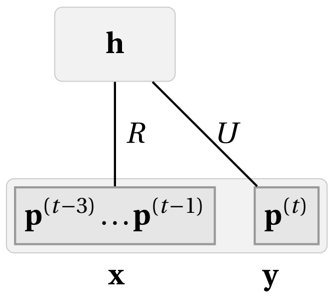


Fig. 4.4 The architecture of a discriminative RBM melody model. It models a sequence of 4 musical events so as to predict the fourth event given the first three. While its structure is identical to that of the RBM, the difference in its learning algorithm leads to the split of the weight matrix W of the RBM into matrices R and U.

The posterior probability din this Discriminative RBM can then be inferred as

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp\left(-E_{free}(\mathbf{x},\mathbf{y})\right)}{\sum_{\mathbf{y}^*} \exp\left(-E_{free}(\mathbf{x},\mathbf{y}^*)\right)} \tag{4.3}$$

where $\mathbf{x}$ is the input vector, and $\mathbf{y}$ is the one-hot encoding of the class-label. The denominator sums over all class-labels $\mathbf{y}^*$ to make $P(\mathbf{y}|\mathbf{x})$ a probability distribution. In the original RBM, $\mathbf{x}$ and $\mathbf{y}$ together make up the visible layer $\mathbf{v}$. The model is learned discriminatively by maximizing the log-likelihood function derived from the distribution

$$P(y|\mathbf{x}) = \frac{\exp(d_y)\prod_j \left(1+\exp(\mathbf{r}_j^\top \mathbf{x} + u_{yj} + c_j)\right)}{\sum_{y^*}\exp(d_{y^*})\prod_j \left(1+\exp(\mathbf{r}_j^\top \mathbf{x} + u_{y^*j} + c_j)\right)} \,. \tag{4.4}$$

The gradient of this function, for a single input-label pair $(\mathbf{x}_i, y_i)$ with respect to its parameters θ can be computed analytically. It is given by

$$\begin{aligned}\frac{\partial \log P\left(y_i|\mathbf{x}_i\right)}{\partial\theta} &= \sum_j \sigma\left(o_{yj}\left(\mathbf{x}_i\right)\right)\frac{\partial o_{yj}\left(\mathbf{x}_i\right)}{\partial\theta} \\ &\quad - \sum_{j,y^*} \sigma\left(o_{y^*j}\left(\mathbf{x}_i\right)\right) P\left(y^*|\mathbf{x}_i\right)\frac{\partial o_{y^*j}\left(\mathbf{x}_i\right)}{\partial\theta}\end{aligned}$$

where $o_{yj}(\mathbf{x}) = c_j + \mathbf{r}_j^\top \mathbf{x} + u_{yj}$. This gradient can be computed efficiently, and the cost optimised as described in Section 3.3. Note that in order to compute the conditional distribution in (4.4) the model does not have to be learned discriminatively, and one can also use the above generatively learned RBM as it learns the joint distribution $P(\mathbf{y},\mathbf{x})$ from which $P(\mathbf{y}|\mathbf{x})$ can be inferred (Salakhutdinov et al., 2007; Larochelle and Bengio, 2008).

4.4 Experiments

Evaluation of the above described non-recurrent connectionist models was carried out on the chorale and folk melody corpus described in Section 3.2. This facilitated comparison with melody models based on n-grams from previous work (Pearce and Wiggins, 2004). The evaluation was carried out on 10 resampling sets of each of the 8 datasets in the corpus, thus making it possible to also test the significance (Dietterich, 1998) of the differences in performance between the models where required. Training and test folds identical to those in (Pearce and Wiggins, 2004) were used here as well. We extracted a small part of the training set (5% of the total number of samples) in each case as the validation set. A grid search was carried out to determine the best set of hyperparameters for the corpus, the details of which differ slightly between models and are described below in their respective sections. While carrying out the iterative gradient-based learning on a model at each grid-point, its performance on the validation set was checked after every 10 iterations. The model corresponding to the iteration with the best validation set performance was chosen to be evaluated on the test set, and this performance reported.

Comparison with Previous Work

The connectionist models evaluated here have been compared with n-gram models from (Pearce and Wiggins, 2004). There, two different types of models were evaluated both individually and in combination. The first of these was a Long-Term Model (LTM), that was governed by structure and statistics induced from a large corpus of sequences from the same musical style. And the other was a Short-Term Model (STM) which relied on structure and statistics particular to the melody being predicted. From a machine learning perspective, the LTM is a model whose parameters are learned offline from a dataset of melodies. The parameters of the STM are learned online while making predictions on the testing data, without any sequence learning occurring in it beforehand. The connectionist prediction models considered here deal only with effects that are induced from a corpus, and are thus compared with the two best performing LTMs in (Pearce and Wiggins, 2004) of unbounded order and order bound 2 respectively. Moreover, as n-grams rely explicitly on the occurrence frequencies of sequences, it is often the case that the model comes across a never-before-encountered context on which to predict the future event, and this is more common in higher order models. This issue has been dealt with by using *smoothed n-grams* (Chen and Goodman, 1999) that use lower-order transition probabilities for generating approximations (through interpolation with, or scaling) of higher-order probabilities. This also applies to events that lack a valid context, i.e. $\{s^{(t)} \mid 1 \leq t \leq n\}$. The n-gram model of bounded order that is used for comparison in the present work has been coined *C2I*. In this name, the 2 indicates the order of the model (the number of past musical events taken into consideration at the time of prediction). The *I* refers to the use of *interpolated smoothing* in it which means that the generated probability distribution over the possible values of the next event are obtained through a weighted sum of the highest order (in this case 2) and the next highest order (in this case 1). The particular interpolated strategy used is labelled C and was proposed in (Moffat et al., 1998). An enhanced variant of this model is the unbounded C^*I model in which, the C and the I hold the same significance as in the case of the bounded order $C2I$ model. The * means that, when available, the shortest deterministic context is always chosen to generate the prediction probabilities for the next event. Otherwise these values are determined using the longest available matching context (with the type C interpolated smoothing applied). As the name suggests, the *deterministic context* is one that results in a single possible continuation of the context in the entire corpus used to train the model. We refer the interested reader to (Pearce and Wiggins, 2004) for further details on these two models. To facilitate a direct comparison between the two approaches, the melodies have not been transposed to a default key.

Boundary Conditions

One assumption to be noted here is regarding the handling of events at the beginning of a melody. When the model relies on a preceding context of length n, the first n events in a melody, i.e. $\{s^{(t)} \mid 1 \leq t \leq n\}$, do not have a valid context. One may rely on a task-independent prior assumption to represent these missing events. Since we deal here with one-hot encodings of musical events which may be interpreted as our knowledge of the probability of occurrence of an event, we represent each non-existent context event by a vector of dimensionality $|[\tau_{pitch}]|$ of any other input vector (which is the number of pitches occurring in the dataset), but with each element of value $1/|[\tau_{pitch}]|$. This signifies a uniform distribution, and thus a lack of any knowledge whatsoever about the nature of these missing events. Consider for the sake of example a simple case where the musical pitch of an event $s^{(t)}$ can assume one of only three values. Here, these would be represented by $[\tau_{pitch}] = \{[1,0,0],[0,1,0],[0,0,1]\}$. Thus, $|\tau_{pitch}| = 3$, and an event belonging to a missing context would be represented by the vector $[1/3,1/3,1/3]$. While this assumption is as good as any other (such as a vector containing only zeros, or one with an additional dimension added to it to account for a missing context event) with regards to the feed-forward neural networks, its probabilistic interpretation is also meaningful in the case of the restricted Boltzmann machine in which each musical event is represented as a set of units in the visible layer which together make up a multinomial distribution. This allows a uniform choice of representation across both classes of models.

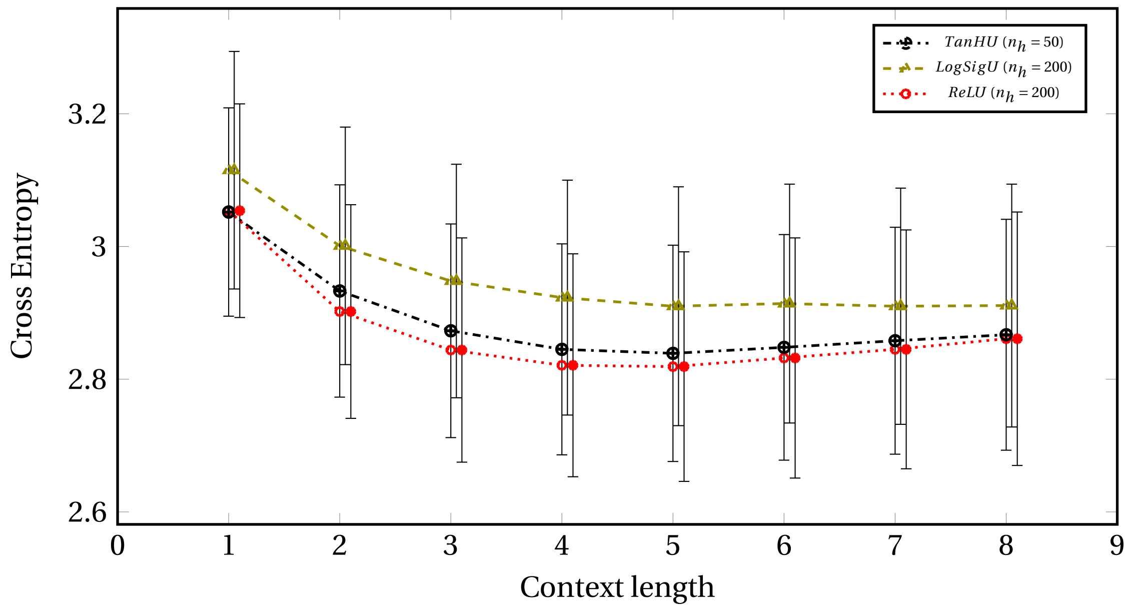


Fig. 4.5 Figure with plots corresponding to the performance of feed-forward neural networks with different types of hidden layer activations, at different context lengths. Each plot corresponds to the best number of hidden units for that type of activation. The vertical error bars represent standard deviation of the cross entropies.

4.4.1 Feed-forward Neural Networks

Each candidate model of the grid search was trained upto a maximum of 250 iterations, and that with the best validation set error was chosen for evaluation on the test set. Based on a preliminary grid search on the datasets where only the learning rate was varied, the initial learning rate η_{init} was set to 0.05. During training, this value decayed with the number of iterations according to the schedule given by $\eta_i = \eta_{init}/(1 + i/\beta)$, where $\beta = 50$ is the number of iterations after which each step of decay occurs (Senior et al., 2013). The number of hidden units in the models n_{hid} was varied as $\{25, 50, 100, 200\}$. Both L_1 and L_2 weight-decay were set to identical values $\lambda_1 = \lambda_2 = \lambda$ and were either on ($\lambda = 0.0001$) or off ($\lambda = 0$). Additionally, the three different hidden layer activation types listed in Table 4.1 were considered while evaluating the feed-forward neural networks. The length of the preceding context of musical events was varied between 1 event and 8 events. The results of the evaluation are shown in Figure 4.5. The horizontal axis corresponds to the number of immediately preceding pitches used for predicting the next one (the context length). The vertical axis corresponds to prediction cross entropy (*cf.* Section 3.6). It should be noted that this is a quantity which is to be minimised, and thus a smaller value reflects better predictive performance. The values of cross entropy plotted in the figure are the averages across all 8 datasets of the chorale and folk melody corpus. Detailed results for each of the 8 datasets can be found Tables A.1, A.2 and A.3 of Appendix A.

It was observed that three key factors influenced the performance of the models. These are (1) the size of the hidden layer, (2) the hidden unit activation type, and (3) the length of the preceding context. Increasing the size of the hidden layer resulted in improved predictions for all models except those containing hyperbolic tangent (TanHU) hidden units, where a hidden layer of 50 units performed better than those with more. In networks with the TanH activation, performance consistently worsened with increase in hidden layer size when models used context lengths greater than 5 for prediction. Increasing the context length generally resulted in an improvement in predictive performance in the case of all the models upto a certain point beyond which the performance deteriorated. This deterioration was less notable in the case of the LogSigU networks. Of the three hidden layer activation types and the various context lengths, the best predictive performance was observed in the case of the ReLU networks at a context length of 5, closely followed by the TanHU and LogSigU networks. An additional advantage of ReLUs happens to be that this type of activation is faster to compute, and also has a simpler gradient which facilitates faster learning (Krizhevsky et al., 2012). Moreover, the results here are also in agreement with previous work which has demonstrated the effectiveness of ReLUs on various other machine learning tasks, particularly when used in deep neural networks (Nair and Hinton, 2010; Zeiler et al., 2013).

4.4.2 Restricted Boltzmann Machines

We examined the two variants of the restricted Boltzmann machine (RBM) described above — those trained generatively and discriminatively. In both cases, each candidate model of the grid search was trained up to a maximum of 500 iterations, and that corresponding to the iteration with the best validation set error was chosen for evaluation on the test set. The result of this can be considered equivalent to early-stopping where learning is terminated according to a certain heuristic when the performance of a model does not improve in successive iterations during training. An initial search on the learning rate η_{init} was carried out by varying it as $\{0.01, 0.05, 0.1\}$. Based on this search, a learning rate of 0.01 was found to be suitable for the generative RBM, and 0.1 for the discriminative one. Unlike the FNNs, a schedule was not used here and the learning rate was kept constant throughout the training process. The number of hidden units in the models n_{hid}, was initially

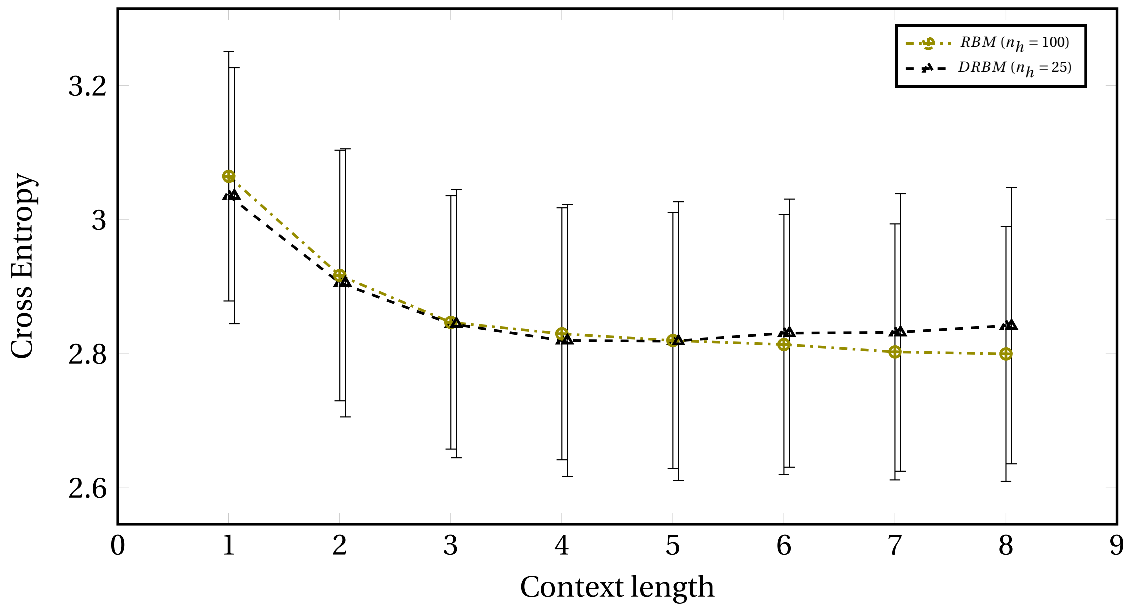


Fig. 4.6 Figure with plots corresponding to the performance of Restricted Boltzmann Machines with different types of hidden layer activations, at different context lengths. Each plot corresponds to the best number of hidden units for that type of activation. The vertical error bars represent standard deviation of the cross entropies.

varied as $\{25, 50, 100\}$. On observing that the best performing model in the case of the generative RBM contained 100 hidden units, the search was extended to 200 hidden units as well. On the other hand, it was extended in the opposite direction to 10 hidden units for the discriminative RBM whose best case contained 25 hidden units. Weight decay (only L_1) λ was either on ($\lambda = 0.0001$) or off ($\lambda = 0$). The length of the preceding context of musical events was varied between 1 event and 8 events, as in the case of the FNN above. The results of the evaluation are shown in Figure 4.6. The horizontal axis corresponds to the number of immediately preceding pitches used for predicting the next one (the context length). The vertical axis corresponds to prediction cross entropy (*cf.* Section 3.6).It should be noted that this is a quantity which is to be minimised, and thus a smaller value reflects better predictive performance. The values of cross entropy plotted in the figure are the averages across all 8 datasets of the chorale and folk melody corpus. The key factors influencing the performance of the models here are (1) the size of the hidden layer and (2) the length of the preceding context. It was observed here as well that increasing the size of the hidden layer resulted in improved predictions for the generative RBM, with the best model containing 100 hidden units. Near identical performance was observed with models containing 200 hidden units but

Model	Context	Cross Entropy
n-gram (b)	2	2.957
n-gram (u)	N/A	2.878
FNN	5	2.819($\pm$0.200)
DRBM	5	2.819($\pm$0.208)
RBM	8	2.799($\pm$0.168)

Table 4.2 Table comparing the best predictive performance of the different models in the evaluation.

these have been considered worse as they are less parsimonious than those containing 100 units. On the other hand, a better predictive performance was oberved in discriminative RBMs with a smaller-sized hidden layer containing 25 units. Similarly, increasing the context length also resulted in better predictive performance. In the generative RBMs, no deterioration in performance within the range of context lengths examined was observed, unlike the the case of the FNN and the discriminative RBM. However, it is to be noted that in the case of 3 of the 8 datasets (`bach`, `kinder` and `shanxi`) the performance did either remain the same or slightly worsen at longer context length, but this effect was averaged out when considering the remaining datasets. Detailed results for each of the 8 datasets for both the RBM variants can be found in Tables A.4 and A.5 of Appendix A.

4.4.3 Discussion

Table 4.2 contains the best predictive performance of each of the models considered in the comparison here. As mentioned previously, the results are averaged across all 8 datasets. One will notice the progressive improvement in the best-case performance from the n-gram models, to the non-recurrent connectionist models, RBM performing better than the rest. One can possibly attribute the difference observed here between the n-grams and the connectionist models to the difference in their respective learning mechanisms. In the n-gram models, probabilities are determined using counts of the occurrence of a certain event after various event contexts. When a context for prediction does not match any of those that the model has already come across in the training data, back-off and interpolated smoothing techniques are applied to make up for this lack of information in the model (Pearce and Wiggins, 2004). In contrast, the connectionist models learn a

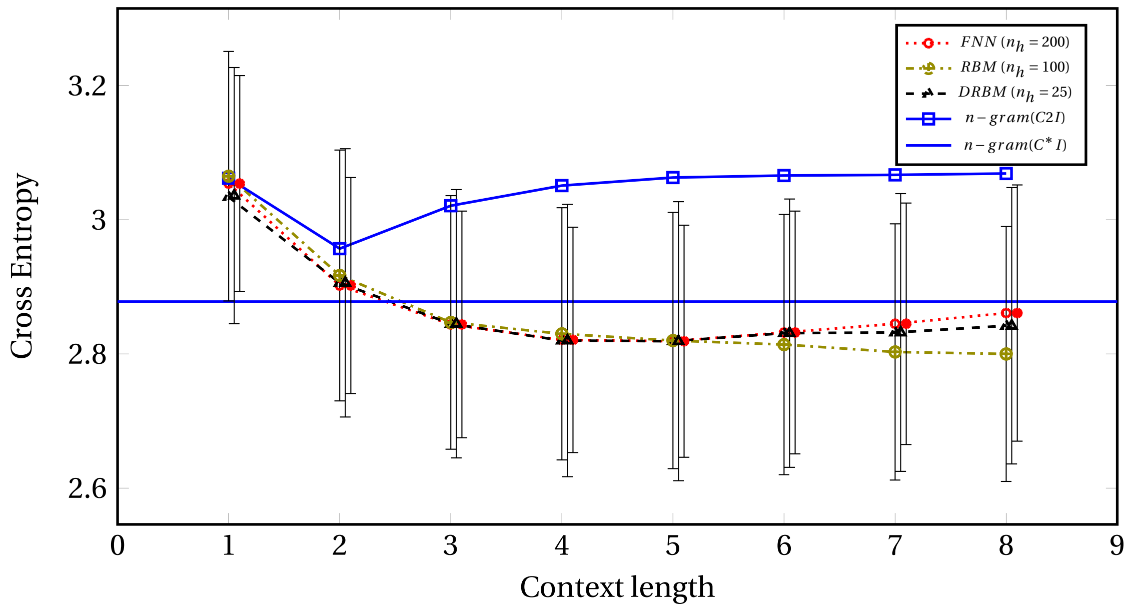


Fig. 4.7 Figure comparing the predictive performance of the different connectionst models of melody with that of the two best equivalent n-gram models of bounded and unbouded order in (Pearce and Wiggins, 2004). The vertical error bars represent standard deviation of the cross entropies.

smooth function between the inputs and the outputs, and thus use an approximation that is also learned from data to possibly make better informed predictions when encountering unseen inputs. Paired t tests were carried out between pairs of the best of each class of connectionist model listed in Table 4.2 i.e., DRBM v RBM, FNN v DRBM and FNN v RBM, and it was found that only the difference in performance between the RBM and the FNN was significant [$t(79) = 6.56, p < 0.001$]. The error bars in Figures 4.5, 4.6 and 4.7 represent the average of the standard deviation of the prediction cross entropies over the different datasets in the IDyOM corpus. One will note the large overlap between these error bars. However, this standard deviation is not directly related to the statistical significance of the difference between the performance of the models as evaluated by the paired t test.

Figure 4.7 illustrates the improvement in predictive performance with context length of the bounded order n-gram models and non-recurrent connectionist models. The performance of the n-gram models peaks at a context length of 2 and worsens thereafter. On the other hand, each of the connectionist models performs progressively better with increasing context length, typically upto 5 events or more. A similar observation was made in (Bengio et al., 2003) in language modelling , where neural network language models also showed progressive improvement in performance up to longer context lengths than n-gram models. It is supposed in

(Pearce and Wiggins, 2004) that the U-shape cross entropy curve of the n-gram model is due to the fact that, while increasing the order bound provides the model with more specific contextual information for making predictions, the higher order contexts are also more likely to fail to produce a prediction. Therefore, the model will escape down to lower order models more frequently, thereby wasting more of the probability mass available on assigning escape probabilities.

Among the connectionist models, the performance of the DRBM and the FNN worsens at context lengths greater than 5. This can be explained as follows. A machine learning model addresses the task of approximating the true function which explains observed data. The process of learning its parameters given the data may be viewed as a search in what is known as the *Hypothesis Space* (Mitchell, 1982). This space contains the set of all possible solutions to the approximation task where each solution, also referred to as a hypothesis, is a different combination of the values of the model's parameters. One of the key factors that influences the result of this search is the dimensionality of the hypothesis space, which is nothing but the number and nature (discrete or continuous) of the parameters. The effective size of the hypothesis space increases exponentially with its dimensionality. It was demonstrated in (Hughes, 1968) using PAC-learners (a type of machine learning model), and now a generally agreed upon fact that, that the accuracy of the final hypothesis arrived at is substantially reduced as the dimensionality of the hypothesis space increases with the number of data samples being the same.

In the present case, the increase in context length automatically results in an increase in the number of model parameters (the size of the weight-matrix between the input units and the hidden units) of the FNN, the RBM and the DRBM. This leads to an increase in the dimensionality of the hypothesis space, while at the same time the number of data samples in the training set remains the same making the models susceptible to the above mentioned *Hughes phenomenon*. One way to counter this effect is to reduce the size of the hidden layer which in turn reduces the number of parameters of the model. On the other hand, the Hughes phenomenon can only be mitigated to the extent to which this reduction in size of the hidden layer does not adversely affect the performance of the model since the hidden layer comprises the features used by it in order to make predictions. This explanation is also supported by the observation that with increasing context length, smaller hidden layer sizes are preferred in the grid search upto a certain point beyond which a further reduction does not afford any benefit.

Alternatively, the worsening or relatively marginal improvment in the performance of these non-recurrent models at longer contexts can also be attributed to the nature of the corpus. It was noted in work by Pearce and Wiggins (2004), where the same corpus was used and similar trend observed, that this trend could have been due to the short length of the melodies in the dataset which limited the models from exploiting information in contexts greater than 6 events. This is exactly the context length at which the DRBM and FNN begin to display signs of worsening in performance while that of the RBM continues to improve but at a slower rate than when the context length is shorter.

Finally, one will also notice that while at lower context lengths the discriminative RBM has slightly better predictive performance than the generative one, the case is the opposite at higher context lengths. While the Hughes phenomenon does indeed explain the worsening of performance of the DRBM at longer contexts, it does not comment about the relative performance of the generative-discriminative RBM pair. In order to explain the latter we consider the following perspective. As context length increases linearly, so does the dimensionality of the input vector. And given a constant hidden layer size, so do the number of model parameters to be learned in both models. On the other hand, the amount of data available to learn these parameters remains constant despite the increase in context length. This can be viewed as an effective decrease in the amount of data available to train a model as context length increases i.e., that there is progressively less data available than the *ideal* amount required to estimate the parameters of a model as context length increases. Now, in a study conducted by Ng and Jordan (2001) it was found that given a generative-discriminative model pair, as the number of training examples increases there can be two distinct regimes of performance. In the first regime, the generative model has already approached its asymptotic error and is thus doing better. And in the second which follows, the discriminative model approaches its lower asymptotic error and does better. This explanation seems plausible here. In other words, the progressive increase in the input dimensionality of the model with context length (but not in the amount of data) may not be permitting the discriminative model to reach its lower asymptotic error at longer context lengths, as a result of which its performance is sub-optimal. This is not a concern at smaller context lengths where the DRBM performs better than its generative counterpart, the RBM. The explanation is also supported by the fact that there is a preference in the former for a smaller hidden layer in the grid search as context length increases while this is not so in the latter.

4.5 Summary

This chapter introduced the application of two classes of non-recurrent Connectionist melody models to monophonic music modelling. The models learn to generate a probability distribution over the possible musical pitch of the next note given the pitches of a fixed number of immediately preceding notes (in sequence). The predictive performance of the models was assessed with the help of their respective prediction cross-entropies over an unseen test dataset. This performance was compared against those of *n*-gram models from previous work (Pearce and Wiggins, 2004). The results indicate that the best case of each connectionist model outperforms the *n*-gram models of both bounded and unbounded order. The former were also found to be able to use information available in longer sequences to make predictions than the latter. While during the evaluation issues pertaining to insufficient data and possibly sub-optimal model parameters were encountered, as discussed above, results demonstrate the efficacy of non-recurrent connectionist models for modelling sequences of musical pitch. The next chapter presents further improvements using their recurrent counterparts.

Chapter 5

Recurrent Melody Models

This chapter continues the discussion on connectionist melody models intitiated in the previous one with the introduction of recurrent extensions to the models covered there. These recurrent models rely on an implicit representation of time, wherein a real-valued vector summarising information pertaining to previous time-steps is propagated forward through time. In the case of the models considered here, this vector contains the activations of the model's hidden layer, as illustrated in Figure 5.1. The passage of information is achieved through what are known as recurrent connections which serve to implement the notion of memory in these models. This recurrence affords the key representational advantage that the model's memory is no longer limited to the events explicitly given as input to it. This is in contrast to the non-recurrent models described in Chapter 4. Owing to this advantage, recurrent connectionist models have been a popular choice for modelling sequential data. We consider three such models, all of which depend on both the musical event at the current time-step and the state of their respective hidden layer from the previous time-step. The first of these is the Recurrent Neural Network (Elman, 1990), which can be considered the recurrent extension of the Feed-forward Neural Network from the previous chapter. Likewise, the second is the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) (Sutskever et al., 2009) - the recurrent extension of the generatively learned RBM. Furthermore, we propose a new model that we refer to as the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM). It is obtained by carrying out a novel discriminative learning (and inference) procedure on the RTRBM. The parameters of these models can be learned using entire sequences (not just windowed subsequences), and it was found that they outperform their non-recurrent counterparts, and thus the n-grams, on the folk and chorale melody corpus introduced in Section 3.2 with the RTDRBM outperforming the rest.

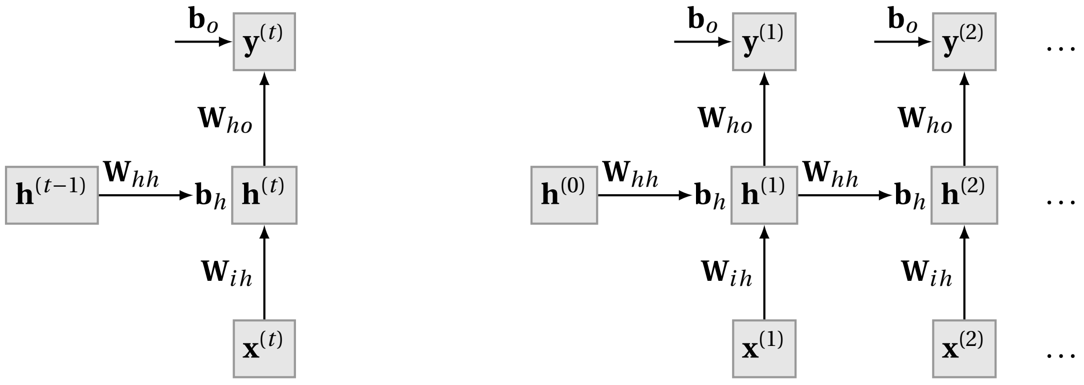


Fig. 5.1 An illustration of the passing of state information from previous time-steps to the future in a simple recurrent neural network. The figure on the right shows how the recurrent connection in the hidden layer between $\mathbf{h}^{(t-1)}$ and $\mathbf{h}^{(t)}$ in the compact representation of the model on the left can be unfolded in time to better illustrate the passage of temporal state information in it.

5.1 Recurrent Neural Network

The recurrent neural network (RNN) can be seen as an extension of the feedforward neural network (FNN) to model sequences more efficiently. The idea is to introduce the notion of memory in the network which is propagated forward through time. The architecture of a simple RNN is illustrated in Figure 5.2. The key difference to note in this figure is the presence of an additional set of input units denoted by $h^{(t-1)}$ which corresponds to the state of the hidden layer of the network in the previous (most recent) time-step (Elman, 1990). This additional input serves as the network's memory which, in theory, represents longer-term history than just the n ($n \in \mathbb{Z}^+$) immediately preceding events. The remainder of the input layer corresponds to the event of the most recent time-step in the sequence. While several variants of the RNN exist for supervised learning depending on the modelling task in question (Graves, 2012), here we describe the specific case where the true output at time-step t is re-introduced as input at time-step $(t+1)$. Given an input sequence $\mathbf{x}^{(1:T)}$ with a corresponding output label sequence $y^{(1:T)}$, at each time-step t, the RNN learns a mapping between the input $\mathbf{x}^{(t)}$ and its corresponding output label $y^{(t)}$ while additionally taking into account its memory of the past represented by the state of its hidden layer $\mathbf{h}^{(t-1)}$ in the previous time-step. This is done in a manner similar to the FNN by first mapping the inputs to a hidden layer

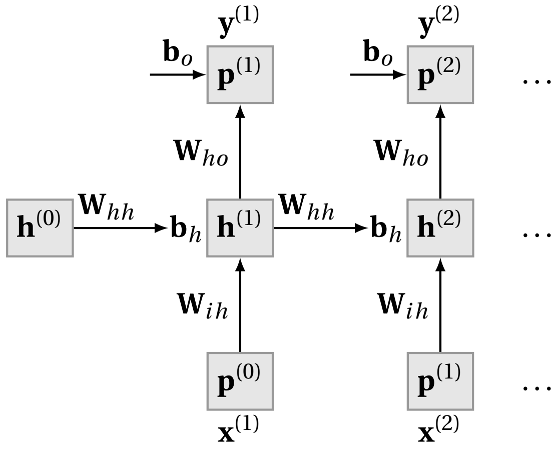


Fig. 5.2 Architecture of the RNN unfolded in time. At a given time-step t, the state of the hidden layer from the previous time-step $\mathbf{h}^{(t-1)}$ is responsible for propagating the information pertaining to the model's state in previous time-steps forward in time.

$h^{(t)}$ as

$$\begin{aligned}\mathbf{u}_h^{(t)} &= W_{ih}^{\top}\mathbf{x}^{(t)} + W_{hh}^{\top}\mathbf{h}^{(t-1)} + \mathbf{b}_h \\ \mathbf{h}^{(t)} &= f_h\left(\mathbf{u}_h^{(t)}\right)\end{aligned} \tag{5.1}$$

with the weight matrices $W_{ih} \in \mathbb{R}^{n_i \times n_h}$, $W_{hh} \in \mathbb{R}^{n_h \times n_h}$, and hidden bias vector $\mathbf{b}_h \in \mathbb{R}^{n_h}$. The hidden layer is, in turn, mapped to the output layer $\mathbf{y}^{(t)}$ as given by

$$\begin{aligned}\mathbf{u}_o^{(t)} &= \mathbf{b}_o + W_{ho}^{\top}\mathbf{h}^{(t)} \\ \mathbf{y}^{(t)} &= f_o\left(\mathbf{u}_o^{(t)}\right) .\end{aligned}$$

with the weight matrix $W_{ho} \in \mathbb{R}^{n_h \times n_o}$, and the output bias vector $\mathbf{b}_o \in \mathbb{R}^{n_o}$. In order to predict the first event $y^{(1)}$ in the sequence given the input $\mathbf{x}^{(1)}$, the initial state of the hidden layer activation vector $\mathbf{h}^{(0)}$ are learned from data and count as parameters of the model. Thus, the RNN has as its parameters the input-to-hidden weights W_{ih}, the hidden-to-hidden weights W_{hh}, the hidden-to-output weights W_{ho} and the hidden and output layer biases $\mathbf{b}_h$, $\mathbf{b}_o$ respectively and the initial state of the hidden layer $\mathbf{h}^0$.

As before in the case of the FNN, we experiment with the three different types of hidden layer non-linearities listed in Table 4.1, i.e. logistic sigmoid, hyperbolic tangent and rectified linear. Furthermore, the optimization criterion for learning the model parameters is once again the negative log-likelihood function given by

(4.2). The network thus has as many output units as the number of predictable musical events, and its output is a probability distribution $P\left(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}\right)$. The choice of the function f_o applied at the output layer remains unchanged from that used in the case of the FNN of Chapter 4 as the prediction task is the same here as well, i.e. a C-class classification task where each class corresponds to a value of musical pitch. The network in this case also has C output units where the value of each unit $y_c^{(t)}$ is obtained from the corresponding activation of that unit $u_{oc}^{(t)}$ by the application of the *softmax* function σ_{max}, given by (4.1) which allows one to interpret the outputs of the network as class-conditional probabilities $P\left(y_c^{(t)}|\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}\right)$ (Bridle, 1990).

5.1.1 Learning

The model parameters can be learned using a temporal variant of the backpropagation algorithm, known as Backpropagation Through Time (Werbos, 1990). The BPTT algorithm is essentially an extension of the original backpropagation algorithm (Rumelhart et al., 1988) applied to an RNN, under the assumption that the RNN is equivalent to a deep FNN with as many hidden layers as its intended temporal memory. This analogy becomes clear if one unfolds the RNN in time as shown in Figure 5.1. Just as the error E^p can be computed for each training example pair $(\mathbf{x}^p, \mathbf{y}^p)$ in the case of the Feed-forward Neural Network, here the error E^t can be computed at each time-step t for the $(\mathbf{x}^t, \mathbf{y}^t)$. The gradient of the error w.r.t. a given model parameter (weights) in the case of the RNN is computed differently according to whether or not the parameter is included in the recursive definition of the model. In the case of weights between the hidden and the output layers W_{ho}, these are computed as before in the case of the FNN as

$$\frac{\partial E^t}{\partial W_{ho}} = \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{u}_o^{(t)}} \frac{\partial \mathbf{u}_o^{(t)}}{\partial W_{ho}} .$$

However, this is not the case with W_{ih} and W_{hh} which are defined recursively as given in (5.1). The derivatives of the error w.r.t. these, which are written as

$$\begin{aligned}\frac{\partial E^t}{\partial W_{ih}} &= \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial W_{ih}} \\ \frac{\partial E^t}{\partial W_{hh}} &= \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial W_{hh}}\end{aligned}$$

contain the term $\mathbf{h}^{(t)}$ which depends on $\mathbf{h}^{(t-1)}$ and so on. For these weight matrices, the gradients are thus given by

$$\frac{\partial E^t}{\partial W_{ih}} = \sum_{k=0}^{t} \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \frac{\partial \mathbf{h}^{(k)}}{\partial W_{ih}}$$
$$\frac{\partial E^t}{\partial W_{hh}} = \sum_{k=0}^{t} \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \frac{\partial \mathbf{h}^{(k)}}{\partial W_{hh}} .$$

The same applies to the hidden layer biases, whose derivatives are given by

$$\frac{\partial E^t}{\partial W_{hh}} = \sum_{k=0}^{t} \frac{\partial E^t}{\partial \mathbf{y}^{(t)}} \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{b}_h^{(t)}} .$$

The exact expressions for the partial derivative depend on the non-linearity in use in the hidden layer of the model. Here as well, the cost function is negative log-likelihood just as with the FNNs, and is given by (4.2). Using this cost function and the above expressions of the gradients, the optimal parameters of the model corresponding to a given dataset can be obtained using the iterative gradient-based optimisation technique described in Section 3.3. In the present work, each sequence is treated as a single mini-batch while computing gradients of the error with respect to the model parameters.

5.2 Recurrent Temporal Restricted Boltzmann Machine

The Recurrent Temporal Restricted Boltzmann Machine (RTRBM) is a generative model proposed for modelling high-dimensional temporal data (Sutskever et al., 2009) which contains a sequence of RBMs, such that the RBM at time-step t is conditioned on that at $(t-1)$ through a set of time-dependent model parameters. In the case of the RTRBM, these parameters are the biases to the visible and hidden layers. The model is based on the idea originally introduced in conditional RBMs (CRBMs) (Taylor et al., 2007), where the biases of the RBM at time t are conditioned (through a linear function) on the values of the visible layers $v^{(\tau)}$ of RBMs at time $\tau < t$. As the CRBMs are limited by the Markov assumption, and cannot account for long-term dependencies, the temporal RBM (Sutskever and Hinton, 2007) (TRBM) was proposed in which the same time-dependent biases are now conditioned on the hidden states $h^{(\tau)}$ of the RBMs in previous time-steps $\tau < t$. Finally, in order to simplify the heuristic learning procedure involved with the TRBM, the RTRBM was proposed where these biases are conditioned on the hidden states $h^{(\tau)}$ of the

RBMs in previous time-steps $\tau < t$.

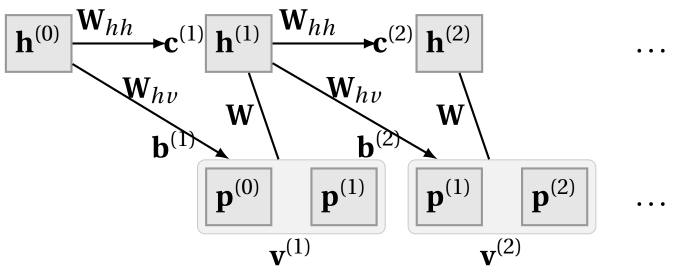


Fig. 5.3 Architecture of the RTRBM unfolded in time. At any given time-step t, the mean-field state of the hidden layer from the previous time-step $\hat{\mathbf{h}}^{(t-1)}$ contributes to the dynamic biases $\mathbf{b}^{(t)}$ and $\mathbf{c}^{(t)}$ of the visible and hidden layers respectively, and is responsible for propagating information pertaining to the model's state in previous time-steps forward in time. In the melody modelling application, it models the joint probability of events $p^{(t-1)}$ and $p^{(t)}$ at any time-step t.

The RTRBM models the joint probability distribution of its visible and hidden layers $\mathbf{v}^{(t)}$ and $\mathbf{h}^{(t)}$ respectively at any time-step t. This joint probability takes the form

$$P\left(\mathbf{v}^{(t)}, \mathbf{h}^{(t)} | \hat{\mathbf{h}}^{(t-1)}\right) = \frac{\exp\left(-E\left(\mathbf{v}^{(t)}, \mathbf{h}^{(t)}; \hat{\mathbf{h}}^{(t-1)}\right)\right)}{Z_{\hat{\mathbf{h}}^{(t-1)}}}$$

where $E\left(\mathbf{v}^{(t)}, \mathbf{h}^{(t)}; \hat{\mathbf{h}}^{(t-1)}\right) = -\left(\mathbf{h}^{(t)\top} W \mathbf{v}^{(t)} + \mathbf{c}^{(t)\top}\mathbf{h}^{(t)} + \mathbf{b}^{(t)\top}\mathbf{h}^{(t)}\right)$ and $Z_{\hat{\mathbf{h}}^{(t-1)}}$ is a normalisation factor with the same interpretation as in the case of the RBM in Section 4.3. Here we work with a variant of the RTRBM in which both the visible and hidden layer biases $\mathbf{b}^{(t)}$ and $\mathbf{c}^{(t)}$ (the superscripts indicate time) depend on the mean-field values of the hidden units $\hat{\mathbf{h}}^{(t-1)}$ at the previous time-step as follows (Boulanger-Lewandowski et al., 2012):

$$\begin{aligned} \mathbf{b}^{(t)} &= W_{hv}\hat{\mathbf{h}}^{(t-1)} + \mathbf{b} \\ \mathbf{c}^{(t)} &= W_{hh}\hat{\mathbf{h}}^{(t-1)} + \mathbf{c} \end{aligned}$$

where $\hat{\mathbf{h}}^{(t)}$ is given by

$$\begin{aligned} \hat{\mathbf{h}}^{(t)} &= \sigma\left(W\mathbf{v}^{(t)} + \mathbf{c}^{(t)}\right) \\ &= \sigma\left(W\mathbf{v}^{(t)} + W_{hh}\hat{\mathbf{h}}^{(t-1)} + \mathbf{c}\right) \end{aligned} \tag{5.2}$$

It is to be noted that (5.2) is the defining equation of an RNN with hidden units $\hat{\mathbf{h}}^{(t)}$ and a logistic sigmoid non-linearity. This RTRBM is characterised by six parameters in all: W, W_{hh}, W_{hv}, $\hat{\mathbf{h}}^{(0)}$, $\mathbf{b}$ and $\mathbf{c}$, where W represents the undirected weights between the visible and the hidden layers of the constituent RBM, W_{hh} and W_{hv}

the directed weights between the hidden layer at time $(t-1)$ and the hidden and visible layers at time t respectively, and $\hat{\mathbf{h}}^{(0)}$ is a vector of initial mean-field values of the hidden units. Here as well **b** and **c** are respectively the time-invariant visible and hidden layer biases of the RBM. The joint distribution of a sequence according to this model is formalised as

$$\begin{aligned} P\left(\mathbf{v}^{(1:T)}, \mathbf{h}^{(1:T)} | \hat{\mathbf{h}}^{(0:T-1)}\right) &= \prod_{t=1}^{T} P(\mathbf{v}^{(t)}, \mathbf{h}^{(t)} | \hat{\mathbf{h}}^{(t-1)}) \\ &= \frac{\exp\left(\sum_{t=1}^{T} -E(\mathbf{v}^{(t)}, \mathbf{h}^{(t)}; \hat{\mathbf{h}}^{(t-1)})\right)}{\prod_{t=1}^{T} Z_{\hat{\mathbf{h}}^{(t-1)}}} \end{aligned}$$

5.2.1 Learning

In order to learn the parameters of the RTRBM, one must maximise the log-likelihood function $\log P\left(\mathbf{v}^{(1:T)}, \mathbf{h}^{(1:T)}\right)$. This involves computing the partial derivatives of this function with respect to the various model parameters. Given $\hat{\mathbf{h}}^{(1)}, \ldots, \hat{\mathbf{h}}^{(T-1)}$ all the RBMs at different time-steps are de-coupled, thus allowing one to carry out Gibbs sampling for each RBM independently. This property can be exploited to compute the approximate gradients of the parameters as given in the Contrastive Divergence (CD) algorithm as if the RBM at a certain time-step t was independent of those at other time-steps. The RTRBM can be thought of as a sequence of conditional RBMs whose parameters are the output of a deterministic RNN. So once the gradients are computable at each time-step, this can be extended to sequences with the help of the Backpropagation Through Time algorithm as described in (Sutskever et al., 2009; Mittelman et al., 2014). This is given by:

$$\begin{aligned} \frac{\partial \log P\left(\mathbf{v}^{(1:T)}, \mathbf{h}^{(1:T)}\right)}{\partial \theta} &= \sum_{t=1}^{T} \frac{\partial \log P(\mathbf{v}^{(t)}, \mathbf{h}^{(t)} | \hat{\mathbf{h}}^{(t-1)})}{\partial \theta} \\ &= \frac{\partial \log P(\mathbf{v}^{(1)}, \mathbf{h}^{(1)} | \hat{\mathbf{h}}^{(0)})}{\partial \hat{\mathbf{h}}^{(0)}} + \sum_{t=1}^{T-1} \frac{\partial \hat{\mathbf{h}}^{(t)}}{\partial \theta} \frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t)}} \end{aligned}$$

where θ is the general notation for the parameters, and $\frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t)}}$ is defined recursively as

$$\frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t)}} = W_{hh}^{\top} \hat{\mathbf{h}}^{(t+1)} (1 - \hat{\mathbf{h}}^{(t+1)}) \frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t+1)}} + \frac{\partial \log P(\mathbf{v}^{(t+1)}, \mathbf{h}^{(t+1)} | \hat{\mathbf{h}}^{(t)})}{\partial \hat{\mathbf{h}}^{(t)}}$$

5.2.2 Discriminative Inference in the RTRBM

In order to address the prediction task which involves computing a conditional distribution, discriminative inference can be carried out in the RTRBM as explained in Section 4.3.1. This will result in the following expression for the posterior probabilities at time t:

$$P(\mathbf{y}^{(t)}|\mathbf{x}^{(1:t)}) = P(\mathbf{y}^{(t)}|\mathbf{x}^{(t)},\hat{\mathbf{h}}^{(t-1)}) \\ = \frac{\exp(-E_{free}(\mathbf{x}^{(t)},\mathbf{y}^{(t)}))}{\sum_{\mathbf{y}^*}\exp(-E_{free}(\mathbf{x}^{(t)},\mathbf{y}^*))} . \tag{5.3}$$

This equation is reminiscent of (4.3), with the distinction that it takes into account temporal information carried forward from the previous time-step through $\hat{\mathbf{h}}^{(t-1)}$. This inference can be carried out on the generatively learned RTRBM. However, in order to accurately model the distribution in the visible layer of the model, the sampling step of the Contrastive Divergence algorithm would assume two sets of multinomial units, one corresponding to the one-hot representation of the musical pitch at time-step $(t-1)$, and the other to that at time-step t.

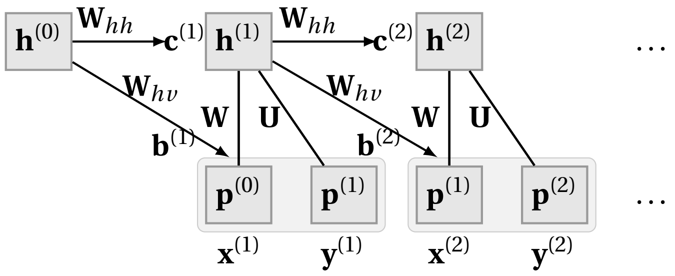


Fig. 5.4 Architecture of the RTDRBM unfolded in time, in which discriminative learning and inference is carried out for classification under the assumption that one set of visible units $\mathbf{y}^{(t)}$ represents a multinomial distribution in the number of classes, and the rest are the inputs $\mathbf{x}^{(t)}$. It is equivalent to the RTRBM used for the same task when learned generatively. At any given time-step t, the mean-field state of the hidden layer from the previous time-step $\hat{\mathbf{h}}^{(t-1)}$ contributes to the dynamic biases $\mathbf{b}^{(t)}$ and $\mathbf{c}^{(t)}$ of the visible and hidden layers respectively, and is responsible for propagating information pertaining to the model's state in previous time-steps forward in time. In the melody modelling application, it models the joint probability of events $p^{(t-1)}$ and $p^{(t)}$ at any time-step t.

5.3 Recurrent Temporal Discriminative RBM

The Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM) that we propose here is depicted in Figure 5.4. One will notice that it is identical in structure to the RTRBM described in Section 5.2, where the visible and hidden layers of the RBM at time t are conditioned on the mean-field values of the hidden layer at time $(t-1)$ (Boulanger-Lewandowski et al., 2012). The difference between the two is in the cost function that is optimized during learning in each case. The RTDRBM uses a discriminative cost function, while the RTRBM a generative one. The main motivation for proposing the RTDRBM here is to directly learn the distribution $P\left(\mathbf{y}^{(1:T)}|\mathbf{x}^{(1:T)}\right)$ over a sequence of input-label pairs $\{\mathbf{x}^{(1:T)}, \mathbf{y}^{(1:T)}\}$, rather than the joint probability $P\left(\mathbf{y}^{(1:T)}, \mathbf{x}^{(1:T)}, \mathbf{h}^{(1:T)}\right)$ like the RTRBM.

The first step in this direction is to carry out discriminative inference in the RTRBM, in a manner similar to (5.3). It can be extended to an entire sequence of T events as follows (mathematical proof in (5.4), (5.5), and (5.6) provided by Son N. Tran):

$$\begin{aligned}
P\left(\mathbf{y}^{(1:T)}|\mathbf{x}^{(1:T)}\right) &= \frac{P\left(\mathbf{y}^{(1:T)}, \mathbf{x}^{(1:T)}\right)}{\sum_{\mathbf{y}^{(1:T)}} P\left(\mathbf{y}^{(1:T)}, \mathbf{x}^{(1:T)}\right)} \\
&= \frac{\prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|\mathbf{y}^{(1:t-1)}, \mathbf{x}^{(1:t-1)}\right)}{\sum_{\mathbf{y}^{(1:T)}} \prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|y^{(1:t-1)}, \mathbf{x}^{(1:t-1)}\right)} \\
&= \frac{\prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|\hat{\mathbf{h}}^{(t-1)}\right)}{\sum_{\mathbf{y}^{(1:T)}} \prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|\hat{\mathbf{h}}^{(t-1)}\right)} \\
&= \frac{\prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|\hat{\mathbf{h}}^{(t-1)}\right)}{\prod_{t=1}^{T} \sum_{\mathbf{y}^{(t)}} P\left(\mathbf{y}^{(t)}, \mathbf{x}^{(t)}|\hat{\mathbf{h}}^{(t-1)}\right)} \\
&= \prod_{t=1}^{T} P\left(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, \hat{\mathbf{h}}^{(t-1)}\right)
\end{aligned} \tag{5.4}$$

One will notice that, just as in the case of the RTRBM, given $\hat{\mathbf{h}}^{(1)}, \ldots, \hat{\mathbf{h}}^{(T-1)}$, this leads to a de-coupling of each time-step in this case as well, except that here the model at each time-step is a DRBM and thus the name *Recurrent Temporal DRBM*.

One might question here the need for a discriminative variant of the RTRBM, as the inference in (5.3) can be carried out just as easily in the RTRBM. While this is indeed true, our reason for experimenting with the RTDRBM is justified by a previous study on generative and discriminative learning (Ng and Jordan, 2001). A result from that study which is particularly relevant here is that given sufficient training examples, a discriminative model tends to do better on the task it is optimized for than its generative counterpart. Moreover, it was also found in (Larochelle and Bengio, 2008) that RBMs learned discriminatively were both efficient in their number of parameters and better at a classification task than those learned generatively. Thus, we considered it of value to explore the idea of discriminative learning in the RTRBM (which gives the RTDRBM). As we shall see later in Section 5.4, there are indications of this type of behaviour here as well.

5.3.1 Learning

The parameters of the model can be learned by maximizing its log-likelihood function which is given by

$$\begin{aligned}\mathscr{L} &= \log P\left(\mathbf{y}^{(1:T)}|\mathbf{x}^{(1:T)}\right)\\ &= \sum_{t=1}^{T} \log P\left(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, \hat{\mathbf{h}}^{(t-1)}\right) \ .\end{aligned} \tag{5.5}$$

Similar to the RTRBM, the learning algorithm here involves the back-propagation through time (Werbos, 1990) update of the model parameters (assuming a general notation θ for the parameters) given by

$$\begin{aligned}\nabla\theta = \frac{\partial \mathscr{L}}{\partial \theta} &= \sum_{t=1}^{T} \frac{\partial \log P(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, \hat{\mathbf{h}}^{(t-1)})}{\partial \theta}\\ &= \sum_{t=1}^{T-1} \frac{\partial \hat{\mathbf{h}}^{(t)}}{\partial \theta} \frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t)}} + \frac{\partial \log P(\mathbf{y}^{(1)}|\mathbf{x}^{(1)}, \hat{\mathbf{h}}^{(0)})}{\partial \hat{\mathbf{h}}^{(0)}}\end{aligned} \tag{5.6}$$

where

$$\begin{aligned}\frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t)}} &= W_{hh}^{\top} \hat{\mathbf{h}}^{(t+1)} (1 - \hat{\mathbf{h}}^{(t+1)}) \frac{\partial \mathscr{A}}{\hat{\mathbf{h}}^{(t+1)}}\\ &+ \frac{\partial \log P(\mathbf{y}^{(t+1)}|\mathbf{x}^{(t+1)}, \hat{\mathbf{h}}^{(t)})}{\partial \hat{\mathbf{h}}^{(t)}}\end{aligned}$$

5.4 Experiments

An evaluation of the recurrent models described above was carried out on the corpus of chorale and folk melody datasets introduced in Section 3.2. This facilitated comparison with melody models based on n-grams from previous work (Pearce and Wiggins, 2004) and the non-recurrent melody models of Chapter 4. The evaluation was carried out on 10 resampling sets of each of the 8 datasets in the corpus, thus making it possible to also test the significance (Dietterich, 1998) of the differences in performance between the models where required. Training and test folds identical to those in (Pearce and Wiggins, 2004) were used here as well. We extracted a small part of the training set (5% of the total number of samples) in each case as the validation set.

A grid search was carried out to determine the best set of hyperparameters for the corpus. Each model was learned up to a maximum of 250 iterations, and that with the best validation set error was evaluated on the test set. A grid search was carried out to determine the best set of hyperparameters for each model. While carrying out the iterative gradient-based learning on a model at each grid-point, its performance on the validation set was checked after every 10 iterations. The model corresponding to the iteration with the best validation set performance was chosen to be evaluated on the test set, and this performance reported. Based on a preliminary search on the datasets, the initial learning rate η_{init} was set to 0.05. This value decayed with the number of iterations according to the schedule given by $\eta_t = \eta_{init}/(1 + t/\tau)$, where $\tau = 50$ is the iterations after which each step of decay occurs. The number of hidden units in the models n_{hid} was varied as $\{25, 50, 100, 200\}$. Both L_1 and L_2 decay were set to identical values $\lambda_1 = \lambda_2 = \lambda$ which was either on ($\lambda = 0.0001$) or off ($\lambda = 0$). This procedure was followed with all the three recurrent connectionist models employed here. The parameters of each of the models were learned over entire sequences, rather than on shorter truncated sequences which is also common practice. This is done by treating each melody as a single batch over which the parameters are updated within each iteration of training. Thus the notion of a fixed context length does not apply to these recurrent models as in the case of the non-recurrent ones. Additionally, in the case of the RNNs, the three types of hidden layer activations listed in Table 4.1 were also examined.

Comparison with Previous Work

As was the case with the non-recurrent prediction models of Chapter 4, the recurrent ones here also deal only with long-term effects that are induced from a corpus, and are thus compared with the two best performing n-gram LTMs in (Pearce and Wiggins, 2004). The first of the said n-gram models is of unbounded order, and is referred to as C^*I (where the * refers to unbounded order), and uses the interpolated smoothing method proposed in (Moffat et al., 1998) to account for unfamiliar contexts. The second is of bounded order, and uses the same interpolated smoothing method, but with a bounded order of 2 (and thus named $C2I$). These have been explained earlier in Section 4.4. We refer the interested reader to (Pearce and Wiggins, 2004) for further details on these two models.

Boundary Conditions

The recurrent models described in this chapter rely on a single musical event immediately preceding the one to be predicted, and so only the first event in a melody $s^{(1)}$ does not have a valid context. To handle this case, we rely on the same as-

Model	Context	Cross Entropy
n-gram (b)	2	2.957
n-gram (u)	N/A	2.878
FNN ($n_h = 200$, $h_{act} = ReLU$)	7	2.819($\pm$0.200)
DRBM ($n_h = 25$, $h_{act} = LogSigU$)	5	2.819($\pm$0.208)
RBM ($n_h = 100$, $h_{act} = LogSigU$)	8	2.799($\pm$0.168)
RNN ($n_h = 50$, $h_{act} = TanHU$)	N/A	2.787($\pm$0.174)
RTRBM ($n_h = 100$, $h_{act} = LogSigU$)	N/A	2.738($\pm$0.182)
RTDRBM ($n_h = 100$, $h_{act} = LogSigU$)	N/A	2.712($\pm$0.168)

Table 5.1 A comparison between the best predictive cross entropy scores of each type of recurrent and non-recurrent model examined in the present work, and n-gram models from previous work.

sumption as the non-recurrent models described in Section 4.4 wherein each non-existent context event is represented by a vector of dimensionality $|[\tau_{pitch}]|$ equal to that of any other input vector (which is the number of pitches occurring in the dataset), but with each element of value $1/|[\tau_{pitch}]|$. As explained there, this signifies a lack of any knowledge whatsoever about the nature of these missing events. The reader is referred to Section 4.4 for an example illustrating this assumption and further details on it. 4.4.

5.4.1 Results

Table 5.1 contains the best predictive performance of each of the connectionist models (both non-recurrent and recurrent) considered here, together with the equivalent n-grams from (Pearce and Wiggins, 2004). The results are averaged across all 8 datasets. One will notice the progressive improvement in the best-case performance from the n-gram models, to the non-recurrent and recurrent connectionist models, with the RTDRBM outperforming the rest. The main influencing factor with regard to the predictive performance of the different classes of models was the hidden layer size. It was found both in the case of the RTRBM and the RTDRBM that a hidden layer size of 100 units resulted in the best predictive performance. In the case of the RNNs, there was an additional factor which influenced the performance of the models, namely the activation type of the hidden units. It was observed that both hyperbolic tangent (TanH) and rectified linear (ReL) hidden layer activations resulted in the best performance of 2.787, however, the model corresponding to the former had a hidden layer size of 50 as opposed to the latter whose hidden layer contained 100 units. This result is reminiscent of the case of the FNNs in Chapter 4 where the best case of the TanH hidden layer FNN contained fewer units than that of the networks with other hidden layer activations. Detailed (dataset-wise) results of the best cases of the three recurrent models examined here are available in Tables A.6, A.7 and A.8.

5.4.2 Discussion

When it comes to the recurrent models (the RNN and RTRBM), the recurrent connections in these can be seen as propagating forward in time a "summary" of long-term contextual information (Elman, 1990), and thus one only needs to explicitly input the most recent time-step to them. They are thus free from the Markov assumption and can take into account this long-term information while at the same time limiting the number of model parameters between the input and hidden layers of the models. This is in contrast with the non-recurrent models whose input space increases in size with context length. Furthermore, the latter are also unaware of the position of their fixed context subsequences in a melody as these are essentially obtained by sliding time-windows over the melody.

As discussed in Chapter 4, while at lower context lengths the discriminative RBM has slightly better predictive performance than the generative one, it is the opposite case at higher context lengths. This was attributed to an increase in input feature dimensionality with context length in relation to the amount of data, which is equivalent to an effective reduction of the amount of available data. given a constant feature dimensionality. It also resulted in a preference in the discriminative case for a smaller number of hidden units from the grid search as context length, and correspondingly the model input dimensionality increased while this was not so in the generative case. This explanation is further supported in the results of this chapter, where we find in the case of both the RTRBM and the RTDRBM, in which the model input dimensionality is independent of the context length, that the latter which is discriminatively learned performs better than the former which is learned generatively. A paired t test between the two confirmed that the difference in performance was indeed significant [$t(79) = 5.54, p < 0.001$]. This was also the case for the difference in performance between the RNN and the RTRBM [$t(79) = 5.03, p < 0.001$].

Table 5.2 lists the performance of best of each class of models on each of the datasets. One will notice a consistent trend across different models in the best pos-

Dataset	FNN	RBM	DRBM	RNN	RTRBM	RTDRBM	Average
`jugoslav`	2.715	2.722	2.705	2.681	2.676	2.655	2.692
`elsass`	3.054	3.002	3.060	2.970	2.945	2.897	2.988
`schweiz`	3.035	3.024	3.017	2.988	2.961	2.932	2.993
`oesterrh`	3.365	3.310	3.418	3.309	3.296	3.259	3.326
`kinder`	2.389	2.377	2.398	2.424	2.313	2.301	2.367
`nova-scotia`	2.750	2.727	2.738	2.679	2.635	2.609	2.690
`bach`	2.420	2.441	2.423	2.451	2.378	2.362	2.413
`shanxi`	2.827	2.798	2.795	2.792	2.703	2.685	2.767
Average	2.819	2.799	2.819	2.787	2.738	2.712	

Table 5.2 Cross entropies of the best instance of each model class on the different datasets, and the average of these values in the bottom row. The rightmost column lists the average cross-entropy of all the models on each of the datasets.

sible cross entropies achievable by each of them on the datasets. Given a dataset, the prediction cross entropy can be used to compare the performance of different models. On the other hand, given a model, the best achievable cross entropy on a given dataset by it sheds light into the nature of the dataset as well. For instance, the greater the number of symbols ($|[\tau_{pitch}]|$) the greater the number of permutations of these that are possible in the sequences that make up the dataset. Those of the permutations that appear in the melodies are determined by style or musical tradition represented by the dataset. It was also mentioned above that the present work does not carry out any key-normalisation (transposition of all melodies to the same key) in order to compare the models evaluated here with those from previous work. Key variability in data creates ambiguity regarding the symbol that is expected to follow a given context as this could differ across keys. Thus, normalisation of melodies to a common key would be expected to increase predictability in the data and thus reduce the cross entropy on the corpus. In Chapter 6, it will be demonstrated how introducing key-signature information as input to a model can amount to a similar effect and result in improved predictive performance.

5.5 Summary

This chapter introduced recurrent extensions of the models in Chapter 4. Each of these models uses the musical pitch of only the most recent note together with a real-valued vector representing the model's state in the previous time-step which serves as a memory of its past (from the start of the melody) in order to predict a probability distribution over the value of the pitch of the next note. The predictive performance of the models was evaluated using cross entropy as the measure and compared against those of n-gram models from previous work (Pearce and Wiggins, 2004). The results indicate that the best case of each connectionist model outperforms that of the n-gram models of both bounded and unbounded context lengths, as well as the non-recurrent connectionist models for melody modelling described in Chapter 4. Of all the models considered here, the RTDRBM performs best. Furthermore, the observations regarding the comparative performance of the RTDRBM proposed here and its generative counterpart the RTRBM can be explained using the same theory underlying discriminative and generative learning (Ng and Jordan, 2001) which was used to explain the relative performance between the generative and the discriminative RBMs in Chapter 4.

Chapter 6

Ensembles of Melody Models

The prediction models described in the previous chapters are, in effect, probabilistic classifiers which assign a class-label (the value of the pitch of the note $s^{(t)}$ at time-step t) to a given input (the sequence of musical pitches leading up to $s^{(t)}$) by first generating a probability distribution over all class-labels (the possible values of musical pitch $[\tau]_{pitch}$ in a dataset). Given two or more such classifiers wherein each is learned differently, it is possible to combine their predictions in what is known as an *ensemble* of classifiers (Valentini and Masulli, 2002). The technique is commonly referred to as *ensembling*. It has been demonstrated that ensembling can often improve the performance of a classification system (Opitz and Maclin, 1999; Dietterich, 2000; Zhou et al., 2002) by effectively combining the different sources of information that the constituent classifiers represent. The advantage of ensembling in predictive models of musical melody was first demonstrated in (Conklin and Witten, 1995), and further studied in (Pearce, 2005). This was done using aggregation rules which weight the prediction of each model in the ensemble as an inverse of the Relative Entropy of its prediction. Models were combined at two different levels. In the first, the ensemble contained models learned on different input features. And in the second, models whose parameters were continuously updated during the prediction task (online models) were combined with those whose parameters weren't (offline models). In this chapter, we first introduce the idea of ensembling and its use in machine learning. This is followed by a description of the entropy-weighted combination rules of (Conklin and Witten, 1995; Pearce, 2005). We employ these rules to combine the predictions of multiple models, all predicting musical pitch while using different melodic features as their respective inputs. The results of the initial experiments presented here are positive and encourage further exploration of this approach.

6.1 Ensembles

An ensemble is a set of machine learning models whose decisions are combined to improve the performance of the overall system (Valentini and Masulli, 2002). The goal of ensembling is to effectively combine predictions of models which have independently learned from different sources of information. Thus, a key consideration in creating a successful ensemble is the diversification of the sources of information that are brought together in it, and there exist ways to measure this diversity (Kuncheva and Whitaker, 2003). A machine learning algorithm searches a hypothesis space $\mathscr{H}$ to find the best possible hypothesis of a function $\mathscr{F}$ underlying the data given to it. An appropriate class of models is chosen to approximate $\mathscr{F}$. However, it is sometimes the case that the available data might not lead to a good enough hypothesis with a given model. This can be due to various reasons such as the size of the chosen model's parameter set with respect to that of the dataset, limited representational capability of the model, or a sub-optimal parameter set solution resulting from the non-convexity of the error surface along which the model is optimised. It is under such conditions that ensembles of models which are by themselves not comprehensive tend to offer an improvement in performance. There exist different ways in which models can be combined into ensembles (Opitz and Maclin, 1999; Moreno-Seco et al., 2006).

Ensemble methods can be broadly classified into two types — *non-generative* and *generative* (Valentini and Masulli, 2002). A non-generative ensemble puts together a set of several pre-determined models trained with their own respective learning algorithms. This involves no modification to the constituent models, but only the optimization of the method used to combine them. It is the outputs of the individual models which are combined and in focus while optimizing the overall system, while the models themselves remain untouched. Ensembles of this type make use of the Dempster-Shafer combination rule, the Bayesian decision rule, aggregation rules such as Minimum, Maximum, Average and Product and Ordered Weight Averaging; in other words, rules which are learned on top of set of constituent models. On the other hand, a generative ensemble aims to improve the overall accuracy of the ensemble by directly boosting the accuracy and the diversity of the constituent models. It achieves this in one of many ways, either by (1) modifying the input data through resampling methods and feature selection, or (2) by manipulating the aggregation of the classes by output coding methods, or (3) by selecting subsets of the constituent models, or (4) by randomly modifying the learning algorithms of the constituent models.

6.2 The Mean and Product Combination Rules

In the present work, we employ two combination rules from the suite of non-generative ensemble methods. These are the *mean* and *product* rules respectively. These rules combine the probability distributions predicted by each of the models that are contained in the ensemble. As it is the predicted distributions which are combined, this approach is independent of the types of models involved. While there is no single optimal way to weight the different predictions, one intuitive way to go about it is to assign greater weights to those whose predictions reflect greater certainty. It was demonstrated in (Conklin and Witten, 1995; Pearce, 2005) when combining n-gram and variable order Markov models that an entropy-weighted combination of the predictions that captures this intuition typically resulted in a system with better predictive performance than any of the individual models. Here, we describe these two rules for creating ensembles. Let M be a set of models and $P_m(s)$ be the probability assigned to symbol $s \in S$ by model m, where S is the discrete alphabet over which the prediction $P_m(s)$ is being made.

The first approach involves taking a weighted arithmetic mean of their respective predictions. This is the *mean combination rule* (Tax et al., 2000), and is defined as

$$P(s) = \frac{\sum_{m \in M} w_m P_m(s)}{\sum_{m \in M} w_m}$$

where each of the weights w_m depends on the entropy of the distribution P_m predicted by the corresponding model m in the combination such that greater entropy (which implies greater uncertainty) is associated with a lower weight (Conklin, 1990). The weights are given by the expression $w_m = H_{rel}(P_m)^{-b}$, where the relative entropy $H_{rel}(P_m)$ is

$$H_{rel}(P_m) = \begin{cases} H(P_m)/H_{max}(P_m), & \text{if } H_{max}(P_m) > 0 \\ 1, & \text{otherwise .} \end{cases}$$

The bias b is a hyperparameter whose value is determined with the help of a validation set. The quantities H and H_{max} are respectively the Shannon entropy of the prediction and its maximum possible value given the discrete alphabet, and are defined as

$$H(P) = -\sum_{s \in S} P(s) \log_2 P(s)$$

$$H_{max}(S) = \log_2 |S|.$$

where $P(s \in S) = P(\chi = s)$ is the probability mass function of a random variable χ distributed over the discrete alphabet S such that the individual probabilities are independent and sum to 1.

The second combination method — the *product combination rule*, is computed by taking the weighted geometric mean of the probability distributions of the models contained in the ensemble. This is given by

$$P(s) = \frac{1}{R}\left(\prod_{m \in M} P_m(s)^{w_m}\right)^{\frac{1}{\sum_{m \in M} w_m}}$$

where R is a normalisation constant which ensures that the resulting distribution over S sums to unity. The weights w_m in this case are obtained in the same manner as in the mean rule. It was observed in a previous application of these two combination methods to melody modelling (Pearce, 2005), that the product rule resulted in a greater improvement in predictive performance.

6.3 Combining Models that Utilise Different Features

The connectionist models described in Chapters 4 and 5 dealt purely with musical pitch sequences. It is often the case that other musical features (referred to as *types* in the multiple viewpoints framework) available in notated music are informative and can help in improving the predictive performance of these models. In this section, we incorporate types other than `pitch` to demonstrate this improvement. This is motivated by previous work on the subject. In (Conklin and Witten, 1995) various combinations of types, corresponding to features such as musical pitch, interval, inter-onset interval, scale degree and metrical location were used to progressively improve the quality of predictions of the pitch of the next note. This was extended further in (Pearce, 2005), with the inclusion of additional types corresponding to note duration, first note in a bar and in a piece, etc. and the proposal of an iterative feature selection method which selected the optimal set of features in order to improve the quality of predictions in each iteration. While the experiments carried out here do not go into the same level of depth as (Conklin and Witten, 1995; Pearce, 2005), we carry out some initial experiments to demonstrate similar improvement on the addition of new features as inputs to the existing pitch-only connectionist models.

We first briefly review the theory on Multiple Viewpoints described in Section 3.1. A musical event s refers to the occurrence of a note in a melody. A *type* τ

refers to any of a set of musical features that can be used to describe an event. The domain of a *type*, denoted by $[\tau]$ is the set of possible values of that type. A *basic type* is a directly observable or given feature such as `pitch` (chromatic pitch), `dur` (note duration), `keysig` (key signature)or `timesig` (time signature). A *derived type* can be derived from any of the basic types or other derived types. Examples of derived types are `ioi` (inter-onset interval, derived from the basic type `onset`), `int` (interval, derived from `pitch`), etc. Two or more types can be "linked" by taking the Cartesian product of their respective domains, thus creating a *linked type*, and this type is also treated in the same way as any other basic or derived type. A *viewpoint* modelling a type τ is a partial function which maps sequences of events onto elements of type τ. A *multiple viewpoint system* (MVS) is a set of models, each of which is trained on subsequences of one *type*, whose individual predictions are combined in some way to influence the prediction of the next event in a sequence. This is a key proposal of the framework which allows one to incorporate useful information from sequences of any arbitrary type in the context (an *input* type) derived from a corresponding sequence of events to influence the predicted probability distribution over a certain type (a *target* type) corresponding to the next event in the sequence. This is realised through the introduction of multiple models, each of which relies on a different source of information (its respective input types) to make a prediction about the target type. The accuracy of the prediction depends on how informative the input type is of the target type. It is possible to combine the information provided by different input types for possibly better predictive performance on the target type.

In previous work implementing this idea using n-gram models (Conklin and Witten, 1995; Pearce, 2005) it was first required to make a prediction about the input type itself and then map this prediction to the domain of the target type. In other words, these approaches first model the distribution $P\left(\Psi_{\mathtt{in}}\left(s^{(t)}\right)|\Psi_{\mathtt{in}}\left(s^{(1:t-1)}\right)\right)$, and then rely on a deterministic mapping to derive $P\left(\Psi_{\mathtt{tgt}}\left(s^{(t)}\right)|\Psi_{\mathtt{in}}\left(s^{(1:t-1)}\right)\right)$. In contrast to this, the connectionist models described in Chapters 4 and 5 can be made to adopt a more direct approach, wherein the prediction task described in 3.5 may be viewed generally as one in which the input of a prediction model is the sequence of events of a certain input type $\Psi_{in}\left(s^{(1:t-1)}\right)$ and the model predicts a probability distribution $P\left(\Psi_{\mathtt{tgt}}\left(s^{(t)}\right)|\Psi_{\mathtt{in}}\left(s^{(1:t-1)}\right)\right)$ over a target type $\Psi_{\mathtt{tgt}}\left(s^{(t)}\right)$, given a sequence $s^{(1:T)}$ of musical events, such that $\Psi_{\mathtt{in}}\left(s^{(t)}\right) \in [\tau_{\mathtt{in}}]$ and $\Psi_{\mathtt{tgt}}\left(s^{(t)}\right) \in [\tau_{\mathtt{tgt}}]$. This differs from previous work employing n-gram models (Conklin and Witten, 1995; Pearce, 2005) in that given an input type sequence, one does not have to make a prediction about the input type itself first and then map this prediction to the domain of the target type. Furthermore, information from multiple input types can be combined in two different ways (1) in a single model which is trained using a set of input types (known as *early fusion*), and (2) with an ensemble containing multiple models (known as *late fusion*). While the latter is only a special case of what has been described in Section 6.2 so far, we describe the former below.

Representing Multiple Input Types in the Same Model

As previously explained in Section 3.1.2, in order to input the type sequences to the neural network models, we first convert each input type value into a $|[\tau]|$-dimensional binary one-hot encoded vector, where $|[\tau]|$ is the size of the syntactic domain of the type τ. When a context event is missing or undefined, each element of the vector is initialized to $1/|[\tau]|$ (*cf.* Section 4.4). Now when there is more than one type in the input to a model, the one-hot vectors corresponding to each of these types extracted from a musical event are concatenated along their lengths to obtain an input vector for that event. This is similar to the idea of *linking* types. And in the case of non-recurrent models, such vectors belonging to successive events in the context are in-turn concatenated as described in Section 4.1. For example, say we rely on types τ_1 and τ_2 of syntactic domain sizes $|[\tau_1]| = 4$ and $|[\tau_2]| = 3$ respectively, and the length of the context used by the model for prediction is $n = 2$, the effective input feature vector to the model at a certain time-step would have the dimensionality $n(|[\tau_1]| + |[\tau_2]|) = 2 \times (4 + 3) = 14$. In doing so, we are effectively bypassing the need to compute a Cartesian product to use multiple types within the same model, which has been the practice when when linking types while using n-gram and variable order Markov models.

6.3.1 Experiments

In order to evaluate the combination of types, we selected one type which is related to the "what" in music — scale-degree (`intfref`), and another which is related to the "when" — inter-onset interval (`ioi`), from the many available choices. Our target type i.e. the one being predicted, is musical pitch (`pitch`) here as well. Through the experiments in this section, we wish to first determine whether or not and to what extent the addition of the above two *types* results in an improvement in the predictive performance of a model that previously relied only on the type `pitch` as input. This would require answering the first set of the following three questions:

1. Does the model predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi}}\left(s^{(1:t-1)}\right)\right)$ show any improvement over $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch}}\left(s^{(1:t-1)}\right)\right)$?

2. Does the model predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,intfref}}\left(s^{(1:t-1)}\right)\right)$ show any improvement over $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch}}\left(s^{(1:t-1)}\right)\right)$?

3. Does the model predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi,intfref}}\left(s^{(1:t-1)}\right)\right)$ similarly also show any improvement over $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch}}\left(s^{(1:t-1)}\right)\right)$, $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi}}\left(s^{(1:t-1)}\right)\right)$ and $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,intfref}}\left(s^{(1:t-1)}\right)\right)$?

And second, we wish to determine how the results of combining the influences of the two additional types through ensembling techniques compare with that obtained when the same is done in a single model. Also relevant here is a comparison between the mean and product rule ensembles. Previous work by Pearce (2005) which compared the two concluded that the latter tended to outperform the former in the majority of cases, and recommended its use in practice. We consider then a second set of questions:

1. How does combining the models predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi}}\left(s^{(1:t-1)}\right)\right)$ and $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,intfref}}\left(s^{(1:t-1)}\right)\right)$ using the mean rule compare with the single model predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi,intfref}}\left(s^{(1:t-1)}\right)\right)$?

2. How does combining the models predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi}}\left(s^{(1:t-1)}\right)\right)$ and $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,intfref}}\left(s^{(1:t-1)}\right)\right)$ using the product rule compare with the single model predicting $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right)|\Psi_{\texttt{pitch,ioi,intfref}}\left(s^{(1:t-1)}\right)\right)$?

3. How do the mean and product rules compare against each other?

The Prediction Model

To answer the above questions, we employ a variant of the basic feed-forward network, which was originally introduced in (Bengio et al., 2003) as a language model for word sequences. The choice of this model was motivated by its demonstrated success on that task. It consists of a feed-forward network such as the one described in Section 4.2, with an additional *embedding* layer (the bottom layer of the network illustrated in Figure 6.1). A given input vector $\mathbf{x} \in \mathbb{R}^{n_i}$ (*cf.* 6.3)is first mapped onto the embedding layer $\mathbf{v} \in \mathbb{R}^{n_e}$ (where typically $n_i > n_e$). The mapping operation is replicated at each time-step for the vector corresponding to $\tau_{in}^{(\mathrm{t})}$ in the input layer, wherein it is mapped onto a corresponding subset of units $\mathbf{e}^{(\mathrm{t})}$ in the

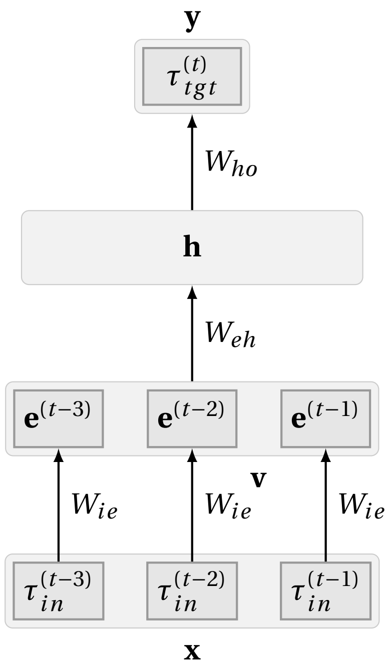


Fig. 6.1 The architecture of the feed-forward neural network with an embedding layer **v** and hidden layer **h**. It models sequences of 4 musical events at a time so as to predict the fourth event given the first three.

embedding layer as shown in Figure 6.1, according to the equation

$$\mathbf{e}^{(t)} = W_{ie}^{\top}\tau_{in}^{(t)}$$

and $\mathbf{v} = [\mathbf{e}^{(t-n+1)} \dots \mathbf{e}^{(t-1)}]$ is the concatenation of the transformed vectors corresponding to the input events of the context. Each mapping is known as a *distributed representation* of its corresponding one-hot vector and, given a fully-trained network, contains information relating to the co-occurrence of that one-hot vector with others. The weight matrix W_{ie} involved in generating this mapping is also learned just as the other weight matrices of the model. From here, the forward-propagation steps are identical to those in the feed-forward neural network introduced in Chapter 4. The embedding layer is then mapped onto the hidden layer by multiplying it with a weight-matrix $W_{eh} \in \mathbb{R}^{n_e \times n_h}$ and applying an element-wise non-linearity f_h to the result $\mathbf{u}_h$ of this product:

$$\mathbf{u}_h = \mathbf{b}_h + W_{eh}^{\top}\mathbf{v}$$
$$\mathbf{h} = f_h(\mathbf{u}_h) \ .$$

where $\mathbf{b}_h \in \mathbb{R}^{n_h}$ is the hidden layer bias. Here, f_h is the Hyperbolic Tangent non-linearity. Likewise, the hidden layer activations vector $\mathbf{h} \in \mathbb{R}^{n_h}$ thus obtained undergoes the same process to produce the outputs.

$$\mathbf{u}_o = \mathbf{b}_o + W_{ho}^{\top}\mathbf{h}$$
$$\mathbf{y} = f_o\left(\mathbf{u}_o\right)$$

where $\mathbf{b}_o \in \mathbb{R}^{n_o}$ is the output layer bias, and the projection happens via the weight-matrix $W_{ho} \in \mathbb{R}^{n_h \times n_o}$. The output layer applies the softmax non-linearity, thus making it a probabilistic model. The input-to-embedding weights W_{ie}, embedding-to-hidden weights W_{eh}, and hidden-to-output weights W_{ho}, together with the hidden and output layer biases $\mathbf{b}_h$ and $\mathbf{b}_o$ respectively make up the parameters of the model.

This model was employed to predict the following six distributions relevant to the above questions, and these are referred to in the rest of this section as:

1. **Instance A** which predicts the type `pitch` using a context of the same type as its input, i.e. $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right) | \Psi_{\texttt{pitch}}\left(s^{(1:t-1)}\right)\right)$.

2. **Instance B** which predicts the type `pitch` using a context of this and the type `ioi` as its input, i.e. $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right) | \Psi_{\texttt{pitch,ioi}}\left(s^{(1:t-1)}\right)\right)$.

3. **Instance C** which predicts the type `pitch` using a context of this and the type `intfref` as its input, i.e. $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right) | \Psi_{\texttt{pitch,intfref}}\left(s^{(1:t-1)}\right)\right)$.

4. **Instance D** which predicts the type `pitch` using a context of this and the types `ioi`, `intfref` as its input, i.e. $P\left(\Psi_{\texttt{pitch}}\left(s^{(t)}\right) | \Psi_{\texttt{pitch,ioi,intfref}}\left(s^{(1:t-1)}\right)\right)$.

5. **Instance $\mathbf{E_m}$** which is an ensemble that combines Instance B and Instance C using the mean rule with the entropy-based weighting scheme.

6. **Instance $\mathbf{E_p}$** which is an ensemble that combines Instance B and Instance C using the product rule with the entropy-based weighting scheme.

Evaluation & Methodology

The six prediction models listed above (Instance A through Instance E_p) were evaluated on a subset of the folk and chorale melody corpus described in Section 3.2, namely, the Chinese folk melody dataset. This particular dataset was chosen due to it being the largest of the 8 contained in the corpus. The extension of this evaluation to the rest of the corpus has been left as future work. In order to evaluate the prediction models here, we once again use cross entropy (see Section 3.7) as in the previous chapters. We remind the reader that, since cross entropy represents the divergence between the empirical distribution and the model's predicted distribution, it is a quantity to be minimised and thus a lower value on the test data reflects better performance. Different neural network configurations were evaluated by a grid search with a learning rate $\eta = 0.01$, the number of hidden units $n_{hid} = \{25, 50, 100, 200, 400\}$, number of embedding units $n_{emb} = \{5, 10, 20\}$, and weight decay $w_{decay} = \{0.0000, 0.0001\}$. The combination bias parameter b for computing the entropy-based weights w_m was varied as $b = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 16, 32\}$, as in (Pearce, 2005). This range was used for both combination rules. Each model was trained using mini-batch gradient descent up to a maximum of 1000 epochs with a batch size of 100 samples, and evaluated over 10-folds, with folds identical to those used in (Pearce and Wiggins, 2004) as before. A small part of the training set (5%) in each fold is extracted as the validation set for model selection over the various hyperparameters.

6.3.2 Results & Discussion

Figure 6.2 compares the predictions of Instances *A*, *B*, *C* and *D* to answer the first set of three questions pertaining to the addition of new types as input into a single model. The first observation here is that only the addition of the `intfref` type leads to any improvement. This is reflected in the similarity between the plots of pairs of model instances which differ only in their use of the `ioi` type (1) Instance *A* and Instance *B*, and (2) Instance *C* and Instance *D*. Paired t-tests between models in both (1) and (2) at different context lengths confirmed that differences in performance were not significant. The tests were performed over all 10 folds of the dataset ($n = 10$). On the other hand, the addition of the type `intfref` does contribute to an improvement in predictive performance from the pitch-only model (Instance *A*). However, paired t-tests revealed that this difference was only significant at lower context lengths of up to 3 in the case of the first pair consisting of Instance *A* and Instance *C*. Likewise, the difference in performance between Instance *B* and Instance *D* was found to be significant upto a context length of 4. The lack of significant improvement in the longer context lengths can possibly be attributed to the Hughes phenomenon, whose effect was observed at longer context lengths in the case of the prediction models discusseds in Section 4.4.3. Its effect is expected to be more pronounced in the case of models which use higher number of input features, and thus rely on a larger number of parameters. Alter-

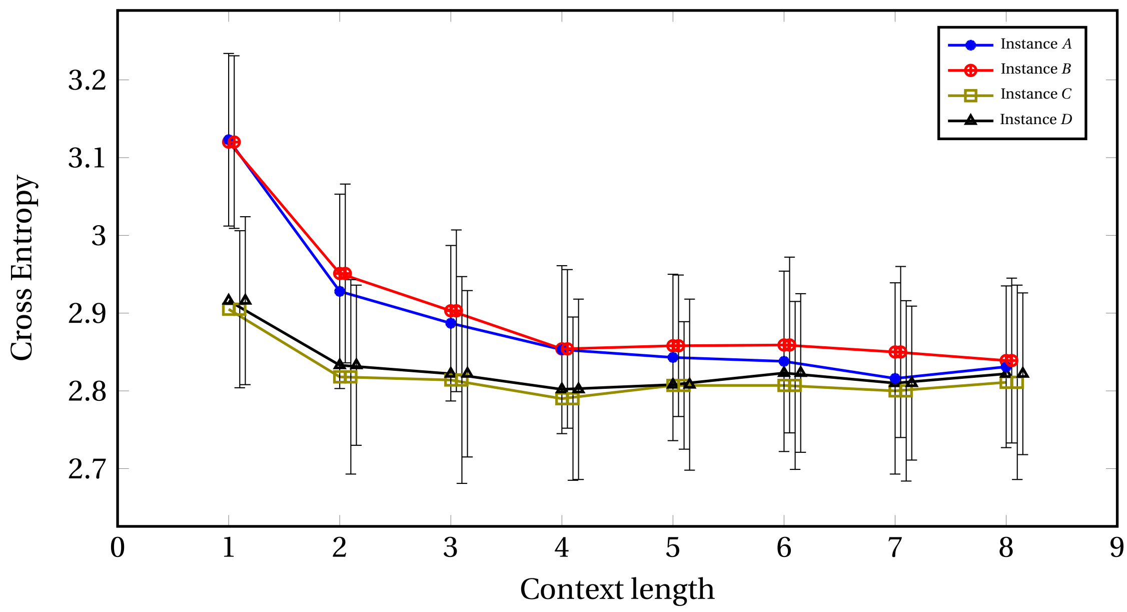


Fig. 6.2 Comparison between the predictive performances, on the Chinese folk melody dataset, of the pitch-only NPMM (Instance *A*), its extensions which use the *intfref* type (Instance *B*) and *ioi* type (Instance *C*) as additional input, and both these types together in a single model (Instance *D*).

natively, it might also be the case that the `intfref` type does not contribute any useful information beyond a certain number of immediately preceding events in this particular dataset.

These observations lead to the conclusion that, while it is indeed possible to improve the predictive performance with the inclusion of new musical features in the same model, it might not always be so. This could be confirmed in those cases where the context length is small enough to not lead to any adverse effects pertaining to data insufficiency. The experiment confirmed the prior expectation that the explicit inclusion of the `intfref` type would help improve predictive performance since scale degree information was originally absent in the input data due to the lack of any transposition of the melodies. The importance of scale degree in musical pitch prediction was also observed in previous work involving *n*-gram models (Pearce and Wiggins, 2006). The expectation was not the same with the type `ioi` since it does not belong to the type set (Section 3.1) of `pitch`, i.e. it is not derived from the type `pitch`. To what extent `ioi` influences the predictions of `pitch` would depend on the correlation between variations in both features in a given dataset, and the results here suggest that this wasn't the case.

In regard to the second set of questions listed above which meant to compare the two ensembling techniques with a single model using the same features as in-

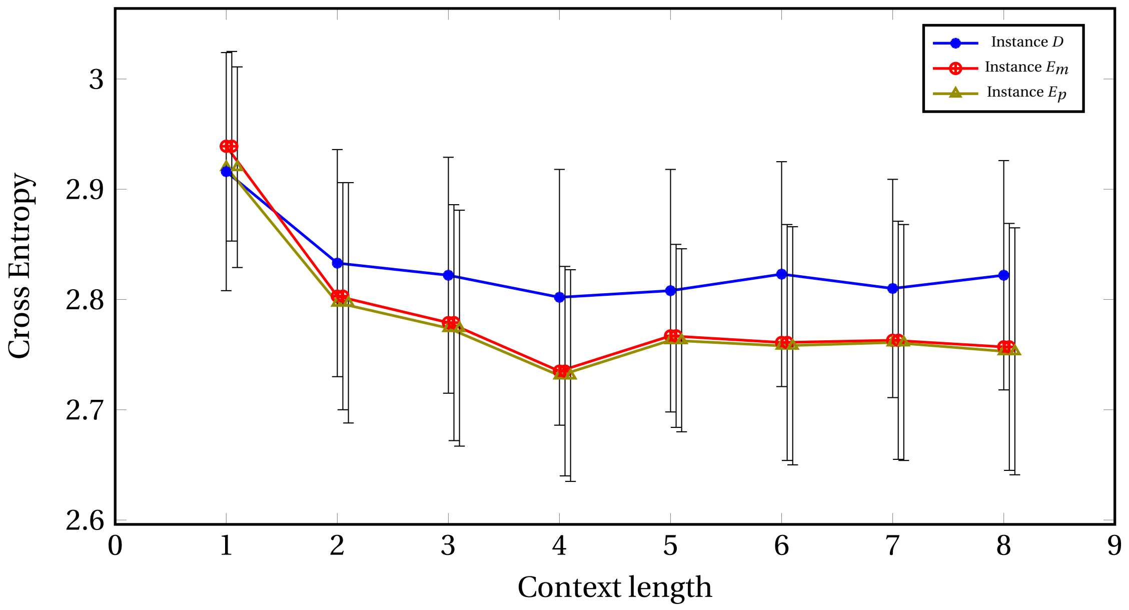


Fig. 6.3 Comparison between the predictive performances, on the Chinese folk melody dataset, of the three models that incorporate the types `pitch`, `ioi` and `intfref` as inputs to a single model (Instance D), and as ensembles of multiple models using the mean and product combination rules (Instance E_m and Instance E_p respectively).

put, the result is illustrated in Figure 6.3. While it might seem from this figure that the average predictive performance of the two ensembles is slightly better than that of the single model incorporating all three types, a paired t-test also revealed that this difference was not significant in all context lengths except 6. This result provides an initial confirmation that a single connectionist model incorporating several different types as opposed to multiple models combined in an ensemble could indeed be a feasible alternative.It was also observed that there was no significant difference in the outcome of combining the said models using either the mean or the product rules. This differs from the result in (Pearce, 2005) where it was shown that the product rule resulted in better predictive performance than the mean rule. When using the mean rule (Instance E_m), a bias value of 1 resulted in better prediction on a validation set with most of the context lengths, and was hence chosen. The value of the bias was 0 in the case of the product rule (Instance E_p). It is the results corresponding to these bias values that have been plotted in Figure 6.3. Finally, as more types were added as input to the models, it was found that smaller embedding layers were being favoured in the grid search.

6.4 Summary

This chapter presented experiments on combining different models that predict the musical pitch of the next note, each given different input melodic features. This aimed to determine the extent to which the addition of new features within the same model improves predictive performance and how this compares with a combination of multiple models, each of which contains a subset of all the features contained in the former. This was explored with a variant of the Feed-forward Neural Network. It was found that out of the two new features that were added as input to the model, there was an improvement in the quality of predictions only in the case of one of these. Moreover, it was found that the addition of both the features as inputs to the same model resulted in similar performance in comparison to ensembles that combine multiple models incorporating the same features. The models were combined using two combination rules, namely the entropy-weighted mean and the product rules introduced in (Conklin and Witten, 1995; Pearce, 2005). It was observed that in the majority of cases, the performance of the mean and product rules was very similar with the key difference being the choice of combination bias value which was lower for the product rule. The reader is referred to Appendix B for more details on the results reported in this Chapter.

Chapter 7

Extending the DRBM and RTDRBM

While the objective of this dissertation at the outset was to demonstrate the efficacy of connectionist models for sequences of musical pitch, which led to the proposal of the novel Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM) architecture in Chapter 5, here we present two additional new directions of work that came about in the process. In the first, the application of the RTDRBM is extended to sequence labelling in general, of which music modelling is only a special case. This involves a small modification to the originally proposed prediction algorithm based on the relaxation of an assumption specific to the latter. This generalisation is evaluated on a benchmark dataset for Optical Character Recognition (Kassel, 1995; Taskar et al., 2004). Results show that the RTDRBM outperforms a set of baseline models by a clear margin. Second, we propose extensions to the Discriminative Restricted Boltzmann Machine (DRBM) which was employed as a non-recurrent melody model in Chapter 4. In the work that originally proposed the DRBM (Larochelle and Bengio, 2008; Larochelle et al., 2012), its hidden layer activations were of the Logistic Sigmoid type. Here, we prove the validity of the generalisation of the said activations to three other types commonly encountered in connectionist literature - Hyperbolic Tangent, Binomial, and Rectified Linear. We carry out experiments on the benchmark MNIST digit dataset (LeCun et al., 2006) to demonstrate the efficacy of the first two of these extensions. Results show that these two variants of the original DRBM perform as well as the logistic sigmoid. An evaluation of the latter (rectified linear) has been left as work to be carried out in the future. This chapter serves to answer the fourth and final research question in Section 1.2.

7.1 Generalising the RTDRBM

This section proposes the extension of the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM) to the general supervised learning problem of sequence labelling. A formalism for sequence labelling, much of which has been adopted from parts of (Graves, 2012), is presented first. This is followed by an explanation of how the RTDRBM can be used to address this task. The extended model is evaluated on a dataset for Optical Character Recognition where it outperforms a set of 8 baseline models from a previous study (Nguyen and Guo, 2007).

7.1.1 Sequence Labelling

The goal of sequence labelling is to assign sequences of labels, drawn from a fixed alphabet, to sequences of input data (Graves, 2012). We limit our attention to cases where the alignment between the input sequence and the corresponding label sequence is predetermined so that only learning the relationship between the inputs and labels is of importance. Furthermore, it is also assumed that the sequences are independent and identically distributed so as to be able to apply the basic machine learning framework (*cf.* Section 3.3.1) where this assumption must hold.

Sequence labelling tasks belong to one of three classes which make progressively looser assumptions about the relationship between the input and label sequences (Graves, 2012). The first and most restrictive case is known as *Sequence Classification*, where a single label corresponds to each sequence. As the name suggests, this involves classifying an entire sequence (as opposed to individual elements in it) as belonging to a class. The entire sequence can be processed before the classification is made. In the second, less restrictive case known as *Segment Classification*, the target sequence consists of multiple labels but the locations of the labels are known in advance. As this involves assigning class-labels to individual elements in a sequence, i.e. the segments as a model comes across them, the effective use of context (from either side of the segments to be classified) by the model is vital to segment classification algorithms. And the third, least restrictive case is known as *Temporal Classification*, wherein no assumptions are made about the label sequences, except that their length is less than or equal to that of the input sequences. They may even be empty. The key distinction between temporal classification and segment classification is that the former requires an algorithm that can decide where in the input sequence the classifications should be made.

The sequence labelling tasks in this chapter belong to the segment classification category, with the added constraint that the length of a label sequence is equal to that of its corresponding input sequence, i.e. a prediction is made at every time-step of the input sequence and thus the segment length is 1. Here we formalise the task of segment classification. Let S_{train} be the set of training examples drawn independently from a fixed distribution $\mathscr{D}_{\mathscr{X}\times\mathscr{Y}}$. The input space $\mathscr{X} = \left(\mathbb{R}^M\right)^*$ is the set of all sequences of size M real-valued vectors. The target space $\mathscr{Y} = L^*$ is the set of all sequences over the finite alphabet L of labels. We refer to elements of L^* as label sequences. Each element is a pair of sequences (X, Y). The target sequence $Y = [y^{(1)}, \ldots, y^{(T)}]$ is as long as the input sequence $X = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}]$. The task is to use S_{train} to train a prediction model $h : \mathscr{X} \mapsto \mathscr{Y}$ to label the sequences in a test set $S_{test} \subset \mathscr{D}_{\mathscr{X}\times\mathscr{Y}}$ as accurately as possible.

7.1.2 The RTDRBM for Segment Labelling

The Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM) was introduced in Section 5.3 where its application to the task of melody modelling was described. In the RTDRBM, the probability distribution over the output variable $y^{(t)}$ at time-step t is given by

$$P(\mathbf{y}^{(t)}|\mathbf{x}^{(1:t)}) = P(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, \hat{\mathbf{h}}^{(t-1)})$$

where the vector $\mathbf{y}^{(t)}$ contains the conditional probabilities of the different possible values of the class-label $y^{(t)}$. According to this, the output $\mathbf{y}^{(t)}$ depends on the input $\mathbf{x}^{(t)}$ and the state of the hidden layer $\hat{\mathbf{h}}^{(t-1)}$ from the previous time-step. The latter is given by

$$\begin{aligned}\hat{\mathbf{h}}^{(t-1)} &= \sigma(W\mathbf{x}^{(t-1)} + U\mathbf{y}^{(t-1)} + \mathbf{c}^{(t-1)})\\ &= \sigma(W\mathbf{x}^{(t-1)} + U\mathbf{y}^{(t-1)} + W_{hh}\hat{\mathbf{h}}^{(t-2)} + \mathbf{c}) \qquad (7.1)\end{aligned}$$

Thus, the state of the hidden layer $\hat{\mathbf{h}}^{(t-1)}$ from the previous time-step which conditions the output $\mathbf{y}^{(t)}$ is itself conditioned on the output $\mathbf{y}^{(t-1)}$ of that time-step. In the melody modelling task, the one-hot encoding of the musical pitch $p^{(t)}$ (which is to be predicted) substitutes the label $\mathbf{y}^{(t)}$ in (7.1), whereas that of the most recent event from the context $p^{(t-1)}$ substitutes the input $\mathbf{x}^{(t)}$. A key assumption there is that, at time-step t the true values $p^{*(1:t-1)}$ of the preceding time-steps are known, using which the value $p^{(t)}$ can be predicted. This means that while predicting $y^{(t)}$, the true values of the previous outputs $y^{*(1:t-1)}$ are available. This corresponds to the intuition that one has already heard the pitch of a note while anticipating the next note, which is justified by the application.

Generally, however, this assumption may not always be satisfied as in the case of the OCR dataset considered in this section in which the goal is to predict a sequence of output labels given a sequence of input features. Thus, in contrast to the melody modelling task we cannot assume any knowledge of the correct label at the previous time-step. This being the case, we directly use the probability distribution predicted by the model at each time-step in order to generate the corresponding hidden layer activations, and as the results in this section demonstrate, this can lead to good quality predictions.

7.1.3 Experiments: Optical Character Recognition

Optical Character Recognition (OCR) involves the conversion of images of typed, handwritten or printed text into machine-encoded text. OCR systems have been commercially available since the middle of the 1950s, and used in postal address reading, license plate recognition, publishing, billing systems and reading aids for the blind (Govindan and Shivaprasad, 1990). The common goal for such applications is the digitisation of printed or handwritten text, which is motivated by the need to index, search and retrieve from increasing number of sources of text. An algorithm for OCR can either recognise each character independent of other characters in the document or as a sequence, the latter of which would also exploit the grammatical information present in character sequences. Here we consider the latter, wherein sequences of individually segmented handwritten characters which make up words serve as the inputs for an RTDRBM which outputs corresponding symbols from the character vocabulary.

Dataset

The MIT OCR dataset[1] originally compiled by Kassel (1995), and later refined by Taskar et al. (2004) is a widely used benchmark dataset for evaluating sequence labelling algorithms. It contains $6,877$ words, which corresponds to $52,152$ English characters. The data are sequences of isolated characters, where an image of a character, of size 16×8 is represented by a vector $\mathbf{x} \in \{0,1\}^{128}$ and belongs to one of 26 classes. The refinement referred to above involved the removal of capitalised leading characters, rasterising and normalising the images of each character. The

[1]http://www.seas.upenn.edu/~taskar/ocr/

dataset is divided into 10 cross-validation folds and has one hold-out test set. We train each RTDRBM, initialised with a different set of values of its hyperparameters, on 9 folds and evaluate it on the final fold during model selection, and then evaluate the best model on the test set (i.e. 10 times).

Baseline Models

We compare the performance of the RTDRBM on the sequence labelling task with the 8 baseline models from the comparative study in (Nguyen and Guo, 2007). These include the Multiclass Support Vector Machine (SVM^{multiclass}) (Crammer and Singer, 2002), Structured SVM (SVM^{struct}) (Tsochantaridis et al., 2005), Max-Margin Markov Network (M^3N) (Taskar et al., 2004), Averaged Perceptron (Collins, 2002), SEARN (Daumé III et al., 2009), Conditional Random Field (CRF) (Lafferty et al., 2001; Peng and McCallum, 2006), Hidden Markov Model (HMM) (Rabiner, 1989), and an ensemble known as the *Structured Learning Ensemble* (SLE) (Nguyen and Guo, 2007) which selects and combines predictions by a subset of all the other models. We refer the reader to the cited work for further details on the models.

Methodology

In order to determine the best model for the task, a grid search was carried out. The initial learning rate η_{init} was set to 0.05. Early-stopping was enabled. For this, the performance of the model on a validation set was determined after every epoch. If the performance happened to be worse than the previous best one for ten consecutive checks, the parameters were reverted back to their values in the previous best model, and training was resumed with a reduced learning rate. And if this happened five times, training was terminated. The learning rate reduction was according to a linear schedule where it is progressively scaled by the factors $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and so on at each reduction step. The number of hidden units n_{hid} was varied as $\{100, 200, 300, 400\}$. Both L_1 and L_2 decay were set to identical values $\lambda_1 = \lambda_2 = \lambda$ and were either both on ($\lambda = 0.0001$) or both off ($\lambda = 0$). The maximum number of training epochs was set to 1000, but it was found that training did not exceed 200 epochs in any of the examined cases. Sequence learning was carried out using mini-batch gradient descent with Backpropagation Through Time (BPTT), where each batch contained data belonging to one sequence. The RTDRBM generates a probability distribution over the different possible values of the classes at each time-step. The label corresponding to the greatest probability class is chosen as the prediction.

Evaluation Measure

In this task, each model is expected to predict the correct label corresponding to the image of a character. All the models are evaluated using the average loss per sequence $E(\mathbf{y}, \mathbf{y}^*)$, given by:

$$E(\mathbf{y}, \mathbf{y}^*) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{L_i} \sum_{j=1}^{L_i} \mathscr{I}\Big((y_i)_j \neq (y_i^*)_j \Big) \right]$$

where $\mathbf{y}$ and $\mathbf{y}^*$ are the predicted and the true sequence respectively, N is the total number of test examples, L_i is the length of the i^{th} sequence, the index j refers to the event in a sequence L_i, and $\mathscr{I}$ is the $0-1$ loss function. This measure was also used in evaluating the various models in (Nguyen and Guo, 2007), which serve as baselines here.

Results

Table 7.1 shows the comparative performance of the RTDRBM against a set of various baseline structured learning models evaluated in a previously published study (Nguyen and Guo, 2007). It is evident that the RTDRBM clearly outperforms the

Model	**Error (%)**
RTDRBM	**15.95**(±**0.0009**)
SLE	20.58
SVM^{struct}	21.16
HMM	23.70
M^{3}N	25.08
Perceptron	26.40
SEARN	27.02
SVM^{multiclass}	28.54
CRF	32.30

Table 7.1 Comparison between the prediction error (%) of the RTDRBM (containing 400 hidden units and trained with a learning rate of 0.05 and weight-decay of 0.0001) and models evaluated in (Nguyen and Guo, 2007).

other models. The best performance was obtained with an RTDRBM containing 400 hidden units, with weight-decay enabled. Table 7.2 compares the RTDRBM's error-rate on this dataset with published state-of-the-art, which use Neural Conditional Random Fields (NCRF) (Do et al., 2010) and Gradient Boosted Conditional Random Fields (GBCRF) (Chen et al., 2015). Since the best performance corresponds to a model with the highest number of hidden units in the considered hy-

Model	**Error (%)**
NCRF	4.44
GBCRF	4.64(±0.0027)
RTDRBM	**15.95(±0.0009)**

Table 7.2 Comparison between the prediction error (%) of the RTDRBM (containing 400 hidden units and trained with a learning rate of 0.05 and weight-decay of 0.0001) and state-of-the-art on the OCR dataset which use Neural Conditional Random Fields (NCRF) (Do et al., 2010) and Gradient Boosted Conditional Random Fields (Chen et al., 2015).

perparameter grid, there is reason to believe that increasing the number of hidden units can further improve the performance on this dataset. Previous work has also shown that unsupervised pre-training of the weights of connectionist models can lead to better initialisation of their parameters and thus help with obtaining more accurate models when this is followed by supervised learning (Bengio, 2009; Boulanger-Lewandowski et al., 2012; Larochelle et al., 2012). These, and other refinements to the results have been left as future work.

7.2 Extensions to the DRBM

The Discriminative Restricted Boltzmann Machine (DRBM) (Larochelle and Bengio, 2008; Larochelle et al., 2012) was proposed as a variant of the Restricted Boltzmann Machine (RBM) which could be used as a supervised learning model, in contrast to the RBM's original purpose of unsupervised feature learning. The DRBM is structurally identical to the RBM, and contains Logistic Sigmoid units (LogSigU) in its hidden layer, with the exception that a part of its visible layer is always a set of multinomial units corresponding to the class-labels, while the rest is assigned to input features. This section describes the theory for generalising the originally proposed LogSigU-DRBM to other hidden layer types (mathematical proof in Sections 7.2.1 and 7.2.2 provided by Son N. Tran). These consist of Hyperbolic Tangent units (TanHU), Binomial units (BinU) (Teh and Hinton, 2001) and Rectified Linear units (ReLU) (Nair and Hinton, 2010). Experiments are also carried out to on the benchmark MNIST dataset (LeCun et al., 2006) in order to evaluate the efficacy of the first two of these extensions and determine how they compare with the original LogSigU-DRBM.

7.2.1 Generalising the Conditional Probability

We begin with the expression for the conditional distribution $P(y|\mathbf{x})$, as derived in (Larochelle et al., 2012). This is given by

$$P(y|\mathbf{x}) = \frac{\sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{y}, \mathbf{h})}{\sum_{\mathbf{y}^*} \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{y}^*, \mathbf{h})} = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{y}, \mathbf{h}))}{\sum_{\mathbf{y}^*} \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{y}^*, \mathbf{h}))} \tag{7.2}$$

where $\mathbf{y}$ is the one-hot encoding of a class label y, and $-\log\sum_{\mathbf{h}}\exp(-E(\mathbf{x}, \mathbf{y}, \mathbf{h}))$ is the Free Energy E_{free} of the RBM. We consider the term containing the summation over $\mathbf{h}$ in (7.2):

$$\begin{aligned}
\exp\left(E_{free}(\mathbf{x}, \mathbf{y})\right) &= -\sum_{\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{y}, \mathbf{h})\right) \\
&= -\sum_{\mathbf{h}} \exp\left(\sum_{i,j} x_i w_{ij} y_j + \sum_j u_{yj} h_j + \sum_i a_i x_i + b_y + \sum_j c_j h_j\right) \\
&= -\exp\left(\sum_i a_i x_i + b_y\right) \sum_{\mathbf{h}} \exp\left(\sum_j h_j \sum_i x_i w_{ij} + u_{yj} + c_j\right)
\end{aligned} \tag{7.3}$$

Now consider only the second term of the product in (7.3). We simplify it by re-writing $\sum_i x_i w_{ij} + u_{yj} + c_j$ as α_j. Thus, we have

$$\begin{aligned}
\sum_{\mathbf{h}} \exp\left(\sum_j h_j \sum_i x_i w_{ij} + u_{yj} + c_j\right) &= \sum_{\mathbf{h}} \exp\left(\sum_j h_j \alpha_j\right) \\
&= \sum_{\mathbf{h}} \prod_j \exp\left(h_j \alpha_j\right) \\
&= \prod_j \sum_k \exp\left(s_k \alpha_j\right)
\end{aligned} \tag{7.4}$$

where s_k is each of the k states that can be assumed by each hidden unit j of the model. The last step of (7.4) results from re-arranging the terms after expanding the summation and product over $\mathbf{h}$ and j in the previous step respectively. The summation $\sum_{\mathbf{h}}$ over all the possible hidden layer vectors $\mathbf{h}$ can be replaced by the summation $\sum_k$ over the states of the units in the layer. The number and values of these states depend on the nature of the activation type (for instance $\{0, 1\}$ in case of LogSigU, $\{-1, +1\}$ in case of TanHU, and so on). The result in (7.4) can be applied to (7.3) and, in turn, to (7.2) to get the following general expression of the conditional probability $P(y|\mathbf{x})$:

$$\begin{aligned}
P(y|\mathbf{x}) &= \frac{\exp\left(b_y\right)\prod_j \sum_k \exp\left(s_k \alpha_j\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \sum_k \exp\left(s_k \alpha_j^*\right)} \\
&= \frac{\exp\left(b_y\right)\prod_j \sum_k \exp\left(s_k \sum_i x_i w_{ij} + u_{yj} + c_j\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \sum_k \exp\left(s_k \sum_i x_i w_{ij} + u_{y^*j} + c_j\right)} \qquad (7.5)
\end{aligned}$$

The result in 7.5 generalises the conditional probability of the DRBM first introduced in (Larochelle and Bengio, 2008). The term inside the summation over k can be viewed as a product between α_j corresponding to each hidden unit j and each possible state s_k of this hidden unit. Knowing this makes it possible to extend the original model with the logistic sigmoid hidden layer activation type to other types of activations.

7.2.2 Extensions to other Hidden Layer Types

We first use the result in (7.5) to derive the expression for the conditional probability $P(y|\mathbf{x})$ in the original LogSigU-DRBM (Larochelle and Bengio, 2008). This will be followed by its extension to Hyperbolic Tangent, Binomial and Rectified Linear hidden layer activations. Section 7.2.3 presents a comparison between the performance of the DRBM with these different activations.

Logistic Sigmoid (LogSigU-DRBM)

The LogSigU-DRBM corresponds to what was originally introduced in (Larochelle and Bengio, 2008). In this case, each hidden unit h_j can either be a 0 or a 1, i.e. $s_k = \{0,1\}$. This reduces $P(y|\mathbf{x})$ in (7.5) to

$$\begin{aligned}
P_{\text{logsig}}\left(y|\mathbf{x}\right) &= \frac{\exp\left(b_y\right)\prod_j \sum_{s_k\in\{0,1\}} \exp\left(s_k \alpha_j\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \sum_{s_k\in\{0,1\}} \exp\left(s_k \alpha_j^*\right)} \\
&= \frac{\exp\left(b_y\right)\prod_j \left(1+\exp\left(\alpha_j\right)\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \left(1+\exp\left(\alpha_j^*\right)\right)}
\end{aligned}$$

which is identical to the result obtained in (Larochelle and Bengio, 2008).

Hyperbolic Tangent (TanHU-DRBM)

A straightforward adaptation to the original LogSigU-DRBM involves replacing its hidden units with the Hyperbolic Tangent (TanH) non-linearity. We refer to this here as the TanHU-DRBM. This is straightforward because, just like the logistic sigmoid, the hyperbolic tangent also assumes two values. However, in this case each hidden unit h_j can either be a -1 or a $+1$, i.e. $s_k = \{-1, +1\}$. Applying this property to (7.5) results in the following expression for $P(y|\mathbf{x})$:

$$\begin{aligned} P_{\tanh}\left(y|\mathbf{x}\right) &= \frac{\exp\left(b_y\right)\prod_j \sum_{s_k \in \{-1,+1\}} \exp\left(s_k \alpha_j\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \sum_{s_k \in \{-1,+1\}} \exp\left(s_k \alpha_j^*\right)} \\ &= \frac{\exp\left(b_y\right)\prod_j \left(\exp\left(-\alpha_j\right) + \exp\left(\alpha_j\right)\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \left(\exp\left(-\alpha_j^*\right) + \exp\left(\alpha_j^*\right)\right)} . \end{aligned}$$

Binomial (BinU-DRBM)

It was demonstrated in (Teh and Hinton, 2001) how groups of N (where N is a positive integer greater than 1) stochastic units of the standard RBM can be combined in order to approximate discrete-valued functions in its visible layer and hidden layers to increase its representational power. This is done by replicating each unit of one layer N times and keeping the weights of all connections to each of these units from a given unit in the other layer identical. The number of these "replicas" of the same unit whose values are simultaneously 1 determines the effective integer value (in the range $[0, N]$) of the composite unit, thus allowing it to assume multiple values. The resulting model was referred to there as the Rate-Coded RBM (RBMrate).

The intuition behind this idea can be extended to the DRBM by allowing the states s_k of each hidden unit to assume integer values in the range $[0, N]$. The summation in (7.5) would then be $S_N = \sum_{s_k=0}^{N} \exp\left(s_k \alpha_j\right)$, which simplifies as below

$$\begin{aligned} S_N &= \sum_{s_k=0}^{N} \exp\left(s_k \alpha_j\right) \\ &= 1 + \exp\left(\alpha_j\right) \sum_{s_k=0}^{(N-1)} \exp\left(s_k \alpha_j\right) \\ &= \frac{1 - \exp\left(\left(N+1\right)\alpha_j\right)}{1 - \exp\left(\alpha_j\right)} \end{aligned}$$

in (7.5) to give

$$P_{\text{bin}}\left(y|\mathbf{x}\right) = \frac{\exp\left(b_y\right)\prod_j \sum_{s_k=0}^{N} \exp\left(s_k \alpha_j\right)}{\sum_{y^*} \exp\left(b_{y^*}\right)\prod_j \sum_{s_k=0}^{N} \exp\left(s_k \alpha_j^*\right)}$$

$$= \frac{\exp\left(b_y\right)\prod_j \frac{1-\exp\left((N+1)\alpha_j\right)}{1-\exp\left(\alpha_j\right)}}{\sum_{y^*}\exp\left(b_{y^*}\right)\prod_j \frac{1-\exp\left((N+1)\alpha_j^*\right)}{1-\exp\left(\alpha_j^*\right)}} . \tag{7.6}$$

We shall henceforth refer to this model as the BinU-DRBM.

Rectified Linear (ReLU-DRBM)

The work of Nair and Hinton (2010) introduced what is known as the Rectified Linear Unit (ReLU) for the hidden layer of the RBM. This type of unit is inspired by the aforementioned Binomial units (Teh and Hinton, 2001) and is a result of increasing the number of replicas of a single binary unit to infinity. Adopting the same intuition here in the case of the DRBM, this would mean that we allow the states s_k to assume integer values in the range $[0,\infty)$ and thus extend the summation S_N in the case of the above BinU-DRBM to an infinite sum S_∞ resulting in the following simplification

$$\begin{aligned} S_\infty &= \sum_{s_k=0}^{\infty} \exp\left(s_k\alpha_j\right) \\ &= 1+\exp\left(\alpha_j\right)\sum_{s_k=0}^{\infty} \exp\left(s_k\alpha_j\right) \\ &= \frac{1}{1-\exp\left(\alpha_j\right)} \end{aligned}$$

with the equation for the ReLU-DRBM posterior probability in (7.5) becoming

$$\begin{aligned} P_{\text{relu}}\left(y|\mathbf{x}\right) &= \frac{\exp\left(b_y\right)\prod_j \sum_{s_k=0}^{\infty}\exp\left(s_k\alpha_j\right)}{\sum_{y^*}\exp\left(b_{y^*}\right)\prod_j \sum_{s_k=0}^{\infty}\exp\left(s_k\alpha_j^*\right)} \\ &= \frac{\exp\left(b_y\right)\prod_j \frac{1}{1-\exp\left(\alpha_j\right)}}{\sum_{y^*}\exp\left(b_{y^*}\right)\prod_j \frac{1}{1-\exp\left(\alpha_j^*\right)}} . \end{aligned}$$

7.2.3 Experiments: Handwritten Digit Recognition

In this section we compare the performance of the first two variants of the Discriminative RBM introduced above (TanHU-DRBM and BinU-DRBM) with that of the originally proposed one with logistic sigmoid hidden units (LogSigU-DRBM) (Larochelle and Bengio, 2008) on the MNIST handwritten digit dataset. It was observed that the classification performances of both the TanHU-DRBM and BinU-DRBM were on par with that of the LogSigU-DRBM. Evaluation of the ReLU-DRBM was not carried out, and has been left to be done in the future. We provide details of the evaluation procedure and the available results in this section.

Dataset

We evaluated the said extensions to the Discriminative RBM on the MNIST dataset which serves as a standard benchmark for classification machine learning algorithms. The MNIST dataset consists of optical characters of handwritten digits. Each digit is a 28×28 pixel gray-scale image (or a vector $\mathbf{x} \in [0,1]^{784}$). Each pixel of the image corresponds to a floating-point value lying in the range $[0,1]$ after normalisation from an integer value in the range $[0,255]$. The dataset is divided into pre-determined training, and test folds containing $60,000$ and $10,000$ images respectively. Of the training images, $10,000$ are separated as the validation set determine the training progress.

Methodology

A grid search was carried out to determine the best hyperparameters. The initial learning rate η_{init} was varied as $\{0.001, 0.01, 0.1\}$. Early-stopping was enabled. For this, the performance of the model on a validation set was determined after every epoch. If the performance happened to be worse than the previous best one for ten consecutive checks, the parameters were reverted back to their values in the previous best model, and training was resumed with a reduced learning rate. And if this happened five times, training was terminated. The learning rate reduction was according to a linear schedule where it is progressively scaled by the factors $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and so on at each reduction step. The number of hidden units n_{hid} was varied as $\{100, 500, 1000, 5000\}$. Both L_1 and L_2 decay were set to identical values $\lambda_1 = \lambda_2 = \lambda$ and were either both on ($\lambda = 0.0001$) or both off ($\lambda = 0$). The maximum number of training epochs was set to 2000, but it was found that training ended before this limit in anything between 50 and 150 epochs in all the examined cases. The negative log-likelihood error criterion was used to optimise the model parameters, and it was found that stochastic gradient descent produced better results than batch gradient descent with a batch size of 400. The DRBM generated a probability distribution over the different possible values of the classes. The class-label corresponding to the greatest probability value was chosen as the predicted class. As the MNIST dataset contains only a single data split, i.e. only one set of training, validation and test sets, the results reported here are each an average over those obtained with 10 different model parameter initialisations using different randomisation seeds.

In addition to the above described methodology, which was common to both the TanH-DRBM and the BinU-DRBM, a few additional considerations were necessary for the latter. Firstly, there is the choice of the number of bins n_{bins}, corresponds to the number of states that can be assumed by each hidden unit of the model. When the value of this hyperparameter is 2, the BinU-DRBM is equivalent to the LogSig-DRBM. The value of n_{bins} was initially varied as $\{2, 4, 8, 16, 32, 64\}$ in our experiments. It was observed that in the case of the BinU-DRBM, there was a tendency for arithmetic overflow errors, particularly when the value of n_{bins} is large. This is expected, if one considers the expression (7.6) for computing the conditional probability in this model. In order to counter this, a lower learning rate of 0.0001 was also included in the grid search, and excluded $n_{bins} = \{16, 32\}$ in the final run whose results are presented in Table 7.4.

Evaluation Measure

In this task, each model is expected to predict the correct label corresponding to the image of a digit. All the models are evaluated using the average loss $E(\mathbf{y}, \mathbf{y}^*)$, given by:

$$E(\mathbf{y}, \mathbf{y}^*) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{I}\left(y_i \neq y_i^*\right)$$

where $\mathbf{y}$ and $\mathbf{y}^*$ are the predicted and the true labels respectively, N is the total number of test examples, and $\mathscr{I}$ is the $0-1$ loss function. This measure was also used in evaluating other models on the MNIST dataset, including the LogSigU DRBM (Larochelle and Bengio, 2008) with which the models here are being compared.

Results

Table 7.3 lists the classification performance on the MNIST dataset of the different variants of the DRBM introduced here along with that of the LogSigU-DRBM introduced in (Larochelle and Bengio, 2008). On this dataset, a difference of 0.2% in classification error is considered statistically significant. As one can see, the performance of the TanHU-DRBM and each of the BinU-DRBMs is equivalent to that of the LogSigU-DRBM. All variants perform best with 500 hidden units. It was observed, in the case of the BinU-DRBM, that the number of bins didn’t play as significant a role as first expected in the performance of the model. There seemed to be a slight deterioration in accuracy with an increase in the number of bins, but

Model	**Error-rate (%)**
LogSigU-DRBM ($n = 500$, $\eta = 0.05$)	1.81
TanHU-DRBM ($n = 500$, $\eta_{init} = 0.01$)	1.84($\pm$0.0007)
BinU-DRBM ($n = 500$, $\eta_{init} = 0.01$)	1.86($\pm$0.0016)

Table 7.3 A comparison between the originally proposed DRBM in (Larochelle and Bengio, 2008) and two of the three extensions proposed in the present work. The BinU-DRBM in this table is the one with $n_{bins} = 2$.

this difference cannot be considered significant given the MNIST dataset. This is illustrated in Table 7.4. It remains to be investigated whether this equivalent performance of all variants of the DRBM is indeed something to be expected in theory, or whether it is a result specific to the chosen dataset.

n_{bins}	η_{init}	**Error-rate (%)**
2	0.01	1.86%
4	0.01	1.88%
8	0.001	1.90%
16	0.001	1.92%

Table 7.4 Classification performance of the BinU-DRBM with different values of n_{bins}. All the models listed in the table use a hidden layer size of 500.

7.3 Summary

In this chapter, we presented extensions to two of the models previously used for the task of melody modelling in Chapters 4 and 5. These are the DRBM and the RTDRBM Respectively. In the case of the RTDRBM, we relaxed a constraint that previously applied to the melody modelling scenario where the model was originally introduced. This constraint states that while predicting a class-label $y^{(t)}$ at time-step t, the true values of the labels $y^{*(1:t-1)}$ of the previous time-steps are available. Relaxing this constraint adapts the RTDRBM to general sequence labelling tasks. Here we directly use the probability distribution generated by the model at each time-step instead of the true label in order to generate the corresponding hidden layer activations. We evaluated the model on the benchmark OCR dataset for sequence labelling and found that it outperformed all the baseline models evaluated on this dataset in a previous study (Nguyen and Guo, 2007). In the case of the DRBM, we first derived a general expression for the posterior probability distribution $p(y|\mathbf{x})$ (where y is the class label and $\mathbf{x}$ is an input feature) according to this model, which allows one to use hidden layer activations other than Logistic Sigmoid in this model with which it was originally introduced. We then evaluated two variants of the model, with hidden layer activations Hyperbolic Tangent and Binomial, and found that, on the benchmark MNIST dataset, all these had a classification performance equivalent to the original Logistic Sigmoid DRBM (Larochelle and Bengio, 2008).

Chapter 8

Conclusions & Future Work

8.1 Conclusions

At the outset, the goal of this dissertation was to demonstrate the efficacy of connectionist models for sequences in monophonic music in following an information theoretic approach that was previously exemplified through the use of n-gram models in (Conklin and Witten, 1995; Pearce, 2005; Whorley et al., 2013). This approach extended the statistical modelling techniques, originally applied to text and language analysis (Bell et al., 1989; Manning and Schütze, 1999), to music domain based on two key observations: (1) that music has a rich internal structure and can be expressed in terms of directly observable or derived musical features, each of which can potentially influence the accuracy of predicting how a piece of music evolves (2) that a predictive model of music can benefit from both (global) information encountered in a large corpus, and (local) information available in an individual piece of music during prediction. The Multiple Viewpoint Systems (MVS) framework which resulted, incorporated these observations to combine n-gram models that (1) utilise different musical features in order to predict other related ones and (2) take into account global or local sequential information in the form of Long and Short Term Models respectively (LTM and STM).

During the first phase of the work summarised in this thesis, (Chapters 4 and 5) a comparative evaluation of six different connectionist LTMs was carried out. This included a set of non-recurrent architectures — the Feed-forward Neural Network (FNN), the Restricted Boltzmann Machine trained both generatively (RBM) and discriminatively (DRBM), and their recurrent counterparts — the Recurrent Neural Network (RNN), the Recurrent Temporal Restricted Boltzmann machine (RTRBM) and the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RT-

DRBM). This evaluation addressed research question 1 on the effectiveness of connectionist models for melodic prediction. It was demonstrated that these architectures can perform on par with, or better than the best of n-gram models previously evaluated as LTMs on a corpus of folk and chorale melodies (Pearce and Wiggins, 2004). Moreover, the recurrent models can be trained on entire melodies free from the Markov assumption that binds their non-recurrent counterparts, allowing the former to potentially leverage valuable information contained in longer sequences in order to make even better predictions. These results also allow one to draw parallels with similar results in connectionist language modelling (Schwenk and Gauvain, 2002; Bengio et al., 2003; Mikolov et al., 2010).

The aforementioned experiments dealt purely with sequences of musical pitch, and can be thought of as the simplest case of Multiple Viewpoints (Conklin and Witten, 1995). In an attempt to further generalise these and to address research question 2 regarding the effectiveness of combining models that use different input features, some preliminary experiments were carried out in Chapter 6 which aimed at extending the set of *types* (musical features) used as input by the models while still predicting the `pitch` type as before. It was found firstly that the addition of new types — `intfref` and `ioi`, as input together with `pitch` did indeed improve the predictive performance of a given model (a variant of the FNN), when compared to the case where only `pitch` was used as input. Moreover, whether or not there was any improvement also depended on the type that the model was augmented with. It was also observed that ensembles of models, each containing one extra type in addition to `pitch` performed as well as a single model that incorporated all these types.

The last of the models listed above — the RTDRBM — is a novel contribution of the present work. It was motivated by the observation that while discriminative inference can indeed be carried out in the generatively learned RTRBM to make it suitable for the music prediction task considered here, there hasn't until now been a proposal for learning the parameters of this recurrent model in a discriminative fashion as demonstrated in the case of non-recurrent RBMs (Salakhutdinov et al., 2007; Larochelle and Bengio, 2008). As noted by previous studies on discriminative and generative learning, the former can result in more robust models than the latter under various circumstances dictated by the nature and quantity of the available data (Ng and Jordan, 2001; Bishop and Lasserre, 2007). Formulating a discriminative learning criterion for the RTRBM leads to the RTDRBM, with an expression for the error gradient at each time-step which can be computed exactly, thus allowing one to use standard iterative gradient-based learning techiniques to learn its parameters.

The second phase of this dissertation saw a shift in focus from the original music modelling goals towards machine learning. These were covered in Chapter 7. It was decided, as a first step, to further extend the application of the RTDRBM to more general sequence labelling tasks of which the music modelling task is only a special case. This involved a small modification to the originally proposed prediction algorithm based on the relaxation of an assumption specific to the music modelling application. The generalised model was evaluated on the OCR dataset (Kassel, 1995; Taskar et al., 2004) and compared against a set of 8 baseline models (Nguyen and Guo, 2007) where it faired better than the rest.

A review of literature related to neural networks and restricted Boltzmann machines led to the observation that, while the performance of a variety of hidden layer activations (Logistic Sigmoid, Hyperbolic Tangent, Rectified Linear, etc.) has been explored with these models and their validity theoretically or empirically verified, the same had not been done with the Discriminative RBM (DRBM) which has relied solely on Logistic Sigmoid units. This led to another novel contribution of the present work. On further investigation into the possibility of making the same extensions to the DRBM, it could be proved in theory that Logistic Sigmoid, Hyperbolic Tangent, Binomial and Rectified Linear units in the DRBM are all special cases of the same general form and can be derived in a very similar manner using this underlying form. The first two out of these three theoretical extensions were also experimentally evaluated, and it was found that DRBMs containing either Hyperbolic Tangent or Binomial hidden layer perform on par with that containing a Logistic Sigmoid hidden layer. The proposed extensions to the DRBM and the invention of the RTDRBM can thus be seen as contributions that provide an answer in affirmation to research question 3, i.e. whether the pursuit of answers to the first two questions could lead to novel contributions that go beyond the said application domain into the area of machine learning.

8.2 Future Work

In this section, we outline three main directions for future work. The first of these involves extending the experiments carried out in Chapter 6 which combine multiple prediction models. The second is the application of the said melody models to music classification, demonstrated here with the help of Restricted Boltzmann Machines. And the third involves further extending some of the prediction models considered here to predict multiple *types* and not just `pitch`, as done here.

8.2.1 Experiments with Other Melodic Features

Chapter 6 addressed research questions 2 with encouraging results. It demonstrated an improvement in the predictive performance of models previously relying only on the type `pitch` as input on the inclusion of the `intfref` type as additional input. On the contrary, the addition of the `ioi` type did not have any positive effect on the performance. The extended models were evaluated on the Chinese folk melody dataset. These first steps towards a more comprehensive evaluation of various types to determine their importance in predicting the type `pitch` may be further extended in three ways. The first involves carrying out the same evaluation on the remaining datasets of the folk melody corpus of Section 3.2 of which the Chinese folk melody dataset is a part. Secondly, Chapter 5 showed how recurrent connectionist models are more effective at modelling musical pitch sequences in monophonic music than their non-recurrent counterparts. It is thus expected that combining the extended input types with a recurrent model, say an RNN would result in improved predictive performance when compared to the FNN used in Section 6.3. Thirdly, the present work considered two additional types to augment the input to the models with, from the numerous other types available for use. The results are encouraging, and lead the way to a more comprehensive study as carried out in (Conklin and Witten, 1995; Pearce, 2005; Whorley et al., 2013). Among these, (Pearce, 2005; Whorley et al., 2013) also utilise a feature selection algorithm that determines the optimal set of types for a given dataset or style of music. Such a comprehensive study examining the effect of additional types is also considered to be a necessary continuation of the present work, and has been left for the future.

8.2.2 Experiments with Models Updated During Prediction

One aspect of Multiple Viewpoints as introduced previously in (Conklin and Witten, 1995) has not been explored in the present work. This the idea of the Short-Term Model (STM). A distinction can be made between two types of prediction models on the basis of how their parameters are learned. While the parameters of both types of models are learned through exposure to appropriate data, those of the first are learned on a corpus of melodies (a training set), its parameters thus being finalized beforehand and kept constant during the prediction stage (on a test set). This model is known as a Long-Term Model (LTM). From a music cognition perspective (Pearce and Wiggins, 2004), it represents more global stylistic characteristics acquired by a listener over a longer time-span. All the models examined in Chapters 4 and 5 are examples of the LTM. The parameters of the second type of model are learned only while making predictions (on a test set), without any sequence learning occurring in it beforehand. This is what is known as the Short-Term Model (STM). It highlights the importance of context-specific information, available in a melody while it is being processed by a listener, in the generation of expectations. A variant of the LTM which lies between the two, introduced in (Pearce, 2005), is the LTM+. In addition to being learned offline on a corpus of melodies like the LTM, it is also updated while making predictions just like the STM. Another distinction between the LTM+ and the STM is that the former is continuously updated across melodies, while the latter is re-initialized after each melody in the test set.

Predictions (in the form of probability distributions) made by each model about a certain musical event in a sequence are combined using the two ensemble methods described in Section 6.2, and this has been shown to improve the quality of predictions over individual models in the past (Conklin and Witten, 1995; Pearce, 2005). The idea of combining corpus-based long-term and context-sensitive short-term predictions from different models was originally a feature of cache-based language models (Kuhn and De Mori, 1990). It was introduced in the context of music in (Conklin and Witten, 1995). The distinction between the long- and short-term models is also akin to the that made in (Justus and Bharucha, 2001) between "schematic" (LTM) and "veridical" (STM) knowledge in a modular view on music processing.

To address this prediction task, we employed the Recurrent Temporal Discriminative Restricted Boltzmann Machine (RTDRBM) introduced in Chapter 5. We began by evaluating its utility as an STM by carrying out online learning in it. Experiments revealed that, while learning did indeed take place, it did not progress quickly enough (as a function of the number of data samples presented to the RTDRBM) to outperform existing state-of-the-art models where both the LTM and STM are based on n-grams (Pearce and Wiggins, 2004). Contrary to our expectations, and the evidence from similar experiments in previous work by Pearce (2005), it was found that the LTM+ performed worse than the LTM. These results are summarised in Table 8.1. It is clear that further work is required in this regard, firstly to improve the performance of the STM, and secondly to verify whether the LTM+ can indeed be made to perform better than the LTM and if not why this is so. Both the negative results can be attributed to poor online learning in the RTDRBM,

Model	Cross Entropy
RTDRBM (LTM)	2.712
RTDRBM (LTM+)	2.756
n-gram (LTM)	2.878
n-gram (LTM+)	2.614

Model	Cross Entropy
RTDRBM (STM)	3.363
n-gram (STM)	3.147

Table 8.1 Predictive performance of the LTM and LTM+ (left) and the STM (right) implemented using the RTDRBM, and their comparison with the corresponding best LTM, LTM+ and STM in the work of (Pearce and Wiggins, 2004). As one will observe, both the LTM+ and the STM which are updated during prediction fall short of achieving a performance competitive to the n-gram models as the case is with the LTM.

which is the issue that is to be addressed here. It was surprising to observe these results despite adhering to the methodology adopted in (Mikolov and Zweig, 2012) where the application (language modelling) bears a great resemblance to the one here and the results were positive. One difference, however, between the two is in the use of the Backpropagation Through Time (BPTT) algorithm. While (Mikolov and Zweig, 2012) employed a variant of the BPTT algorithm, known as truncated BPTT, which limits the length of the past sequence taken into account in updating the model parameters to one event, the present works makes no such simplification and takes into account the longest available sequence until a certain time-step. Whether or not this may be resposible for the difference in results is to be investigated.

8.2.3 Further Experiments with the DRBM

Section 7.2.2 introduced theory that facilitated the extension of the LogSigU-DRBM, originally proposed in (Larochelle and Bengio, 2008) to three new variants, namely the TanHU-DRBM, the BinU-DRBM and the ReLU-DRBM. The first two variants were evaluated on the benchmark MNIST datasets, and found to perform on par with the original LogSigU-DRBM. It remains to be seen whether there exist any applications where these variants can outperform the LogSigU-DRBM. For example, the efficacy of the RBMrate proposed in (Teh and Hinton, 2001) which inspired the BinU-DRBM was demonstrated on the FERET face recognition database (Phillips et al., 1998). Alternatively, it may also be investigated whether it can be theoretically proven that the equivalence in performance is indeed something to be expected. Moreover, the ReLU-DRBM was not evaluated in this dissertation owing to arithmetic overflow errors that resulted during training. Similar errors encountered in the case of the BinU-DRBM could be resolved by employing a smaller learning rate and enabling weight-decay. The latter, however, resulted in poorer performance. Work is also currently underway to optimise these extensions with other error criteria, namely Average Cross Entropy and Mean Squared Error to compare these with the Negative Log-Likelihood criterion used here.

8.2.4 Folk Melody Classification

Music classification involves assigning one or more class-labels to musical pieces based on their content. A system for music classification can be evaluated according to the accuracy of class-labels assigned to unseen pieces. Such systems are relevant to studies in systematic musicology. In the present work, a set of RBM-based prediction models was employed as a one-vs-all classifier in the classification of 7 different folk-melody styles from the corpus described in Section 3.2. This approach is inspired by previous work using statistical language models for this task (Pérez-Sancho et al., 2008; Conklin, 2013). It also bears a resemblance to the approach in (Hörnel and Menzel, 1998) where a committee of neural networks, each of which has learned a specific harmonisation style, was used to recognise different styles according to how *expected* it is w.r.t. each network.

With one model trained on pitch sequences from each class, a given test melody was assigned to the class whose model returned the lowest average cross entropy value over that melody. As in Chapter 4, each of the models was trained on sequences of lengths ranging from 2 to 9. A grid search was carried out by varying the number of hidden units as $\{100, 200, 400\}$, the weight-decay coefficient as $\{0.0000, 0.0001\}$, and setting the learning rate to 0.01. It was found that the models corresponding to sequence length 6 had the overall best classification performance, the results of which are shown in Table 8.1. The overall classification accuracy is 61.74%. There is a relatively high degree of confusion between classes corresponding to geographically close regions of Europe (Alsace, Yugoslavia, Switzerland, Austria, Germany), suggesting the need to further optimize the models, or add more musical features.

Our approach towards classification of folk melodies has the following intuition. Each model trained on a certain type of folk music can be viewed as an "expert" of that type of music. The analogy here is that of someone familiar with a certain type of music being better equipped to expect how it might evolve in time. This translates into the expectation for a model trained on a certain type of folk melodies to make predictions with lower average cross entropy when presented

	Nova-Scotia	Alsace	Yugoslavia	Switzerland	Austria	Germany	China	Total
Nova-Scotia	117	6	2	2	2	13	10	152
Alsace	8	33	11	7	15	15	2	91
Yugoslavia	15	14	54	9	17	7	3	119
Switzerland	6	9	10	33	22	11	2	93
Austria	5	16	10	14	41	14	4	104
Germany	14	23	10	15	14	132	5	213
China	11	3	2	2	5	1	213	237

Fig. 8.1 Confusion Matrix with results of the folk melody classification task.

with similar melodies. The confusion observed between some of the classes using this approach seems to support this intuition.

This work is interesting from a musicological perspective. A system for music classification can be used for assigning style/genre labels to unlabelled pieces of music based on their musical content (Conklin, 2013). And as observed in the above preliminary result, it can also offer interesting insights into the regional and cultural origins of different styles of music (Tzanetakis, 2014). Given the state of this experiment, two extensions are being considered. Firstly, the inclusion of other musical features (musical intervals, note durations, and so on) can be expected to further improve the classification system by incorporating new information. Furthermore, its comparison with existing systems of folk melody classification such as (Li et al., 2006; Hillewaere et al., 2009; Conklin, 2013) is due.

8.2.5 Multiple Type Prediction

The present work has mainly focused on predicting just one type, namely `pitch`. In order for the model to generate music, i.e. melodies, at least one additional type is of interest. This is either `ioi` (inter-onset interval) and `dur` (note duration) which introduces a sense of rhythm. There are multiple ways of doing this. The most straightforward option would be to take the Cartesian product of the concerned types and predict them with no change to the architecture of the model. Another is to use a different model for each type to be predicted. However, these two approaches do not exploit the benefits of using connectionist architectures. We consider an alternative way of addressing this, through multiple sets of softmax outputs in the model. This is illustrated in Figure 8.2 in the case of a Feed-forward Neural Network (FNN), and can be extended similarly to an RNN. As shown here,

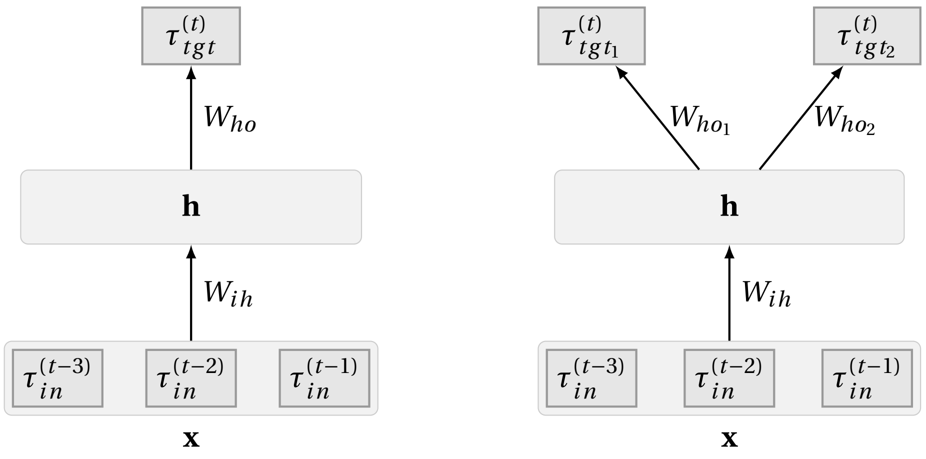


Fig. 8.2 Adaptation of a model predicting a single type `pitch` to one that predicts both `pitch` and `ioi`. It involves the addition of a second set of softmax units in the output layer. This approach can be extended to other types of interest.

subsets of units belonging to the output layer of a network are treated as independent sets of softmax units. This does not alter the learning procedure of the model in any way for FNNs or RNNs. This can be considered to be an instance of Multi-Task Learning (MTL) (Caruana, 1997) where a single network is used to address two different tasks, one of predicting types τ_{tgt_1} and τ_{tgt_2} simultaneously, by using a shared intermediate representation in its hidden layer that suits both tasks. In theory, this is expected to improve predictive performance in both tasks when there is useful information that can be transferred between them. The extension of this idea to the RBM-based models is not as simple, since such an extension would still involve carrying out inference over the same number of possibilities as the product of the domains of the target types involved. Thus, we don't comment on it here.

We carried out an initial experiment with the FNN in this direction. by looking into the simultaneous prediction of the types `pitch` and `ioi`. We compared three networks, all of which take as inputs the types `pitch` and `ioi`, and predict (1) the type `pitch` (2) the type `ioi`, and (3) both `pitch` and `ioi` respectively. The evaluation was carried out only on the Chinese folk melody subset of the folk and chorale melody corpus introduced in Section 3.2. The context length of the models was varied as $\{1,2,3,4,5,6,7,8\}$, the number of hidden units as $\{25,50,100,200\}$, weight decay regularisation coefficient as $\{0.0000,0.0001\}$. The models were trained up to a maximum of 1000 epochs with early-stopping enabled. In the case of the MTL model, the overall performance was the equally-weighted average of the cross-entropies of its individual constituent models during training. This influences the

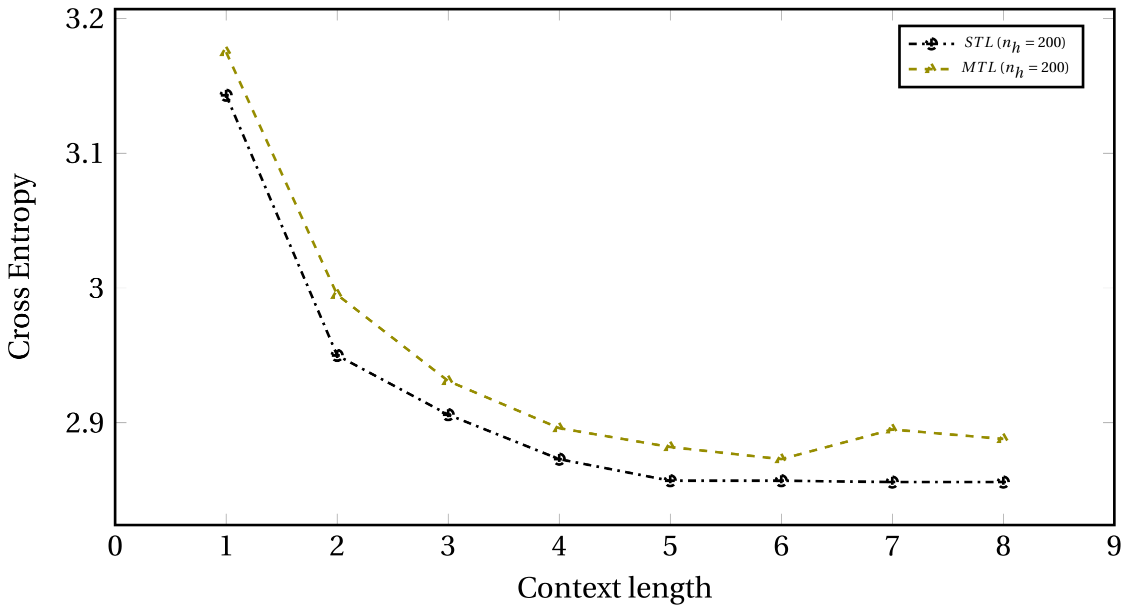


Fig. 8.3 A plot comparing the performance in predicting the type `pitch` by a model predicting only this type (STL model), and another predicting both `pitch` and `ioi` (MTL model).

computation of the error gradient in the model in order to update its parameters. The results are illustrated in Figures 8.3 and 8.4 respectively.

The performance of the MTL model in predicting each of the two types is slightly worse than that of the corresponding STL (Single-Task Learning) models. This is contrary to our initial expectation. However, they are indeed close and exhibit a similar trend, which we view this as an encouraging first step. It must be noted that the results of this experiment were meant to serve as only a feasibility test for the MTL model, which in our opinion has been successful. There is still scope for optimising the performance of the models. Firstly, it is possible to experiment with different schemes to weight the predictions of the two (or in general, more) predictors that are contained in the MTL model. This could even be dynamic, and adjusted during training based on the performance of the model on a validation set. Furthermore, the early-stopping schedule employed here, when used elsewhere in other experiments, was found to be too conservative, thus leading to sub-optimal performance in those experiments, which led to the schedule employed in Chapters 4 and 5. We suspect that re-running this experiments on this and other datasets with these factors (and possibly others not mentioned here) taken into account might lead to different observations, and leave this to be explored in the future.

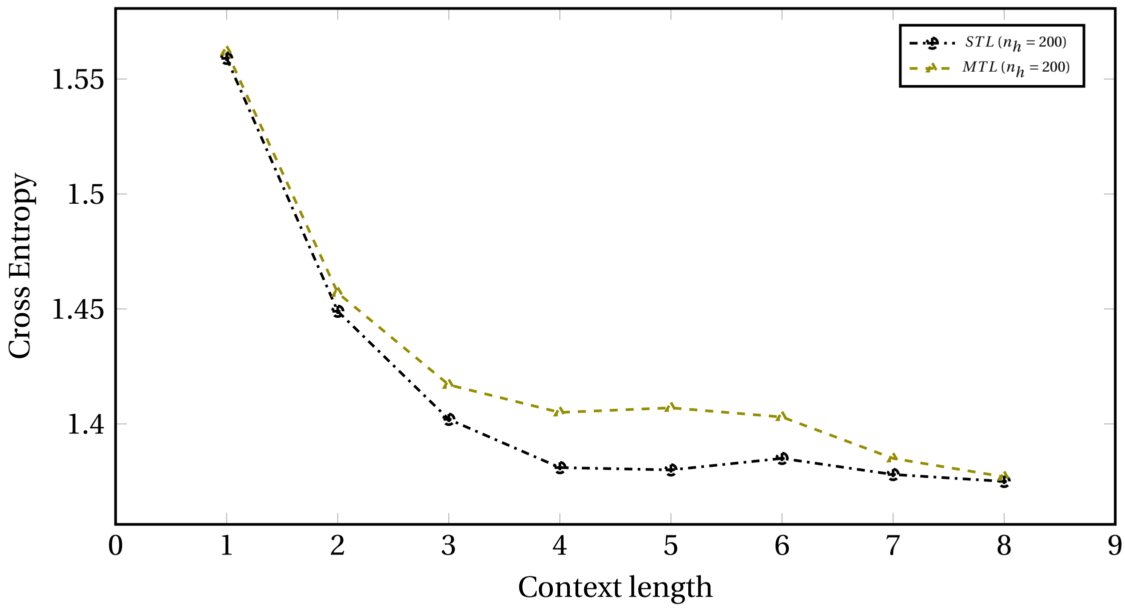


Fig. 8.4 A plot comparing the performance in predicting the type `ioi` by a model predicting only this type (STL model), and another predicting both `pitch` and `ioi` (MTL model).

8.3 Reflections

The work presented in this thesis demonstrates how application-driven research can lead to valuable insights at the level of more basic research that it is based on. In particular, what started here with a study of models purely aimed at musical pitch prediction in symbolic music data led to the development of a novel machine learning architecture for sequence labelling in general and its application demonstrated in the context of handwritten character recognition. Work improving the proposed architecture and its extension to other application domains is currently under way.

The results of the present work mirror the success of recurrent connectionist models in a wide range of applications, most notably speech recognition (Graves et al., 2013), natural language processing (Mikolov et al., 2010), temporal classification (Graves, 2012) and game AI (Silver et al., 2016). Given this success and the high level of research activity surrounding these models, one can only expect their use to be more widespread in the future and their success even greater. Of particular interest to the author are the many variants of the Long Short Term Memory (LSTM) networks and Neural Turing Machines (Graves et al., 2014) which have been shown to successfully overcome the memory limitations of traditional recurrent neural networks used here.

And while music data in audio form is the current the mainstay of the music technology industry which is driven by music streaming and recommendation services, the future is nevertheless bright for musical information available in the symbolic form as well as models that analyse this information. AI-based music creation (or computational musical creativity) and music education are two important applications expected to drive the interest in the latter. It is the author's hope that the research carried out as a part of this thesis and its results will be considered valuable in these applications.

References

Ambience music solutions. http://www.ambsol.net/AMS/. Accessed: 2015-12-04.

Holdcom. http://www.holdcom.com/business-music-services/. Accessed: 2013-01-22.

Jukedeck. http://www.jukedeck.com. Accessed: 2015-12-04.

Makemusic, inc. http://www.makemusic.com. Accessed: 2015-12-04.

Mood media. http://www.moodmedia.com/. Accessed: 2013-01-22.

Rapidcomposer. http://www.musicdevelopments.com/rapidcomposer.html. Accessed: 2013-01-22.

Moray Allan and Christopher KI Williams. Harmonising Chorales by Probabilistic Inference. *Advances in Neural Information Processing Systems*, 17:25–32, 2004.

Charles Ames. The Markov Process as a Compositional Model: A Survey and Tutorial. *Leonardo*, 22(2):175–187, 1989.

Gérard Assayag, Shlomo Dubnov, and Olivier Delerue. Guessing the Composer's Mind: Applying Universal Prediction to Musical Style. In *International Computer Music Conference*, pages 496–499, 1999.

Eric Battenberg and David Wessel. Analyzing Drum Patterns using Conditional Deep Belief Networks. In *International Society for Music Information Retrieval Conference*, 2012.

Ron Begleiter, Ran El-Yaniv, and Golan Yona. On Prediction using Variable Order Markov Models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

Timothy Bell, Ian H Witten, and John G Cleary. Modeling for Text Compression. *ACM Computing Surveys*, 21(4):557–591, 1989.

Matthew I. Bellgard and C P Tsang. Harmonizing Music the Boltzmann Way. *Connection Science*, 6(2):281–297, 1994.

Pierfrancesco Bellini, Paolo Nesi, and Giorgio Zoia. Symbolic Music Representation in MPEG. *IEEE MultiMedia*, (4):42–49, 2005.

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

Yoshua Bengio. Practical Recommendations for Gradient-based Training of Deep Architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy Layer-wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19:153, 2007.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

Greg Bickerman, Sam Bosley, Peter Swire, and Robert Keller. Learning to Create Jazz Melodies using Deep Belief Nets. In *International Conference On Computational Creativity*, 2010.

John Biles. GenJam: A Genetic Algorithm for Generating Jazz Solos. In *International Computer Music Conference*, pages 131–131, 1994.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Christopher M Bishop and Julia Lasserre. Generative or Discriminative? Getting the Best of Both Worlds. *Bayesian Statistics*, 8:3–24, 2007.

Léon Bottou. Stochastic Learning. In *Advanced Lectures on Machine Learning*, pages 146–168. Springer, 2004.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences : Application to Polyphonic Music Generation and Transcription. In *International Conference on Machine Learning*, 2012.

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

Daniel Brown. Mezzo: An Adaptive, Real-time Composition Program for Game Soundtracks. In *AIIDE Workshop on Musical Metacreativity*, 2012.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479, 1992.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Pietro Casella and Ana Paiva. MAgentA: An Architecture for Real Time Automatic Composition of Background Music. In *Intelligent Virtual Agents*, pages 224–232. Springer, 2001.

Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing Multiple Parameters for Support Vector Machines. *Machine learning*, 46(1-3):131–159, 2002.

Stanley F Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

Stanley F Chen, Douglas Beeferman, and Roni Rosenfield. Evaluation Metrics for Language Models. 1998.

Tianqi Chen, Sameer Singh, Ben Taskar, and Carlos Guestrin. Efficient Second-Order Gradient Boosting for Conditional Random Fields. In *International Conference on Artificial Intelligence and Statistics*, 2015.

Srikanth Cherla, Hendrik Purwins, and Marco Marchini. Automatic phrase continuation from guitar and bass guitar melodies. *Computer Music Journal*, 37(3): 68–81, 2013.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Don D Coffman. Measuring Musical Originality using Information Theory. *Psychology of Music*, 20(2):154–161, 1992.

Joel E Cohen. Information theory and music. *Behavioral Science*, 7(2):137–163, 1962.

Michael Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *ACL Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, 2002.

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, 2011.

Darrell Conklin. Prediction and entropy of music. 1990.

Darrell Conklin. Representation and Discovery of Vertical Patterns in Music. In *Music and Artificial Intelligence*, pages 32–42. Springer, 2002.

Darrell Conklin. Multiple Viewpoint Systems for Music Classification. *Journal of New Music Research*, 42(1):19–26, 2013.

Darrell Conklin and John G Cleary. Modelling and generating music using multiple viewpoints. 1988.

Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

Edgar Coons and David Kraehenbuehl. Information as a Measure of Structure in Music. *Journal of Music Theory*, pages 127–161, 1958.

David Cope. *Experiments in musical intelligence*, volume 12. AR Editions Madison, WI, 1996.

Greg Cox. On the Relationship Between Entropy and Meaning in Music: An Exploration with Recurrent Neural Networks. In *Annual Conference of the Cognitive Science Society*, pages 429–434, 2010.

Koby Crammer and Yoram Singer. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

Helen Creighton. *Songs and ballads from nova scotia*, volume 1703. General Pub., 1966.

Michael Scott Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*, pages 637–642, 2010.

George E Dahl, Ryan P Adams, and Hugo Larochelle. Training Restricted Boltzmann Machines on Word Observations. In *International Conference on Machine Learning*, pages 679–686, 2012.

Rachel Darnley-Smith and Helen M Patey. *Music therapy*. Sage, 2003.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based Structured Prediction. *Machine learning*, 75(3):297–325, 2009.

Thomas G Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

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

Christian Dittmar, Estefanía Cano, Jakob Abeßer, and Sascha Grollmisch. Music information retrieval meets music education. *Multimodal Music Processing*, 3: 95–120, 2012.

Trinh Do, Thierry Arti, et al. Neural Conditional Random Fields. In *International Conference on Artificial Intelligence and Statistics*, pages 177–184, 2010.

Kemal Ebcioğlu. An Expert System for Harmonizing Four-Part Chorales. *Computer Music Journal*, pages 43–51, 1988.

Douglas Eck and Juergen Schmidhuber. Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756. IEEE, 2002.

Hauke Egermann, Marcus T Pearce, Geraint A Wiggins, and Stephen McAdams. Probabilistic Models of Expectation Violation Predict Psychophysiological Emotional Responses to Live Concert Music. *Cognitive, Affective, & Behavioral Neuroscience*, pages 1–21, 2013.

Jeffrey L Elman. Finding Structure in Time. *Cognitive science*, 14(2):179–211, 1990.

J Erkkilä, O Lartillot, G Luck, K Riikkilä, and P Toiviainen. Intelligent Music Systems in Music Therapy. *Music Therapy Today*, 5(5), 2004.

Johannes Feulner and Dominik Hörnel. MELONET: Neural Networks that Learn Harmony-based Melodic Variations. In *Proceedings of the International Computer Music Conference*, pages 121–121. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994.

Judy A Franklin. Recurrent Neural Networks for Music Computation. *INFORMS Journal on Computing*, 18(3):321–338, 2006.

Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A Survey of Audio-Based Music Classification and Annotation. *Multimedia, IEEE Transactions on*, 13(2):303–319, 2011.

Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic Music Generation by Modeling Temporal Dependencies using a RNN-DBN. In *Artificial Neural Networks and Machine Learning–ICANN 2014*, pages 217–224. Springer, 2014.

Michael Good. MusicXML for Notation and Analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001.

VK Govindan and AP Shivaprasad. Character Recognition — A Review. *Pattern recognition*, 23(7):671–683, 1990.

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

Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385. Springer, 2012.

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

Niall Griffith and Peter M Todd. *Musical Networks: Parallel Distributed Perception and Performance.* MIT Press, 1999.

James Hendler. Avoiding Another AI Winter. *IEEE Intelligent Systems*, 2(23):2–4, 2008.

Walter B Hewlett. MuseData: Multipurpose Representation. In *Beyond MIDI*, pages 402–447. MIT Press, 1997.

Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A Neural Net for Harmonizing Chorales in the Style of JS Bach. In *Advances in Neural Information Processing Systems*, pages 267–274, 1992.

Lejaren Hiller and Calvert Bean. Information Theory Analyses of Four Sonata Expositions. *Journal of Music Theory*, pages 96–137, 1966.

Lejaren Hiller and Ramon Fuller. Structure and Information in Webern's Symphonie, Op. 21. *Journal of Music Theory*, pages 60–115, 1967.

Lejaren A Hiller and Leonard M Isaacson. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

Lejaren A Hiller and Leonard M Isaacson. *Experimental Music; Composition with an electronic computer.* Greenwood Publishing Group Inc., 1979.

Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. Global Feature Versus Event Models for Folk Song Classification. In *ISMIR*, volume 2009, page 10th. Citeseer, 2009.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

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. A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade*, pages 599–619. Springer, 2012.

Geoffrey E Hinton, Terrence J Sejnowski, and David H Ackley. *Boltzmann Machines: Constraint Satisfaction Networks that Learn.* Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

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

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

Dominik Hörnel and Wolfram Menzel. Learning Musical Structure and Style with Neural Networks. *Computer Music Journal (CMJ)*, 22(4):44–62, 1998.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

Gordon P Hughes. On the Mean Accuracy of Statistical Pattern Recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, 1968.

Eric J Humphrey, Juan Pablo Bello, and Yann LeCun. Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics. In *International Society for Music Information Retrieval Conference*, pages 403–408, 2012.

David Huron. Humdrum and Kern: Selective Feature Encoding. In *Beyond MIDI*, pages 375–401. MIT Press, 1997.

David Huron. Music Information Processing using the Humdrum Toolkit: Concepts, Examples, and Lessons. *Computer Music Journal*, 26(2):11–26, 2002.

David Brian Huron. *Sweet anticipation: Music and the psychology of expectation.* MIT press, 2006.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential Model-based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

Michael I Jordan. Serial Order: A Parallel Distributed Processing Approach. Technical report, Institute for Cognitive Science, University of California San Diego, 1986.

James M Joyce. Kullback-leibler Divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

Timothy Justus and Jamshed Bharucha. Modularity in Musical Processing: The Automaticity of Harmonic Priming. *Journal of Experimental Psychology: Human Perception and Performance*, 27(4):1000–1011, 2001.

Robert H Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition.* PhD thesis, Massachusetts Institute of Technology, 1995.

Robert M Keller and David R Morrison. A Grammatical Approach to Automatic Improvisation. In *Sound and Music Computing Conference*, pages 11–13, 2007.

Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription.* Springer Science & Business Media, 2007.

Leon Knopoff and William Hutchinson. Information Theory for Musical Continua. *Journal of Music Theory*, pages 17–44, 1981.

Leon Knopoff and William Hutchinson. Entropy as a Measure of Style: The Influence of Sample Length. *Journal of Music Theory*, pages 75–97, 1983.

Ron Kohavi et al. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

David Kraehenbuehl and Edgar Coons. Information as a Measure of the Experience of Music. *Journal of Aesthetics and Art Criticism*, pages 510–522, 1959.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, volume 1, page 4, 2012.

Roland Kuhn and Renato De Mori. A Cache-based Natural Language Model for Speech Recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(6):570–583, 1990.

Ludmila I Kuncheva and Christopher J Whitaker. Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy. *Machine learning*, 51(2):181–207, 2003.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*, pages 282–289, 2001.

Andrew Lambert, Tillman Weyde, and Newton Armstrong. Beyond the Beat: Towards Metre, Rhythm and Melody Modelling with Hybrid Oscillator Networks. In *Joint 40th International Computer Music Conference and 11th Sound & Music Computing conference*, Athens, Greece, 2014a.

Andrew Lambert, Tillman Weyde, and Newton Armstrong. Studying the Effect of Metre Perception on Rhythm and Melody Modelling with LSTMs. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014b.

Andrew J. Lambert, Tillman Weyde, and Newton Armstrong. Perceiving and Predicting Expressive Rhythm with Recurrent Neural Networks. In *12th Sound & Music Computing conference*, Maynooth, Ireland, 2015.

Michael Z Land and Peter N McConnell. Method and Apparatus for Dynamically Composing Music and Sound Effects using a Computer Entertainment System, May 24 1994. US Patent 5,315,057.

Edward W. Large, Felix V. Almonte, and Marc J. Velasco. A Canonical Model for Gradient Frequency Neural Networks. *Physica D: Nonlinear Phenomena*, 239 (12):905–911, June 2010.

Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted Boltzmann machines. In *International Conference on Machine Learning*, pages 536–543. ACM Press, 2008.

Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. Learning Algorithms for the Classification Restricted Boltzmann Machine. *The Journal of Machine Learning Research*, 13(1):643–669, 2012.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 2006.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, (521): 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.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616. ACM, 2009.

Xiao Li, Gang Ji, and Jeff Bilmes. A Factored Language Model of Quantized Pitch and Duration. In *International Computer Music Conference*, pages 556–563, 2006.

Qi Lyu, Zhiyong Wu, and Jun Zhu. Polyphonic Music Modelling with LSTM-RTRBM. In *ACM Conference on Multimedia*, pages 991–994. ACM, 2015.

Christopher D Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT press, 1999.

Marco Marchini and Hendrik Purwins. Unsupervised Generation of Percussion Sound Sequences from a Sound Example. In *Sound and Music Computing Conference*, 2010.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

James Martens and Ilya Sutskever. Learning Recurrent Neural Networks with Hessian-free Optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040, 2011.

David A Medler. A Brief History of Connectionism. *Neural Computing Surveys*, 1: 18–72, 1998.

Leonard B Meyer. *Emotion and Meaning in Music*. University of Chicago Press, 1956.

Leonard B Meyer. Meaning in music and information theory. *The Journal of Aesthetics and Art Criticism*, 15(4):412–424, 1957.

Tomas Mikolov and Geoffrey Zweig. Context Dependent Recurrent Neural Network Language Model. In *SLT*, pages 234–239, 2012.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent Neural Network based Language Model. In *INTERSPEECH*, pages 1045–1048, 2010.

George A Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

Tom M Mitchell. Generalization as Search. *Artificial intelligence*, 18(2):203–226, 1982.

Roni Mittelman, Benjamin Kuipers, Silvio Savarese, and Honglak Lee. Structured Recurrent Temporal Restricted Boltzmann Machines. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1647–1655, 2014.

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2008.

Alistair Moffat, Radford Neal, and Ian Witten. Arithmetic Coding Revisited. *ACM Transactions on Information Systems (TOIS)*, 16(3):256–294, 1998.

Francisco Moreno-Seco, José M Inesta, Pedro J Ponce De León, and Luisa Micó. Comparison of classifier fusion methods for classification in pattern recognition tasks. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 705–713. Springer, 2006.

Frederic Morin and Yoshua Bengio. Hierarchical Probabilistic Neural Network Language Model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

Michael C Mozer. Connectionist music composition based on melodic, stylistic and psychophysical constraints. *Music and Connectionism*, pages 195–211, 1991.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective.* MIT press, 2012.

Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Eugene Narmour. *The analysis and cognition of melodic complexity: The implication-realization model.* University of Chicago Press, 1992.

Andrew Y Ng and Michael I Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems*, pages 841–848, 2001.

Nam Nguyen and Yunsong Guo. Comparisons of Sequence Labeling Algorithms and Extensions. In *International Conference on Machine Learning*, pages 681–688, 2007.

Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a System for Automated Music Engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, volume 1. Citeseer, 2003.

Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation.* Springer, 2009.

Harry Ferdinand Olson. *Music, Physics and Engineering*, chapter 9, page 340. Courier Corporation, 1967.

Diana Omigie, Marcus T Pearce, Victoria J Williamson, and Lauren Stewart. Electrophysiological correlates of melodic processing in congenital amusia. *Neuropsychologia*, 2013.

David Opitz and Richard Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, pages 169–198, 1999.

Nicola Orio. *Music Retrieval: A Tutorial and Review*, volume 1. Now Publishers Inc., 2006a.

Nicola Orio. *Music Retrieval: A Tutorial and Review*, volume 1, chapter 2, pages 21–22. Now Publishers Inc., 2006b.

Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

Jean-Francois Paiement. *Probabilistic Models for Music.* PhD thesis, Ecole Polytechnique Federale de Lausanne, 2008.

Douglas B Paul and Janet M Baker. The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

Marcus Pearce. Early Applications of Information Theory to Music. http://webprojects.eecs.qmul.ac.uk/marcusp/notes/music-information-theory.pdf, 2007. Last accessed: 2016-02-12.

Marcus Pearce and Geraint Wiggins. Towards a Framework for the Evaluation of Machine Compositions. In *Symposium on Artificial Intelligence and Creativity in the Arts and Sciences (AISB)*, pages 22–32, 2001.

Marcus Pearce and Geraint Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.

Marcus T Pearce and Geraint A Wiggins. Expectation in melody: The influence of context and learning. *Music Perception*, 23(5):377–405, June 2006.

Marcus Thomas Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, City University London, 2005.

Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4): 963–979, 2006.

Carlos Pérez-Sancho, David Rizo, and José M Iñesta. Stochastic Text Models for Music Categorization. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 55–64. Springer, 2008.

P Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J Rauss. The FERET Database and Evaluation Procedure for Face-Recognition Algorithms. *Image and vision computing*, 16(5):295–306, 1998.

Boris Teodorovich Polyak. Some Methods of Speeding Up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, 4 (5):1–17, 1964.

Keith Potter, Geraint A Wiggins, and Marcus T Pearce. Towards Greater Objectivity in Music Theory: Information-dynamic Analysis of Minimalist Music. *Musicae Scientiae*, 11(2):295–324, 2007.

Lutz Prechelt. Early Stopping — But When? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Albert Riemenschneider. 371 harmonized chorales and 69 chorale melodies with figured bass. *New York: G. Schirmer*, 1941.

Martin A Rohrmeier and Stefan Koelsch. Predictive Information Processing in Music Cognition. A Critical Review. *International Journal of Psychophysiology*, 83 (2):164–175, 2012.

Perry Roland. The Music Encoding Initiative (MEI). In *International Conference on Musical Applications*, volume 1060, pages 55–59, 2002.

Dana Ron, Yoram Singer, and Naftali Tishby. The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length. *Machine learning*, 25(2-3):117–149, 1996.

Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65(6):386, 1958.

David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* MIT Press, 1986.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning Representations by Back-Propagating Errors. *Cognitive Modeling*, 5, 1988.

Matti P Ryynänen and Anssi P Klapuri. Automatic Rranscription of Melody, Bass Line, and Chords in Polyphonic Music. *Computer Music Journal*, 32(3):72–86, 2008.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. In *International Conference on Machine Learning*, pages 791–798. ACM, 2007.

Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek. Automatic Genre Classification of Music Content: A Survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, 2006.

Helmut Schaffrath and D Huron. The essen folksong collection in the humdrum kern format. *Menlo Park, CA: Center for Computer Assisted Research in the Humanities*, 1995.

Holger Schwenk and Jean-Luc Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–765. IEEE, 2002.

Alan Senior, Georg Heigold, Marc'Aurelio Ranzato, and Ke Yang. An Empirical Study of Learning Rates in Deep Neural Networks for Speech Recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6724–6728. IEEE, 2013.

Joan Serra, Emilia Gómez, and Perfecto Herrera. Audio Cover Song Identification and Similarity: Background, Approaches, Evaluation, and Beyond. In *Advances in Music Information Retrieval*, pages 307–332. Springer, 2010.

Claude E Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Claude E Shannon. Prediction and Entropy of Printed English. *Bell system technical journal*, 30(1):50–64, 1951.

Roger N Shepard. Geometrical Approximations to the Structure of Musical Pitch. *Psychological Review*, 89(4):305, 1982.

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 *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

Paul Smolensky. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, pages 194–281. MIT Press, 1986.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136, 2011.

Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

Athina Spiliopoulou and Amos Storkey. Comparing probabilistic models for melodic sequences. In *Machine Learning and Knowledge Discovery in Databases*, pages 289–304. Springer, 2011.

Ilya Sutskever and Geoffrey E Hinton. Learning Multilevel Distributed Representations for High-dimensional Sequences. In *International Conference on Artificial Intelligence and Statistics*, pages 548–555, 2007.

Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. The Recurrent Temporal Restricted Boltzmann Machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-Margin Markov Networks. *Advances in Neural Information Processing Systems*, 16:25, 2004.

David Tax, Martijn Van Breukelen, Robert Duin, and Josef Kittler. Combining Multiple Classifiers by Averaging or by Multiplying? *Pattern Recognition*, 33(9):1475–1485, 2000.

Graham W Taylor and Geoffrey E Hinton. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *International Conference on Machine Learning*, pages 1025–1032. ACM, 2009.

Graham W Taylor, Geoffrey E Hinton, and Sam Roweis. Modeling Human Motion using Binary Latent Variables. In *Advances in Neural Information Processing Systems*, pages 1345–1352, 2007.

Yee Whye Teh and Geoffrey E Hinton. Rate-Coded Restricted Boltzmann Machines for Face Recognition. *Advances in Neural Information Processing Systems*, pages 908–914, 2001.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning*, pages 1064–1071. ACM, 2008.

Peter M Todd. A Connectionist Approach to Algorithmic Composition. *Computer Music Journal*, 13(4):27–43, 1989.

Peter M Todd and D Gareth Loy. *Music and Connectionism.* Mit Press, 1991.

Petri Toiviainen. Modeling the target-note technique of bebop-style jazz improvisation: An artificial neural network approach. *Music Perception*, pages 399–413, 1995.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large Margin Methods for Structured and Interdependent Output Variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

George Tzanetakis. Computational Ethnomusicology: A Music Information Retrieval Perspective. In *International Computer Music Conference*, pages 69–74, 2014.

Giorgio Valentini and Francesco Masulli. Ensembles of Learning Machines. In *Neural Nets*, pages 3–20. Springer, 2002.

Vladimir Naumovich Vapnik and Samuel Kotz. *Estimation of Dependences based on Empirical Data*, volume 40. Springer-verlag New York, 1982.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

Chris Walshaw. The ABC Music Notation. http://www.abcnotation.com. Accessed: 2015-12-04.

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*, 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.

Tillman Weyde and Klaus Dalinghaus. Design and optimization of neuro-fuzzy-based recognition of musical rhythm patterns. *Journal of Smart Engineering Design*, 5(2):67–79, 2003.

Raymond Peter Whorley et al. *The Construction and Evaluation of Statistical Models of Melody and Harmony*. PhD thesis, 2013.

Geraint A Wiggins, P Rebuschat, M Rohrmeier, JA Hawkins, and I Cross. Computer Models of (Music) Cognition. *Language and Music as Cognitive Systems*, pages 169–188, 2011.

Ronald J Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989.

Joseph E Youngblood. Style as Information. *Journal of Music Theory*, pages 24–35, 1958.

Matthew D Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On Rectified Linear Units for Speech Processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling Neural Networks: Many Could Be Better than All. *Artificial intelligence*, 137(1):239–263, 2002.

Appendix A

Detailed Results for Chapters 4 and 5

This appendix contains further details of the results of the musical pitch prediction task on which the non-recurrent and recurrent connectionist models were evaluated in Chapters 4 and 5. In those chapters, the prediction scores (cross-entropy) of the various models averaged across all 8 datasets of the melody corpus described in Section 3.2 were presented. The prediction scores specific to each dataset are listed here.

A.1 Feed-forward Neural Network

In chapter 4, feed-forward neural networks (FNNs) with three different hidden layer activation types were examined, namely those with logistic sigmoid (LogSig), hyperbolic tangent (TanH), and rectified linear (ReL) activations. The prediction scores of the three models, averaged across the 8 datasets of the folk and melody corpus were summarised in Figure 4.5. This section lists the dataset-specific prediction scores for the three models. Table A.1 contains the results corresponding to the network using logistic sigmoid hidden unit activations with 200 hidden units, Table A.2 those of the network using hyperbolic tangent hidden unit activations with 50 hidden units, and Table A.3 those of the network using rectified linear unit activations with 200 hidden units.

A.2 Restricted Boltzmann Machines

In chapter 4, restricted Boltzmann machines (RBMs) learned in two different ways — generatively and discriminatively — were evaluated on the 8 datasets of the folk and chorale melody corpus. This section details the dataset-wise predictive per-

Dataset	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$	$l = 7$	$l = 8$
`jugoslav`	2.934	2.837	2.826	2.806	2.835	2.839	2.836	2.852
`elsass`	3.323	3.236	3.170	3.167	3.146	3.168	3.176	3.160
`schweiz`	3.361	3.267	3.207	3.179	3.156	3.160	3.174	3.169
`oesterrh`	3.618	3.521	3.462	3.460	3.455	3.481	3.462	3.470
`kinder`	2.740	2.643	2.583	2.539	2.517	2.494	2.459	2.460
`nova-scotia`	3.049	2.930	2.875	2.847	2.830	2.827	2.828	2.839
`bach`	2.702	2.570	2.527	2.485	2.467	2.461	2.473	2.475
`shanxi`	3.195	3.003	2.936	2.898	2.877	2.885	2.871	2.860
Average	3.115	3.001	2.948	2.923	2.910	2.914	2.910	2.911

Table A.1 Prediction cross-entropy of Feed-forward Neural Networks with 200 hidden units, and Logistic Sigmoid activations on each of the 8 datasets of the folk and chorale melody corpus.

Dataset	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$	$l = 7$	$l = 8$
`jugoslav`	2.867	2.766	2.737	2.745	2.749	2.759	2.773	2.792
`elsass`	3.205	3.123	3.065	3.050	3.056	3.078	3.103	3.109
`schweiz`	3.273	3.178	3.099	3.065	3.068	3.071	3.096	3.103
`oesterrh`	3.534	3.458	3.420	3.402	3.395	3.444	3.448	3.476
`kinder`	2.709	2.583	2.496	2.439	2.418	2.398	2.389	2.393
`nova-scotia`	3.013	2.876	2.809	2.779	2.770	2.774	2.784	2.779
`bach`	2.683	2.547	2.496	2.451	2.440	2.437	2.441	2.446
`shanxi`	3.136	2.937	2.865	2.831	2.815	2.824	2.830	2.835
Average	3.052	2.933	2.873	2.845	2.839	2.848	2.858	2.867

Table A.2 Prediction cross-entropy of Feed-forward Neural Networks with 50 hidden units, and Hyperbolic Tangent activations on each of the 8 datasets of the folk and chorale melody corpus.

formance of these two variants of the Restricted Boltzmann Machine (RBM) summarised in Figure 4.5. Table A.4 contains the results corresponding to the generative RBM containing 100 units in its hidden layer, and Table A.5 those of the discriminative RBM containing 25 hidden units

A.3 Recurrent Neural Networks

In chapter 5, recurrent neural networks (RNNs) with three different hidden layer activation types were examined, namely those with logistic sigmoid (LogSig), hyperbolic tangent (TanH), and rectified linear (ReL) activations. The prediction score of the best of these was listed in Table 5.1. This section lists the dataset-specific prediction scores for these three models.

Dataset	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$
`jugoslav`	2.862	2.737	2.711	2.707	2.715	2.740	2.741	2.777
`elsass`	3.216	3.113	3.060	3.050	3.054	3.098	3.124	3.123
`schweiz`	3.278	3.144	3.078	3.029	3.035	3.023	3.064	3.057
`oesterrh`	3.536	3.413	3.376	3.379	3.365	3.422	3.410	3.476
`kinder`	2.708	2.545	2.444	2.390	2.389	2.369	2.373	2.384
`nova-scotia`	3.010	2.832	2.770	2.758	2.750	2.757	2.767	2.789
`bach`	2.682	2.516	2.454	2.421	2.420	2.409	2.425	2.442
`shanxi`	3.137	2.917	2.856	2.832	2.827	2.841	2.852	2.841
Average	3.054	2.902	2.844	2.821	2.819	2.832	2.845	2.861

Table A.3 Prediction cross-entropy of Feed-forward Neural Networks with 200 hidden units, and Rectified Linear activations on each of the 8 datasets of the folk and chorale melody corpus.

Dataset	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$
`jugoslav`	2.883	2.765	2.708	2.728	2.745	2.738	2.723	2.722
`elsass`	3.211	3.111	3.052	3.050	3.037	3.018	3.005	3.002
`schweiz`	3.290	3.161	3.073	3.041	3.034	3.041	3.040	3.024
`oesterrh`	3.533	3.431	3.379	3.380	3.351	3.338	3.312	3.310
`kinder`	2.727	2.567	2.469	2.420	2.393	2.409	2.375	2.377
`nova-scotia`	3.020	2.856	2.783	2.771	2.761	2.740	2.738	2.727
`bach`	2.690	2.517	2.457	2.423	2.429	2.425	2.429	2.441
`shanxi`	3.149	2.925	2.852	2.829	2.809	2.803	2.798	2.798
Average	3.063	2.917	2.847	2.830	2.820	2.814	2.803	2.799

Table A.4 Prediction cross-entropy of generatively trained Restricted Boltzmann Machines with 100 hidden units, and Logistic Sigmoid hidden layer activations on each of the 8 datasets of the folk and chorale melody corpus.

A.4 Recurrent Temporal Restricted Boltzmann Machine

This section details the dataset-specific predictive performance of the best Recurrent Temporal Restricted Boltzmann Machine on each of the 8 individual datasets of the folk and melody corpus, the average of which was listed in Table 5.1 in Chapter 5.

A.5 Recurrent Temporal Discriminative Restricted Boltzmann Machine

This section details the dataset-specific predictive performance of the best Recurrent Temporal Discriminative Restricted Boltzmann Machine on each of the 8 in-

Dataset	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$
`jugoslav`	2.864	2.744	2.707	2.708	2.705	2.720	2.703	2.707
`elsass`	3.196	3.123	3.042	3.031	3.060	3.069	3.100	3.114
`schweiz`	3.209	3.123	3.045	3.019	3.017	3.029	3.031	3.066
`oesterrh`	3.511	3.435	3.412	3.404	3.418	3.466	3.483	3.499
`kinder`	2.707	2.553	2.439	2.400	2.398	2.377	2.363	2.350
`nova-scotia`	3.011	2.850	2.792	2.763	2.738	2.753	2.745	2.751
`bach`	2.676	2.530	2.476	2.439	2.423	2.433	2.441	2.448
`shanxi`	3.115	2.897	2.846	2.795	2.795	2.802	2.793	2.804
Average	3.063	2.906	2.845	2.820	2.819	2.831	2.832	2.842

Table A.5 Prediction cross-entropy of discriminatively trained Restricted Boltzmann Machines with 25 hidden units, and Logistic Sigmoid hidden layer activations on each of the 8 datasets of the folk and chorale melody corpus.

Dataset	LogSig	TanH	ReL
	$(n_h = 200)$	$(n_h = 50)$	$(n_h = 100)$
`jugoslav`	2.958	2.681	2.757
`elsass`	3.628	2.970	2.992
`schweiz`	3.587	2.988	2.994
`oesterrh`	3.829	3.309	3.278
`kinder`	2.507	2.424	2.383
`nova-scotia`	3.180	2.679	2.665
`bach`	2.599	2.451	2.440
`shanxi`	2.876	2.792	2.786
Average	3.146	2.787	2.787

Table A.6 Prediction cross-entropy of the best cases of Recurrent Neural Networks with three different types of hidden-layer activations, on each of the 8 datasets of the folk and chorale melody corpus.

dividual datasets of the folk and melody corpus, the average of which was listed in Table 5.1 in Chapter 5.

Dataset	LogSig ($n_h = 100$)
jugoslav	2.676
elsass	2.945
schweiz	2.961
oesterrh	3.296
kinder	2.313
nova-scotia	2.635
bach	2.378
shanxi	2.703
Average	2.738

Table A.7 Prediction cross-entropy of the best cases of Recurrent Temporal Restricted Boltzmann Machine with logistic sigmoid hidden layer activations on each of the 8 datasets of the folk and chorale melody corpus.

Dataset	LogSig ($n_h = 100$)
jugoslav	2.655
elsass	2.897
schweiz	2.932
oesterrh	3.259
kinder	2.301
nova-scotia	2.609
bach	2.362
shanxi	2.685
Average	2.712

Table A.8 Prediction cross-entropy of the best cases of Recurrent Temporal Discriminative Restricted Boltzmann Machine with logistic sigmoid activations on each of the 8 datasets of the folk and chorale melody corpus.

Appendix B

Detailed Results for Chapter 6

This appendix describes the results presented in Chapter 6 in greater detail. These are listed in four tables. The first one (Table B.1) corresponds to the results plotted in Figures 6.2 and 6.3, and is related to the experiments carried out in Section 6.3 to test the effects of adding the *types* `intfref` and `ioi` as inputs to a prediction model that originally relied only on the type `pitch`.

Model	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$	$l = 7$	$l = 8$
Model A	3.123	2.928	2.887	2.853	2.843	2.838	2.816	2.831
Model B	3.120	2.951	2.903	2.854	2.858	2.859	2.850	2.839
Model C	2.905	2.818	2.814	2.790	2.807	2.807	2.800	2.811
Model D	2.916	2.833	2.822	2.802	2.808	2.823	2.810	2.822
Model E_s	2.939	2.803	2.779	2.735	2.767	2.761	2.763	2.757
Model E_p	2.920	2.797	2.774	2.731	2.763	2.758	2.761	2.753

Table B.1 Numerical values corresponding to the plots in Figures 6.2 and 6.3. The reader is referred to Section 6.3.1 for descriptions of the model in each row of the table. The columns correspond to the different context length at which the model was evaluated as it is a variant of the non-recurrent feed-forward neural network.