CITY UNIVERSITY LONDON
EST 1894

City Research Online

STRUCTURING LUTE TABLATURE AND MIDI DATA

MACHINE LEARNING MODELS FOR VOICE SEPARATION IN SYMBOLIC MUSIC REPRESENTATIONS

Reinier Franciscus de Valk

A thesis submitted for the degree of
Doctor of Philosophy

City University London
Department of Computer Science
Music Informatics Research Group

September 2015

CITY UNIVERSITY LONDON

EST 1894

Declaration

The research presented in this thesis was conducted within the Music Informatics Research Group housed at the Department of Computer Science at City University London. It was carried out from October 2011 to March 2015 under the supervision of Dr Tillman Weyde, head of the Music Informatics Research Group, and Professor Tim Crawford, affiliated with the Department of Computing at Goldsmiths College, University of London. In submitting this thesis, I declare the research presented in it to be my original work. Wherever contributions of others are concerned, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Abstract

This thesis concerns the design, development, and implementation of machine learning models for voice separation in two forms of symbolic music representations: lute tablature and MIDI data. Three modelling approaches are described: MA1, a note-level classification approach using a neural network, MA2, a chord-level regression approach using a neural network, and MA3, a chord-level probabilistic approach using a hidden Markov model. Furthermore, three model extensions are presented: backward processing, modelling voice and duration simultaneously, and multi-pass processing using an extended (bidirectional) decision context.

Two datasets are created for model evaluation: a tablature dataset, containing a total of 15 three-voice and four-voice intabulations (lute arrangements of polyphonic vocal works) in a custom-made tablature encoding format, `tab+`, as well as in MIDI format, and a Bach dataset, containing the 45 three-voice and four-voice fugues from Johann Sebastian Bach's *Das wohltemperirte Clavier* (BWV 846–893) in MIDI format. The datasets are made available publicly, as is the software used to implement the models and the framework for training and evaluating them.

The models are evaluated on the datasets in four experiments. The first experiment, where the different modelling approaches are compared, shows that MA1 is the most effective and efficient approach. The second experiment shows that the features are effective, and it demonstrates the importance of the type and amount of context information that is encoded in the feature vectors. The third experiment, which concerns model extension, shows that modelling backward and modelling voice and duration simultaneously do not lead to the hypothesised increase in model performance, but that using a multi-pass bidirectional model does. In the last experiment, where the performance of the models is compared with that of existing state-of-the-art systems for voice separation, it is shown that the models described in this thesis can compete with these systems.

For my mother

Acknowledgements

Armed with an interdisciplinary research proposal that was perhaps a bit heavy on the musicological side, in September 2011 I moved to London, where I was to embark on a research project upon completion of which I would be awarded a doctorate in computer science. Not for a single moment did I doubt that, once I had reached the other end of the journey I was about to begin, and regardless of my newly acquired title, I would continue to consider myself a musicologist. So I immersed myself in the new discipline. Four years and countless hours of modelling, programming, conducting experiments, and writing later, upon submitting the final version of my thesis as you have it before you, I must admit that I am now in fact equally—if not more—comfortable with the title *computer scientist*.

For this conversion I hold responsible my supervisor, Tillman Weyde, whom I would like to thank first. Without his advice and guidance, this thesis would never have seen the light of day. From the innumerable inspiring discussions and brainstorming sessions we had—the door to his office was always open—my research has benefitted greatly, and whenever I strayed away from the path we had set out, he was there to steer me back into the right direction. Moreover, as the head of the Music Informatics Research Group, he succeeded in creating an informal and collegial environment within which it was very pleasant to work.

I also thank my second supervisor, Tim Crawford, for many stimulating meetings about a wide range of topics related directly (and sometimes not-so-directly) to my research, and, especially, for reminding me from time to time not to lose touch with the musicological aspect of my work. I am grateful to him for offering me the opportunity to be part of the Goldsmiths team and work as a research assistant for the ECOLM III project, and, later, for the SLICKMEM project. Both experiences have broadened my horizon

Contents

Tables

Figures

Symbols, abbreviations, and terms

ε margin, Section 3.1.3.2
λ regularisation parameter, Section 5.1.1

acc   accuracy, Section 3.2.2
application mode Section 3.2.4.1
AVC average voice consistency, Section 5.6.2

B (B$'$) bidirectional model as used in MA1, Section 3.1.6.1
bidirectional Section 3.1.6
bwd backward processing mode, Section 3.1.5

C   model as used in MA2, Section 3.1.3.5
c   conflict percentage, Section 5.7.1
cmp completeness, Section 3.2.2
conflict Section 3.3

decision context Section 3.1.6
deviation threshold Section 3.1.2.1
duration class Section 3.1.2.3

error propagation Section 3.2.4.1

fwd   forward processing mode, Section 3.1.5

ground truth feature vector Section 3.1.3.2

H   model as used in MA3, Section 3.1.4.2
half voice assignment Section 3.2.3
HMM hidden Markov model, Section 3.1.1.2

in-voice chord Section 4.2.1

1

Introduction

The representations of music used in the field of music information retrieval (MIR) can be audio-based or symbol-based. Audio representations include recorded sound signals, while the symbolic representations comprise any form of musical notation, text, or any of the "myriad discrete computer encodings" (Downie, 2003, p. 302) such as the Musical Instrument Digital Interface (MIDI) format, the Music Encoding Initiative (MEI) format, the MusicXML format, or the Humdrum `**kern` format.[1] MIR is a genuinely multidisciplinary field, combining areas such as information science, musicology, music theory, music perception and cognition, audio engineering, and computer science (Downie, 2003; Byrd and Crawford, 2002; Orio, 2006). Traditionally, symbolic music representations have been the tools of musicologists, and it is indeed in the musicological areas of MIR—areas relating to music analysis, music theory, music perception and cognition—where they are used most. Although strong in the early years of MIR, the role of symbolic music representations has become less significant over time, and the past decade has seen a clear shift towards signal processing and audio representations. This shift can be ascribed to the constant development of the internet in conjunction with the invention of audio encoding formats such as MP3 (MPEG-1 or MPEG-2 Audio Layer III) and their increasing (online) availability (Orio, 2006; Volk et al., 2011). As a result, the audio engineering areas of MIR have witnessed a considerable growth, while the musicological areas now constitute a clear minority. This shift in focus is one reason why the exploration of a number of complex problems relating to symbolic music representations has stagnated.

One of such complex problems is *voice separation* in symbolic music rep-

---

[1]See `http://www.music-encoding.org`, `http://www.musicxml.com`, and `http://www.music-cog.ohio-state.edu/Humdrum`.

resentations, the subject of this thesis. Voice separation, simply put, is the task of identifying individual melodic lines (*voices*) in a music representation in which these lines are not directly identifiable (a more formal definition is given in Section 2.1.3). Voice separation in symbolic music representations is a research topic that has received relatively little attention hitherto. Only a handful of systems designed for this task exist. All these systems have been reported to perform reasonably well, but in all cases room for improvement is evident. A possible explanation for the abandonment of some existing research and the lack of new initiatives may very well be the fact that voice separation has been recognised as a difficult task even for expert musicians—let alone for a computer (Orio, 2006). An adequate separation into constituent voices, however, is an essential precondition for the successful execution of other complex MIR tasks that can be of direct use for musicological research, such as automatic transcription, pattern retrieval, melody matching and the detection of concordances (identical pieces occurring in more than one source), voice-based content analysis, etc. It is thus well worthwhile revisiting the topic.

In this thesis, the problem of voice separation is tackled using a supervised machine learning approach, where models are trained on example data to learn mappings of notes to voices, and then applied to new data to provide such mappings. The use of machine learning models to tackle the problem of voice separation is still relatively novel. Only in two previous studies have such models been implemented and evaluated; an extended and more systematic approach, however, has hitherto not been undertaken. In this thesis, three new modelling approaches are presented, two of which are neural network-based, while in the third a hidden Markov model is used. The models are specifically designed for voice separation in lute tablature, or, more precisely, sixteenth-century lute intabulations, instrumental arrangements of polyphonic vocal works. However, as a sole focus on this highly specialist symbolic representation would limit their practical applicability significantly, each model is also implemented in an adapted version, to be applied to a much more widely used symbolic representation: MIDI data.

1.1   Motivation

Two questions now come to the fore: why the focus on lute tablature, and why the limitation to sixteenth-century intabulations? The answer to the first question is that there is a gap in musicological research where music written in lute tablature is concerned, which can begin to be bridged with the help of MIR tools. Lute tablature is a *Griffschrift*: a practical notation

that merely prescribes the physical actions a lute player must perform in order to produce sound from the instrument. It provides no direct information about pitch, only limited information about note duration, and, although it was used to notate at times considerably complex polyphonic music, it reveals very little information about polyphonic structure. This renders lute tablature hard to interpret for the majority of musicologists. As a consequence, the corpus of music written in lute tablature, which spans more than two-and-a-half centuries and contains some 860 individual sources harbouring invaluable musicological information, has been largely overlooked in musicological research. In order to unlock this corpus for further research, it must thus be presented in a more familiar format. The traditional musicological solution is transcription into modern music notation, and such work indeed has been ongoing—but it is a time-consuming and specialist undertaking. The situation sketched above is a typical example of an opportunity for interdisciplinary interaction (Neubarth et al., 2011; Honingh et al., 2014): a musicological problem exists, for which MIR research can provide a solution. MIR models for voice separation as presented in this thesis, together with MIR models for key detection, pitch spelling, etc., can be integrated in (interactive) systems for automatic transcription of tablature. Such systems, which enable fast and large-scale transcription, can then provide non-specialists the tools to overcome problems that previously prevented them from researching music written in lute tablature.

The second question in fact consists of two parts: why only *sixteenth-century* lute music, and why only *intabulations*? The reason for the focus on sixteenth-century lute music, or, as it is commonly known, Renaissance lute music, is mostly a practical one. Limits to the scope of the research have to be set, and the choice for Renaissance music is motivated by several factors. These include, but are not limited to, the fact that this subcorpus is (i) large, as the lute was extremely popular in the sixteenth century,[2] (ii) stylistically fairly homogeneous, (iii) mostly highly contrapuntal (that is, the music consists of different voices sounding simultaneously, each of which moves independently of the others and has a strong individual character), and (iv) not complicated by a large number of tuning variants being used, as is the case for later corpora. The reason for the focus on one particular genre, sixteenth-century intabulations, is threefold (and explained in more detail in Section 2.2.2). Apart from the fact that intabulations are representative and challenging, using them facilitates the process of labelling the data. This can

---

[2]To give an impression of the size of the subcorpus: approximately 200 print sources and a similar number of manuscript sources from the sixteenth century are extant today (see Brown, 1965 and Ness and Kolczynski, 2001, respectively).

be achieved relatively quickly, and with minimal need for interpretation, by using the vocal models as polyphonic blueprints.

The research presented in this thesis connects well to ongoing efforts towards digitisation of lute sources. Since its inception in 1999, the Electronic Corpus of Lute Music (ECOLM) project has aimed to make lute music available for research and performance (Lewis et al., 2004). To this end, machine-readable encodings, tablature renderings thereof, images of original sources, and different kinds of metadata have been stored in a database that is accessible through the project website.[3] Recently, research into the application of optical music recognition methods to tablature (Dalitz and Karsten, 2005; Dalitz and Crawford, 2013) has enabled the encoding process, which had hitherto been manual, to be carried out (semi-)automatically. A large—and growing—body of encodings is thus available for research and analysis; among the tools to start tackling this body are models such as the ones presented in this thesis.

1.2 Aims and objectives

In this thesis, the following two-fold research question is explored:

> How can supervised machine learning models be used for voice separation in polyphonic music in symbolic representations, and which modelling approaches are the most effective?

This question is considered in terms of a main aim and four objectives. The main aim underlying the research is to design, implement, and evaluate supervised machine learning models for voice separation in polyphonic music written in lute tablature and polyphonic music in MIDI format. To this end, the following objectives are formulated:

Objective 1 To design and implement models.
1.1 To design three different modelling approaches:
    MA1 A note-level classification approach using a neural network model, where notes are classified into classes representing voices.
    MA2 A chord-level regression approach using a neural network model, where mappings of chords to voices are rated.
    MA3 A chord-level probabilistic approach using a discrete hidden

---

[3]See `http://www.ecolm.org`.

Markov model, where the most likely sequence of mappings of chords to voices is determined.

1.2 To design three model extensions:

    X1 Simultaneous voice and duration modelling.

    X2 Backward processing.

    X3 Multi-pass processing using a bidirectional decision context.

1.3 To define a set of features, numerical representations of properties of notes or chords in their polyphonic context, relevant to the task of voice separation.

1.4 To implement the models, the model extensions, the feature extraction algorithms, and the framework for training and evaluating the models.

Objective 2 To create datasets.

2.1 To create a corpus of sixteenth-century lute intabulations, where each intabulation is represented as a machine-readable encoding of the tablature in plain text format and a set of monophonic MIDI files.

2.2 To create a corpus of music written in (early) modern staff notation —keyboard fugues by Johann Sebastian Bach—, where each fugue is represented as a set of monophonic MIDI files.

Objective 3 To evaluate the models.

3.1 To evaluate, on both datasets, the three modelling approaches.

3.2 To evaluate, on both datasets, the relevance of the features.

3.3 To evaluate, on both datasets, the effect of the model extensions.

3.4 To compare the models' performance with that of existing systems for voice separation.

As is explained in more detail in Section 1.4, the individual objectives together with their sub-objectives are addressed in the successive chapters of this thesis; they thus determine its overall structure.

1.3   Contributions

The key contributions of this thesis to the existing research are:

▶   Three new machine learning models for voice separation in lute tablature and MIDI data:

    ▶   A neural network model for classifying notes into classes

representing voices.

► A neural network model for rating mappings of chords to voices.

► A hidden Markov model for determining the most likely sequence of mappings of chords to voices.

► A set of features relevant to the task of voice separation.

► The implementation as open source software of the models, the three model extensions, the feature extraction algorithms, and the framework for training and evaluating the models.

► `tab+`, an extensible encoding format for rendering lute tablature machine-readable that supports all lute tablature systems.

► Two datasets:

► A tablature dataset, consisting of encodings and sets of monophonic MIDI files.

► A Bach dataset, consisting of sets of monophonic MIDI files. This dataset, which is an adaptation of an existing dataset, can be used as a standardised benchmark for research into voice separation in symbolic representations, which is currently lacking.

Additionally, for each piece in the tablature dataset a transcription in modern music notation is devised.

► Experimental results enabling assessment of the models.

1.4   Thesis overview

Chapter 2 opens with the definition of three key terms in Section 2.1: *voice*, *polyphonic music*, and *voice separation*. Following that, in Section 2.2, the musicological background against which the research is carried out is sketched, where the sixteenth-century lute, its music, and the notational format of this music and the problems it entails, are introduced. Subsequently, in Section 2.3, the computational background to the research is given, where existing research into voice separation in symbolic representations is discussed.

Chapter 3 describes the methodology followed, and addresses Objective 1. First, in Section 3.1, the three modelling approaches (Objective 1.1), the model extensions (Objective 1.2) and the feature set (Objective 1.3) are described. Subsequently, in Section 3.2, the evaluation procedure, the evaluation metrics, and the different evaluation modes are explained. Following that, in Section 3.3, the subject of conflicts and conflict resolution, which is

specific to MA1, is discussed. The chapter closes with a description of the implementation details in Section 3.4.

In Chapter 4, which in its entirety addresses Objective 2, the datasets are described. In Section 4.1, first, the tablature dataset is presented, and the data formats, the data creation, and the internal representation used are discussed. Furthermore, a detailed description of the encoding format devised for this thesis, `tab+`, is given. In Section 4.2, then, the same is done for the Bach dataset.

Chapter 5 deals with model evaluation, and thus covers Objective 3. Sections 5.1 and 5.2, first, describe two preliminary experiments concerning hyperparameter optimisation. In Experiment 1, described in Section 5.3, the three modelling approaches are evaluated and compared (Objective 3.1). In Experiment 2, described in Section 5.4, the relevance of the features is evaluated in two sub-experiments (Objective 3.2). In Experiment 3, described in Section 5.5, the effect of the three model extensions is evaluated in three sub-experiments (Objective 3.3). In Experiment 4, described in Section 5.6, a comparison between the performance of a selection of the models presented in this thesis and the performance of a number of existing voice separation systems is made (Objective 3.4). Chapter 5 is concluded with a discussion of a number of overarching issues that stem from limitations of the modelling approaches, presented in Section 5.7.

In Chapter 6, finally, first a summary of the preceding chapters is given in Section 6.1. Following this, in Section 6.2 conclusions are presented, and in Section 6.3, perspectives for future work are outlined.

2

Background and related work

In this chapter, the background to the research presented in this thesis is outlined. In Section 2.1, which deals with terminology, a number of important recurring terms are defined: *voice*, *polyphonic music*, and *voice separation*. In Section 2.2, then, the musicological background to the research is given. The sixteenth-century lute is introduced, its notation—lute tablature—and the problems it entails with respect to the reflection of polyphonic structure are described, and the rationale behind the focus on intabulations, instrumental arrangements of polyphonic vocal works, is explained. In Section 2.3, lastly, the computational background to the research is provided. The musicological problem described in Section 2.2 is linked to the problem of voice separation, existing research into voice separation in music in symbolic formats is discussed, and systems for voice separation are compared.

2.1   Terminology

There are three important terms, all of which can be considered to be key terms in the research carried out in this thesis: *voice*, *polyphonic music*, and *voice separation*. Before giving any musicological or computational background, these terms must be defined clearly.

2.1.1   Voice

In the field of musicology, when not referring to the human voice, the term *voice* denotes an individual musical line, either sung or played, that "contribute[s] to one or more elements of the music" (Drabkin, 2001, p. 164). Often-used alternatives are *part*, or, less frequently, *voice part*. Cambouropoulos (2008) acknowledges ambiguity of the term, and distinguishes between (i)

9

a music-theoretical meaning, where a voice is defined as a "*mono*phonic sequence of successive nonoverlapping [sic] musical tones" (p. 78), and (ii) a perceptual meaning, where it is defined as a "perceptually independent sequence of notes *or multi-note simultaneities*" (p. 78, emphasis mine). The former meaning corresponds to the musicological definition of the term; in the latter meaning, where a voice is not necessarily monophonic, it is regarded as an equivalent of an *auditory stream*, a phenomenon described in the field of music psychology. The term is coined by Bregman and Campbell (1971), who define an auditory stream as "a sequence of auditory events whose elements are related perceptually to one another, the stream being segregated perceptually from other co-occurring auditory events" (p. 244).

Throughout this thesis, the term *voice* is used in its musicological (or music-theoretical) meaning, denoting a monophonic sequence of notes. Where a sequence of notes or multi-note simultaneities is intended, the term *stream* is preferred.[1]

2.1.2   Polyphonic music

In MIR research, the term *polyphonic music* is generally used to denote any kind of music in which "more sounds are playing at the same time" (Orio, 2006, p. 10), or, more specific, in which "multiple notes sound at a time" (Byrd and Crawford, 2002, p. 255). By this definition, many genres of music—ranging from a Renaissance motet to a twenty-first-century rock song—may be categorised as polyphonic music.

The definition used in this thesis is somewhat more restricted, and remains closer to the musicological notion of what polyphony is (see, for example, Frobenius et al., 2001). Music is considered to be polyphonic if it satisfies three criteria: (i) it must consist of multiple parts, (ii) the individual parts must be of equal importance, and (iii) the individual parts must, at least to some extent, move independently.

2.1.3   Voice separation

Voice separation (also encountered is *voice segregation*) is defined by Cambouropoulos (2008) as "the task of separating a musical work consisting of multi-note sonorities into independent constituent voices" (p. 75). Although a suitable starting point, this definition leaves some room for interpretation. First, it does not specify what *multi-note sonorities* are. A multi-note sonority shall therefore be defined as a multi-note musical event occurring at time

---

[1]Additionally, when referring to others' work, the original terminology is followed.

t, at least one note of which has an onset time that is equal to t. If all the notes have an onset time equal to t, the multi-note sonority is said to be *left-aligned*. From this definition, then, it follows that there can also exist *single-note* sonorities. A single-note sonority shall be defined as a single-note event occurring at time t, the only note of which has an onset time that is equal to t. In polyphonic music, single-note sonorities generally constitute the minority. However, their use, as well as the use of non-left-aligned multi-note sonorities, increases the rhythmic salience of the individual voices, which promotes their independence. Second, Cambouropoulos's definition does not specify whether the term *voice* is intended in its music-theoretical or in its perceptual meaning (see Section 2.1.1).

In this thesis, a slightly elaborated version of Cambouropoulos's definition is therefore used. Voice separation is taken to denote:

> The task of separating a musical work consisting of single-note and multi-note sonorities into independent constituent voices, that is, monophonic sequences of successive non-overlapping notes.

2.2 Musicological background: the lute and its music in the sixteenth century

From its rise in the late fifteenth century to its gradual decline in the second half of the eighteenth century, the lute remained among the most popular solo instruments in Western Europe. Played in social settings ranging from the church to the court to the tavern, throughout all social classes, and by professionals and amateurs alike, a vast and diverse amount of music was written for it. The instrument's historical significance is attested by over 360 printed and over 500 manuscript sources extant today, containing approximately 60,000 compositions (Ness and Kolczynski, 2001).

In this thesis, the focus is on sixteenth-century Renaissance lute music. Figure 2.1 shows a typical Renaissance lute with six *courses*, that is, string pairs, and eight *frets*, pieces of gut string tied around the neck. The vibrating part of a course can be shortened by pressing a finger against a fret; because the frets are placed at semitone intervals, stopping a course at the nth fret thus raises the pitch of the open (that is, unstopped) course by n semitones. As in Figure 2.1, the lute's first (that is, highest-sounding) course was generally single-strung, while the others were doubled, generally tuned in unisons (second and third courses) and octaves (the remaining courses). The double
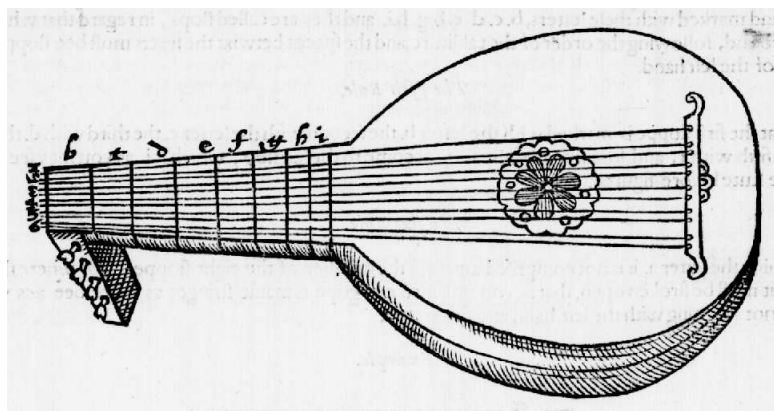
Figure 2.1 A typical Renaissance lute with six courses and eight frets. Image taken from William Barley, *A nevv Booke of Tabliture* (London, 1596₄).

stringing served to brighten the tone of a course, and is reflected neither in the tablature nor in transcription in modern music notation. The standard tuning used for the most part of the sixteenth century is the so-called *Renaissance tuning*, where the courses are tuned in perfect fourths with a major third in the middle. For an instrument as depicted in Figure 2.1, this thus yields a range of approximately two octaves and a fifth.

2.2.1   Lute tablature

The music in the surviving sources is notated exclusively in lute tablature. Lute tablature, a practical notation designed by and for lutenists, is a Griffschrift that instructs the player where on the fretboard to place the left-hand fingers and which courses to pluck with the right hand. It was born out of necessity for a new form of notation when in the late fifteenth century a shift from playing monophonic music with a plectrum to playing polyphonic music using the right-hand fingers took place. Throughout the sixteenth century, various tablature systems were in use simultaneously, conceptually all very similar to each other. Generally, four main systems are distinguished: on the one hand there are the so-called French, Italian, and Spanish systems, and on the other the German system. The former three all make use of a six-line (or, sometimes, five-line) staff whose lines represent the courses of the lute, where the lowest line may represent either the lowest-sounding (French and Spanish tablature) or the highest-sounding (Italian tablature) course. The frets at which the courses must be stopped are indicated either by means of letters (French tablature) or numerals (Italian and Spanish tablature), where *a* (or *0*) indicates an open course, *b* (or *1*) the first fret, etc. In the German

system, then, a unique symbol—encountered are letters, numerals, and various other symbols—is used for each course-fret combination. In this system, a staff is therefore not needed. Unlike the other systems, the German tablature system is not uniform: because the system was originally invented for the five-course lute, the symbols used to denote the frets on the later-added sixth course can vary from source to source. In all four systems, rhythm symbols indicating a minimum duration (more on this below) that applies to all notes in a chord are placed above the tablature chords. Two practices of rhythm symbol placement are discerned: in the one, symbols are given for all tablature chords, while in the other, they are only provided when there is a change in minimum duration.[2] In Figure 2.2, fragments of music notated in respectively Italian, French, and German lute tablature are shown.[3]

Although used to notate highly polyphonic music, lute tablature conveys no—or at best very limited—information about polyphonic structure, which was assumed to be filled in by the player. Moreover, as improvisation and ornamentation were an important part of sixteenth-century performance practice, the notation must not be considered prescriptive, but rather descriptive, to be deviated from at one's own discretion. Concretely, lute tablature lacks two notational clues necessary for the visualisation of polyphonic structure. First, it does not indicate to which polyphonic voices the tablature notes belong, and second, it does not indicate individual note durations, but rather gives a minimum duration that applies to all notes in a chord (and thus enables only the duration of the shortest notes in the chord to be notated precisely). Notational devices by means of which the former is achieved in mensural and modern music notation are, for example, stem direction or placement of the notes on separate staves; notational devices by means of which the latter is achieved are the use of various note head shapes or colours, as well as different flaggings of the note stems.

An additional problem with tablature—although one not directly related to polyphonic structure—concerns pitch, which depends on the tuning of the lute. Contemporary treatises show that the sixth (lowest-sounding) course was generally tuned to nominal G or A, although other pitches are mentioned as well (Wachsmann et al., 2001; Radke, 1963). Additionally, *scordatura* tunings deviating from the standard tuning were in use—the lowering of the sixth course by a whole tone being the most common example. All of this, however, is generally left unspecified in the tablature.

---

[2]Minor variants of the four main tablature systems exist; for more details, the reader is referred to the existing literature. Suitable starting points are Dart et al. (2001) and Smith (2002).

[3]More information on the pieces shown in this figure, as well as in subsequent figures, is given in Tables 4.1 and 4.2.

(a) Italian lute tablature. Abondante (1548_1), 'Mais mamignone', opening bars.

(b) French lute tablature. Phalèse (publ.) (1547_7), 'Tant que uiuray' [a4], opening bars.

(c) German lute tablature. Judenkünig (1523_2), 'Elslein liebes Elslein', opening bars.

Figure 2.2 Examples of staff-based and staffless lute tablature systems.

As has been acknowledged by researchers from various fields (see, for example, Charnassé and Stepien, 1986; Griffiths, 2002; Rhodes and Lewis, 2006), its notational conciseness renders tablature difficult to interpret for non-specialists, which in turn has led to the corpus having been researched only sparsely. In the words of Griffiths (2002): "[i]t is the alien nature of the lute's tablature notation, marvellously practical and comprehensible to players but seemingly impenetrable to others, that creates a psychological and mechanical barrier and has inhibited many of even the finest scholars" (p. 90). In Figure 2.3, a transcription in modern music notation of the fragment shown in Figure 2.2a is given, which may serve to illustrate how complex the polyphonic structure of the music notated in this deceivingly simple-looking format can be.

Figure 2.3 Transcription in modern music notation of a tablature fragment. Abondante (1548_1), 'Mais mamignone', opening bars.

2.2.2   Intabulations

The surviving corpus of instrumental sixteenth-century lute music can be subdivided into three main genres: dances or dance pairs, fantasias and ricercares, and intabulations, that is, instrumental arrangements of polyphonic vocal pieces. In this thesis, the focus is exclusively on intabulations. This choice is motivated by three considerations. First, intabulations formed the predominant lute genre in the sixteenth century (see, for example, Brown, 1965), and thus are highly representative of the contemporary lute practice. All kinds of vocal pieces, secular and sacred, were arranged for the lute: chansons, lieder, madrigals, motets, and even complete masses, varying in number of voices from two to five (and in very rare cases even six), and

ranging in structure from simple homophony to strict imitative polyphony. Second, since the densest polyphonic structures in lute music are found in intabulations, they constitute a challenging subcorpus. Third and most important, using intabulations facilitates the process of labelling the tablature data (that is, mapping the individual data points—the tablature notes—to voices), as the vocal models, whose polyphonic structure is unambiguous, can be used as polyphonic blueprints. This thus presupposes that intabulations are close arrangements of their vocal models. Studies of intabulations and the intabulation process (Ward, 1952; Brown, 1973–1974, 1976; Thibault, 1976; Göllner, 1984) and studies of contemporary treatises on the subject (Minamino, 1988; Canguilhem, 2001) indeed show that, although often a significant amount of ornamentation was added, it was generally of great concern to the intabulator to remain faithful to the vocal model. The process of labelling the tablature data and the role of the vocal models therein is described in further detail in Sections 4.1.1 and 4.1.2.

Although intabulators strove to remain faithful to the vocal models they arranged, intabulations are generally not literal, verbatim transcriptions of their vocal counterparts. First, the technical limitations of the instrument had to be taken into account: often reworkings were necessary because passages were otherwise not playable. Second, in some cases, intabulators simply *chose* to change the original notes. In general, two categories of adaptations can be discerned, be they by necessity or by choice: changes and simplifications.[4] Among the most common changes are included:

- ▶ Addition of notes, ornamental or other.
- ▶ Addition of complete bars or longer sections.
- ▶ Alteration of notes, in terms of pitch, duration, onset time, or combinations of these.
- ▶ Omission of notes.
- ▶ Omission of complete bars or longer sections.

Furthermore, the following simplifications are encountered:

- ▶ The representation of unisons by a single note rather than by two notes of the same pitch. Four examples of this practice can be found in the fragment shown in Figure 2.3: two in the second half of bar 3 (notes D_4 and E_4), and two in the first half of bar 4 (notes

---

[4]A more complete overview of such adaptations, most of them due to the technical limitations of the lute, can be found in Minamino (1988), where two categories of reworkings are discerned: alteration of counterpoint and alteration of rhythm (pp. 89–101).

D_4 and G_4). A single note belonging to two voices shall henceforth be referred to as a *single-note unison*.

► The omission of complete voices. Although a fairly drastic adaptation, it is nevertheless a common one. It is generally a structurally less important inner voice, such as the altus in a four-voice piece or one of the tenors in a five-voice piece, that is left out.

2.2.3 Instrumental idiom: three characteristics

In instrumental music, the musical idiom is often determined by the particular possibilities of the instrument on the one hand, and by its technical limitations on the other. This is no different in lute music. There are three characteristics of lute music, all due to instrumental limitations, that deserve special mention as they have a direct influence on the modelling approaches. First, unisons are more often than not represented as a single note rather than as two notes of the same pitch (as discussed in Section 2.2.2). Second, although six-note chords are encountered from time to time, lute music rarely contains more than five consistently independent voices. For practical reasons, in this thesis it is therefore assumed that the maximum number of voices possible on the lute is five. Third, as shown by Minamino (1988), there was a general consensus among intabulators that a note played on the lute could not be sustained beyond what was called a *semibreve*, a duration that corresponds to a modern half note when a reduction of the values by a factor of two is assumed (as is common practice when transcribing various forms of early music).

2.3 Computational background: voice separation

The musicological problem sketched in Section 2.2—how to determine polyphonic structure in lute tablature—can be interpreted as a problem of voice separation. In this section, an overview of existing research into voice separation in symbolic music representations is presented. Two phases are discerned: Phase 1 (1980s–1990s), where the research focused on the modelling of perceptual phenomena relating to polyphonic structure and auditory stream segregation, and Phase 2 (2000s–2010s), where various systems for voice separation were developed. Both the models from Phase 1 and the systems from Phase 2 are discussed in detail. The section is concluded with a discussion of two early applications of what can be considered rule-based

voice separation systems, intended specifically for the automatic transcription of German lute tablature. These applications were developed more or less concurrently with, but independently of, the models developed in Phase 1.

2.3.1   Voice separation, Phase 1: modelling of perceptual phenomena

In the first phase of research into voice separation (1980s–1990s), the research focused on the modelling of perceptual phenomena relating to polyphonic structure and *auditory stream segregation*, the process of grouping and separating components of sound (Bregman and Campbell, 1971; Bregman, 1990). Different approaches are taken.[5]

Huron (1989b) focuses on the perceptual independence of individual polyphonic voices, and presents a model for measuring pseudo-polyphony in Bach fugues. The model predicts the number of concurrent streams perceived at a certain time. A related experiment, focusing on listeners' ability to identify correctly the number of simultaneously sounding voices in polyphonic music, is described in Huron (1989a).

Marsden (1992) applies a rule-based approach to modelling the perception of voices in polyphonic music, and describes six models that are, again, tested on Bach fugues. The first two of these models are production systems based on production rules, mostly related to pitch and the *Gestalt* principle of *good continuation*, a principle that allows established patterns to be continued. The latter four are competition rule models analogous to neural networks, where rules compete with each other and the outcome of the competition is determined by the global functioning of a model through a system of weights.

The perception of *apparent motion* in music is modelled by Gjerdingen (1994). The term is borrowed from psychology, and denotes how a succession of static individual events (such as musical notes) are perceived as a moving, fluid whole (a melodic line). Gjerdingen uses an adaptation of the Grossberg-Rudd neural network model (Grossberg and Rudd, 1989), originally intended to model apparent motion in vision, and applies it to pitch, resulting in a model able to simulate several musical phenomena of auditory stream segregation.

McCabe and Denham (1997), lastly, present a two-layer leaky integrator neuron model of the early stages of auditory stream segregation. The model segregates acoustic input into a foreground stream and a background stream,

---

[5]The overview presented here is by no means exhaustive; it is limited to the studies cited most frequently in the research conducted in Phase 2.

and is capable of replicating the results from a number of psychophysical experiments relating to auditory stream segregation.

2.3.2 Voice separation, Phase 2: voice separation systems

The first phase of research into voice separation, as discussed above, can be regarded as a preliminary phase or theoretical background for the second phase (2000s–2010s), where systems for voice separation were developed. Two categories can be discerned: rule-based systems (the majority, discussed first below) and machine learning systems. The behaviour of rule-based systems is governed and formally described by a set of pre-defined rules. Machine learning systems, on the other hand, do not solely rely on knowledge encoded in them, but instead learn from training data to adapt their behaviour.

A common element that links the systems in both categories together is that all of them, in one way or another, are based on at least one of two fundamental perceptual principles associated with auditory stream segregation. These two principles are presented by Huron (2001) as the Pitch Proximity Principle and the Principle of Temporal Continuity. The former dictates that "[t]he coherence of an auditory stream is maintained by close pitch proximity in successive tones within the stream" (p. 24), and the latter that "[i]n order to evoke strong auditory streams, use continuous or recurring rather than brief or intermittent sound sources" (p. 12). Put differently: the closer two notes are to one another in terms of pitch or time, respectively, the more likely they are perceived as belonging to the same voice.

2.3.2.1 Rule-based systems (1)

Cambouropoulos (2000) briefly describes an elementary version of a streaming algorithm that uses path length minimisation. The algorithm, which is part of a larger system for automatic transcription of MIDI data into modern music notation, is based on the Gestalt principle of *proximity* and attempts to find, after the music is segmented into beats, the shortest streams that connect all onsets within the beats. It is acknowledged, however, that the problem is not trivial, and that the algorithm presented can be improved.

Temperley (2001) presents a *preference rule* system for contrapuntal analyis. Preference rules are criteria to evaluate a possible analysis; the system is based on five of such rules. Two of them, the Pitch Proximity Rule and the White Square Rule, match the Pitch Proximity Principle and the Principle of Temporal Continuity; the other two prescribe to minimise the number of voices (New Stream Rule) and to avoid shared notes (Collision Rule).

The fifth rule, added only following testing, is the Top Voice Rule, which prescribes to maintain a single voice as the top voice. Numerical scores, reflecting to which extent the preference rules are satisfied, are assigned to individual analyses; the analysis that gets the highest score is selected. Dynamic programming techniques are used to limit the amount of possible analyses that are to be evaluated. In later work, Temperley (2009) presents a unified probabilistic model of polyphonic music analysis. This model integrates three aspects of music analysis—metrical analysis, harmonic analysis, and stream segregation—and captures the complex interactions between them. The stream segregation analysis process presented can be regarded as a probabilistic version of the system presented in Temperley (2001).

Chew and Wu (2005) present a *contig* mapping approach that bears similarities to the saturated chord approach adopted by Charnassé and Stepien (1992) and discussed in Section 2.3.3. In addition to the Pitch Proximity Principle, a second perceptual principle underlying their approach is defined: the Stream Crossing Principle, prohibiting voice crossing. Their algorithm divides the music into segments where a constant number of voices is active, the contigs (the term is borrowed from computational biology). The *maximal voice contigs*, in which the number of voices that is active equals the maximum number of voices, act as starting points. The notes they contain are first assigned to voices, where, following the Stream Crossing Principle, the lowest note is simply assigned to the lowest voice, etc. The notes in the maximal voice contigs are then connected to the notes in the neighbouring (left and right) contigs, where the cost of the connection of two available notes is determined by their absolute pitch difference (cf. the Pitch Proximity Principle). After the connections have been established, the neighbouring contigs are connected to their neighbours, etc. This crystallisation process is completed when all contigs have been connected. Worth mentioning, lastly, is that a highly thorough evaluation method, for which three tailor-made metrics are defined (see also Section 5.6.2), is used to evaluate the algorithm.

A modified version of Chew and Wu's algorithm is presented by Ishigaki et al. (2011), who adapt the original algorithm by prioritising the connection of contigs that share a boundary at which the number of voices *increases*. This adaptation is based on the assumption that, in order to stand out, new voices (or voices re-entering after a period of rest) enter at a substantial distance from the already active voices (cf. the Pitch Proximity Principle). The idea is that when two contigs are connected, this distinguishability of the new voice prevents it from being connected incorrectly to the already active voices. Thus, in repeated iterations through all contigs, the algorithm connects adjacent contigs that share a boundary at which the number of voices increases. In each iteration, the individual contigs grow in size; the

connecting process is terminated when there are no more contigs to connect.

Madsen and Widmer (2006) present an algorithm for separating voices in MIDI data that is, according to themselves, highly inspired by Temperley (2001). The algorithm consists of a voice configuration unit and a note assignment unit. The music is processed in a linear fashion, where voice assignments are calculated locally, using a small lookahead. For groups of notes, the voice configuration unit generates well-formed solutions by ensuring that neither too few nor too many voices are available. The notes are then assigned to voices by the note assignment unit, which evaluates all possible voice assignments and calculates a preferred solution using a parametrised cost function. The cost of assigning a note to a voice depends on its pitch distance to the previous note in that voice (cf. the Pitch Proximity Principle); furthermore, costs are assigned to starting or ending a voice, and to adding a rest, that is, not assigning a note to an available voice.

Szeto and Wong (2006) consider streams to be clusters containing events proximal in the pitch and time dimensions, and model stream segregation as a clustering problem. Notes are defined as events, vectors consisting of a start time, an end time, and a pitch. Relations between events are defined as sequential (meaning that the notes do not overlap) and simultaneous (meaning that they do overlap). Simultaneous events cannot belong to the same cluster; therefore, relations between clusters are also defined as sequential and simultaneous. The starting point is an initial clustering in which each event is a separate cluster. By means of an an agglomerative single-link clustering algorithm, in an iterative process all clusters are combined into larger clusters until only n simultaneous sequential clusters—the streams—remain. It must be noted, lastly, that for Szeto and Wong stream segregation is only a means to an end—as can be deduced form the title of their study. Stream segregation, it is suggested, should be included as a preprocessing step in systems for pattern matching in polyphonic music databases, as this is expected to improve the quality of retrieved patterns.

2.3.2.2 Rule-based systems (2): non-monophonic voices

In all of the studies discussed so far, the term *voice* is used in its music-theoretical meaning. Kilian and Hoos (2002) present a stochastic local search algorithm for calculating an optimal voice separation. In their study, the term *voice* is used in its perceptual rather than its music-theoretical meaning. First, a piece is partitioned into slices of overlapping notes. The slices are processed iteratively, where the notes in them are assigned to voices. A stochastic local search algorithm is used to find assignments that minimise a parametric cost function that assesses the quality of these assignments. This

cost function is a weighted sum of four terms that penalise, respectively, large pitch intervals between successive notes in a voice (cf. the Pitch Proximity Principle), large gaps between successive notes in a voice (cf. the Principle of Temporal Continuity), large pitch differences within a chord, and overlap between successive notes in a voice. By weighting these terms differently, different results can be achieved, where voices can be strictly monophonic but may also contain multiple notes simultaneously. This is in line with their aim to be able to create "reasonable and flexible score-notation" rather than a "correct analysis" (p. 40), or even one that is valid in terms of auditory stream segregation.

The perceptual concept of a voice as a stream that is not necessarily monophonic is also fundamental to the Voice Integration/Segregation Algorithm (VISA) presented by Karydis et al. (2007a,b). Two types of perceptual principles underly their approach: principles responsible for *horizontal integration* (the Pitch Proximity Principle and the Principle of Temporal Continuity), and principles responsible for *vertical integration*. Of the latter kind are defined the Synchronous Note Principle, which is based on Huron's Onset Synchrony Principle (Huron, 2001), and which dictates that notes that have the same onset time and duration tend to be merged into a single sonority, and the Principle of Tonal Fusion, which is based on Huron's Tonal Fusion Principle (Huron, 2001), and which dictates that concurrent notes are perceptually less independent when they form intervals that promote *tonal fusion*. Tonal fusion is the phenomenon where two notes tend to be perceived as one (Huron, 1991; DeWitt and Crowder, 1987). It is suggested that vertical integration is prior to horizontal integration, and that an algorithm for voice separation should thus first identify concurrent notes that merge into single sonorities, and only thereafter, guided by the horizontal integration principles, break all notes (or sonorities) down into separate streams. This is reflected in the functioning of VISA. The music is processed in a linear fashion and divided into sweep line sets, sets of notes with the same onset time. For each sweep line set, it is determined whether to merge notes with the same onset and duration (*synchronous notes*) or not. This is done by looking at the context: if, within a definable window around the sweep line set, the proportion of synchronous notes exceeds a definable threshold, the context is assumed to be homophonic, and synchronous notes in the sweep line set are merged; otherwise, they are not. The sweep line set is thus partitioned into clusters, where a cluster may consist of a single note or of multiple notes. A bipartite matching algorithm that minimises a cost function is then used to match voices to clusters (if the number of detected voices is smaller than the number of clusters in the sweep line set) or clusters to voices (if the number of voices is equal or greater).

Lastly, a revised version of VISA is described in Rafailidis et al. (2009). The general functioning of the algorithm is left unchanged; revisions have been made with respect to the enumeration of voices, the partitioning of the sweep line sets into clusters, and the matching of notes to voices as well as the cost calculation.

2.3.2.3   Machine learning systems

In addition to the rule-based systems described above, two machine learning approaches have been tried. Kirlin and Utgoff (2005) present VoiSe, a system designed to separate voices in both explicit and implicit polyphony. The system consists of two components. The first is a *same-voice predicate*, implemented as a learned decision tree. Based on the examination of a number of features—5 measurements of pitch distance and 10 rhythmic features, among which several forms of time distance (cf. the Pitch Proximity Principle and the Principle of Temporal Continuity)—the predicate determines whether or not two notes belong to the same voice. It does so for all note pairs within two types of windows: one based on a maximum number of beats between the offset of the first note and the onset of the second note of a note pair, and one based on a maximum number of intervening notes. Per window, one predicate is learned. The second component is a hard-coded algorithm that maps notes to voices, and that is applied to a different excerpt of music than the predicate is trained on. The algorithm moves through the music in a linear fashion. If a note pair within a similar window as used for training the predicate satisfies the learned predicate, the notes are mapped to the same integer; otherwise, they are mapped to different integers.

Jordanous (2008) presents a probabilistic system that learns, based on pitch only, how likely a note is to occur for a voice, as well as how likely a transition between two notes is to occur. The system is partly inspired by the approach taken in Chew and Wu (2005), notably with respect to the way the music is processed. First, the music is broken down in smaller sections by searching for *marker points*, chords in which all voices are present (cf. Chew and Wu's maximal voice contigs) and whose pitches are far apart. Second, windows are defined around the marker points, extending both to the left and the right; the window around a marker point meets the window around the next marker point in the middle between the two. Third, starting at the marker points and working towards their left and right window boundaries, each note is assigned to a voice using the probabilities learned in the training to maximise a cost function. The notes to the left of the first marker point and subsequently those to the right of the last marker point are assigned in two completing steps.

2.3.2.4   A priori modelling assumptions

In all except two cases—the system by Kilian and Hoos (2002) and VISA (Karydis et al., 2007a,b)—voices are assumed to be monophonic. Three more of such a priori modelling assumptions can be distinguished: (i) voice crossing is allowed (or not), (ii) voices are allowed to share notes (or not), and (iii) the number of voices is known at the start of the voice assignment process (or not). Note that assumptions (i) and (ii) need only be made a priori when using a rule-based system. When using a machine learning system, they *can* be made—but it can also be left to the system to learn whether voice crossing occurs or whether voices are allowed to share notes.

Based on findings from music psychology showing that crossing streams are difficult to perceive (Deutsch, 1975; van Noorden, 1975), in most of the rule-based systems voice crossing is either not allowed or strongly discouraged. Only Madsen and Widmer (2006) explicitly allow them, but add that for their algorithm solutions involving voice crossing are more costly and therefore not the preferred choice. Voices sharing notes, second, is generally also discouraged, on the grounds that the unison is the interval that most promotes tonal fusion (DeWitt and Crowder, 1987; Huron, 1991). Third, with regard to the number of voices, two approaches are discerned. In the majority of the systems the number of voices is determined during the process. Generally, it is preferred to be kept to a minimum. In the systems by Chew and Wu (2005) and Jordanous (2008), then, the number of voices is known at the start of the voice assignment process, as it is determined by the maximal voice contigs from which the voice assignment process is seeded.

2.3.3   Applications for automatic transcription of lute tablature

A series of articles published over the course of two decades (Charnassé and Ducasse, 1971, 1973; Charnassé and Stepien, 1986; Charnassé, 1988; Charnassé and Stepien, 1991a,b, 1992) report the research conducted by the Équipe de Recherche sur l'Analyse et Transcription des Tablatures par Ordinateur (ERATTO), a research team that was based at the Centre National de la Recherche Scientifique in Ivry-sur-Seine, France. Despite the use of the plural *tablatures* in the team name, the research focused specifically on automatic transcription of *German* lute tablature—a choice motivated by the observation that this tablature style is the most challenging to interpret. In a summarising article (Charnassé and Stepien, 1992) published around the time of discontinuation of the team, the authors provide the history of the project in a nutshell, as well as some of the methods proposed along the

way. Additionally, two applications of rule-based voice separation systems (where ad hoc rules are used rather than perceptual principles) intended for automatic transcription of German lute tablature are described.

The first is a so-called *one-pass restructuring exploration* that uses *saturated chords*—chords containing as many notes as there are voices—as anchor points. The music is processed in linear fashion (both left-to-right and right-to-left are tried), in segments delimited by the saturated chords. Using a self-defined Principle of Continuity dictating that melodic lines in lute music tend to be composed of small intervals, notes between two framing saturated chords are connected to the contiguous notes closest in pitch. A last-assigned note concept is used to improve connections. In this straightforward approach, a fair number of problematic situations are encountered—for example, when large pitch distances between contiguous notes or voice crossing (which is said to be mainly due to imitation) lead to what is called *continuity disruption*, or, conversely, when multiple acceptable solutions lead to ambiguity with respect to continuity. Such situations are handled with additional ad hoc rules and heuristics, and the parametrisation of threshold values for continuity decisions.

The second system, which again makes use of saturated chords and the principle of continuity, is a fixed-rules inference system implemented in Prolog. It consists of two stages: first, imitative entries or other motifs are identified and frozen, thus exempting them from any further continuity analysis. Second, the remaining notes are grouped together in chains based on the principle of continuity (it is not entirely clear whether the frozen motifs are included in the chains or whether they are considered separate chains). Once all notes have been assigned to a chain, ambiguities and complex disruptions in continuity (which determine the beginnings and endings of the chains) are resolved. Each chain is connected to a saturated chord or another chain; production rules are used to determine good connections. Unambiguous cases are resolved first; problematic cases lead to new production rules and are resolved iteratively.

Although no numerical evaluation is given, it is concluded that the second system yields "far better" (Charnassé and Stepien, 1992, p. 168) results than the first. However, it is noted that the system is too dependent on the concept of saturated chords forming a reliable framework to start from, and, furthermore, that the results may improve when in addition to the principle of continuity other rules—pertaining, for example, to harmony or counterpoint—are modelled as well.

3

Method

In this thesis, the problem of voice separation is tackled using a supervised machine learning approach: models are trained on example data (pairs of input and desired output) to learn underlying rules or principles that map the inputs to the outputs, and then applied to new, unseen data (inputs) to make output decisions. More concretely, the models presented take as input feature vector representations of musical entities—in this case, notes or chords—and produce as output voice decisions. Three modelling approaches are described:

MA1 A note-level classification approach using a neural network model.
MA2 A chord-level regression approach using a neural network model.
MA3 A chord-level probabilistic approach using a discrete hidden Markov model.

This chapter, which describes the methodology used, is organised as follows. A fully detailed description of the modelling approaches and the models, the model extensions—simultaneous voice and duration modelling, backward processing, and multi-pass processing using a bidirectional decision context—, and the feature set is given in Section 3.1. In Section 3.2, the evaluation procedure, the evaluation metrics, and the evaluation modes are discussed, followed in Section 3.3 by a detailed description of conflicts occurring in MA1 and their resolution. Section 3.4, finally, gives the implementation details.

3.1   Modelling

The different modelling approaches entail different learning methods. In MA1, the data points are notes. Each note is represented as an n-dimensional feature vector encoding properties of that note in its polyphonic context, with which is associated a label encoding the voice class (the term is explained in Section 3.1.2) the note belongs to. The model is trained to classify the data correctly. In MA2, then, the data points are chords. Each chord is represented as a set of m n-dimensional feature vectors, each of them encoding properties of that chord in its polyphonic context under one of m possible mappings of the chord notes to voices. The model is trained to rate the correct mappings for the data the highest. In MA3, lastly, the data points are also chords. Each chord is now represented as an n-dimensional feature vector encoding the pitches in the chord, with which is associated the mapping of the chord notes to voices. The model is trained to find the most likely sequence of mappings for the data.

3.1.1   Learning models

3.1.1.1   Neural networks

Neural networks are adaptive mathematical models that model complex relations between input values and output values. In the research presented in this thesis, standard three-layer feed-forward neural networks with resilient backpropagation (Rprop) as the learning algorithm are used. Resilient backpropagation (Riedmiller and Braun, 1993) is preferred over the standard backpropagation algorithm (Rumelhart et al., 1986), firstly because it has been shown to outperform the latter (Schiffmann et al., 1993; Riedmiller and Braun, 1993; Riedmiller, 1994), and secondly because it does not have any parameters that must be set (backpropagation requires the setting of a learning rate and a momentum). Not the classic algorithm, but the later variant iRprop$^+$ (Igel and Hüsken, 2000, 2003), which shows improvements in terms of both robustness and convergence speed, is used.[1] As the activation function for the neurons in the hidden layer and the output layer the sigmoid function is chosen, where, given input z, the activation value of a neuron is calculated as follows:

$$\text{sigm}(z) = \frac{1}{1 + e^{-z}}. \tag{3.1}$$

---

[1]Training with resilient backpropagation (or another backpropagation algorithm) is susceptible to the *flat spot problem* (Fahlman, 1989). In the iRprop$^+$ implementation used (see Section 3.4), the flat spot problem is addressed by default; this is left unchanged.
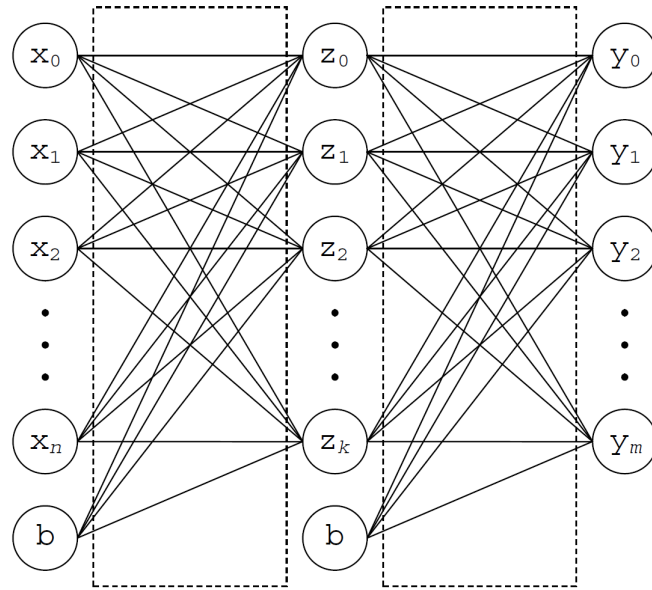
Figure 3.1 A three-layer neural network with n input neurons x, k hidden neurons z, m output neurons y, and two bias neurons b.

Multilayer feed-forward neural networks are universal approximators (Hornik et al., 1989; Hornik, 1991), meaning that they can represent any function with arbitrary precision, depending on the number of neurons in the network. This has the advantage that no assumptions need to be made about the nature of the function that is modelled. On the other hand, training a network is a non-convex optimisation problem, which means that there can be multiple local minima where the gradient descent learning algorithm can get stuck.

Figure 3.1 gives a schematic representation of a three-layer neural network as used for the research presented in this thesis. The sizes of the input and output layer vary per modelling approach, as different feature vectors and different output representations are used. The size of the hidden layer, then, is a hyperparameter that is optimised for the various models in a preliminary experiment (see Section 5.1.1). Both the input and the hidden layer contain a bias neuron, which receives no inputs and outputs only the value 1.

All neural network models used in this thesis are trained using batch training, where the network weights are initialised randomly, and the weight update is performed after each iteration of the learning algorithm over the complete training set. The number of iterations required for convergence is determined by checking the voice decision accuracy (see Section 3.2.2) and the network error every 10 iterations; when a stabilisation in both is
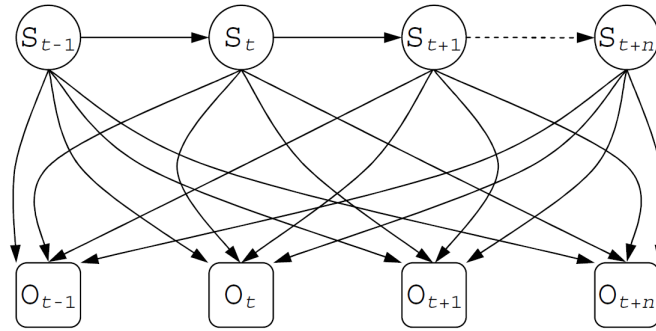
Figure 3.2 A hidden Markov model with n hidden states S and observations O. Horizontal arrows indicate transition probabilities; vertical arrows indicate observation probabilities.

witnessed, convergence is assumed to be achieved.

3.1.1.2   Hidden Markov models

Hidden Markov models (HMM) are probabilistic models that define probability distributions over sequences of observations. A sequence of observations is generated by a sequence of hidden states, which cannot be observed directly. The transitions between the hidden states follow a Markov process, where the probability distribution of a state S_t depends only on the state(s) S_{t-n}, and not on those before $t - n$. Knowing the transition probabilities (the probabilities of transitions between hidden states) and the observation probabilities (the probabilities of an observation given a hidden state), the most likely sequence of hidden states can be computed using the Viterbi algorithm (Viterbi, 1967; Forney, 1973).

Hidden Markov models have proven to be very useful for modelling time series data (Ghahramani, 2001)—a typical example is their application in speech and language processing (Rabiner, 1989)—, and are being used in many areas of MIR. In this thesis, a first-order ($n = 1$) hidden Markov model is used. A schematic representation of such a model is given in Figure 3.2.

3.1.2   MA1: a note-level classification approach using a neural network model

In MA1, an approach on the note level, the task of voice separation is modelled as a multi-class classification problem, where the classes are the voices the notes can be classified into. Each note in the dataset is represented as an n-dimensional feature vector encoding properties of that note in its poly-

phonic context, and with this feature vector is associated a label encoding the *voice class*—a number between and including 0 and 4—the note belongs to. There are thus five voice classes, a number that corresponds to the maximum number of voices assumed to be possible on the lute (see Section 2.2.3). The goal is to have a model that, for each note in a piece, takes as input the feature vector representing that note, and, based on that input, classifies the note into the correct voice class. To this end, the model is trained so that given a set of feature vectors, its outputs approximate the set of associated labels as closely as possible.

The model used in this approach is a three-layer feed-forward neural network model as described in Section 3.1.1.1. It has an output layer containing five neurons, each of which represents one of the possible voice classes. Because the sigmoid activation function is used, the individual output neurons have an activation value ranging between and including 0 and 1. This value reflects the approximate average activation of the output neuron for a given input. The model output for a given input is thus a five-dimensional vector containing the activation values of the output neurons. The label associated with an input, then, is a five-dimensional binary vector. Each of its elements again represents one of the voice classes; the position of the 1 determines the voice class encoded. The voice class decision for a given input is made by determining the position of the highest activation value in the model output. If this position corresponds to the position of the 1 in the label associated with the input, correct classification is achieved. For each given input, the model is thus trained to activate a certain output neuron more strongly than the others. Ideally, the network output will be identical to the label; in practice, however, as observed in the experiments, it is always an approximation (although often a very close one).

3.1.2.1   Single-note unisons

There is one characteristic of music written in tablature that complicates the voice class decision slightly: the fact that each note can be a single-note unison, a note belonging to two voice classes (see Sections 2.2.2 and 2.2.3). A label encoding two voice classes contains the number 1 twice. Thus, theoretically, given an input encoding a single-note unison, the output of a well-trained model will contain two high activation values that are close to each other, and that are notably higher than the other activation values. For each note in the tablature data, when making the voice class decision both the highest and the second-highest value in the output are therefore always considered. The output is assumed to represent a classification into two voice classes if the second-highest activation value deviates no more than a defined

percentage, the *deviation threshold*, from the highest. For all experiments conducted in this thesis, this deviation threshold is set to 5%. However, the deviation threshold heuristic is not entirely without problems; this is discussed in more detail in Section 5.7.3.1.

3.1.2.2   Sigmoid versus softmax activation function

It can be argued that theoretically, it would be more appropriate to use the softmax (normalised exponential) function as the activation function for the output layer. With this function, given input z_j, the activation value of the jth neuron in the layer is calculated as follows:

$$\text{softm}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{3.2}$$

where K is the total number of neurons in the layer. With this function, the activation values of the output neurons will also range between and including 0 and 1, but will now sum to 1. This means that the network output can be interpreted as a probability distribution, which in the case of a classification task with binary vectors—also some sort of probability distributions—as class labels is a highly convenient interpretation. However, in this thesis the sigmoid function is preferred over the softmax function, as two observations speak against the use of the latter. First, the elements of a label do not always sum to 1 (that is, they cannot always be interpreted as a probability distribution): in the case of single-note unisons, which account for approximately 1.5% of the tablature data, they sum to 2.[2] This is likely to cause some loss in the learning, as the softmax function cannot replicate this. Second, unlike the sigmoid function, the softmax function cannot be used out-of-the-box for simultaneous voice and duration modelling, as described in Section 3.1.2.3.

3.1.2.3   Simultaneous voice and duration modelling

The classification approach as described above can easily be adapted to include the modelling of full note duration, which, like voice information, is not provided in the tablature but must be inferred. Note duration plays a significant role in the voice separation process: because each voice is assumed to be monophonic, the offset time of a note n_t determines whether or not the voice that note is assigned to is available for a following note n_{t+x}. Modelling voice and duration simultaneously is therefore hypothesised to be beneficial to model performance.

---

[2]The tablature data consists of 11641 notes, 158 of which are single-note unisons.

The durations are discrete; just like the voices, they can thus be interpreted as classes. Moreover, all durations used in tablature notation—the *semibreve* (| or ♭), *minim* (⌐ or ⌡), *semiminim* (⌐ or ⌡), *fusa* (⌐ or ⌡), and *semifusa* (⌐ or ⌡)—are relative: each duration is the double of the next smaller duration, and a multiple of the shortest duration encountered, the semifusa.[3] 32 *duration classes* are used, where class 0 represents a semifusa and class 31 a breve, the double of a semibreve (cf. the voice classes as defined in Section 3.1.2). As discussed in Section 2.2.3, the longest sustainable duration on the lute was considered to be the semibreve, in which there are only 16 semifusae. This seems to render duration classes 16–31 superfluous; however, durations exceeding the semibreve are sometimes justifiable—for instance when a *corona* (fermata) is used at an important point of closure in a piece. A duration class is encoded as a binary vector in a similar fashion as a voice class; the result now is a 32-dimensional vector.

When modelling voice and duration simultaneously, the goal is to have a model that, given as input a feature vector representing a note, classifies that note simultaneously into the correct voice class and into the correct duration class. The model's output layer now contains 37 neurons: the first five of these represent the voice classes, and the others the duration classes. Similarly, the labels are now 37-dimensional; each of them is a concatenation of a vector encoding a voice class and one encoding a duration class. Again, the model is trained so that given a set of feature vectors, its outputs approximate the set of associated labels as closely as possible. The model output is interpreted as two subvectors; the class decisions are made by determining the position of the highest activation value(s) in the respective subvectors.

Simultaneous voice and duration modelling is the first model extension (X1) proposed under Objective 1.2 (see Section 1.2); it is implemented for MA1 only. It should be noted, lastly, that simultaneous voice and duration modelling only applies to the tablature dataset—in the Bach dataset, the notes' full durations are always given, and thus need not be inferred.

3.1.2.4 Application to unseen data and conflicts

When the trained model is applied to unseen data, it makes the class decisions for each note successively. The music is processed in linear fashion; chords are always processed from bottom to top, as they are organised in the internal representation used (see Sections 4.1.2 and 4.2.2).[4] It may now occur that a

---

[3]To avoid terminological confusion, in this thesis the American nomenclature (*whole note*, *half note*, etc.) is used when referring to duration in music in non-tablature formats.

[4]The reason for processing chords bottom-to-top is thus a purely practical one; it may be worthwhile experimenting with processing them top-to-bottom as well, as in lute music

voice class decision is made that conflicts with a previous voice class decision, meaning that two simultaneous or overlapping notes will be assigned to the same voice. This is a situation that is to be avoided if a voice is assumed to be monophonic. Conflicts can occur between the decision for a note n in a chord c and (i) the decision for another (lower) note in c, (ii) the decision for a *sustained previous note*, that is, a note in a previous chord whose offset time is greater than the onset time of n, and (iii) the decision for an *interrupting next note*, that is, a note in a next chord whose onset time is less than the offset time of n. Which conflicts can occur when depends both on the processing mode (forward or backward; see Section 3.1.5) and on the decision context (unidirectional or bidirectional; see Section 3.1.6). Conflict resolution is a topic on its own, which is discussed in detail in Section 3.3.

3.1.2.5   Number of voices

The model used in MA1, which has an output layer of five neurons allocated to voice classes (and, when modelling voice and duration simultaneously, an additional 32 allocated to duration classes) can be applied to music with any number of voices smaller or equal to five (and any number of durations ranging from a semifusa to a breve). Neurons corresponding to non-existing voice (or duration) classes will never be activated. The *actual* number of voices in a dataset is determined prior to the training by finding the largest chord in the set (where sustained notes, if applicable, are also considered when determining the size of a chord). This number is needed for the resolution of certain conflicts; this is discussed in more detail in Section 3.3.1.1.

3.1.2.6   Nomenclature

In this thesis, the model used in MA1 is referred to as N when modelling only voice, and as N′ when it is extended with X1, that is, when modelling voice and duration simultaneously.

3.1.3   MA2: a chord-level regression approach using a neural network model

In MA2, an approach on the chord level, the task of voice separation is modelled as a regression problem, where mappings of chord notes to voices are rated. Each chord in the dataset is represented as a set of m n-dimensional feature vectors, each of them encoding properties of that chord in its polyphonic context under one of m possible mappings of the chord notes to voices.

---

the top voice, which tends to be more ornamented than the others, has some prominence.

The goal is to have a model that, for each chord in a piece, takes as input the feature vector set representing that chord, and, based on that input, rates the correct mapping the highest. This is unlike a standard regression approach, where desired outputs—which in this case would be ratings—are given. To this end, the model is trained so that given a set of feature vector sets, for each feature vector set the vector resulting from the correct mapping gets a higher network output—in other words, is rated higher—than all the other vectors in that feature vector set. This technique is called *relative training*, and is described in more detail in Section 3.1.3.2.

To suit this modelling approach, the model, again a three-layer feed-forward neural network model, has an output layer containing only a single neuron, the comparator neuron. The activation value of this neuron ranges between and including 0 and 1, and reflects the rating for a mapping.

3.1.3.1   Mappings: enumeration and pruning

For each chord, given the number of notes in the chord and the number of voices in the dataset, all possible *mappings* of notes to voices are enumerated. A mapping is encoded as an n-dimensional vector whose individual elements represent a voice (where the first element is the top voice) and contain a discrete value between -1 and $n - 1$, where -1 indicates that no note is mapped to the voice, 0 that the first chord note (whose index in the chord is 0; sustained previous notes are not included) is mapped to the voice, etc. For reasons explained below, n is set not to the maximum number of voices possible on the lute (five), but to the highest number of voices encountered in the datasets used, four. An example of this representation is given in Figure 3.3. Note that the vectorial representation used enables one note to be mapped to two voices (see chords 13–15 and 18, which all contain a single-note unison), but does not enable multiple notes to be mapped to one voice (in which case the voice is no longer monophonic).

Only mappings that meet the following criteria are enumerated: (i) all chord notes are mapped to a voice, (ii) each chord note is not mapped to more than one (Bach dataset) or two (tablature dataset) voices, and (iii) each chord note is not mapped to a voice already taken by a sustained previous note (Bach dataset only). Both in order to avoid the method to become computationally too expensive and to improve the model's performance, from the list thus generated all mappings containing more than two *voice inversion pairs* are then pruned. A voice inversion pair is an instance of two voices being inverted within a chord. The first chord of bar 4 in Figure 3.3a, for example, where the tenor voice (T) lands above both the superius (S) and the altus (A) after a leap of a fourth, contains two such pairs: TS (= ST)

(a) Transcription in modern music notation.

```
10 [ 1,  0, -1, -1]        15 [ 1,  1,  2,  0]
11 [-1,  1, -1,  0]        16 [ 0, -1, -1, -1]
12 [ 0, -1, -1, -1]        17 [ 0, -1, -1, -1]
13 [ 1,  0, -1,  0]        18 [ 1,  0,  1, -1]
14 [ 2,  1,  2,  0]        19 [ 3,  1,  2,  0]
                           20 [ 0, -1, -1, -1]
                           21 [ 2, -1,  1,  0]
```

(b) Mappings for chords 10–21.

Figure 3.3 Vectorial representation of mappings of chord notes to voices. Abondante (1548_1), 'Mais mamignone', bars 3–4 (chords 10–21).

and TA (= AT). When counting voice inversion pairs, sustained previous notes (if applicable) are included, and unisons are not considered to be voice inversions. The limit of two is chosen as this is the maximum number of voice inversion pairs per chord encountered in the datasets.

That the pruning can be very effective in reducing the number of mapping possibilities is shown in Table 3.1, which gives, both for the tablature dataset and for the Bach dataset, the number of possibilities for a chord of n notes in a context of v voices before and after pruning (all numbers provided are for the case where there are no sustained previous notes). Note that where n is smaller than v, the number of mapping possibilities is always higher on the tablature dataset than on the Bach dataset. This is because on the tablature dataset, up to $v - n$ notes can in this case be mapped to two voices. The table shows that the effect of the pruning increases as the number of notes

Table 3.1 Number of mapping possibilities for a chord of n notes in a context of v voices before and after pruning of mappings containing more than two voice inversion pairs. Tablature and Bach datasets. B = before pruning, A = after pruning.

n	Tablature dataset						Bach dataset					
	$v = 3$		$v = 4$		$v = 5$		$v = 3$		$v = 4$		$v = 5$	
	B	A	B	A	B	A	B	A	B	A	B	A
1	6	6	10	10	15	15	3	3	4	4	5	5
2	12	12	42	40	110	100	6	6	12	12	20	20
3	6	5	60	38	330	161	6	5	24	20	60	50
4			24	9	360	85			24	9	120	45
5					120	14					120	14

in a chord grows, and that it is stronger when there are more voices.

3.1.3.2 Feature vector set generation and relative training

For each mapping in the list of enumerated mappings for a chord, a feature vector is calculated. In each mapping the chord notes are distributed differently among the voices; thus, each of them leads to a different polyphonic embedding of the chord, and therefore results in a different feature vector. (Exactly how the polyphonic embedding is encoded in the features is described in Section 3.1.7.2.) Each feature vector constitutes an element of the feature vector set that represents the chord; the feature vector that results from the correct mapping—henceforth referred to as the *ground truth feature vector*—is placed as the first element of the set. It must be noted that not all feature vector sets have the same size, as not all chords have the same number of mapping possibilities.

For the training, a list of *relative training pairs* is created from the feature vector sets representing the training data. A relative training pair is a pair of feature vectors, the first element of which is always a ground truth feature vector. For each feature vector set, the first element (the ground truth feature vector) is paired up with every other element (feature vector), and each thus created pair is appended to the list of relative training pairs. The entire process of mapping enumeration and pruning, feature vector set generation, and creation of the list of relative training pairs is outlined in Figure 3.4.

The list of relative training pairs is used to extract the actual training data from. The model is trained using a method that is based on relative ratings, where one item is to be rated higher than another. This method is similar to the one proposed by Zheng et al. (2007), which, in turn, is a

```
01   method enumPruneAndList(training set t) returns list
02       create empty list of RTPs l
03       for each chord c in t
04           enumerate and prune mappings
05           create empty feature vector set F
06           create empty ground truth feature vector g
07           for each mapping m for c
08               calculate feature vector f(c, m)
09               if m is correct for c
10                   g := f(c, m)
11                   insert f(c, m) at top of F
12               else
13                   append f(c, m) to F
14           for each feature vector f != g in F
15               create RTP (g, f) and append to l
16       return l
17   end method
```

Figure 3.4 Mapping enumeration and pruning, feature vector set generation, and creation of the list of relative training pairs. RTP = relative training pair.

generalisation of a method suggested earlier by Braun et al. (1991). The latter has previously been applied successfully to tackle musical problems by Weyde and Dalinghaus (2003) and Hörnel (2004). The model is trained so that for each chord, the ground truth feature vector is rated higher than any of the other feature vectors for that chord—in other words, so that the first element of each relative training pair extracted from the feature vector set representing the chord is rated higher than the second. Thus, in each training iteration, the list of relative training pairs is presented to the model. The model then evaluates both elements of each relative training pair; if the model output (rating) for the first element does not exceed that for the second by at least *margin* ε (a suitable value for which is determined in a preliminary grid search; see Section 5.1.2), the relative training pair is turned into two training examples, consisting of the respective feature vectors and their ratings. Because the ground truth feature vector is to be rated higher, the original ratings are switched and adapted: the ground truth feature vector gets the higher rating, to which ε is added, while the other gets the lower, from which ε is subtracted. When all relative training pairs have been checked, the model is trained with the training set thus created. The procedure as described is repeated in the next training iterations. In the

```
01   method createTrainingEx(list of RTPs l) returns list
02       create empty list of TEs m
03       for each RTP (g, f) in l
04           if rating(g) < (rating(f) + epsilon)
05               append TE (g, rating(f) + epsilon) to m
06               append TE (f, rating(g) – epsilon) to m
07       return m
08   end method
```

Figure 3.5 Creation of the list of training examples for each training iteration. TE = training example.

case when at the beginning of a training iteration all relative training pairs are rated as desired, the training is fully successful. The process of creating the training data for each training iteration is outlined in Figure 3.5.

3.1.3.3   Application to unseen data

When applying the trained model to unseen data, a mapping decision is made for each chord successively. As in MA1, the music is processed in linear fashion. For each chord, all possible mappings are enumerated and pruned, and a feature vector set is created. (Note that the ground truth feature vector now cannot be placed at the top of the set, as the correct mapping is unknown.) Each feature vector in the set is then rated; the mapping that goes with the feature vector that receives the highest rating is the mapping given to the chord.

Lastly, three things must be noted. First, when applying the trained model to unseen data, due to incorrect mapping decisions for previous chords, the rare case may arise where the list of enumerated mappings contains *only* mappings with more than two voice inversion pairs. This is a situation that can only occur on the Bach data, where it can be caused by sustained previous notes mapped to the incorrect voice. Because pruning would now result in an empty list, in this case all remaining mappings are simply retained. Second, as opposed to MA1, MA2 is conflict-free. As mentioned in Section 3.1.2.4, in MA1, conflicts can occur between the voice class decision for a note n in chord c and (i) the decision for another note in c, (ii) the decision for a sustained previous note, and (iii) the decision for an interrupting next note. In MA2, mappings to which (i) applies simply do not occur, as the mapping representation used does not enable multiple chord notes to be mapped to one voice. Mappings to which (iii) applies also do not occur, as the model used

in MA2 is only implemented in forward processing mode (see Section 3.1.5). Mappings to which (ii) applies, then, are eliminated in the enumeration (as explained in Section 3.1.3.1). Third, both when training the model and when applying it to unseen data, the possibility exists that when rating the feature vectors in a feature vector set representing a chord, the highest rating is received by more than one feature vector (the *multiple-highest-rating problem*). In this case, rather than opting for a possibly complicated elimination procedure to determine the most likely candidate, or, alternatively, for random selection (which has the advantage that it is a straightforward solution, but which renders mapping decisions cumbersome to reconstruct), the mapping that goes with the first feature vector in the set that gets this highest rating is simply selected.

3.1.3.4   Number of voices

Theoretically, unlike the model used in MA1, the model used in MA2 can be applied to music with any number of voices. However, a maximum number of voices is nevertheless set. The reason for this is that the size and content of the feature vector depend on the assumed maximum number of voices. As is explained in more detail in Section 3.1.7.3, the higher this number is, the longer the feature vector becomes, and the more unnecessary information it may contain. In MA2, the maximum number of voices is therefore assumed not to be the maximum number of voices possible on the lute, but rather the highest number encountered in the datasets: four. As in MA1, the *actual* number of voices in a dataset—which is now needed in the mapping enumeration—is determined prior to the training by finding the largest chord in the set.

3.1.3.5   Nomenclature

In this thesis, the model used in MA2 is referred to as C.

3.1.4   MA3: a chord-level probabilistic approach using a discrete hidden Markov model

In MA3, also an approach on the chord level, the task of voice separation is modelled as a probability problem, where, given an observation sequence, a discrete hidden Markov model as described in Section 3.1.1.2 is used to estimate hidden states. Here, the observations are the chords, and the hidden states the mappings of chord notes to voices. Each chord c is represented as an n-dimensional feature vector encoding the pitches (MIDI numbers,

ordered from low to high) in the chord, where n thus equals the number of notes in the chord. With this chord is associated a mapping of the chord notes to voices. Each mapping m_t for a given time frame t is an n-dimensional vector whose individual elements represent a voice and contain a discrete value between -1 and $n-1$, where n is equal to the highest number of voices encountered in the datasets, four (this is the exact same representation as used in MA2; see Section 3.1.3.1 and Figure 3.3). The goal is to have a model that finds the most likely sequence of mappings for the data.

To this end, first, a chord dictionary and a mapping dictionary, listing the set of chords C and the set of mappings M contained in the dataset, respectively, are created. For each training set used in the cross-validation procedure (see Section 3.2.1), then, an $|M| \times |M|$ transition probability matrix $P(m_{t+1}|m_t)$, denoting the probability of transitioning to mapping m_{t+1} given mapping m_t, a $|C| \times |M|$ observation probability matrix $P(c_t|m_t)$, denoting the probability of observing chord c_t given mapping m_t, and a $1 \times |M|$ initial state matrix $P(m_0)$, denoting the probability of a mapping being the initial state m_0, that is, the first mapping of a piece, are created. This is done by increasing, for each occurrence in the training set of a mapping transition, a chord-mapping combination, or a mapping as initial state, the corresponding element in the appropriate matrix by 1, and then normalising each element by dividing it by the sum N of the matrix row or column (depending on the layout of the matrix) it is in. The possibility now exists that a mapping transition, a chord-mapping combination, or a mapping as initial state does not occur in the training set, resulting in a probability of 0 for it. This is problematic because on the test set, this mapping transition, chord-mapping combination, or mapping as initial state will now not be considered: the model has learned that it never occurs. To avoid such situations, a form of Laplacian smoothing is applied, where all three matrices are initialised with 1s—in other words, where one extra occurrence of each mapping transition, chord-mapping combination, or mapping as initial state is assumed.

An unwanted side effect of the smoothing, however, is that impossible chord-mapping combinations in the observation probability matrix (combinations where the number of notes in the chord and the number of notes encoded in the mapping are different) now also receive a non-zero probability ($\frac{1}{N}$)—whereas a probability of 0 is actually desired in these cases. This is corrected by setting all elements in this matrix that represent an impossible chord-mapping combination back to 0.

The most likely sequence of mappings in the test set, finally, is computed using the Viterbi algorithm.

3.1.4.1   Number of voices and conflicts

As is the case with the model used in MA2, the model used in MA3 can be applied to music with any number of voices. In MA3, however, the maximum number of voices assumed—which is reflected by the dimension of the mapping vector—does not restrict the model in any way. The dimension of the mapping vector is a parameter that can be set to any desired value; as mentioned in Section 3.1.4, for this thesis it is set to four. Unlike in the other two modelling approaches, then, in MA3 the *actual* number of voices in a dataset is irrelevant and thus needs not be determined.

Furthermore, as in MA1, in MA3 conflicts (see Section 3.1.2.4) can occur —but only on the Bach dataset, where a note in chord c_t may be mapped to a voice that is actually not available because a sustained previous note in a chord c_{t-x} is also mapped to it. Because no duration information is used in the model, in MA3 these conflicts are currently left unresolved.

3.1.4.2   Nomenclature

In this thesis, the model used in MA3 is referred to as H.

3.1.5   Processing modes

The *processing mode* determines the direction in which a piece is processed. Two different processing modes can be discerned: *forward* (fwd), where the piece is processed from left to right, and *backward* (bwd), where it is processed from right to left, starting with the final chord. (As already mentioned in Section 3.1.2.4, in MA1, irrespective of the processing mode, the chords themselves are always processed from bottom to top.)

The forward processing mode is the most intuitive, as it corresponds to the direction in which music unfolds—new notes get their meaning in the context of what is heard before—and in which the listener experiences it. The motivation behind the backward processing mode has to do with what may be called the *transparency* of the polyphonic structure. In much polyphonic music, the polyphonic fabric tends to become texturally saturated towards the end of a piece, whereas at the beginning, the texture tends to be thinner (the latter applies especially to pieces written in imitative counterpoint, where the individual voices usually enter successively). A typical example is given in Figure 3.6.

In the opening bars of a piece, it therefore often becomes clear to which voice the first notes belong only when one or more other voices enter. Towards the end, on the other hand, such problems occur only rarely, as here all

Figure 3.6 Textural saturation. Pisador (1552_7), 'Pleni de la missa misma', opening and closing bars.

voices are generally active. The final chord (and to a certain extent the music leading up to it) usually provides a very stable point of reference: firstly, almost without exception all voices are represented here, and secondly, they tend to be represented in the right order—that is, without crossing one another.[5] In backward processing mode, where a model starts processing the polyphonically most transparent end of a piece—all voices are represented and do not cross one another—, the risk of starting on the wrong foot, potentially leading to a propagation of wrong decisions, is therefore smaller than in forward processing mode.

Backward processing is the second model extension (X2) proposed under Objective 1.2 (see Section 1.2); like the first model extension, it is implemented for MA1 only.

3.1.6 Decision context and double-pass models

The processing mode determines only the direction in which a piece is processed; the information on which the class or mapping decision is based depends on the *decision context*. The decision context is the polyphonic context within which the feature vector for a note (MA1) or the set of feature vectors for a chord (MA2) is calculated (the term does not apply to MA3). Two types are used: a *unidirectional* and a *bidirectional* decision context. As the names suggest, the former extends to only one direction—either left

---

[5]This is the case for all pieces in both the tablature dataset and the Bach dataset used in this thesis.

Figure 3.7 Unidirectional (extending to the left or right) and bidirectional (extending to both directions) decision contexts. Abondante (1548_1), 'Mais mamignone', bars 3–4.

or right—of a note or chord, whereas the latter extends to both directions. A unidirectional decision context thus only partially embeds a note or chord polyphonically, whereas a bidirectional one does so completely. The decision context is bounded by the immediate adjacent notes in all voices; its size, which depends on the onset time of these notes, therefore varies from chord to chord. To illustrate, Figure 3.7 shows, for all notes in a chord, the unidirectional decision contexts extending to the left and to the right. Combined, these form the bidirectional decision context.

For reasons explained below, in this thesis, the unidirectional decision context extending to the left is only used in forward processing mode, and the one extending to the right only in backward processing mode. An approach comparable to one in which a model that uses a unidirectional decision context extending to the right is used in *backward* processing mode, is found in the work of Charnassé and Stepien (1992). They call the processing mode the *scanning direction*, and report to encounter fewer problems when scanning the music from right to left. They state that "[a]lthough this seems very strange, there is some kind of underlying logic to it. Th[e] music is heavily embellished, and ornaments usually lead into a chord or sustained note. Consequently, as an ornament approaches its resolution chord, it seems to narrow down its range" (p. 163).

The rationale behind using a bidirectional decision context is the availability of more comprehensive information on the polyphonic embedding of a note or chord when the class or mapping decision is made. However, a model that uses a bidirectional decision context—henceforth referred to as a

bidirectional model, as opposed to a *unidirectional model*—cannot be used in a first pass through the data. This is because in a first pass, when applying a trained model to unseen data, the *polyphonic information*—that is, voice and, if applicable, duration information—available to the model only grows incrementally with every note or chord that is processed and for which a class or mapping decision is made (more on this in Section 3.2.4.1). Thus, in forward processing mode, only polyphonic information to the left of the note or chord the decision is made for is available, and in backward processing mode only information to the right. (In short: in a first pass, only information in the direction *opposite* to the processing direction is available.) A bidirectional model, however, requires the availability of this information in both directions. The idea is therefore to acquire, in a first pass through the data, polyphonic information from a unidirectional model, and subsequently to annotate the data with it. In a second pass, then, a bidirectional model can be employed. In the calculation of the features that encode a polyphonic relation of a note with another note, either from the direction identical to the processing mode or from the direction opposite to it, information acquired in the first pass is now used. This applies both when the model is trained and when it is applied to unseen data. The actual, correct polyphonic information serves exclusively as training targets. (As opposed to in the first-pass model, where the correct polyphonic information is used in the feature calculation when the model is trained, but the information provided by the model itself when it is applied to unseen data (see also Section 3.2.4.1), in the second-pass model this information thus always comes from the same distribution.[6]) The intuition is that the model learns to correct incorrect decisions from the first pass, or, more precisely, learns to make correct decisions even when given input based on incorrect polyphonic information. The enhanced decision context is expected to be beneficial in this process.

Multi-pass processing using a bidirectional decision context is the third model extension (X3) proposed under Objective 1.2 (see Section 1.2); like the first two model extensions, it is implemented for MA1 only.

3.1.6.1   Nomenclature

In this thesis, the model used in MA1 is referred to as B when it is extended with X3, that is, when a bidirectional decision context is used, and the data is annotated with only voice information, and as B′ when it is extended with

---

[6]An alternative approach, where the polyphonic information with which the data is annotated is updated note for note with the information provided by the second-pass model when this model is applied to unseen data, brings with it several complications and is therefore left for future work.

X3 and the data is annotated with both voice and duration information.

3.1.7   Features

Feature design is an essential aspect of the modelling, as the features must capture information that is relevant to the task at hand. This section concerns the features used in MA1 and MA2 only; the only feature used in MA3, pitch, is exactly the same as its counterpart in these modelling approaches. The features used are taken from the same superset, which contains features belonging to four different categories. With each successive category, the scope of the features contained by it widens:

(1)   Note-level features, capturing individual properties of a note (for example, pitch).

(2)   Note-chord features, capturing aspects of a note's position within a chord (for example, the note's sequence number in a chord).

(3)   Chord-level features, capturing properties shared by all notes in a chord (for example, their metric position).

(4)   Polyphonic embedding features, capturing aspects of the polyphonic relation of a note to another note to the left or to the right (for example, proximity in pitch or time), or of polyphonic relations within the chord (for example, the number of voice inversion pairs).

In what follows, the features contained in each category are described in detail. Most of the features in all categories but the first are used in both modelling approaches, but some of them are specific to one approach. If this is the case, it is indicated before the feature description. Features marked with an asterisk (*), lastly, apply only to the tablature dataset.

3.1.7.1   Feature descriptions, categories (1)–(3)

The following note-level features are defined:

pitch The pitch of the note, expressed as a MIDI number. On the tablature dataset, the same tuning (G) is assumed for all pieces, so that each tablature symbol always corresponds to the same pitch.[7]

---

[7]Of course, this does not hold when the standard tuning is used for one piece and a scordatura tuning for another. The most common scordatura tuning, where only the lowest-sounding course is altered (see Section 2.2.1) is used in only two pieces in the dataset (see Table 4.2). The effect of scordatura tunings on this feature can in this case thus be

course* The course the note is played on. As per convention, the highest-sounding course is numbered 1 (see Section 2.2).

fret* The position on the fretboard where the course is stopped. An open course yields a value of 0.

duration The note's duration. On the tablature dataset, this is the minimum duration, that is, the duration as indicated in the tablature; on the Bach dataset, where the concepts of minimum and maximum (see below) note duration as used on the tablature dataset do not apply, this is the actual, full duration. On the tablature dataset, durations are measured in breves; on the Bach dataset, they are measured in whole notes (see also Section 3.1.2.3, footnote 3).

maxDuration* The note's duration as determined by the next note on the same course.

isOrnamentation (MA1) True (1) if the note's duration is a fusa (or a sixteenth note on the Bach dataset) or shorter and the only note in the chord, false (0) if not.

The note-chord features comprise:

indexInChord The index (based on pitch) in the chord. Any sustained previous notes or interrupting next notes (if applicable) are included.

pitchDistToNoteBelow (MA1) The distance in pitch, measured in semitones, to the note below in the chord. Any sustained previous notes or interrupting next notes (if applicable) are included.

pitchDistToNoteAbove (MA1) The distance in pitch, measured in semitones, to the note above in the chord. Any sustained previous notes or interrupting next notes (if applicable) are included.

The chord-level features comprise:

chordSize The number of notes in the chord. Any sustained previous notes or interrupting next notes (if applicable) are included.

metricPosition The note's metric position in the bar. The metric position is not relative to the meter used; the metric position for a note at onset time 0.5, for example, will be 0.5 both in a $\frac{2}{2}$ meter and in a $\frac{3}{2}$ meter.

numNotesNextChord (MA1) The number of new notes (that is, notes

---

said to be minimal.

that are not sustained from previous chords) in the next chord. This feature is expected to be of use for the duration class decision when modelling voice and duration simultaneously in forward processing mode.

intervals A vector the size of the maximum number of intervals in a chord (that is, the maximum number of voices -1), encoding the intervals, measured in semitones, in the chord. Any sustained previous notes or interrupting next notes (if applicable) are included. For each interval the chord is short of the maximum number of intervals, a default value of -1 is added.

3.1.7.2   Feature descriptions, category (4) (polyphonic embedding features)

The polyphonic embedding features form the largest and most important category. They comprise:

pitchProx A vector the size of the maximum number of voices, encoding for each voice v the proximity in pitch of the note to the adjacent note in v. If a voice is not in use, the corresponding vector element is set to the default value -1. If a voice is in use but there exists no adjacent note (that is, if the note the proximity is calculated for is the first or final note in a voice), the corresponding vector element is also set to this default value.

iOProx As above, now encoding for each voice v the proximity in inter-onset time of the note to the adjacent note in v.

oOProx As above, now encoding for each voice v the proximity in offset-onset time of the note to the adjacent note in v.

These features require a somewhat more elaborate explanation. *Proximity*, first, is defined as the inverted distance, where distance in pitch is measured in semitones, and distance in time in breves (tablature dataset) or whole notes (Bach dataset). Proximity is favoured over distance as it emphasises differences between smaller distances. The position of the *adjacent note* in a voice, second, depends on the processing mode used: in forward processing mode, it is the previous note (the note to the left) in the voice, while in backward processing mode, it is the next (the note to the right). The proximities are calculated as follows. Let n_t be the note the class decision is made for (MA1), n_t^v the note mapped to voice v under mapping m (MA2), and $n_{t\pm1}^v$ the adjacent note in voice v. Furthermore, let $\mathrm{p}(n)$ denote a note's

pitch, $\mathrm{on}(n)$ its onset time, and $\mathrm{off}(n)$ its offset time. For each voice v is calculated:

▶   The proximity in pitch of n_t (or, in MA2, n_t^v) to $n_{t\pm1}^v$

$$\texttt{pitchProx}(v) = \frac{1}{|\mathrm{p}(n_t) - \mathrm{p}(n_{t\pm1}^v)| + 1}, \qquad (3.3)$$

where values fall in the range $(0, 1]$.

▶   The proximity in inter-onset time of n_t (or n_t^v) to $n_{t\pm1}^v$

$$\texttt{iOProx}(v) = \begin{cases} \frac{1}{(\mathrm{on}(n_t) - \mathrm{on}(n_{t\pm1}^v)) + 1} & \text{if } \mathrm{on}(n_{t\pm1}^v) < \mathrm{on}(n_t) \\ \frac{1}{(\mathrm{on}(n_t) - \mathrm{on}(n_{t\pm1}^v)) - 1} & \text{if } \mathrm{on}(n_{t\pm1}^v) > \mathrm{on}(n_t) \end{cases}, \quad (3.4)$$

where values fall in the range $(0, 1)$ if the upper case applies, that is, if the adjacent note is to the left, and in the range $(-1, 0)$ if the lower case applies, that is, if the adjacent note is to the right.

▶   The proximity in offset-onset time of n_t (or n_t^v) to $n_{t\pm1}^v$

$$\texttt{oOProx}(v) = \begin{cases} \frac{1}{(\mathrm{on}(n_t) - \mathrm{off}(n_{t\pm1}^v)) + 1} & \text{if } \mathrm{off}(n_{t\pm1}^v) \leq \mathrm{on}(n_t) \\ \frac{1}{(\mathrm{on}(n_t) - \mathrm{off}(n_{t\pm1}^v)) - 1} & \text{if } \mathrm{off}(n_{t\pm1}^v) > \mathrm{on}(n_t) \end{cases} \quad (3.5)$$

if the adjacent note is to the left, where values fall in the range $(0, 1]$ if the upper case applies and in the range $(-1, 0)$ if the lower applies, and

$$\texttt{oOProx}(v) = \begin{cases} \frac{1}{(\mathrm{off}(n_t) - \mathrm{on}(n_{t\pm1}^v)) - 1} & \text{if } \mathrm{off}(n_t) < \mathrm{on}(n_{t\pm1}^v) \\ \frac{1}{(\mathrm{off}(n_t) - \mathrm{on}(n_{t\pm1}^v)) + 1} & \text{if } \mathrm{off}(n_t) \geq \mathrm{on}(n_{t\pm1}^v) \end{cases} \quad (3.6)$$

if the adjacent note is to the right, where values fall in the range $(-1, 0)$ if the upper case applies, and in the range $(0, 1]$ if the lower applies. It should be noted that on the tablature dataset, the offset time is determined by the minimum duration of a note when modelling only voice, but by the full duration of a note when modelling voice and duration simultaneously. On the Bach dataset, it is always determined by the full duration.

With respect to the features $\texttt{pitchProx}$, $\texttt{iOProx}$, and $\texttt{oOProx}$, three things should be noted. As shown above, the proximities have a slightly different meaning in the different modelling approaches: in MA1, the proximities of n_t to the adjacent notes in *all* voices are calculated, whereas in

MA2, for each voice v the proximities of n_t^v to the adjacent note in v are calculated. Second, as becomes clear from the above, the size of the decision context (see Section 3.1.6) for a note or chord, which varies from note to note and from chord to chord, is determined in the calculation of the proximities. Third, the proximity features correspond directly to the two perceptual principles that most of the existing voice separation systems are based on (see Section 2.3.2)—the Pitch Proximity Principle and the Principle of Temporal Continuity (Huron, 2001), which dictate that the closer two notes are to one another in terms of pitch or time, respectively, the more likely they are perceived as belonging to the same voice.

Furthermore, the following polyphonic embedding features are used:

adjNoteOnSameCourse* (MA1) A binary vector the size of the maximum number of voices, encoding all voices with an adjacent note on the same course by setting the corresponding vector elements to 1. If a voice is not in use, the corresponding vector element is set to 0. This feature is expected to be useful because on the lute there is often a preference to play, if technically possible, notes belonging to the same voice on the same course.

voicesOccupied A binary vector the size of the maximum number of voices, encoding all voices that are already occupied in the chord by setting the corresponding vector elements to 1. Any sustained previous notes or interrupting next notes (if applicable) are included. If a voice is not in use, the corresponding vector element is set to 0.

pitchMovements (MA2) A vector the size of the maximum number of voices, encoding for each voice v the pitch movement (measured in semitones, $+$ or $-$) of n_t^v with respect to $n_{t\pm1}^v$. If a voice is not in use, the corresponding element in the vector is set to 0. If a voice is in use but there exists no adjacent note, the corresponding vector element is also set to 0.

pitchVoiceRelation (MA2) The Pearson product-moment correlation coefficient r, encoding the relation between the chord's pitch ordering and voice ordering:

$$r = \frac{n(\sum_{i=1}^n P_i V_i) - \sum P \sum V}{\sqrt{n(\sum_{i=1}^n (P_i)^2) - (\sum P)^2}\sqrt{n(\sum_{i=1}^n (V_i)^2) - (\sum V)^2}}, \quad (3.7)$$

where n is the number of data samples (that is, the number of pitches, or, as the case may be, voices in the chord), P the set of pitches in the chord, and V the set of voices in the chord. Any sustained previous notes (if applicable) are included.

numVoiceInversionPairs (MA2) The number of voice inversion pairs in the chord. Any sustained previous notes (if applicable) are included.

sumDistVoiceInversionPairs (MA2) The distance in pitch (measured in semitones) between the inverted voices in a voice inversion pair, summed over all pairs. Any sustained previous notes (if applicable) are included.

avgDistVoiceInversionPairs (MA2) The distance in pitch (measured in semitones) between the inverted voices in a voice inversion pair, averaged over all pairs. Any sustained previous notes (if applicable) are included.

mapping (MA2) The current mapping m of notes to voices. Including the mapping is useful as it is possible that two different mappings yield the exact same feature vector. By simply including the mapping—each of which is unique—in the feature vector, this problem is solved.

3.1.7.3   Organisation in feature vectors

From the superset described above, the feature vectors for both modelling approaches are constructed. Depending on the approach, the feature vectors are organised differently. In MA1, they are organised along the main feature categories as described in Section 3.1.7: note-level features, note-chord features, chord-level features, and polyphonic embedding features. Table 3.2 shows the feature vector for the unidirectional model, which for the tablature dataset is 41-dimensional, and for the Bach dataset 33-dimensional. The same features and organisation are used for the bidirectional model—only now the polyphonic embedding features 16–35 (tablature dataset) or 13–27 (Bach dataset) are calculated in their entirety twice: first for the adjacent notes to the left, and then again for the adjacent notes to the right. For the tablature dataset this yields an additional 20 features, resulting in a 61-dimensional feature vector; for the Bach dataset, the 15 additional features result in an 48-dimensional feature vector.

Note that for the tablature dataset, where each note in a chord has the same minimum duration as given in the tablature, the `duration` feature is a chord-level feature. However, for the Bach dataset, where chord notes can have different durations, it must be a note-level feature. For the latter dataset, it thus replaces the non-applicable `maxDuration` feature.

In MA2, the feature vectors are organised slightly differently. The only note-chord feature that applies here, `indexInChord`, is added to the note-

Table 3.2 MA1, feature vector for the unidirectional model: note-level features (top), note-chord features (upper middle), chord-level features (lower middle), and polyphonic embedding features (bottom).

Tablature dataset	Bach dataset	Feature
0	0	`pitch`
1		`course`
2		`fret`
3		`maxDuration`
	1	`duration`
4	2	`isOrnamentation`
5	3	`indexInChord`
6	4	`pitchDistToNoteBelow`
7	5	`pitchDistToNoteAbove`
8	6	`chordSize`
9		`duration`
10	7	`metricPosition`
11	8	`numNotesNextChord`
12–15	9–12	`intervals`
16–20		`adjNoteOnSameCourse`
21–25	13–17	`pitchProx`
26–30	18–22	`iOProx`
31–35	23–27	`oOProx`
36–40	28–32	`voicesOccupied`

level features; the original fourth category, the polyphonic embedding features, now forms the third category. The features in the first category are different for each note in the chord; those in the second category are the same for each note in the chord but different from chord to chord, and those in the last category are different for each mapping m that is enumerated for the chord, as each mapping results in a different polyphonic embedding. An additional difference with MA1, as already mentioned in Section 3.1.3.4, concerns the assumed maximum number of voices. The note-specific features are calculated for each note in the chord, where a default value of -1 is added for each note the chord is short of its maximum size (which, when a voice is considered to be monophonic, equals the maximum number of voices). In order to avoid having unnecessary default values in each feature vector, in MA2 the maximum number of voices is therefore set to the highest number of voices in the datasets used—four. As shown in Table 3.3, in MA2 a 54-dimensional feature vector is thus used for the tablature dataset, and a 45-dimensional feature vector for the Bach dataset.

Note that for the tablature dataset, as in the feature vector for MA1, the

Table 3.3 MA2, feature vector: note-level features (top), chord-level features (middle), and polyphonic embedding features (bottom).

Tablature dataset	Bach dataset	Feature
0, 5, 10, 15	0, 3, 6, 9	`indexInChord`
1, 6, 11, 16	1, 4, 7, 10	`pitch`
2, 7, 12, 17		`course`
3, 8, 13, 18		`fret`
4, 9, 14, 19		`maxDuration`
	2, 5, 8, 11	`duration`
20	12	`chordSize`
21		`duration`
22	13	`metricPosition`
23–25	14–16	`intervals`
26–29	17–20	`pitchProx`
30–33	21–24	`iOProx`
34–37	25–28	`oOProx`
38–41	29–32	`pitchMovements`
42–45	33–36	`voicesOccupied`
46	37	`pitchVoiceRelation`
47	38	`numVoiceInversionPairs`
48	39	`sumDistVoiceInversionPairs`
49	40	`avgDistVoiceInversionPairs`
50–53	41–44	`mapping`

`duration` feature is a chord-level feature, while for the Bach dataset it is again a note-level feature.

3.1.7.4   Scaling

The individual feature values can vary considerably. Pitch, for example, is a discrete feature whose value moves roughly between 30 and 90 in steps of 1, the various proximities are continuous features whose values move between -1 and 1, and the binary and boolean features can only take values of 0 and 1. These differences can hinder the learning performance of the model. In both modelling approaches, each feature f is therefore scaled so that it falls in the range $[0, 1]$:

$$f'_i = \frac{f_i - \min(F_i)}{\max(F_i) - \min(F_i)},$$ (3.8)

where f_i is the original value of the feature at index i in the feature vector, f'_i the scaled value, and F_i the set of all values at index i over all feature vectors in the training set. (This approach has the consequence that on the

test set, scaled values can occasionally exceed the range $[0, 1]$, namely when $f_i < \min(F_i)$ or when $f_i > \max(F_i)$. The excess, however, is typically only small and therefore not detrimental to model performance.) Features that have been given a default value are not scaled and retain this default value; furthermore, when $\max(f_i)$ and $\min(f_i)$ are equal (causing division by 0), f_i' is set to 0.

3.2   Evaluation

The models are evaluated on the three-voice and four-voice subdatasets separately, and, irrespective of the modelling approach, always on the note-level. Evaluation is carried out using cross-validation, where three different evaluation metrics are used: accuracy (the main evaluation metric), soundness, and completeness. Accuracy is a per-note metric; soundness and completeness measure transitions between note pairs.

3.2.1   Cross-validation procedure

Evaluation is carried out using k-fold cross-validation, where a dataset is divided into k subsets, and a model is trained on the total content of $k - 1$ subsets and then tested on the remaining subset. This is repeated k times until all subsets have served as test set once. k is set equal to the number of pieces in a dataset, making each piece a subset. This procedure is preferred over a random division into subsets, as samples from the same piece may contain regularities, possibly even verbatim repetitions. If such samples end up both in the training and the test set, the results are unrepresentative for unseen data. By performing piecewise cross-validation, on the other hand, the results can be used as estimates for results on unseen data of the same type.

The evaluation metrics, measured on both the training and the test sets, are averaged over all folds. The per-fold percentages for each evaluation metric m are weighted by the number of notes (or, for soundness and completeness, note pairs) in the subset in that fold, so that the average values over all folds are always per-note (or per-pair) rates:

$$\mathrm{avg}(m) = \frac{\sum_{i=1}^{k}(m_i \cdot |N_i|)}{\sum_{i=1}^{k} |N_i|},\tag{3.9}$$

where k is the number of folds, and N the set of notes in a subset. The average is calculated in the same fashion for all evaluation metrics.

3.2.2   Evaluation metrics

The evaluation metrics used—*accuracy*, *soundness*, and *completeness*—are defined as follows:

Accuracy Accuracy (acc) measures the percentage of notes that have been assigned to the correct voice:

$$\text{acc} = \frac{|C|}{|N|} \cdot 100, \qquad (3.10)$$

where C is the set of notes assigned to the correct voice in a subset, and N the set of all notes in that subset. When modelling voice and duration simultaneously, accuracy is also used to measure the percentage of notes that have been assigned to the correct duration. Therefore, *voice accuracy* and *duration accuracy* are discerned.

Soundness Soundness (snd) measures correct transitions between notes. The term is borrowed from the field of logic and has been redefined for use in voice separation research by Kirlin and Utgoff (2005). Let f be an assigned voice and g a correct voice; Kirlin and Utgoff then consider a pair of adjacent notes (n_t, n_{t+1}) in f to be *sound* if $g(n_t) = g(n_{t+1})$ holds. Note that, according to the definition, f and g need not be the same voice. Extending this definition, in this thesis soundness is taken to be the percentage of sound pairs in *all* voices:

$$\text{snd} = \frac{|S|}{|P|} \cdot 100, \qquad (3.11)$$

where S is the set of sound pairs in a subset, and P the set of all pairs in all assigned voices f in that subset. Soundness thus reflects the percentage of adjacent note pairs whose elements have been assigned to the same voice and actually belong to the same voice.

Completeness Completeness (cmp) is a metric complementary to soundness; the term is again borrowed from logic. A pair of adjacent notes (n_t, n_{t+1}) in correct voice g is *complete* if assigned voice $f(n_t) = f(n_{t+1})$ holds (Kirlin and Utgoff, 2005). Again, f and g need not be the same voice. Analogous to soundness, completeness is thus taken to be the percentage of complete pairs in all voices:

$$\text{cmp} = \frac{|C|}{|P|} \cdot 100, \qquad (3.12)$$

where C is the set of complete pairs in a subset, and P the set of all pairs in all correct voices g in that subset. Completeness

reflects the percentage of adjacent note pairs whose elements actually belong to the same voice and have been assigned to the same voice (or, in other words, the percentage of correct adjacent note pairs retrieved).

3.2.2.1 Statistical significance

In this thesis, the Wilcoxon signed-rank test (Wilcoxon, 1945) is used to determine whether the medians of two sets of accuracy measurements are, statistically speaking, significantly different from each other (a set of accuracy measurements consists of the per-fold accuracy values obtained in cross-validation). The outcome of the test can be used to validate statements about how these two results compare. The significance criterion is set to 0.05 (this is one of the standard values); if the test yields a p-value that is smaller, the difference is considered to be statistically significant.

The three tests used most commonly to test for statistical significance are the Wilcoxon signed-rank test, the sign test, and the Student's t-test (Student, 1908). The former two are both non-parametric tests, while the latter assumes the data to be normally distributed. The fact that this assumption is generally speaking not valid makes the Student's t-test less attractive. The Wilcoxon signed-rank test is preferred over the sign test, then, for the simple reason that it is more powerful.

It must be noted that using a statistical significance test is only one way of validating statements about results, and that the outcome of such a test is not to be interpreted too rigidly.

3.2.3 Evaluation of single-note unisons

The evaluation of voice and duration assignments for single-note unisons needs to be discussed in some more detail. Each note can be a single-note unison; as is shown in Table 3.4, for each note there are thus nine possible voice assignment scenarios. Five assignment categories, then, cover all these scenarios. The most straightforward scenarios are (1) and (7), which always fall in the *correct* voice assignment category, and scenarios (2), (4), (6), and (9), which always fall in the *incorrect* category. The remaining scenarios (3), (5), and (8) yield neither fully correct nor fully incorrect voice assignments; therefore, three additional assignment categories are needed. Scenario (3), where a note that is a single-note unison has been assigned to only one voice and that voice is a correct voice, falls in the *overlooked* category; scenario (8), where a note that is a single-note unison has been assigned to two voices but

Table 3.4 Voice assignment scenarios and corresponding voice assignment categories. AV = assigned voice(s), CV = correct voice(s).

Scenario	Voices		Description	Voice assignment
	AV	CV		category
(1)	1	1	AV is CV	*correct*
(2)			AV is not CV	*incorrect*
(3)	1	2	AV is one of CV	*overlooked*
(4)			AV is none of CV	*incorrect*
(5)	2	1	one of AV is CV	*superfluous*
(6)			none of AV is CV	*incorrect*
(7)	2	2	both AV are CV	*correct*
(8)			one of AV is CV	*half*
(9)			none of AV are CV	*incorrect*

only one of these is a correct voice, falls in the *half* category; and scenario (5), where a note that is *not* a single-note unison has been assigned to two voices, one of which is its correct voice, falls into the *superfluous* category. When determining the accuracy, such voice assignments count as halves (meaning that they increase the number of notes assigned to the correct voice only by 0.5).

The evaluation of duration assignments for single-note unisons proceeds similarly. Although single-note unisons with two different durations do occur, in their current form the models only classify a note into a single duration class. This yields three duration assignment categories: *correct*, where a note that is either not a single-note unison or a single-note unison with only one duration has been assigned to the correct duration; *overlooked*, where a note that is a single-note unison with two different durations has been assigned to only one correct duration; and *incorrect*, which applies in all other cases. Again, the overlooked duration assignments count as halves.

3.2.4 Evaluation modes and error propagation

The evaluation metrics on the training data are informative about a model's capacity to adapt to that data, that is, to learn mappings from inputs to outputs. The evaluation metrics on the test data, on the other hand, are informative about a model's ability to generalise on unseen data, that is, to perform the task on unseen data. In MA1 and MA2, the models are evaluated on both the training and the test data, where the evaluation on the test data differs depending on whether a unidirectional or a bidirectional model is used. In MA3, the models are only evaluated on the test data.

3.2.4.1   Evaluation of unidirectional models

A unidirectional model, first, is evaluated on the test data in two different modes, called *test mode* and *application mode*:

Test mode In test mode, the feature vectors for note n or chord c are calculated using the correct polyphonic information (that is, the information that is also used as targets in the training) for all notes or chords processed before n or c.

Application mode In application mode, the feature vectors for n or c are calculated using the polyphonic information that is generated by the model for all notes or chords processed before n or c. Two things should be noted: first, as already mentioned in Section 3.1.6, the generated information grows incrementally with every note or chord that is processed and for which a class or mapping decision is made. Second, the generated information may contain errors.

Application mode corresponds to the real-world situation where the correct polyphonic information is not available, and where all voice class or mapping decisions are based on previous decisions. The metrics in application mode thus reflect the *actual* performance of the model on unseen data. Why then test mode? First, the metrics in test mode, where the correct polyphonic information is always encoded in the feature vector, reflect the *optimal* performance of the model on unseen data. Test mode thus serves a kind of gauging function. Second, the results in test mode are instrumental in quantifying *error propagation*. Error propagation is the phenomenon in which an incorrect voice or duration assignment influences the voice class or mapping decision for the following notes negatively. In test mode, where the polyphonic information generated by the model is not used in the feature calculation, and incorrect assignments thus have no further consequence, this effect is suppressed. In application mode, however, where the polyphonic information as generated by the model is used in the feature calculation, an incorrect assignment can set in motion a chain of incorrect assignments.

Error propagation can be quantified by comparing results in test and application mode. Suppose, for example, that when a trained model is applied to a piece consisting of 100 notes, in test mode five of these notes are assigned to the wrong voice. This thus yields a voice accuracy of 95%. In application mode, the same initial misassignments will be made—but because the polyphonic information generated by the model is now used in the feature calculation, each misassignment runs the risk of causing further misassignments. Suppose now that in application mode, each initial misas-

signment indeed leads to another misassignments. This then leads to a total of 10 misassignments, which yields a voice accuracy of 90%. The percentage of misassignments due to error propagation in application mode, or *error propagation percentage*, m, is calculated as follows:

$$m = \frac{\text{acc}_\text{T} - \text{acc}_\text{A}}{100 - \text{acc}_\text{A}} \cdot 100, \tag{3.13}$$

where acc_T and acc_A are the accuracies in test and application mode, respectively.

3.2.4.2   Evaluation of bidirectional models

A bidirectional model, on the other hand, is only evaluated in a single mode on the test data. This mode shall henceforth be referred to as (bidirectional) application mode, but it can be said to be midway between test and (unidirectional) application mode. As explained in Section 3.1.6, both on the training and the test data, the feature vectors are now calculated using the polyphonic information as provided by the first-pass model. Bidirectional application mode is thus similar to test mode in that it uses pre-existing polyphonic information in the feature calculation, but it is also similar to unidirectional application mode in that the results now reflect model performance on unseen data in the real-world situation where the correct polyphonic information is not known. Lastly, because when applied to unseen data, pre-existing polyphonic information—and not information generated by the model—is used in the feature calculation, the model does not suffer from error propagation.

3.3   Conflicts

As mentioned in Section 3.1.2.4, when applying a trained model to unseen data in MA1, it may occur that a voice class decision is made that *conflicts* with a previous voice class decision—meaning that two simultaneous or overlapping notes (further specification follows below) will be assigned to the same voice. Such conflicts must be resolved because, first, each voice is assumed to be monophonic, and second, since the implementation of the feature calculation is based on this assumption, the calculation of a number of polyphonic embedding features (see Section 3.1.7.2) becomes problematic if it is not true.[8]

---

[8]The main issue concerns inconsistencies with respect to which of the two simultaneous or overlapping notes in a voice is considered when the feature vector for a note that is processed later is calculated.

When applying a trained model to unseen data in *test mode*, however, the polyphonic information generated by the model is not used in the feature calculation. Conflicting voice class decisions thus have no direct implications: although voices may be non-monophonic after the voice assignment process has terminated, the process itself is not influenced by any conflicts. Because test mode primarily serves a gauging function and the actual outcome is of lesser importance, in this mode conflicts are not resolved. When applying a trained model to unseen data in *unidirectional application mode*, then, where the polyphonic information generated by the model is used in the feature calculation and conflicting voice class decisions thus create problems, conflicts are resolved *during* the voice assignment process. Conflict resolution in unidirectional application mode is described first, in Section 3.3.1. When applying a trained model to unseen data in *bidirectional application mode*, lastly, as in test mode, the polyphonic information generated by the model is not used in the feature calculation (see Section 3.1.6). However, because application mode corresponds to the real-world application situation, conflicts must be resolved. This is now done *after* the voice assignment process. This has the advantage that more context information is available, which makes conflict resolution decidedly easier, and enables the use of a more straightforward postprocessing heuristic. Conflict resolution in bidirectional application mode is described in Section 3.3.2.

Conflicts can occur between the decision for a note n and:

►   The decision for a preceding note in the same chord.
►   The decision for a sustained previous note.
►   The decision for an interrupting next note.

In unidirectional application mode, conflicts with sustained previous notes are specific to the forward processing mode, while those with interrupting next notes are specific to the backward processing mode. In bidirectional application mode, both these conflicts may occur in either processing mode—but as will become clear in Section 3.3.2, because of the conflict resolution heuristic used, only those with sustained previous notes are relevant.

In the following Sections 3.3.1 and 3.3.2, a systematic overview of all conflicts that can occur is presented, and for each conflict the resolution procedure is described. The conflict resolution procedures used are mostly pragmatic; although they ensure a musically acceptable outcome, this outcome is not always optimal. Finding the optimal resolution for a conflict often requires context information that is not available to the model; additionally, it can complicate the resolution procedure greatly (for example, multiple decisions may have to be remade, and any new conflicts emerging

in the process resolved). Because conflicts occur only with low frequency and the effect of conflicts on model performance is thus marginal (see Section 5.7.1), a further optimisation of the conflict resolution procedures is left for future work.

It should be noted, lastly, that a full understanding of the material presented in Sections 3.3.1 and 3.3.2, both of which provide highly detailed descriptions of the different conflicts and their (at times fairly complex) resolution procedures, is not required for a full understanding of the remainder of the thesis.

3.3.1   Conflict resolution in unidirectional application mode

When applying a trained model to unseen data in unidirectional application mode, three basic types of conflicts are discerned. Type (i) applies to both the tablature and the Bach dataset; type (ii) and (iii), which involve single-note unisons, are specific to the tablature dataset. Where a single-note unison is concerned in a conflict—be it a type (ii), (iii), or any other conflict as described below—, a distinction is made between the *primary voice class*, that is, the voice class corresponding to the highest activation value in the model output, and the *secondary voice class*, that is, the one corresponding to the position of the second-highest activation value (see also Section 3.1.2.1). The three basic conflict types are as follows:

(i)   The (primary) voice class under the current decision is equal to the (primary) voice class under the previous decision (voice or duration conflict).

(ii)   The primary voice class under the current decision is equal to the secondary voice class under the previous decision (voice or duration conflict).

(iii)   The secondary voice class under the current decision is equal to the primary or secondary voice class under the previous decision (voice or duration conflict).

Examples of type (i), (ii), and (iii) conflicts are shown in Figures 3.8–3.10. For each type, a situation where there is a conflict with the decision for a preceding chord note, a situation where there is a conflict with the decision for a sustained previous note, and a situation where there is a conflict with the decision for an interrupting next note is shown.

```
3(A)----                        5(A)----        5(A)------------
2(A)----            2(A)------------               2(A)----
1(T)----            1(T)----4(T)----    4(T)----1(T)----
0(B)----            0(B)----3(B)----    3(B)----0(B)----
```

(a) With decision for a preceding chord note.

(b) With decision for a sustained previous note.

(c) With decision for an interrupting next note.

Figure 3.8 Schematic view of type (i) conflicts in a four-voice piece. Notes in order of processing; the conflict occurs at the note printed in bold.

```
3(S)----                        5(S)----        5(S)------------
2(A,S)--            2(A,S)---------               2(A,S)--
1(T)----            1(T)----4(T)----    4(T)----1(T)----
0(B)----            0(B)----3(B)----    3(B)----0(B)----
```

(a) With decision for a preceding chord note.

(b) With decision for a sustained previous note.

(c) With decision for an interrupting next note.

Figure 3.9 Schematic view of type (ii) conflicts in a four-voice piece. Notes in order of processing; the conflict occurs at the note printed in bold.

```
3(S,A)--                        5(S,A)--        5(S,A)----------
2(A)----            2(A)------------               2(A)----
1(T)----            1(T)----4(T)----    4(T)----1(T)----
0(B)----            0(B)----3(B)----    3(B)----0(B)----
```

(a) With decision for a preceding chord note.

(b) With decision for a sustained previous note.

(c) With decision for an interrupting next note.

Figure 3.10 Schematic view of type (iii) conflicts in a four-voice piece. Notes in order of processing; the conflict occurs at the note printed in bold.

Table 3.5 shows how the different types of conflicts are resolved. In the case where a voice class decision is *adapted* (type (i) conflicts), the highest activation value in the model output for the note is set to 0, and the decision is remade by determining the new highest activation value. In the case where a voice class decision is *reduced* (type (ii) and (iii) conflicts), meaning that a classification into two voice classes is reduced to one into a single voice

Table 3.5 Conflict resolution per conflict type in unidirectional application mode. Tablature and Bach datasets. Conflicts with sustained previous notes and interrupting next notes are specific to forward and backward processing mode, respectively; on the tablature dataset, they apply only when modelling voice and duration simultaneously. n = current note, c = chord note, s = sustained previous note, i = interrupting next note, VC = voice class decision, DC = duration class decision.

Decision for n conflicts with decision for	Tablature dataset			Bach dataset
	Type (i)	Type (ii)	Type (iii)	Type (i)
c	VC(n) adapted	VC(c) reduced	VC(n) reduced	VC(n) adapted
s	DC(s) adapted	DC(s) adapted	VC(n) reduced	VC(n) adapted
i	DC(n) adapted	DC(n) adapted	VC(n) reduced	VC(n) adapted

class, the second-highest activation value—to whose position the secondary voice class corresponds—in the model output for the note is ignored, and the decision is remade. In the case where a duration class decision is adapted, lastly, the note's maximum duration (its onset time subtracted from the onset time of the next note in the same voice) is determined, and the decision is remade. A duration class decision adaptation is thus always a reduction of the duration.

Conflicts are resolved in a fixed, multi-step sequence:

(1)   Resolve any type (i) conflicts with decisions for any sustained previous notes or interrupting next notes (depending on the processing mode).

(2)   Resolve any type (ii) conflicts with decisions for any sustained previous notes or interrupting next notes.

(3)   Resolve any type (i) conflicts with decisions for any lower chord notes. If such conflicts are encountered, the adapted voice class decision for the current note may result in new conflicts with decisions for sustained previous or interrupting next notes; thus, steps (1) to (3) must be repeated until no more conflicts occur.

(4)   Resolve any type (ii) conflicts with decisions for any lower chord notes.

(5)   Resolve any type (iii) conflicts with decisions for any sustained previous, interrupting next, or lower chord notes.

All steps do not always apply; those necessary depend firstly on the dataset, and secondly, in the case of the tablature data, on whether only voice is modelled or voice and duration simultaneously.

```
4(S)------------
          6( )
      3(A)-----------
1(T)----2(T)----5( )
0(B)--------------------
```
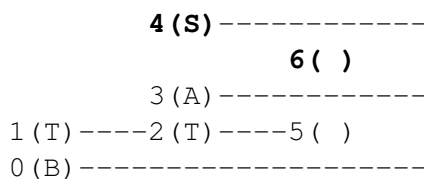
Figure 3.11 Schematic view of a type (iv) conflict in a four-voice piece. Notes in order of processing; the conflict occurs at the second note printed in bold but is resolved at the first. Empty parentheses indicate that a note has not yet been processed.

```
6( )------------
5(A)----2(S)
4(T)----1(T)
3(B)----0(B)
```
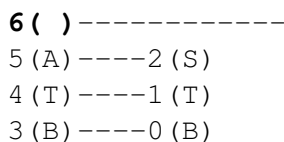
Figure 3.12 Schematic view of a type (v) conflict in a four-voice piece. Notes in order of processing; the conflict occurs at the note printed in bold. Empty parentheses indicate that a note is currently being processed.

3.3.1.1   Special conflicts: type (iv) and (v) conflicts

Apart from the three basic conflict types, there are two special types of conflicts, both of which are resolved in an additional last step of the conflict resolution sequence. They both have to do with the number of voices in a piece being exceeded. Recall that the number of voices in a dataset is determined prior to the training by finding the largest chord in the set (see Section 3.1.2.5).

Type (iv) conflicts, first, only apply to the tablature dataset, and only occur in forward processing mode when modelling voice and duration simultaneously. They can occur to multiple notes simultaneously. After all notes in a chord c_t have been processed, it may be the case that, due to incorrect duration class decisions for notes in this chord (and possibly in previous chords), the next chord c_{t+1} now contains more notes than there are voices in the piece. An example of such a situation is shown in Figure 3.11, where the last chord contains five notes due to incorrect duration class decisions for any of the notes 0, 3, or 4.

Let n be the total number of notes in c_{t+1}, v be the number of voices in the piece, and V be the maximum number of voices (five). If $v < n \leq V$,

this will cause problems when the voice class decisions for the notes in c_{t+1} are made: the voice classes for the last $n-v$ notes will have to be classes not in use in the piece. In Figure 3.11, for example, after the voice class decision for note 5 has been made, for note 6 only the fifth voice class—which is not in use—remains. If $n > V$, this even precludes the calculation of a feature vector for the notes in c_{t+1}, as several features are based on the assumption that there is a maximum number of voices (see Sections 3.1.7.1 and 3.1.7.2). Thus, if after the last note in c_t has been processed, a type (iv) conflict is detected in c_{t+1}, a list is made containing all notes with an onset time lower than that of c_{t+1} whose offset time exceeds the onset time of c_{t+1}. The notes in this list are ordered firstly by onset time (lower first), and secondly by position in the chord (lower first). The duration class decision for the last $n-v$ notes in this list is then remade by determining their minimum duration. In the case of Figure 3.11, the duration class decision is thus only remade for the last note in the list, note 4.

Type (v) conflicts, second, only apply to the Bach dataset, and only occur in backward processing mode. Technically, a type (v) conflict is a reinstated type (i) conflict. Let n_t be the note the voice class decision is made for in chord c_t. If n_t has an offset time that exceeds the onset time of at least one next chord c_{t+x}, due to type (i) conflicts with incorrectly classified notes in c_{t+x} or possibly in c_t, it may happen that only voice classes not in use in the piece (or, in the case of a five-voice piece, no voice classes at all) are left available for n_t. An example of such a situation is shown in Figure 3.12, where because of the misclassification of note 2 to the superius, for note 6 only the fifth voice class—which is not in use in the piece—is left available.

If this situation arises (which is always at the end of the conflict resolution sequence), rather than going through the complicated process of adapting class decisions for interrupting next or lower chord notes—which is likely to result in new conflicts with other previous decisions, which then would have to be adapted, etc.—the original voice class decision is simply restored. This is a temporary solution at best; the conflict remains unresolved, meaning that there is a voice that, at a certain point, contains two simultaneous notes. The systematic resolution of type (v) conflicts is left for future work, firstly because of the complications it entails, and secondly, because they occur fairly rarely and thus do not influence the model performance significantly.

```
                              4(S)----
                   2(A)----3(T)----
                   1(T)-----------
                   0(B)-----------
```

Figure 3.13 Schematic view of a type (vi) conflict in a four-voice piece. Notes in order of processing; the conflict occurs at the note printed in bold.

```
              2(A)----
              1(A)----
              0(A,T)--
```

Figure 3.14 Schematic view of type (vii) conflicts in a four-voice piece. Notes in order of processing; the conflicts occur at the notes printed in bold.

3.3.2   Conflict resolution in bidirectional application mode

In the conflict resolution heuristic following the application of a trained model to unseen data in bidirectional application mode, the music is processed from left to right, chord for chord. To be resolved for each chord c are, firstly, any conflicts between voice class decisions for notes in c and voice class decisions for sustained previous notes, and secondly, any conflicts between voice class decisions for notes within c. These conflicts will henceforth be referred to as type (vi) and (vii) conflicts, respectively. It must be noted that processing from left to right is an arbitrary choice: alternatively, the music could be processed from right to left, in which case type (vi) conflicts would be conflicts between voice class decisions for notes in c and voice class decisions for interrupting next notes. Processing from right to left is not tried, as this is expected to yield similar results as processing from left to right. Yet another possibility is to take into account conflicts between voice class decisions for notes within c and voice class decisions for both sustained previous notes and interrupting next notes *simultaneously*. Although this approach is more comprehensive, it also has the strong disadvantage that it tends to become complicated fairly quickly. It is therefore also not tried.

Type (vii) conflicts can occur on both datasets; type (vi) conflicts only on the Bach dataset. This is because for the bidirectional model, modelling voice and duration simultaneously is not implemented—meaning that on

the tablature dataset sustained previous notes do not exist.[9] Both types of conflicts are resolved in a similar, multi-step process that is repeated until a chord is conflict-free. In the rare case where a chord contains both conflict types, all conflicts of type (vi) are resolved first, and subsequently all conflicts of type (vii).

The conflict resolution process is the simplest in the case of type (vi) conflicts. An example of such a conflict is given in Figure 3.13.

The steps taken to resolve the type (vi) conflict(s) in chord c are as follows:

(1)   List all notes in c whose voice class decision conflicts with that for the sustained notes; let n_0 be the first note in this list.

(2)   List all voices that are available (to all chord notes) prior to conflict resolution; excluded are thus all voices taken up by sustained notes and all voices taken up by chord notes.

(3)   Retrieve the model output for n_0 and compare the activation values for all available voices (as listed in step (2)); reassign n_0 to the available voice that gives the highest activation value.

(4)   Repeat from step (1) until all conflicts are resolved.

In the case of type (vii) conflicts, the conflict resolution process is somewhat more involved. An example of multiple type (vii) conflicts within one chord is given in Figure 3.14, where all notes have been assigned to the same voice.

The steps taken to resolve the type (vii) conflict(s) in chord c are as follows:

(1)   List all notes in c whose voice class decision conflicts with that for other notes in c.

(2)   List all voices to which more than one chord note has been assigned; let v_0 be the first voice in this list.

(3)   List all voices that are available (to all chord notes) prior to conflict resolution; excluded are thus all voices taken up by chord notes, and, on the Bach dataset only, all voices taken up by sustained notes.

(4)   Do:
    (a)   If one of the chord notes is a single-note unison *not* involving

---

[9]Note that this is unrelated to the fact that duration information acquired from a first-pass model may be used in the feature calculation. Like the voice information acquired in the first pass, this duration information serves purely to aid the second-pass model in learning to make the correct voice class decision. It is not information generated by the model in bidirectional application mode, and thus cannot cause conflicts there.

v_0 and there are no available voices because of this single-note unison: retrieve the first chord note that is a single-note unison not involving v_0; reassign the note only to the primary voice class.

(b)   Else if one of the chord notes is a single-note unison involving v_0: retrieve the first chord note that is a single-note unison involving v_0; if the primary voice class for this note is equal to v_0, reassign the note only to the secondary voice class and vice versa.

(c)   Else: retrieve for each chord note listed in (1) that has been assigned to v_0 the model output and compare in each model output the activation value for each available voice (as listed in step (3)); reassign the chord note that gives the highest activation value thus found to the available voice that yields this value.

(5)   Repeat from step (1) until all conflicts are resolved.

Here, conditions (a) and (b) only occur when the model is applied to the tablature dataset.

A concrete example may serve to illustrate the resolution of type (vii) conflicts more clearly. Given the two type (vii) conflicts as shown in Figure 3.14, and assuming that this example is taken from the tablature dataset (where there are no sustained previous notes, but where notes can be single-note unisons), step (1) yields a list containing chord notes 0, 1, and 2, step (2) a list containing the altus, which is thus v_0, and step (3) a list containing the bassus and the superius. Because note 0 is a single-note unison involving v_0, in step (4) condition (b) applies, and note 0 is reassigned to the tenor—therewith resolving the first type (vii) conflict. Starting from the top again, step (1) now yields a list containing only notes 1 and 2, step (2) a list containing the altus, which is again v_0, and step (3) a list containing the bassus and the superius. Because none of the chord notes is a single-note unison anymore, in step (4) condition (c) now applies. The question whether to reassign note 1 or 2 to one of the available voices, the bassus or the superius, is solved by determining for each of these notes the model output for each available voice, and then selecting from the resulting note-voice combinations (note 1 and the bassus, note 1 and the superius, note 2 and the bassus, and note 2 and the superius) the one that gives the highest activation value.

3.4   Implementation details

The models, the model extensions, the feature extraction algorithms, and
the framework for training and evaluating the models are implemented in
Java. For the models used in MA1 and MA2, the neural network implemen-
tation provided by the Encog Machine Learning Framework is used.[10] For the
model used in MA3, which is only partially implemented in Java, addition-
ally the Hidden Markov Model Toolbox for Matlab is used.[11] Furthermore,
the MUSITECH software infrastructure (Weyde, 2005) is used for the cre-
ation of the internal data representations (see Sections 4.1.2 and 4.2.2). The
remainder of the software is created from scratch.[12]

The framework takes as input machine-readable encodings of the tabla-
ture in plain text format together with sets of monophonic MIDI files. On
the Bach dataset, it takes as input only sets of monophonic MIDI files (the
content and function of these files is explained in more detail in Chapter 4).
Each of these encodings and MIDI file sets represents one piece in the dataset.
For each fold in the cross-validation procedure, the framework outputs three
plain text files—one when training a model, one when evaluating it in test
mode, and one when evaluating it in application mode—in MA1 and MA2,
and a single plain text file—when evaluating a model in application mode—
in MA3. These files contain, for that fold, the evaluation metrics, detailed
information about the number and type of misassignments, a list giving for
each note the voice it has been assigned to, and a list of conflicts that have
been encountered and resolved (if applicable). Furthermore, when evaluat-
ing a model in application mode, additional material is created. Firstly, in
all three modelling approaches, a so-called `Transcription` object (more
on this in Sections 4.1.2 and 4.2.2) is created and stored after the voice as-
signment process has ended. This object represents the test set for that
fold—which is always a single piece; see Section 3.2.1—and can be exported
in various formats (such as, for example, MIDI, MEI, or MusicXML), or be
visualised in modern music notation. Secondly, in MA1 and MA2, the poly-
phonic information generated by a model is stored as a list of voice labels
(and, if applicable, duration labels). This is the information used to annotate
the data with when using a bidirectional model.

---

[10]See `http://www.heatonresearch.com/encog`.

[11]See `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`. I am
grateful to Dr Emmanouil Benetos for providing ready-to-use implementations and run-
ning the experiments with the model used in MA3.

[12]The code is made available as open source software and can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

4

Datasets

The models described in the previous chapter are evaluated on two different datasets, which are the subject of this chapter. The tablature dataset, first, consists of a total of 15 three-voice and four-voice intabulations by various well-known sixteenth-century intabulators. Each intabulation is represented as a machine-readable encoding of the tablature in plain text format, capturing all relevant information contained in the tablature, and a set of monophonic MIDI files, each of which contains the pitch and duration information for an individual voice. The Bach dataset consists of the 45 three-voice and four-voice fugues from *Das wohltemperirte Clavier* (BWV 846–893) by Johann Sebastian Bach, music written in (early) modern staff notation. Each fugue is represented only as a set of monophonic MIDI files, each of which again contains the pitch and duration information for an individual voice.

The tablature dataset is created manually as part of this thesis; it is discussed in Section 4.1. Special attention is given to the data creation process, notably to the preceding polyphonic alignment process, where the intabulations are aligned with their vocal models in order to devise transcriptions in modern music notation that are used to extract the sets of MIDI files from. Furthermore, the internal data representation is described, and a detailed specification of the encoding format used to render the tablature machine-readable, `tab+`, is given. The Bach dataset is an adaptation of an existing set. It is discussed in Section 4.2, where the adaptations made are clarified, and the internal data representation is described.

4.1   The tablature dataset

The tablature dataset consists of 15 intabulations—six for three voices and nine for four voices, the most common intabulation formats—and comprises

a total of 11641 notes distributed over 6808 chords. All taken from popular sixteenth-century printed lute books, these intabulations are arrangements of sacred and secular polyphonic vocal works by renowned composers active some generations earlier, roughly between 1450 and 1550. The dataset is aimed to be as heterogeneous as possible, and includes works from different centres of activity, from different decades, arranging different vocal genres, and of different polyphonic textures. It is shown in Tables 4.1 and 4.2.

The terminology used to distinguish difference in polyphonic texture requires some additional explanation. A piece is considered to be *imitative* if the polyphonic structure is governed by motivic imitation, *semi-imitative* if it contains points of imitation but the polyphonic structure is not governed by imitation, and *free* if it contains only very few or no points of imitation. It must be noted that these qualifications, especially the first two, apply primarily to the vocal models. In the intabulations, the distinction between imitative and a semi-imitative textures tends to be less pronounced, as technical problems often complicate maintaining strict motivic imitation.

4.1.1   Formats and data creation

The tablature dataset used for this thesis is created from scratch. Data creation is a laborious and time-consuming process; for this reason, the dataset is still relatively small. Each intabulation is represented, first, as a machine-readable encoding of the tablature in plain text format, which captures all relevant information contained in the tablature, and second, as a set of mono-phonic MIDI files, each of which contains the pitch and duration information for an individual voice. Both the encodings and the MIDI file sets are created manually. In the case of the encodings, this is done by transcribing facsimile reproductions of the original tablature books. Details about the encoding format used are given in Section 4.1.3.[1] The MIDI file sets, then, are created in a two-step process. The first of these steps is the most challenging: using professional music notation software, a polyphonic transcription in modern music notation is made of each intabulation. This is done by aligning the intabulation with its vocal model. The polyphonic alignment process and the issues it entails are discussed in detail in Section 4.1.1.1. In the second step, each individual voice in the transcription thus created is exported as a MIDI file.[2]

---

[1] There exists at least one ready-to-use body of machine-readable encodings of lute tablature—the one stored in the ECOLM database (see Section 1.1). Further explanation as to why none of this material is used as a starting point is given in Section 4.1.3.

[2] The encodings and MIDI file sets, as well as the polyphonic transcriptions can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

Table 4.1 Tablature dataset, three-voice pieces. Subscript indices following publication years refer to entries in Brown (1965).

Intabulation				Vocal model	
Title	Source	Notes	Chords	Composer (edition)	Genre, texture
'Disant adiu madame'	Hans Newsidler, *Der ander theil des Lautenbůchs* (Nuremberg, 1536$_7$)	331	194	Compère (Finscher, 1972)	chanson, semi-imitative
'Meß pensees'	Newsidler, *Der ander theil*	706	481	Compère (Finscher, 1972)	chanson, semi-imitative
'Pleni de la missa misma'	Diego Pisador, *Libro de mvsica de vihvela* (Salamanca, 1552$_7$)	334	208	Josquin (Hudson, 1995)	mass section, imitative
'Elslein liebes Elslein'	Hans Judenkünig, *Ain schone kunstliche vnderweisung* (Vienna, 1523$_2$)	167	81	Senfl (Geering and Altwegg, 1962)	lied, free
'Nun volget Lalafete'	Hans Newsidler, *Das Ander Buch. Ein New künstlich Lautten Buch* (Nuremberg, 1544$_2$)	965	563	Janequin (Merritt and Lesure, 1965–1971, vol. I)	chanson, free
'Tant que uiuray' [a3]	Pierre Phalèse (publ.), *Des Chansons Reduictz en Tabulature de Lvt, Liure premier* (Leuven, 1547$_7$)	246	125	Sermisy (Allaire and Cazeaux, 1974)	chanson, free
		2749	1652		

Table 4.2 Tablature dataset, four-voice pieces. Subscript indices following publication years refer to entries in Brown (1965); in pieces marked with an asterisk (*), the most common scordatura tuning (see Section 2.2.1) is used.

Intabulation				Vocal model	
Title	Source	Notes	Chords	Composer (edition)	Genre, texture
'Absolon fili mi'*	Sebastian Ochsenkun, *Tabulaturbuch auff die Lauten* (Heidelberg, 1558₅)	1184	727	Josquin (Sherr, 2002)	motet, imitative
'In exitu Israel de Egipto'	Ochsenkun, *Tabulaturbuch*	1974	1296	Josquin (Jas, 2008)	motet, imitative
'Qui habitat'	Ochsenkun, *Tabulaturbuch*	2238	1443	Josquin (Perkins, 2011)	motet, imitative
'Bramo morir per non patir piu morte'	Antonio Rotta, *Intabolatvra de lavto, Libro primo* (Venice, 1546₁₅)	708	322	Festa/Arcadelt (Seay, 1978)	madrigal, free
'Tant que uiuray' [a4]	Phalèse, *Des Chansons*	457	228	Sermisy (Allaire and Cazeaux, 1974)	chanson, free
'Herr Gott laß dich erbarmen'*	Ochsenkun, *Tabulaturbuch*	371	195	Isaac (Wolf, 1907)	lied, free
'Mais mamignone'	Iulio Abondante, *Intabolatvra di lavtto, Libro secondo* (Venice, 1548₁)	705	316	Janequin (Merritt and Lesure, 1965–1971, vol. II)	chanson, semi-imitative
'Las on peult'	Pierre Phalèse (publ.), *Theatrvm mvsicvm* (Leuven, 1563₁₂)	777	395	Janequin (Merritt and Lesure, 1965–1971, vol. III)	chanson, semi-imitative
'Il nest plaisir'	Iulio Caesaro Barbetta, *Il terctio libro de intavolatvra de livto* (Strasbourg, 1582)	478	234	Janequin (Merritt and Lesure, 1965–1971, vol. II)	chanson, semi-imitative
		8892	5156		

4.1.1.1   Polyphonic alignment

Conceptually, the *polyphonic alignment* process is straightforward. The polyphonic structure of an intabulation's vocal model is always unambiguous, as each voice is notated on a separate staff. The polyphonic alignment process thus implies tracing the tablature notes in the vocal model and assigning them to a voice accordingly. When their polyphonic position is known, the notes' full duration can then be determined (recall that in the tablature, only their minimum duration is given). A note's full duration is determined by three (interrelated) factors: (i) the onset time of the next note in the same voice, (ii) the onset time of the next note on the same course, and (iii) a theoretical maximum duration of a semibreve (see Section 2.2.3). When for each note both a voice and a duration have been determined, the polyphonic alignment process is completed.[3] For practical reasons, for the polyphonic alignment process scholarly editions of the vocal models are preferred over facsimile reproductions of the original sources.[4] The editions used are those listed in Tables 4.1 and 4.2.

In practice, however, the polyphonic alignment process entails some issues. This is because, as already discussed in Section 2.2.2, intabulations are never literal, verbatim transcriptions of their vocal models: intabulators always made adaptations in their arrangements. Such adaptations can lead to ambiguities with regard to which tablature notes correspond to which notes in the model (if any). A typical example is shown in Figure 4.1. In this rather faithful arrangement, the majority of the original notes have been retained verbatim (unmarked in Figure 4.1b; a note is considered to have been retained verbatim if both its pitch and onset time remain the same), but, following the list of changes and simplifications as presented in Section 2.2.2, 17 notes, most of them ornamental, have been added (marked white); nine original notes have been altered either in terms of pitch, that is, replaced by harmonic variants or alternatives, or in terms of onset time, that is, shifted rhythmically (asterisk-shaped—the four notes to which the latter applies are all found in bar 2; the tenor note has additionally been altered in terms of pitch); three original notes have been omitted (parenthesised); and in two cases two original notes sounding in unison have been replaced by a single-note unison (bracketed).

With regard to a number of these adaptations, several interpretations are contrapuntally plausible. This example, straightforward as it may be, thus

---

[3]The polyphonic alignment process thus yields the basis of the transcription that is created. To complete this transcription, decisions on several further matters on which the tablature not always provides information—key and time signature, pitch spelling, barring, tuning, etc.—must be made. These decisions are *not* part of the polyphonic alignment

(a) Vocal model, edition. Sermisy (Allaire and Cazeaux, 1974), 'Tant que vivray', opening bars.

(b) Intabulation, transcription in modern music notation and schematic overview of adaptations. Phalèse (publ.) (1547_7), 'Tant que uiuray' [a4], opening bars.

Figure 4.1 Polyphonic alignment ambiguities.

---

process.

[4]The polyphonic alignment process is currently carried out manually; (partial) automation of the process is left for future work.
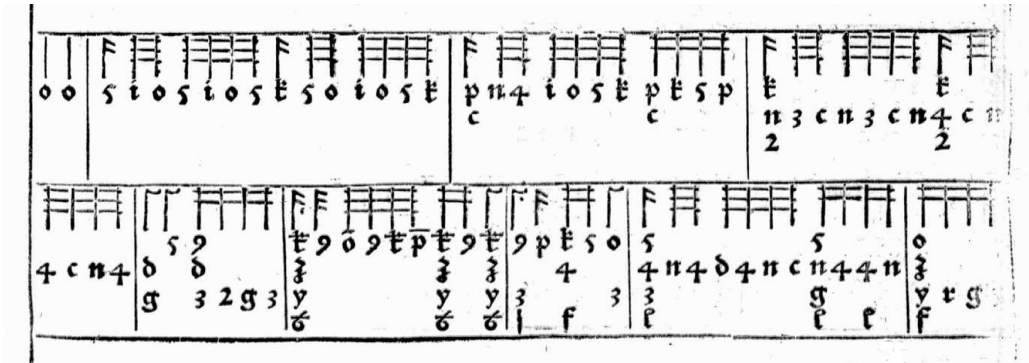
Figure 4.2 Sebastian Ochsenkun's intabulation style. Ochsenkun (1558₅), 'Absolon fili mi', opening bars.

already shows how adaptations can complicate the polyphonic alignment process by introducing an element of subjectivity to it. It is therefore seldom possible to speak of *the only correct* mapping of tablature notes to voices; nevertheless, a procedure in which the vocal models are used as blueprints does guarantee a *valid* mapping.

4.1.1.2 Sebastian Ochsenkun's *Tabulaturbuch auff die Lauten*

To alleviate alignment problems as described above, a number of works from Sebastian Ochsenkun's *Tabulaturbuch auff die Lauten* (1558) are included in the dataset. Ochsenkun, like many other intabulators in German speaking regions, uses the staffless German tablature system, in which each course-fret intersection is represented as a unique symbol (see also Section 2.2.1). What makes him stand out among his contemporaries, however, is the fact that he organises the tablature symbols horizontally, that is, by voice, in a consistent manner—note that this is only possible in German tablature—, which enables him to indicate polyphonic lines.[5] An example is given in Figure 4.2; clearly visible are the successive entries of the superius (bar 1), altus (bar 3), tenor (bar 4), and bassus (bar 6). Note also the difference with the example of German tablature in Figure 2.2c, where all tablature symbols are pushed towards to highest line.

When transcribing Ochsenkun's intabulations, the first part of the polyphonic alignment process—tracing tablature notes in the vocal model and assigning them to a voice—can thus be bypassed by simply adhering to his

---

[5]Several other German intabulators, notably Hans Newsidler, follow a comparable modus operandi—but without exception, these are much more liberal in their horizontal organisation. The consistency found in Ochsenkun's intabulations is unique.

horizontal organisation; only the second part—determining the notes' full duration—still has to be executed.

However, a critical note is in order here. The question remains in how far Ochsenkun's polyphonic interpretations are representative of the contemporary intabulation practice. Although he states in the foreword of the *Tabulaturbuch* that he sets "as much of the substance of the song as this instrument [that is, the lute] can adopt and comfortably manage" (Ochsenkun, 1558, f. Aii^r, translation mine),[6] and that he has put together the book "on the right foundation of the present day's lute art, with four, five, and six voices ... each with its colorations, properly fitted to the lute" (f. Aiii^v, translation mine),[7] he seems to have made some choices that, if not simply misprints, are remarkable at least, and that sometimes lead to significant deviations from the vocal model. This has mostly gone unnoticed in the musicological literature, where there is a general agreement that Ochsenkun follows the models as faithfully as possible (see, for example, Hong, 1984; Dorfmüller, 1967, 2001). Regardless of some peculiarities, however, in general Ochsenkun's intabulation style is close to that of his contemporaries, and his works can therefore safely be included in the dataset.

4.1.2   Internal data representations

The encoding file and the set of MIDI files that represent an intabulation are processed as follows. From the encoding file, a `Tablature` object (an internal representation of the intabulation) is created, and from that a list of `TablatureNote` objects is extracted. As its name suggests, each of the `TablatureNote` objects corresponds to a tablature note, and contains a number of basic attributes of that note: its pitch, course, fret, onset time, minimum duration, maximum duration, the sequence number and the size of the chord it is in, and its sequence number within the chord. The `TablatureNote` objects are added to the list in a fixed sequence, where the object corresponding to the lowest note in the first chord constitutes the first element of the list, and the one corresponding to the highest note in the final chord the last (the objects are thus ordered first according to the onset time, and second according to the pitch of the note they corre-

---

[6]Original wording: "souil diß Instrument von der substantz des gesangs annemmen vnd bequemlich leiden mögen".

[7]Original wording: "auff den rechten grund yetziger zeit Lauttenkunst / mit vier / fünf vnnd sechs stimmen / ... jeden mit sein Coloribus / artlich auff die Lauten zugericht". The six-voice pieces are notated as if they were for five voices.

spond to).[8] From the set of MIDI files, then, a `Transcription` object (an internal representation of the intabulation) is created, from which a list of voice labels is extracted. This list is ordered in similar fashion as the list of `TablatureNote` objects, that is, the label that belongs to the lowest note in the first chord constitutes the first element, and the one that belongs to the highest note in the final chord the last. When there are two labels for a single note—in other words, when this note is a single-note unison—these labels are combined into a single new label (as described in Section 3.1.2.1). When there are two successive notes that have the same pitch and the same onset time, meaning that they form an *actual* unison, the note on the lower course is added first. The labelling of the data needed for MA1 is achieved by mapping each element in the list of `TablatureNote` objects to each element in the list of voice labels. The mappings of notes to voices used in MA2 and MA3, then, are extracted after first mapping the `TablatureNote` objects to the voice labels likewise, and subsequently organising the list of `TablatureNote` objects into chords—that is, grouping objects together based on the onset time of the note to which they correspond.

When training a model, the labels are used during the training process in the feature calculation and as training targets, and after the process to calculate the evaluation metrics. When evaluating it, in test mode the labels are used during the evaluation process in the feature calculation and after the process to calculate the evaluation metrics, while in application mode, they are only used after the process to calculate the evaluation metrics.

The `Transcription` object, together with the `Tablature` object, serve further use in the feature calculation.

4.1.3 Encoding format

The encoding format used for this thesis, `tab+`, is custom-made and inspired by the TabCode format as described in Crawford (1991), which is the format used for the encodings stored in the ECOLM database (see also Section 1.1).[9] The principal difference between the two is that unlike TabCode, which supports only French, Italian, and Spanish tablature, `tab+` also supports German tablature. Although it is good practice to build on existing formats, the simpler option to devise a new encoding format is preferred over the more complicated option of extending TabCode to also include German

---

[8]Strictly speaking, the object corresponding to the note on the lowest *course* in the chord constitutes the first element, etc. Although it is theoretically possible that a note on a lower course has a pitch that is higher than that of note on a higher course in the same chord, this never occurs in the dataset.

[9]A full description of TabCode can be found at `http://www.ecolm.org`.

Table 4.3 Encoding, header section: fields.

Field	Values	Description
AUTHOR		the intabulator of the piece
TITLE		the title of the piece
SOURCE		the source containing the piece
TABSYMBOLSET	FrenchTab, ItalianTab, SpanishTab, extensions of GermanTab	the TabSymbolSet used to encode the piece
TUNING	transpositions of G and G′	the tuning used for the piece
METER_INFO		the meter(s) used in the piece, as well as the barring per meter

lute tablature. Because of their similarity, however, the conversion of either format into the other is a relatively simple task. This works two ways: a conversion of TabCode into tab+ opens up the ECOLM database for further research into voice separation in tablature, while a conversion in the opposite direction enables the inclusion of music in German lute tablature in the ECOLM database.

4.1.3.1   Content and symbols

tab+ is intended to remain simple, intuitive, and easy to read, and is extensible. Each encoding consists of two main sections: a header and a body. The header, first, contains general information about the piece that is encoded and consists of six fields enclosed in curly brackets. The fields are listed in Table 4.3.

As shown in Table 4.3, the TABSYMBOLSET and TUNING fields can only take a range of preset values from which one must be selected. These fields require some additional explanation. A *TabSymbolSet*, first, is a set of *Tab-Symbols*, each of which encodes a position on the fretboard. When encoding a piece, it is convenient to use a representation that is close to the original notation. For this reason, separate TabSymbolSets are defined for the letter-based French tablature system (FrenchTab), for the numeral-based Italian and Spanish systems (ItalianTab and SpanishTab—these are identical), and for the three variants of the symbol-based German system used for this thesis (Judenkuenig1523, Newsidler1536, and Ochsenkun1558—all of which are extensions of the GermanTab TabSymbolSet, which covers only the highest five courses). In the case of the former two TabSymbolSets, each TabSymbol

a1	b1	c1	d1	e1	f1	g1	h1	...
a2	b2	c2	d2	e2	f2	g2	h2	...
a3	b3	c3	d3	e3	f3	g3	h3	...
a4	b4	c4	d4	e4	f4	g4	h4	...
a5	b5	c5	d5	e5	f5	g5	h5	...
a6	b6	c6	d6	e6	f6	g6	h6	...

(a) FrenchTab.

01	11	21	31	41	51	61	71	...
02	12	22	32	42	52	62	72	...
03	13	23	33	43	53	63	73	...
04	14	24	34	44	54	64	74	...
05	15	25	35	45	55	65	75	...
06	16	26	36	46	56	66	76	...

(b) ItalianTab.

5	e	k	p	v	9	e-	k-	...
4	d	i	o	t	7	d-	i-	...
3	c	h	n	s	z	c-	h-	...
2	b	g	m	r	y	b-	g-	...
1	a	f	l	q	x	a-	f-	...
A	B	C	D	E	F	G	H	...
+	A	B	C	D	E	F	G	...
+	2-	3-	4-	5-	6-	7-	8-	...

(c) GermanTab extended to Judenkuenig1523 (third from bottom), Newsidler1536 (second from bottom), and Ochsenkun1558 (bottom).

Figure 4.3 TabSymbolSets. The diagrams visualise the fretboard, where the rows are the courses and the columns the frets. The double vertical line indicates the nut at the beginning of the fretboard.

consist of two characters, indicating a fret and a course, respectively, while in the case of the latter three, each TabSymbol attempts to replicate the original tablature symbol. The five TabSymbolSets that are defined are shown in Figure 4.3.

The tuning that is selected, then, determines the pitch that is associated with each tablature symbol. Although the tuning is generally not specified in the tablature, in the case of intabulations the intended tuning (or, at least, an appropriate tuning) can be deduced from the vocal models. Two tunings are defined: the standard Renaissance tuning (see Section 2.2) with the lowest course tuned to nominal G, and the most common scordatura variant of it, denoted in Table 4.3 as G′, where the sixth course is lowered by a whole tone (see Section 2.2.1). Both these tunings can be transposed up and down as needed, and thus cover all pieces in the tablature dataset.

The body, second, is a string of characters, which, separately or combined, form symbols. Two types of symbols can be discerned: *punctuation symbols*, which have no counterpart in the original tablature and serve encoding parsing purposes, and *musical symbols*, which always have a counterpart in the tablature. To the latter belong (i) the TabSymbols in the different TabSymbolSets defined (see Figure 4.3), (ii), a set of rhythm symbols, (iii) a set of mensuration signs, and (iv) a set of barline variants. The punctuation symbols and the musical symbols are listed in Table 4.4.

Both the rhythm symbols and the mensuration signs require some additional explanation. Each rhythm symbol can be modified into its dotted counterpart by simply adding an asterisk (`mi*`, `fu*`, etc.). The duration of the corona can vary and is parametrised by the number in its encoding, indicating the duration in semibreves. Again, an asterisk can be added (`co1*`, for example, yields a corona with a duration of a dotted semibreve). Furthermore, a dash can be placed after a rhythm symbol to indicate that it is connected to the next rhythm symbol by means of a horizontal beam (`fu-`, `sf-`, etc.).[10] For each mensuration sign, then, a default placement on the third staff line from the top is assumed, but its vertical position can be parametrised by adding a number, indicating the staff line, to its encoding. Individual mensuration signs are often merged into compound mensuration signs (for example, C3 or O3); these can be encoded by simply juxtaposing two mensuration signs (`MC3.M34`, `MO3.M34`).

The five TabSymbolSets and two transposable tunings that are defined, as well as the sets of rhythm symbols, mensuration signs, and barlines, suffice for the tablature dataset used in this thesis. For future work with different datasets, if needed, additional TabSymbolSets and tunings can always be defined, and the sets of musical symbols can always be extended. Further possible extensions may concern, for example, the inclusion of fingering or ornamentation signs.

---

[10]Specifying beaming is not necessary for successful parsing of the encoding; this option is included for future visualisation purposes.

Table 4.4 Encoding, body section: punctuation symbols and musical symbols.

(a) Punctuation symbols.

Symbol	Encoding	Description
symbol separator	.	demarcates a musical symbol
event separator	>	demarcates a vertical event
system break indicator	/	indicates a system break
end break indicator	//	indicates the end of a piece
comment indicators	{ and }	demarcate an inserted comment

(b) Musical symbols: rhythm symbols (top), mensuration signs (middle), and barlines (bottom).

Symbol	Encoding	Description
semibreve	sb	
minim	mi	
semiminim	sm	
fusa	fu	
semifusa	sf	
corona	co1, co1*, co2, etc.	
dot	*	dot of addition
beam	–	beam connecting two rhythm symbols
C	MC	
cut C	McC	
3	M3	
O	MO	
single	|	
single repeat	|: and :|, :|:	
double	||	
double repeat	||: and :||, :||:	

4.1.3.2   Encoding principles

The tablature is interpreted as a sequence of vertical events separated by barlines. Three types of events are distinguished: (i) mensuration sign events, each of which consists of a (compound) mensuration sign, (ii) chord events, each of which consists of one or more tablature symbols, possibly—but not

(a) Tablature.

```
{AUTHOR:PHALESE, Pierre (publ.)}
{TITLE:Tant que uiuray [a4]}
{SOURCE:Des Chansons Reduictz en Tabulature de Lvt, Liure
 premier (1547_7), f. Gii^v}
{TABSYMBOLSET:FrenchTab}
{TUNING:G}
{METER_INFO:2/2 (1-4)}

McC3.>.
{bar 1}mi*.a4.a3.d2.f1.>.
sm.h1.>.
mi.a4.a3.d2.f1.>.
fu-.c4.b3.a2.d1.>.fu-.c2.>.fu-.d2.>.fu.a1.>.|.
{bar 2}sm-.a4.d3.d2.c1.>.sm-.a1.>.sm-.c1.>.sm.d1.>.
sm-.a4.d3.d2.c1.>.sm-.c4.>.sm-.e4.>.sm.g4.a2.>.|.
{bar 3}mi.h4.h3.a1.>.
mi.e4.d3.d2.f1.>.
fu-.c4.a2.a1.>.fu-.d2.>.fu-.a1.>.fu.c1.>.
fu-.e1.>.fu-.f1.>.fu-.c1.>.fu.e1.>.|.
{bar 4}sm.a5.d3.c2.f1.>.{bar 4 cut off}//
```

(b) `tab+` encoding.

Figure 4.4 `tab+` encoding of a tablature fragment. Phalèse (publ.) (1547[7]), 'Tant que uiuray' [a4], opening bars.

necessarily—with a rhythm symbol placed above,[11] and (iii) rest events, each of which consists of a rhythm symbol only. Each event is encoded as a sequence of one or more musical symbols, where the symbols are separated from one another by means of symbol separators. The events themselves, then, are separated by means of event separators. Comments placed within comment indicators can be inserted at any point before the end break indi-

---

[11]In the parsing of the encoding, each chord that has no rhythm symbol is automatically assigned the duration that goes with the rhythm symbol encountered last.

cator, and hard returns can be used to organise the encoding visually as to increase legibility. Lastly, to ensure that no parsing errors occur, the following *encoding rules* must be followed:

Encoding rule 1 The encoding cannot contain space characters other than those occurring within comment indicators.

Encoding rule 2 A chord event must be encoded in a fixed sequence: any rhythm symbol must be encoded first, followed by the lowest-course tablature symbol, the second-lowest-course tablature symbol, etc.

Encoding rule 3 Mensuration sign events, chord events, and rest events must always be succeeded by an event separator.

Encoding rule 4 Barline events must never be succeeded by an event separator.

Encoding rule 5 The encoding must end with an end break indicator.

An example of a piece (the fragment in French tablature also shown in Figure 4.1b) encoded in `tab+`, with some comments concerning barring inserted and hard returns placed at convenient points to increase legibility, is given in Figure 4.4. Note that the mensuration sign is interpreted as a $\frac{2}{2}$ meter, meaning that a semibreve corresponds to a half note (see Section 2.2.3).

4.2   The Bach dataset

The Bach dataset consists of the 26 three-voice and the 19 four-voice fugues contained in books I and II of Johann Sebastian Bach's *Das wohltemperirte Clavier* (BWV 846–893), and comprises a total of 50135 notes distributed over 29787 chords. It is shown in Tables 4.5 and 4.6.

4.2.1   Format and data adaptations

The Bach dataset is retrieved from the MuseData repository of the Center for Computer Assisted Research in the Humanities (CCARH) in the form of MIDI files.[12] In order for the data to be compatible with the system, three kinds of adaptations are made to the original files. First, each MIDI file, in its original form representing a complete fugue, is separated into a set of MIDI files, each containing the pitch and duration information for an individual voice. This is a very straightforward task, as in the original MIDI files the information for each voice is stored in a separate channel. Second, in order

---

[12]See `http://www.musedata.org` and `http://www.ccarh.org`.

Table 4.5 Bach dataset, three-voice pieces.

Fugue	BWV	Book	Notes	Chords
2 in c minor	847	I	747	408
3 in C♯ major	848	I	1432	851
6 in d minor	851	I	835	549
7 in E♭ major	852	I	953	614
8 in d♯ minor	853	I	1478	816
9 in E major	854	I	733	438
11 in F major	856	I	850	574
13 in F♯ major	858	I	911	574
15 in G major	860	I	1779	1071
19 in A major	864	I	1206	702
21 in B♭ major	866	I	952	527
1 in C major	870	II	1066	667
3 in C♯ major	872	II	774	535
4 in c♯ minor	873	II	1381	865
6 in d minor	875	II	805	502
10 in e minor	879	II	1446	1006
11 in F major	880	II	963	597
12 in f minor	881	II	1126	675
13 in F♯ major	882	II	1408	919
14 in f♯ minor	883	II	1478	928
15 in G major	884	II	763	522
18 in g♯ minor	887	II	1630	993
19 in A major	888	II	760	455
20 in a minor	889	II	857	695
21 in B♭ major	890	II	960	554
24 in b minor	893	II	1051	653
			28344	17690

Table 4.6 Bach dataset, four-voice pieces.

Fugue	BWV	Book	Notes	Chords
1 in C major	846	I	736	400
5 in D major	850	I	789	474
12 in f minor	857	I	1456	892
14 in f♯ minor	859	I	860	482
16 in g minor	861	I	878	517
17 in A♭ major	862	I	918	551
18 in g♯ minor	863	I	804	404
20 in a minor	865	I	2402	1201
23 in B major	868	I	897	526
24 in b minor	869	I	1817	1120
2 in c minor	871	II	663	385
5 in D major	874	II	896	415
7 in E♭ major	876	II	707	365
8 in d♯ minor	877	II	1049	551
9 in E major	878	II	784	399
16 in g minor	885	II	1665	890
17 in A♭ major	886	II	1353	775
22 in b♭ minor	891	II	1761	928
23 in B major	892	II	1356	822
			21791	12097
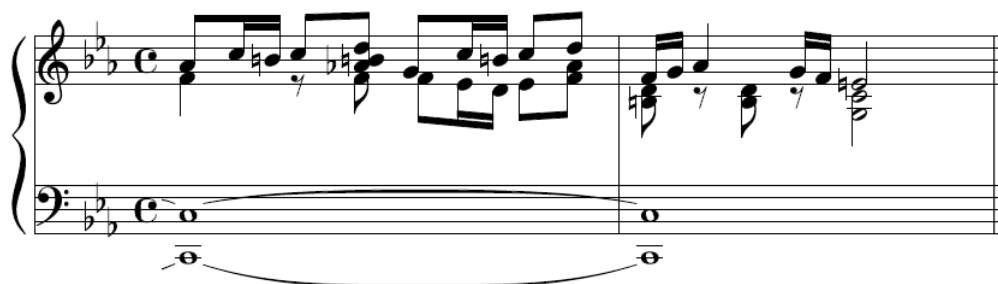
Figure 4.5 In-voice chords. Bach, *Das wohltemperirte Clavier*, book I, Fugue 2 in c minor (BWV 847), closing bars. Example reprinted with permission of CCARH.

Figure 4.6 Added extra voice. Bach, *Das wohltemperirte Clavier*, book II, Fugue 17 in A♭ major (BWV 886), closing bars. Example reprinted with permission of CCARH.

to comply with the definition of voice as a monophonic sequence of notes as used in this thesis, any simultaneous MIDI notes—that is, notes that have the same onset and offset time—within the separated MIDI files are reduced to single notes. Such *in-voice chords*, which are witnessed in a number of fugues, usually occur at the end of the piece and result in enhanced chords that have an embellishing function: they serve to give a sense of finality. An example is given in Figure 4.5, where at least one in-voice chord is encountered in each voice.

Third, some fugues contain temporarily added extra voices, which also serve an embellishment end. An example of this is shown in Figure 4.6, where such an extra voice, descending chromatically from A♭$_3$ (not in the figure) to E♭$_3$, is placed between the bassus and the tenor (note also the enhanced final chord). In order not to exceed the nominal number of voices in a fugue, these added voices are removed wherever they are encountered.

A total of 206 notes are thus removed from the original data. It must be stressed, lastly, that neither of the adaptations as described above impose

serious inflictions on the polyphonic structure of the fugues.[13]

4.2.2   Internal data representations

Each set of MIDI files that represents a fugue is processed in a similar way as a set representing an intabulation. From this set, a `Transcription` object (an internal representation of the fugue) is created, from which then a list of `Note` objects and a list of voice labels are extracted. Each `Note` object correspond to a note, and contains the same basic attributes as a `TablatureNote` object (apart from course and fret). The `Note` objects are added to the list in the same manner as the `TablatureNote` objects, that is, the object corresponding to the lowest note in the first chord first, etc. When there are two successive notes that have the same pitch and the same onset time, meaning that they form a unison, the note with the longer duration is added first; if they have the same duration, the one in the lower voice is added first. Because the `Note` objects and the voice labels stem directly from the same `Transcription` object, the labelling required for MA1 is already provided. Similar to as described in Section 4.1.2, then, the mappings of notes to voices used in MA2 and MA3 are extracted after organising the list of `Note` objects into chords. The use of the labels is identical to as described in Section 4.1.2.

Again, too, the `Transcription` object serves further use in the feature calculation.

---

[13]The MIDI file sets, as well as a list of all notes that are removed from the original data, can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

5

Experimental results and discussion

In this chapter, the results from experiments conducted with the various models described in Chapter 3 on the two datasets described in Chapter 4 are presented and discussed. A total of six experiments are conducted, the first two of which are preliminary experiments concerning model optimisation. In the first of these, which is described and discussed in Section 5.1 and applies to the neural network models used in MA1 and MA2, three hyperparameters—the hidden layer size, the regularisation parameter λ, and the margin ε—are optimised. In the second preliminary experiment, which is described and discussed in Section 5.2 and applies to the hidden Markov model used in MA3, the configuration of the transition probability matrix, the observation probability matrix, and the initial state matrix types is optimised.

The four main experiments, then, are described and discussed in Sections 5.3, 5.4, 5.5, and 5.6, respectively. In Experiment 1, the three modelling approaches—the note-level classification approach (MA1), the chord-level regression approach (MA2), and the chord-level probabilistic approach (MA3)—are evaluated and compared. In Experiment 2, the relevance of the features is evaluated in two sub-experiments. In Experiment 3, which consists of three sub-experiments, the effect of the three model extensions—backward processing, simultaneous voice and duration modelling, and multi-pass processing using a bidirectional decision context—is evaluated. In Experiment 4, lastly, the performance of the models is compared with the performance of existing voice separation systems.

Finally, in Section 5.7, a number of overarching issues encountered throughout the experiments conducted are described: conflicts and conflict resolution, error propagation, and the handling of three complex musical phenomena: single-note unisons, voice crossing, and imitation.

5.1 Neural network models and hyperparameter optimisation

The first preliminary experiment applies to the neural network models used in MA1 and MA2, and concerns hyperparameter optimisation. For both these models, two hyperparameters are optimised in a cross-validated grid search: the size of the hidden layer and the regularisation parameter λ. Furthermore, for the model used in MA2, an additional third parameter is optimised: the margin ε.

5.1.1 Hidden layer size and regularisation

Both the number of hidden neurons and the amount of regularisation applied in the training influence the complexity of a model. A model that is too complex learns random noise in the training data and is thus prone to overfitting (has high variance), meaning that it adapts well to the training data but generalises poorly on new data. A model that is not complex enough, on the other hand, fails to learn relevant relations between input and outputs, and is prone to underfitting (has high bias): it adapts poorly to the training data and also generalises poorly on new data.

The optimal hidden layer size depends on several interrelated factors, such as the size of the input and output layers, the complexity of the function that must be learned, the training algorithm used, the type of activation function used, etc. Generally, a too small hidden layer tends to lead to underfitting, and a too large hidden layer to overfitting (see Geman et al., 1992, for a discussion on the influence of the hidden layer size on the bias/variance trade-off). Finding the optimal size, however, is a well-known problem (Baum and Haussler, 1989; Lawrence et al., 1996; Huang and Huang, 1991; Elisseeff and Paugam-Moisy, 1997). Although several rules of thumb are encountered in the literature—stating, for example, that the size of the hidden layer should be between that of the input layer and that of the output layer, or that it should never be more than twice the size of the input layer—, in this thesis this problem is approached empirically. Thus, for the models used in MA1 six different hidden layer sizes are tried, measuring respectively $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1, 2, and 4 times the size of the input layer (excluding the bias neuron; numbers are rounded half up). For the model used in MA2, only the first five of these hidden layer sizes are tried.

Another measure to prevent overfitting is to use some form of *regularisation* when training a model, that is, to reduce the weights in the weight matrix by a certain amount in the weight update. For all models, L_2 regu-

larisation (or *weight decay*) is used, where larger weights are penalised more than smaller, and each weight θ in the weight matrix is reduced as follows:

$$\theta' = \theta - (\lambda \cdot \theta), \tag{5.1}$$

where θ and θ' are the weight values before and after regularisation, respectively, and λ is the regularisation parameter, a constant that controls the degree of regularisation. 10 different values of λ are tried: 0.1, 0.03, 0.01, 0.003, ..., $1 \cdot 10^{-5}$, and 0.0 (or no regularisation).

The six, or, as the case may be, five different hidden layer sizes together with the 10 values for λ yield a grid of 60 (50) possible combinations of these two parameters. Each model is then trained and evaluated with every parameter combination. Moreover, because the model architecture differs depending on the dataset a model is applied to, each model is trained and evaluated both on the tablature dataset and on the Bach dataset (with the exception of the model variant for modelling voice and duration simultaneously, as used in MA1). This is done on the two four-voice subdatasets only; the complete results are given in Tables A.1–A.10 in the Appendix. The combination giving the highest accuracy in application mode is then selected as the optimal combination.[1] The optimal combinations are shown in Tables 5.1 and 5.2. As in the remainder of this thesis, values in parentheses following accuracy values indicate standard deviations over all folds.[2] Recall, furthermore, that the models used in MA1 are referred to as N (modelling only voice) and N′ (modelling voice and duration simultaneously), and the one used in MA2 as C.

Looking at Tables A.1–A.6, the following general observations can be made with regard to the models used in MA1. First, when moving diagonally through the grids for the N model (see Tables A.1 and A.3), starting in the upper left corner with the combination of the smallest hidden layer and the largest λ value, model performance initially improves, but gradually stabilises. This stabilisation is confirmed by the fact that the highest accuracy value is generally not statistically different from most of the values given by parameter combinations in the lower right corner of the grid (see Tables A.2 and A.4). Thus, as the hidden layer size increases and the amount of regularisation applied decreases, the model tends to become less and less sensitive to change in these parameters. This effect is the strongest—that is,

---

[1]By tracking stabilisation in both the voice decision accuracy and the network error (see Section 3.1.1.1), in this experiment it is also determined that for the models used in MA1 400 iterations are required for convergence, and for that used in MA2 600.

[2]All numerical experimental results presented in this thesis are truncated after two (evaluation metric and standard deviation values) or four (*p*-values) places, and not rounded. Others' results are always copied verbatim.

Table 5.1 Hyperparameter optimisation, N, N′, and C models: optimal hidden layer size and λ value. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

Model	HL	λ	Training	Test	Application
			acc	acc	acc
N	$1 \cdot$ IL	0.001	92.97	92.04	79.63
			(0.29)	(2.46)	(5.96)
N′	$1/2 \cdot$ IL	$3 \cdot 10^{-5}$	97.36	96.61	70.47
			(0.18)	(1.85)	(8.13)
C	$1/4 \cdot$ IL	0.001	84.02	80.85	75.26
			(1.78)	(6.17)	(6.90)

Table 5.2 Hyperparameter optimisation, N and C models: optimal hidden layer size and λ value. Bach dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

Model	HL	λ	Training	Test	Application
			acc	acc	acc
N	$1 \cdot$ IL	$1 \cdot 10^{-5}$	98.04	97.57	80.70
			(0.06)	(0.85)	(7.50)
C	$1/4 \cdot$ IL	$3 \cdot 10^{-5}$	96.95	96.86	79.56
			(0.22)	(1.06)	(6.80)

the stabilisation takes place the earliest—in application mode, and the least strong in training mode. The results for the N′ model (see Tables A.5 and A.6), however, show a different picture. Here, after a similar stabilisation halfway through the grid, the accuracies clearly decrease again (in training and test mode this decrease is statistically significant) as the hidden layer size further increases—indicating that larger hidden layers are in fact detrimental to the model performance. Generally speaking, for the models used in MA1, neither a clear underfitting effect (which should be the strongest for parameter combinations in the upper left corner of the grid) nor a clear overfitting effect (which should be the strongest for combinations in the lower right corner) are witnessed.

The picture for the C model (see Tables A.7–A.10) is roughly the same as that for the N model: accuracies increase when moving diagonally through the grids, and stabilise approximately halfway; a clear underfitting effect is not witnessed. The similarity ends, however, when the largest hidden layer size is reached. While the accuracies in training and test mode keep increasing when moving from the next-to-largest to the largest hidden layer size, those

in application mode drop—in the case when the model is applied to the tablature dataset dramatically. This is a clear sign of overfitting, which can be explained as follows. Tracking the mapping decision during the training reveals that when the hidden layer size is too large, the relative training becomes problematic in that the highest rating is very frequently given to more than one feature vector in the feature vector set representing a chord (the multiple-highest-rating problem; see Section 3.1.3.3).[3] As explained in Section 3.1.3.3, in such cases the mapping is selected that goes with the first feature vector in the set that gets the highest rating. This means that if the ground truth feature vector—which is always placed as the first element of the set—is among the feature vectors receiving the highest rating, the correct mapping is selected—even though the model has not actually *learned* that it is the correct mapping (it gives the same highest rating to other feature vectors as well). Although this increases training accuracy, it is very likely to result in bad generalisation capacity. The current solution to the multiple-highest-rating problem is thus clearly only a stop-gap measure that suffices as long as this problem does not occur too often; a more elegant solution is left for future work. It must be mentioned, lastly, that with the parameter combinations selected as the optimal combinations for the C model (see Tables 5.1 and 5.2), the multiple-highest-rating problem never occurs.[4]

5.1.2   Margin

In MA2 there is a third parameter that is optimised: the margin ε, which serves an important role in the relative training procedure (see Section 3.1.3.2). However, in this modelling approach, where for each chord mappings must be enumerated and sets of voice assignments must be generated and subsequently rated, an exhaustive grid search over the space of three parameters would be computationally very expensive. In the case of the C model, the grid search over the hidden layer size and λ is therefore performed with ε = 0.05, a value that has proven to give good results on the four-voice Bach subdataset in an earlier study (Weyde and de Valk, 2016). In order to test the effect of different ε values, eight additional values are tried, all in combination with the optimal hidden layer size and λ value as determined in the grid search. This is again done on the two four-voice subdatasets only.

---

[3]The multiple-highest-rating problem is not specific to the largest hidden layer size and also occurs when the three smaller hidden layer sizes, and, notably, the next-to-largest hidden layer size, are used. Moreover, on the tablature dataset the model is more susceptible to it. However, as Tables A.7–A.10 show, only when the largest hidden layer size is used the problem results in overfitting.

[4]This is also true when the C model is applied to the three-voice subdatasets.

Table 5.3 Hyperparameter optimisation, C model: effect of different ε values. Tablature dataset, four-voice pieces. The asterisk (*) indicates that the multiple-highest-rating problem occurs for that value.

ε	Training	Test	Application
	acc	acc	acc
0.5*	88.39 (1.74)	**87.48** (5.47)	**75.80** (11.30)
0.16*	**88.81** (4.01)	86.27 (6.88)	74.18 (9.90)
0.05	84.02 (1.78)	80.85 (6.17)	75.26 (6.90)
0.016	79.70 (3.36)	76.79 (8.28)	69.18 (10.39)
0.005	72.01 (4.44)	68.84 (9.47)	66.94 (8.71)
$1.6 \cdot 10^{-3}$	67.95 (3.48)	65.57 (8.12)	63.60 (9.15)
$5 \cdot 10^{-4}$	66.17 (3.85)	66.59 (8.55)	63.84 (7.89)
$1.6 \cdot 10^{-4}$*	66.48 (2.27)	68.11 (10.69)	63.20 (11.42)
0.0	67.52 (3.57)	66.93 (9.14)	62.40 (9.63)

Table 5.4 Hyperparameter optimisation, C model: effect of different ε values. Bach dataset, four-voice pieces. The asterisk (*) indicates that the multiple-highest-rating problem occurs for that value.

ε	Training	Test	Application
	acc	acc	acc
0.5*	**97.82** (0.78)	**97.70** (1.41)	53.04 (18.12)
0.16*	97.01 (0.18)	96.66 (1.36)	77.17 (9.87)
0.05	96.95 (0.22)	96.86 (1.06)	**79.56** (6.80)
0.016	96.55 (0.53)	96.25 (1.17)	76.69 (9.04)
0.005	95.34 (1.62)	94.95 (2.30)	75.67 (9.04)
$1.6 \cdot 10^{-3}$*	94.11 (1.32)	94.03 (2.52)	71.08 (8.23)
$5 \cdot 10^{-4}$*	91.67 (1.91)	91.48 (3.23)	69.30 (7.13)
$1.6 \cdot 10^{-4}$*	91.27 (2.20)	90.80 (3.17)	68.90 (10.62)
0.0*	89.71 (3.62)	90.35 (4.52)	70.01 (8.27)

The results are shown in Tables 5.3 and 5.4. As in the remainder of this thesis, the best results are printed in bold. The ε values giving the highest accuracy in application mode are the ones selected; values with which the multiple-highest-rating problem as described above occurs are not taken into account. The tables show that the initial choice of $\varepsilon = 0.05$ indeed is a good one for the C model.

In all experiments in the remainder of this thesis, the optimal hyperparameter settings for the different models as shown in Tables 5.1 and 5.2 and Tables 5.3 and 5.4 will be used.

Table 5.5 Matrix configuration optimisation, H model: effect of different configurations of matrix types. Tablature dataset, four-voice (top) and three-voice (bottom) pieces.

Configuration	Application
	acc
(1)	74.35 (6.86)
(2)	74.35 (6.86)
(3)	75.17 (7.92)
(4)	**75.19** (7.91)
(1)	**85.70** (8.54)
(2)	**85.70** (8.54)
(3)	82.57 (10.88)
(4)	82.57 (10.88)

Table 5.6 Matrix configuration optimisation, H model: effect of different configurations of matrix types. Bach dataset, four-voice (top) and three-voice (bottom) pieces.

Configuration	Application
	acc
(1)	67.96 (4.89)
(2)	**67.97** (4.89)
(3)	66.89 (4.98)
(4)	66.87 (5.01)
(1)	**80.44** (4.03)
(2)	**80.44** (4.04)
(3)	79.69 (4.12)
(4)	79.70 (4.09)

5.2   Hidden Markov model and matrix configuration optimisation

The second preliminary experiment applies to the hidden Markov model used in MA3, the H model. In what may be called an optimisation process, for this model four different configurations of transition probability matrix (TPM), observation probability matrix (OPM), and initial state matrix (ISM) types are explored, where a matrix is said to be *data-extracted* when the probabilities it contains are extracted from the data as described in Section 3.1.4, and *uniform* when the probabilities it contains are all equal:

- **(1)** A data-extracted OPM, a uniform TPM, and a uniform ISM.
- **(2)** A data-extracted OPM, a uniform TPM, and a data-extracted ISM.
- **(3)** A data-extracted OPM, a data-extracted TPM, and a uniform ISM.
- **(4)** A data-extracted OPM, a data-extracted TPM, and a data-extracted ISM.

This is done on both datasets; the results are shown in Tables 5.5 and 5.6. Note that there is only one set of accuracies; these correspond to the neural network models' accuracies in application mode.

The tables show that the influence of the type of ISM used is negligible (compare the results for configuration (1) versus those for configuration (2)

Table 5.7 Experiment 1: modelling approaches. Tablature dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training, test, and application mode.

Model	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
N	**92.97**	**89.06**	**87.45**	**92.04**	**88.08**	**86.52**	**79.63**	**87.44**	**86.28**
	(0.29)			(2.46)			(5.96)		
C	84.02	83.50	78.18	80.85	83.39	77.46	75.26	84.29	78.95
	(1.78)			(6.17)			(6.90)		
H							74.35	77.95	72.16
							(6.86)		
N	**96.57**	**94.66**	**93.33**	**93.68**	**90.89**	**89.60**	**87.01**	**90.40**	**90.43**
	(0.29)			(2.75)			(7.03)		
C	90.78	88.85	86.98	89.59	90.66	88.51	84.75	90.30	89.63
	(1.09)			(6.02)			(7.93)		
H							85.70	88.43	84.56
							(8.54)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

Model	Training		Test		Application	
	C	H	C	H	C	H
N	*0.0039*		*0.0039*		*0.0078*	*0.0039*
C						0.2500
N	*0.0312*		0.1562		0.0937	0.6875
C						0.6875

(c) Runtimes.

Model	Run-time
N	448
C	3454
N	84
C	308

and the results for configuration (3) versus those for configuration (4)), but that the influence of the type of TPM used is not (compare the results for configurations (1) and (2) versus those for configurations (3) and (4)): on three out of the four subdatasets, clearly better results are achieved using a uniform TPM, while on the remaining subdataset (the four-voice tablature subdataset), better results are achieved using a data-extracted TPM. Given this outcome, it is decided to select configuration (2), where a data-extracted OPM, a uniform TPM, and a data-extracted ISM are used, for the experiments in the remainder of this thesis.

Table 5.8 Experiment 1: modelling approaches. Bach dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training, test, and application mode.

Model	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
N	**98.04** (0.06)	**96.40**	**96.56**	**97.57** (0.85)	**95.86**	**96.02**	**80.70** (7.50)	93.91	93.81
C	96.95 (0.22)	94.54	94.30	96.86 (1.06)	94.79	94.60	79.56 (6.80)	**94.48**	**94.40**
H							67.97 (4.89)	74.35	66.99
N	**98.86** (0.06)	**97.95**	**98.06**	**98.62** (0.95)	**97.68**	**97.91**	**92.49** (5.03)	**97.19**	**97.13**
C	97.91 (0.36)	96.79	96.49	97.87 (1.29)	96.95	96.68	90.02 (5.17)	96.68	96.49
H							80.44 (4.04)	83.45	80.45

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

Model	Training		Test		Application	
	C	H	C	H	C	H
N	*0.0001*		*0.0001*		0.6794	*0.0001*
C						*0.0001*
N	*0.0001*		*0.0001*		*0.0024*	*0.0001*
C						*0.0001*

(c) Runtimes.

Model	Run-time
N	2045
C	4466
N	3946
C	3804

5.3   Experiment 1: modelling approaches

In this first experiment, the three main models—the note-level neural network model, N (MA1); the chord-level neural network model, C (MA2); and the chord-level hidden Markov model, H (MA3)—are compared. The comparison is done separately on the three-voice and four-voice tablature subdatasets and the three-voice and four-voice Bach subdatasets. As explained in Section 3.2.4, the evaluation metrics in training mode are informative about a model's capacity to adapt to the training data, while those in test and application mode are informative about its capacity to generalise on unseen data. The metrics in test mode reflect its optimal performance on unseen data, and those in application mode its actual performance on unseen data. The results for all models on both datasets are shown in Tables 5.7a and

5.8a.[5] As in the remainder of this thesis, p-values for pairwise comparison of models' accuracies in training, test, and application mode are given in a separate table directly below the results table (in this case, Tables 5.7b and 5.8b). p-values < 0.05 (which indicate statistical significance) are printed in italics; p-values < 0.0001 are not further specified. Lastly, runtimes (in seconds)—the runtime is the time it takes for a model to be both trained *and* evaluated—for the N and C models are shown in Tables 5.7c and 5.8c.[6]

Focusing on model performance, the results shown in Tables 5.7 and 5.8 can be summarised as follows:

▶   The N model yields the highest accuracy values in all modes, where the difference is generally the largest on the tablature dataset. In training mode, its accuracy values are statistically significantly better than those for the other two models in all four cases, in test mode, they are statistically significantly better than those for the C and H models in respectively three and four out of the four cases, and in application mode in respectively two and three out of the four cases. The N model also always yields the highest soundness and completeness values in training and test mode; in application mode, it yields higher values than those for the C and H models in respectively three and four out of the four cases.

▶   In terms of accuracy, the C model yields either much higher or similar (and not statistically significantly different) values than the H model. The data type seems to play a role with respect to the partial high performance of the H model; this is discussed below. In terms of soundness and completeness, the C model always yields higher values than the H model.

▶   The H model generally yields the lowest values, both in terms of accuracy, but especially in terms of soundness and completeness.

Furthermore, it is noticed that:

---

[5]Previous results with slightly different versions of the N model and the H model on the four-voice tablature subdataset and on two smaller three-voice and four-voice Bach subdatasets are published in de Valk et al. (2013), with another slightly different version of the N model on the same three-voice and four-voice tablature subdatasets in de Valk and Weyde (2015), and with four different versions of the C model on the four-voice Bach subdataset in Weyde and de Valk (2016).

[6]All experiments are run on an HP ENVY 15 Notebook PC with an Intel Core i5-4210U dual-core processor and 8 GB RAM. Because of its combined implementation (see Section 3.4, footnote 11), runtimes for the H model are not calculated.

▶ The accuracies decrease in increasingly larger steps when moving from training to test to application mode. While the difference between values in training and test mode tends to be small, that between values in test and application mode is large to very large. Correspondingly, the standard deviation of the accuracy grows in increasingly larger steps, starting with a generally very small value in training mode. Overall, the standard deviation values are the lowest for the N model (clearly so in training and test mode), and the highest for the C model.

▶ Soundness and completeness values, on the other hand, remain much more stable when moving from training to test to application mode.

▶ Runtimes are generally lower for the N model; the difference is more pronounced on the tablature dataset.

From these observations, the following can be concluded. Overall, the N model performs best, is the most stable, and is the most efficient. In comparison to the other models, this model performs notably well on the tablature dataset. It is followed by the C model, and, at some distance, the H model. The higher standard deviations in all three modes for the C model show that this model is the most unstable. The generally large differences between accuracies in test and application mode, witnessed for both the N and C models, indicate that in application mode, errors tend to propagate. However, the differences between soundness and completeness (evaluation metrics measuring transitions between notes) in test mode on the one hand and application mode on the other, which are only small, suggest that notes are often assigned to the incorrect voice in groups. Although strictly speaking incorrect (as reflected by the lower accuracy), such assignments are often musically acceptable—that is, both contrapuntally correct and consistent with contemporary musical style. An example of this is given in Section 5.7.2, where the subject of error propagation is discussed in more detail. The smaller differences in runtimes on the Bach dataset, lastly, can be explained by the fact that on this dataset, sustained previous notes reduce the mapping possibilities for the chords (see Section 3.1.3.1). This can then reduce the size of the training set created in each training iteration (see Section 3.1.3.2), and can thus speed up the training considerably.

Taking into account that the datasets differ with respect to two main parameters—textural density, (that is, number of voices), and data type (that is, Renaissance lute music written in tablature or Baroque keyboard music written in (early) modern staff notation), the following is noticed:

- ▶   All three models perform better on the three-voice subdatasets. The differences are generally less pronounced in training and test mode (especially on the Bach dataset), but very well noticeable in application mode.

- ▶   In all modes, the N and C models perform better—at times even considerably—on the Bach dataset than on the tablature dataset (with the possible exception, perhaps, of the N model applied to the four-voice subdataset, where the accuracies in application mode are similar). The performance difference is the most striking in the case of the C model, especially in training and test mode. Conversely, however, the H model always performs considerably better on the tablature dataset than on the Bach dataset.

The first observation, regarding textural density, indicates that the task of voice separation becomes more complicated as the number of voices grows. With respect to the second observation, it is hypothesised, firstly, that the better performance of the N and C models on the Bach dataset is related to the more comprehensive information on note duration available in this dataset, where the full note durations are given (and used in the feature calculation), and where note overlap thus occurs. In the case of the C model, a direct (and observable) reason for the better performance on this dataset is the fact that, as already mentioned above in the context of runtime differences, sustained previous notes reduce the mapping possibilities for the chords (see Section 3.1.3.1). This is clearly beneficial to the learning. In the case of the N model, then, it is more difficult to pinpoint exactly how the more comprehensive duration information relates to the better performance. The encoding in the feature vectors of note overlap, which implies the unavailability of the voices that the notes causing the overlap (that is, sustained previous notes in forward and interrupting next notes in backward processing mode) belong to, is presumed to play a role here. (Of course, this would apply also in the case of the C model.)

Secondly, the reason why the H model, which makes use of pitch information only, performs better on the tablature dataset than on the Bach dataset is hypothesised to be related to the greater uniformity in mode (key) in the former dataset. Because not all modes fit the instrument well, in the tablature dataset—and in lute music in general—only a select number of modes are used. In the Bach dataset, on the other hand, all 24 keys are used. This, and the fact that this dataset is larger, leads to a much greater diversity of chords, and increases the risk of encountering a chord in a test set that

only seldom, or even never, occurs in the complementary training set.[7] This complicates the correct mapping decision for that chord. A possible solution to this issue would be to transpose, in a preprocessing step, all pieces in the dataset to the same key, or to represent each chord as a feature vector encoding relative rather than absolute pitches. It is worth noting, lastly, that on the tablature dataset the accuracy values for the relatively simple H model are similar to those for the much more complicated C model (on the three-voice subdataset they are even higher—although the difference is not statistically significant).

5.3.1   Conclusion

Considering the above, in the remainder of this thesis the focus will be on MA1, the note-level classification approach. The decision to drop MA2, the chord-level regression approach, and MA3, the chord-level probabilistic approach, is mostly a practical one and stems from the boundaries that must be set when deciding on the scope of the thesis; it is not to say that these approaches are not worth further investigation. MA2 already yields good results, especially on the Bach dataset, but has the disadvantage that it is computationally more expensive (training times can be prolonged by up to a factor of approximately eight); MA3, which in its current form is rather straightforward, is expected to yield better results when extended. It is therefore envisaged that these lines of research will be revisited in future work.

5.4   Experiment 2: features

In this experiment, which consists of two sub-experiments, the validity of the features is explored. In Experiment 2.1, the performance of the N model is measured when increasingly more context information is encoded in the feature vectors; in Experiment 2.2, the effect of the inclusion of tablature information in the feature vector is investigated.

5.4.1   Experiment 2.1: context information

The first sub-experiment serves to demonstrate the effect of an increase of context information encoded in the feature vectors. The model is trained

---

[7]The chord dictionaries for the three-voice and four-voice tablature subdatasets contain 190 and 482 unique chords, respectively; those for the three-voice and four-voice Bach subdatasets 2004 and 2471 unique chords.

Table 5.9 Experiment 2.1: features, context information. Tablature dataset, four-voice pieces. FV = feature vector.

(a) Results in training, test, and application mode.

FV	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
A	69.65	71.70	82.53	68.27	71.41	82.05	67.52	76.29	75.77
	(0.54)			(4.58)			(10.58)		
B	79.57	83.76	81.16	78.24	82.72	80.72	78.22	82.86	80.52
	(0.60)			(5.18)			(5.45)		
C	80.24	84.39	82.66	78.73	83.44	81.74	78.87	83.71	81.69
	(0.61)			(5.56)			(5.70)		
D	**92.97**	**89.06**	**87.45**	**92.04**	**88.08**	**86.52**	**79.63**	**87.44**	**86.28**
	(0.29)			(2.46)			(5.96)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

FV	Training			Test			Application		
	B	C	D	B	C	D	B	C	D
A	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
B		*0.0039*	*0.0039*		0.4257	*0.0039*		0.3007	0.1289
C			*0.0039*			*0.0039*			0.1640

and evaluated using four different feature vectors, which are constructed along the lines of the four categories as described in Section 3.1.7. Feature vector A thus contains only the note-level features (that is, 5 features on the tablature dataset or 3 features on the Bach dataset); in feature vector B (8 or 6 features, respectively), the note-chord features are added; in feature vector C (16 or 13 features, respectively), the chord-level features are added; and in feature vector D (41 or 33 features, respectively), lastly, the polyphonic embedding features are added. (Feature vector D is thus equal to the full feature vector as used in all other experiments in this thesis.) The context information encoded thus increases from highly local (feature vector A), via two intermediate stages (feature vectors B and C), to more global (feature vector D).

The experiment is conducted on the two four-voice subdatasets only. Furthermore, in order to guarantee that differences in performance are solely due to different inputs, in this experiment a model with the same, fixed hidden layer size is used for each feature vector. The hidden layer size decided upon is the one determined to be optimal when feature vector D is used (see Tables 5.1 and 5.2), which contains thus either 41 (tablature dataset) or 33 (Bach

Table 5.10 Experiment 2.1: features, context information. Bach dataset, four-voice pieces. FV = feature vector.

(a) Results in training, test, and application mode.

FV	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
A	66.79	70.98	76.99	65.94	71.03	76.79	65.58	83.22	78.97
	(0.21)			(4.15)			(9.49)		
B	83.17	90.80	87.12	82.04	90.91	86.86	78.55	90.35	85.53
	(0.26)			(4.72)			(4.76)		
C	84.68	91.52	87.53	83.51	91.56	87.19	**81.42**	91.96	87.39
	(0.24)			(4.07)			(3.79)		
D	**98.04**	**96.40**	**96.56**	**97.57**	**95.86**	**96.02**	80.70	**93.91**	**93.81**
	(0.06)			(0.85)			(7.50)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

FV	Training			Test			Application		
	B	C	D	B	C	D	B	C	D
A	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*
B		*0.0001*	*0.0001*		*0.0007*	*0.0001*		*0.0001*	0.1336
C			*0.0001*			*0.0001*			0.8287

dataset) neurons. The results for Experiment 2.1 are shown in Tables 5.9 and 5.10.

With regard to the accuracies in training and test mode, two considerable increases are observed: when moving from feature vector A to feature vector B (where information on a note's position in a chord is added), and when moving from feature vector C to feature vector D (where information on a note's polyphonic embedding is added). A small increase is also witnessed when moving from feature vector B to feature vector C (where information on chord-level properties is added); in test mode, however, this increase is only statistically significant on the Bach dataset. Soundness and completeness follow a pattern similar to that seen in the accuracies. Looking at the accuracies in application mode, then, a different picture arises. While a notable (and statistically significant) increase in accuracy is still witnessed when moving from feature vector A to feature vector B, the difference when moving from feature vector B to feature vector C is only statistically significant on the Bach dataset, and the difference when moving from feature vector C to feature vector D never. Soundness and completeness, however, as in training and in test mode, keep increasing when moving from one feature vector to

Table 5.11 Experiment 2.2: features, tablature information. Tablature dataset, four-voice (top) and three-voice (bottom) pieces. The asterisk (*) indicates that the tablature features are excluded from the feature vector. FV = feature vector.

(a) Results in training, test, and application mode.

FV	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
D	**92.97**	**89.06**	**87.45**	**92.04**	**88.08**	**86.52**	**79.63**	**87.44**	**86.28**
	(0.29)			(2.46)			(5.96)		
D*	91.99	87.73	86.30	91.03	86.84	85.53	77.64	86.23	85.70
	(0.28)			(3.44)			(7.08)		
D	**96.57**	**94.66**	**93.33**	**93.68**	**90.89**	**89.60**	**87.01**	**90.40**	**90.43**
	(0.29)			(2.75)			(7.03)		
D*	95.65	93.21	91.96	92.19	90.54	88.73	84.84	89.78	89.05
	(0.31)			(3.80)			(9.47)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

FV	Training	Test	Application
	D*	D*	D*
D	*0.0039*	0.1289	0.0976
D	*0.0312*	0.3125	0.8437

the next.

5.4.2   Experiment 2.2: tablature information

In the second sub-experiment, the effect of the inclusion of tablature information in the feature vector is investigated. The model is therefore trained and evaluated using a version of feature vector D from which all tablature features—that is, the features that encode information that can only be deduced from the tablature (and that relate to properties of the instrument): `course`, `fret`, `maxDuration`, and `adjNoteOnSameCourse`—have been removed. The results are then compared with those yielded by the model that uses the complete feature vector. It should be noted that by removing all tablature features, the feature vector for use on the tablature dataset becomes essentially the same as the one for use on the Bach dataset. The only difference is the position of the duration feature, which in case of the tablature dataset is a chord-level feature, whereas in case of the Bach dataset it is a note-level feature.

This experiment is done on both tablature subdatasets. As in Experiment

2.1, a model with the same, fixed hidden layer size (the one determined to be optimal for the full feature vector, containing 41 neurons) is used for both feature vectors. The results are shown in Table 5.11.

As the table shows, in all modes the model performs better on both subdatasets when the tablature features are included in the feature vector—but the accuracy differences are only statistically significant in training mode.

5.4.3   Conclusion

The following can be concluded. The features are clearly effective: using the smallest feature vector A, containing five features in the case of the tablature dataset and only three in the case of the Bach dataset, already yields accuracies between 65–70% in all modes. Using the largest feature vector D yields good (application mode) to very good accuracies (training and test mode). As clearly shown by the improvement in all evaluation metric values in training mode, an increase in context information encoded in the feature vectors leads to an increasingly better adaptation to the data. Especially the note-chord features (added in feature vector B) and the polyphonic embedding features (added in feature vector D), both of which cause a sizeable performance boost when added to the feature vector, turn out to be highly valuable. As shown by the increasing accuracies in test mode and the increasing soundness and completeness in test and application mode, the availability of more context information also seems to lead to an increasingly better generalisation on new data. The latter, however, is contradicted by a stabilisation in accuracy in application mode. Considering the results in test mode, this is believed to be due to error propagation.

On the tablature dataset, the inclusion of tablature features in the feature vector leads to slightly better results, but the accuracy difference is only statistically significant in training mode. It is therefore hypothesised that, although its inclusion leads to a better adaptation to the data, the tablature information as currently encoded in the features does not convey significant information about polyphonic structure.

5.5   Experiment 3: model extensions

In this experiment, the basic N model as hitherto used—that is, the model that processes the music from left to right, models only voice, and makes use of a unidirectional decision context—is extended in three manners (X1–3) in order to investigate how these extensions affect model performance. This yields three sub-experiments, each of them dedicated to an extension. In

Table 5.12 Experiment 3.1: model extensions, backward processing (X2). Tablature dataset, four-voice (top) and three-voice (bottom) pieces. PM = processing mode.

(a) Results in training, test, and application mode.

PM	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
fwd	92.97	89.06	87.45	92.04	88.08	86.52	**79.63**	**87.44**	86.28
	(0.29)			(2.46)			(5.96)		
bwd	**93.99**	**90.35**	**88.93**	**92.50**	**88.29**	**86.73**	78.62	87.17	**86.88**
	(0.27)			(4.01)			(6.95)		
fwd	96.57	94.66	93.33	93.68	90.89	89.60	87.01	90.40	90.43
	(0.29)			(2.75)			(7.03)		
bwd	**97.10**	**95.54**	**94.19**	**94.88**	**92.47**	**90.97**	**88.61**	**91.75**	**91.15**
	(0.28)			(2.24)			(6.09)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

PM	Training	Test	Application
	bwd	bwd	bwd
fwd	*0.0039*	0.7343	0.4257
fwd	*0.0312*	0.0625	0.2187

Section 3.1.5 it is argued that modelling backward, where the music is processed starting at the polyphonically most transparent end, might be a more promising approach for voice separation than modelling forward. Thus, in Experiment 3.1, in addition to the forward processing mode, a backward processing mode (X2) is tried. In Section 3.1.2.3 it is argued that, because note duration plays a significant role in the voice separation process, additionally modelling duration may be beneficial to model performance. In Experiment 3.2, which applies to the tablature dataset only, instead of modelling only voice, modelling voice and duration simultaneously (X1) is therefore tested.[8] In Section 3.1.6, lastly, it is argued that the availability of more comprehensive information on the polyphonic embedding of a note or chord when the class decision is made can improve model performance. In Experiment 3.3, lastly, multi-pass processing using a bidirectional decision context (X3) is thus tried.

---

[8] For previous experiments with processing modes and modelling voice and duration simultaneously, see de Valk and Weyde (2015). In this study, a slightly different variant of the N model is used, which is applied to the two tablature subdatasets only.

Table 5.13 Experiment 3.1: model extensions, backward processing (X2). Bach dataset, four-voice (top) and three-voice (bottom) pieces. PM = processing mode.

(a) Results in training, test, and application mode.

PM	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
fwd	**98.04**	96.40	**96.56**	**97.57**	**95.86**	**96.02**	80.70	93.91	**93.81**
	(0.06)			(0.85)			(7.50)		
bwd	97.90	**96.43**	96.44	97.00	95.48	95.29	**81.32**	**94.12**	93.46
	(0.10)			(2.06)			(7.32)		
fwd	**98.86**	**97.95**	**98.06**	**98.62**	**97.68**	**97.91**	92.49	**97.19**	**97.13**
	(0.06)			(0.95)			(5.03)		
bwd	98.81	97.93	98.02	98.48	97.64	97.68	**93.33**	97.01	96.85
	(0.08)			(1.23)			(4.20)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

PM	Training	Test	Application
	bwd	bwd	bwd
fwd	*0.0001*	0.1133	0.6507
fwd	*0.0029*	0.8028	0.1650

5.5.1   Experiment 3.1: backward processing (X2)

In the first sub-experiment, the effect of the processing mode on the model performance is investigated. The model is trained and evaluated in backward processing mode, where the music is processed from right to left, starting with the final chord (see Section 3.1.5). This is done on both datasets; the results are then compared with those yielded by the basic forward-processing model as hitherto used. The results are shown in Tables 5.12 and 5.13.

The tables show that in training and test mode, on the tablature dataset all evaluation metric values are slightly higher in backward processing mode, whereas on the Bach dataset, they are either similar in both modes or slightly higher in forward processing mode. The accuracy differences, however, are always only statistically significant in training mode. In application mode, the results show more variation—but the accuracy differences are never statistically significant. The processing mode thus only has a statistically significantly positive effect on the learning, where a backward processing mode is beneficial on the tablature dataset, and a forward processing mode on the Bach dataset.

Table 5.14 Experiment 3.2: model extensions, simultaneous voice and duration modelling (X1), forward processing mode. Tablature dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training, test, and application mode. Values below the dashed line apply to duration.

Model	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
N	92.97	89.06	87.45	92.04	88.08	86.52	**79.63**	**87.44**	**86.28**
	(0.29)			(2.46)			(5.96)		
N′	**97.36**	**96.38**	**94.27**	**96.61**	**95.49**	**93.47**	70.47	77.25	71.36
	(0.18)			(1.85)			(8.13)		
	71.55			68.58			72.33		
	(1.67)			(6.61)			(7.84)		
N	96.57	94.66	93.33	93.68	90.89	89.60	**87.01**	**90.40**	**90.43**
	(0.29)			(2.75)			(7.03)		
N′	**98.57**	**98.12**	**96.63**	**96.90**	**95.61**	**94.67**	75.35	77.78	75.65
	(0.15)			(1.83)			(10.30)		
	75.07			70.58			71.78		
	(2.42)			(4.04)			(3.24)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

Model	Training	Test	Application
	N′	N′	N′
N	*0.0039*	*0.0039*	*0.0078*
N	*0.0312*	*0.0312*	*0.0312*

5.5.2   Experiment 3.2: simultaneous voice and duration modelling (X1)

In the second sub-experiment the effect of modelling voice and duration simultaneously (see Section 3.1.2.3) on the model performance is investigated. Two aspects are considered: the effect of additionally modelling duration on the voice assignment performance, and the effect of the processing mode on the duration assignment performance. For this sub-experiment, which applies to the tablature dataset only, the N′ model is used. This model is trained and evaluated on the two subdatasets, both in forward and in backward processing mode. The results are then compared with those yielded by the basic forward-processing and backward-processing N models and to each other. At this point it should be noted that, as already mentioned in Sec-

Table 5.15 Experiment 3.2: model extensions, simultaneous voice and duration modelling (X1), backward processing mode. Tablature dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training, test, and application mode. Values below the dashed line apply to duration.

Model	Training			Test			Application		
	acc	snd	cmp	acc	snd	cmp	acc	snd	cmp
N	**93.99**	**90.35**	**88.93**	**92.50**	**88.29**	**86.73**	**78.62**	**87.17**	**86.88**
	(0.27)			(4.01)			(6.95)		
N′	93.14	89.01	87.57	91.65	87.10	85.58	77.71	86.93	85.84
	(0.30)			(4.91)			(7.89)		
	80.10			76.37			84.75		
	(1.68)			(6.87)			(6.47)		
N	**97.10**	**95.54**	**94.19**	**94.88**	**92.47**	**90.97**	**88.61**	**91.75**	**91.15**
	(0.28)			(2.24)			(6.09)		
N′	96.38	94.39	92.93	94.32	92.07	90.72	88.35	91.74	90.50
	(0.40)			(2.80)			(7.55)		
	86.59			81.59			87.28		
	(2.04)			(3.21)			(4.71)		

(b) p-values for pairwise comparison of accuracies in training, test, and application mode.

Model	Training	Test	Application
	N′	N′	N′
N	*0.0039*	*0.0390*	0.4960
N	*0.0312*	0.4375	0.4375

tion 3.2.2, for the N′ model, there are two types of accuracy: voice accuracy and duration accuracy. The latter is calculated in the exact same way as the former, and measures the percentage of notes that have been assigned to the correct duration. (See also Section 3.2.3 for issues with the evaluation of voice and duration assignments for single-note unisons.) The results for the second sub-experiment are shown in Tables 5.14 and 5.15.

With regard to the effect of additionally modelling duration on the voice assignment performance, the following is observed. In forward processing mode, when both voice and duration are modelled all evaluation metric values in training and test mode are clearly higher than when only voice is modelled, but those in application mode, conversely, are much lower. Accuracy differences are statistically significant in all modes. In backward processing mode, then, when both voice and duration are modelled all evaluation metric values

in all modes are consistently slightly lower than when only voice is modelled, but here the differences are much smaller than in forward processing mode. Moreover, the accuracy differences are only always statistically significant in training mode. Thus, in forward processing mode additionally modelling duration has a statistically significantly positive effect on the learning, but a statistically significantly negative effect on the generalisation (at least in a real-world application). In backward processing mode, almost the opposite is witnessed: the effect on the learning is statistically significantly negative, while that on the generalisation is negligible.

A possible explanation for these results is as follows. Let n be the note the voice class decision is made for (and the decision context be unidirectional—that is, extending to the left in forward and to the right in backward processing mode). In forward processing mode modelling voice and duration simultaneously, not only the voices to which the notes to the left of n belong are known, but also their full durations. It is hypothesised that this additional duration information facilitates the voice class decision for n. This is because information relating to notes whose offset time exceeds the onset time of n, and, more importantly, the voices these notes belong to, is now encoded in the feature vector (see Section 3.1.7.2). Among other things, this provides the model information about which voices are unavailable for n because they are still occupied (recall that each voice is assumed to be monophonic). This would explain why in forward processing mode modelling voice and duration simultaneously yields better results than modelling only voice (the exception being the results in application mode; this is discussed shortly). It would also explain why in backward processing mode, where duration information to the left of n is never available, modelling voice and duration simultaneously and modelling only voice yield much more similar results. (It does not explain, however, why in backward processing mode modelling voice and duration simultaneously actually consistently yields a slightly *worse* voice assignment performance than modelling only voice. It is hypothesised that this has to do with the more complex model that is used when modelling voice and duration simultaneously.) The results in application mode in forward processing mode, however, do not fit this picture. Here, all evaluation metric values are considerably lower when modelling voice and duration simultaneously than when modelling only voice. It must be noted at this point that the duration accuracy in forward processing mode is consistently relatively low (an explanation for this is given below)—on average between 25–30% of the notes get assigned to the wrong duration. In application mode, then, it is the duration information provided by the model (and not the correct duration information, as in training and test mode) that is used in the feature calculation. It is hypothesised that when this information is partly incorrect, as is the case

in application mode, this has a strong negative effect on the voice assignment performance.

With regard to the effect of the processing mode on the duration assignment performance, then, it is observed, first, that the duration accuracies are consistently much higher in backward processing mode than in forward processing mode. Backward processing thus yields better results than forward processing, both with respect to learning and generalisation. Second, it is observed that both in forward and in backward processing mode, the duration accuracies in test mode are lower than those in training mode, which is to be expected, but that those in application mode are always higher than those in test mode (and in three out of four cases even higher than those in training mode).

The first observation can be explained as follows. Let n be the note the duration class decision is made for (and the decision context be unidirectional). In backward processing mode, the voices to which the notes to the right of n belong are known. It is now hypothesised that having available this information facilitates the duration class decision for n. As explained in Section 4.1.1.1, the duration of n is determined by three (interrelated) factors: (i) the onset time of the next note in the same voice, (ii) the onset time of the next note on the same course, and (iii) a theoretical maximum duration of a semibreve. The third factor can be left out of consideration here—the theoretical maximum duration is ingrained in the target labels, meaning that notes are unlikely to be assigned to durations longer than this value (see also Section 3.1.2.3). The second factor, the onset time of the next note on the same course, determines the note-level feature `maxDuration`. Because this onset time is known in both forward and backward processing mode, `maxDuration` will thus have the same value in both. The first and most significant factor, the onset time of the next note in the same voice, however, is only known in backward processing mode, where, together with the onset times of the next notes in the other voices, it is encoded in the feature vector (in the `iOProx` features, see Section 3.1.7.2). This provides the model information about the maximum duration of n: if n is assigned to voice v, its offset time cannot exceed the onset time of the next note in v. The availability of this important additional information is thought to explain the better duration assignment performance in backward processing mode.

The second observation—that in both processing modes the duration accuracies are always higher in application mode than in test mode (and in three out of four cases also in training mode)—has to do with adaptations made because of conflicts encountered in application mode. Let n be the note the voice and duration class decisions are made for. As explained in

Section 3.3.1, in forward processing mode the duration of a sustained previous note is shortened to its maximum duration if n is assigned to the same voice as this sustained previous note. In backward processing mode, conversely, the duration of n itself is shortened to its maximum duration if it is assigned to the same voice as an interrupting next note. In both processing modes, such adaptations can lead to a considerable increase in accuracy. (The opposite phenomenon, where, due to a conflict, notes initially assigned to the correct duration are reassigned to the incorrect duration, also occurs—but far less often.)

5.5.3   Experiment 3.3: multi-pass processing using a bidirectional decision context (X3)

In the third sub-experiment, the effect of using a bidirectional decision context is investigated. As explained in Section 3.1.6, when using a bidirectional model, not only the polyphonic relation of the note the voice class decision is made for with notes opposite to the processing direction is encoded in the feature vector, but also the polyphonic relation with notes *in* the processing direction. This leads to a more comprehensive polyphonic embedding. As also explained in Section 3.1.6, a bidirectional model therefore cannot be used in a first pass through the data, as necessary information (to be precise, polyphonic information in the processing direction) is lacking. Thus, the data must first be annotated with polyphonic information acquired from a (unidirectional) first-pass model. This information is then used in the calculation of the features, both when training and when testing the model; the actual, correct polyphonic information is used exclusively as training targets.

In this sub-experiment, the polyphonic information yielded by the appropriate best-performing first-pass model is used to annotate the data with. Experiments are carried out annotating the data only with voice information, in which case information generated by the best-performing N model is used for the annotation, and (on the tablature dataset only) annotating it with both voice and duration information, in which case the information generated by the best-performing N′ model is used. In the former case, the four-voice tablature subdataset is thus annotated with voice information generated by the N model in forward processing mode, and the three-voice tablature subdataset as well as both Bach subdatasets with voice information generated by the N model in backward processing mode (see Tables 5.12 and 5.13). In the latter case, both tablature subdatasets are annotated with voice and duration information generated by the N′ model in backward processing mode

Table 5.16 Experiment 3.3: model extensions, multi-pass processing using a bidirectional decision context (X3). Tablature dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training and application mode.

Model	Training			Application		
	acc	snd	cmp	acc	snd	cmp
N (fwd)	**92.97**	**89.06**	**87.45**	79.63	**87.44**	**86.28**
	(0.29)			(5.96)		
B	84.04	86.98	84.25	80.56	86.00	82.57
	(0.88)			(5.58)		
B′	84.98	86.95	84.30	**80.93**	86.30	82.74
	(0.72)			(6.98)		
N (bwd)	**97.10**	**95.54**	**94.19**	88.61	91.75	**91.15**
	(0.28)			(6.09)		
B	93.60	93.73	92.49	88.77	90.63	88.11
	(1.22)			(6.10)		
B′	93.79	93.85	92.59	**89.79**	**91.91**	89.52
	(1.22)			(6.52)		

(b) p-values for pairwise comparison of accuracies in training and application mode.

Model	Training		Application	
	B	B′	B	B′
N (fwd)	*0.0039*	*0.0039*	0.1289	0.3593
B		*0.0039*		0.9101
N (bwd)	*0.0312*	*0.0312*	0.8437	0.0937
B		0.1562		0.0625

(see Tables 5.14 and 5.15).[9]

After the data annotation, in a second pass through the data the bidirectional model is trained and evaluated on both datasets. Because in the case of a bidirectional model the processing mode has no relevance—both in forward and in backward processing mode the exact same information is used in the feature calculation, leading to the exact same results—this is done only

---

[9]Note that the voice accuracy is slightly lower for the N′ model than for the N model (although the difference is not statistically significant). An alternative approach would be to annotate the data with the voice information generated by the model with the highest voice accuracy, and the duration information generated by the model with the highest duration accuracy. One reason for not implementing this approach is that it is more involved; another, more important, reason is that the voice information and duration information generated by the first-pass model are not independent (as discussed in Section 5.5.2).

Table 5.17 Experiment 3.3: model extensions, multi-pass processing using a bidirectional decision context (X3). Bach dataset, four-voice (top) and three-voice (bottom) pieces.

(a) Results in training and application mode.

Model	Training			Application		
	acc	snd	cmp	acc	snd	cmp
N (bwd)	**97.90**	**96.43**	**96.44**	81.32	**94.12**	**93.46**
	(0.10)			(7.32)		
B	87.82	92.40	89.81	**81.94**	93.95	90.50
	(0.41)			(5.94)		
N (bwd)	**98.81**	**97.93**	**98.02**	93.33	97.01	**96.85**
	(0.08)			(4.20)		
B	95.22	96.82	95.72	**94.33**	**97.36**	96.03
	(0.15)			(3.06)		

(b) *p*-values for pairwise comparison of accuracies in training and application mode.

Model	Training	Application
	B	B
N (bwd)	*0.0001*	0.7982
N (bwd)	*0.0001*	*0.0110*

in forward processing mode. For each dataset, the results are then compared with those yielded by the overall best-performing unidirectional model for that dataset. On the tablature dataset, where the bidirectional model yields two sets of results per subdataset (one for when the data is annotated with only voice information, and one for when it is annotated with both voice and duration information), these are also compared with one another. The results are shown in Tables 5.16 and 5.17. Recall that the bidirectional model is referred to as B when the data is annotated with only voice information, and as B′ when it is annotated with both voice and duration information. With regard to Experiment 3.3 two things must be noted. First, because the effect of the values of the hyperparameters hidden layer size and λ is minimal on the performance of the N model from a certain point on (see Section 5.1.1 and Tables A.1–A.4 in the Appendix), for the bidirectional model, which is not very different from the N model in terms of architecture, these hyperparameters are not optimised. Instead, the same values as decided upon for the N model are used (see Tables 5.1 and 5.2). Second, modelling voice and duration simultaneously is not implemented for the bidirectional model.

The tables show that in training mode, the bidirectional model always

yields lower values for all evaluation metrics than the unidirectional model, where the accuracy differences are always statistically significant. This observation can be explained by the fact that, as opposed to the unidirectional model, the bidirectional model takes as input features encoding partly incorrect polyphonic information. This makes learning the mappings from inputs to outputs more difficult; the model does not seem to benefit from the enhanced decision context here. However, despite its poorer capacity to adapt to the data, in application mode the bidirectional model always yields higher accuracies than the unidirectional model (although the difference is only statistically significant on the three-voice Bach subdataset). This apparent contradiction—a statistically significantly worse adaptation to the training data, but nevertheless a similar, if not better, generalisation on new data—seems to give a first indication that having a larger decision context is indeed beneficial to model performance. Interestingly, however, the bidirectional model's higher accuracy values in application mode are not complemented with the expected higher soundness and completeness values—as in training mode, these are generally lower for this model. A clear explanation for this remains to be found.

Table 5.16 also shows that on the tablature dataset, the B′ model generally yields higher values for all evaluation metrics than the B model, both in training and in application mode. The differences, however, are mostly small to very small, and the accuracy differences only statistically significant on the four-voice subdataset in training mode. Regardless, additionally annotating the data with duration information thus has a positive effect on model performance.

Naturally, the performance of the bidirectional model depends on the correctness of the polyphonic information the data is annotated with. The voice information used to obtain the results shown in Tables 5.16 and 5.17 is, on average, roughly 80% (four-voice subdatasets) to 90% (three-voice subdatasets) correct; the duration information roughly 85% correct. As can be seen in Tables 5.16 and 5.17, for the bidirectional model this yields voice accuracy values that are slightly higher, but in the same order of magnitude. An interesting question now is how the correctness of the polyphonic information that the data is annotated with for the second pass influences the bidirectional model's performance. As a proof of concept, the model is thus trained and evaluated on data annotated with the correct voice and duration information. Its performance is then compared with that of the best-performing unidirectional model in *test* mode, where the features are also calculated using the correct polyphonic information.[10] These results are

---

[10]The situation is not entirely similar to unidirectional test mode, as in the latter,

Table 5.18 Experiment 3.3: model extensions, multi-pass processing using a bidirectional decision context (X3), use of correct polyphonic information for data annotation. Tablature dataset, four-voice (top) and three-voice (bottom) pieces. The asterisk (*) indicates that the data is annotated with the correct polyphonic information.

(a) Results in training and test mode.

Model	Training			Test		
	acc	snd	cmp	acc	snd	cmp
N (fwd)	92.97	89.06	87.45	92.04	88.08	86.52
	(0.29)			(2.46)		
B*	95.34	92.65	91.12	94.36	91.49	89.49
	(0.31)			(3.12)		
B′*	**98.05**	**97.46**	**95.50**	**97.42**	**96.79**	**94.41**
	(0.11)			(1.17)		
N (bwd)	97.10	95.54	94.19	94.88	92.47	90.97
	(0.28)			(2.24)		
B*	97.75	96.58	95.28	95.58	93.16	91.77
	(0.27)			(1.99)		
B′*	**99.23**	**99.30**	**97.80**	**97.96**	**97.29**	**95.83**
	(0.18)			(1.28)		

(b) *p*-values for pairwise comparison of accuracies in training and test mode.

Model	Training		Test	
	B*	B′*	B*	B′*
N (fwd)	*0.0039*	*0.0039*	*0.0039*	*0.0039*
B*		*0.0039*		*0.0039*
N (bwd)	*0.0312*	*0.0312*	*0.0312*	*0.0312*
B*		*0.0312*		*0.0312*

shown in Tables 5.18 and 5.19.

As becomes clear from Tables 5.18 and 5.19, when the data is annotated with the correct polyphonic information, both in training and in test mode the B model always outperforms the N model. Moreover, on the tablature dataset the B′ model now also outperforms the B model. The differences are generally considerable, in some cases even large (up to 5–10 percentage points on the four-voice tablature subdataset); moreover, accuracy differences are always statistically significant.

The following general conclusion can now be drawn. A bidirectional model is better equipped to perform the task of voice separation; this ap-

---

conflicts between voice class decisions are not resolved.

Table 5.19 Experiment 3.3: model extensions, multi-pass processing using a bidirectional decision context (X3), use of correct polyphonic information for data annotation. Bach dataset, four-voice (top) and three-voice (bottom) pieces. The asterisk (*) indicates that the data is annotated with the correct polyphonic information.

(a) Results in training and test mode.

Model	Training			Test		
	acc	snd	cmp	acc	snd	cmp
N (bwd)	97.90 (0.10)	96.43	96.44	97.00 (2.06)	95.48	95.29
B*	**99.36** (0.06)	**98.77**	**98.80**	**99.11** (1.01)	**98.56**	**98.48**
N (bwd)	98.81 (0.08)	97.93	98.02	98.48 (1.23)	97.64	97.68
B*	**99.47** (0.04)	**98.99**	**99.10**	**99.50** (0.75)	**99.13**	**99.14**

(b) *p*-values for pairwise comparison of accuracies in training and test mode.

Model	Training	Test
	B*	B*
N (bwd)	*0.0001*	*0.0001*
N (bwd)	*0.0001*	*0.0001*

plies both to its capacity to learn and its capacity to generalise. Its actual, real-world performance, however, depends to a fair extent on the correctness of the polyphonic information that is used to annotate the data with. Results on the tablature dataset, lastly, show that annotating the data with both voice and duration information has a positive effect on the bidirectional model's performance.

5.5.4   Conclusion

With respect to the model extensions, the following can be concluded. Experiment 3.1 shows that when modelling only voice, the processing mode only has a statistically significantly positive effect on the learning, where a backward processing mode is indeed beneficial on the tablature dataset, but a forward processing mode on the Bach dataset.

Experiment 3.2 shows that, in terms of voice assignment performance, in forward processing mode additionally modelling duration has a statistically significantly positive effect on the learning, but a statistically significantly

negative effect on the generalisation. In backward processing mode, on the other hand, almost the opposite situation is witnessed—the difference being that the effect on the generalisation is now negligible. In forward processing mode, additionally modelling duration can thus be beneficial to the voice assignment performance—but under the condition that the duration information used does not contain too many errors (which is hypothesised to explain the poor generalisation when voice and duration is modelled simultaneously). In backward processing mode, where the duration information is not used in the feature calculation, additionally modelling duration is never beneficial to the voice assignment performance. Experiment 3.2 also shows that, in terms of duration assignment performance, backward processing yields better results than forward processing, both with respect to the learning and the generalisation. This is hypothesised to be due to the voice information that is used in the feature calculation.

Experiment 3.3, lastly, shows that a bidirectional model is better equipped to perform the task of voice separation than a unidirectional model. Its actual, real-world performance, however, depends on the correctness of the polyphonic information that is used to annotate the data with. Furthermore, this sub-experiment shows that on the tablature dataset, annotating the data with both voice and duration information is beneficial to model performance.

5.6   Experiment 4: comparison with existing voice separation systems

In this final experiment, the performance of the N and B models is compared with that of each of the existing systems for voice separation for which quantitative results have been documented. These include five of the rule-based systems: the contig mapping approach presented by Chew and Wu (2005), which is modified by Ishigaki et al. (2011), the algorithm by Madsen and Widmer (2006), and the two successive versions of the Voice Integration/Segregation Algorithm (VISA) by Karydis et al. (2007a,b) and Rafailidis et al. (2009), respectively. Moreover, they include both machine learning systems: the learned-predicate-based VoiSe by Kirlin and Utgoff (2005), and the probabilistic system by Jordanous (2008). A full discussion of these (and more) systems is presented in Section 2.3.2.

The approach taken is to train and evaluate, using the different evaluation metrics proposed, the N and B models on the different datasets used. An alternative and more strict approach would be to evaluate existing or newly created implementations of the various systems on the tablature and Bach

datasets used in this thesis; this approach, however, is not followed for the simple reason that it would be a much too laborious and time-consuming process.

In order for pairwise comparisons between systems as performed in this section to be fair, it is imperative, first, that per pair the same dataset is used for the evaluation, and second, that the same evaluation metrics are used. Both, however, are not without problems.

5.6.1   Datasets

As is shown in Table 5.21, six out of seven of the systems are evaluated on one or more different configurations of pieces taken from the same superset of 78 keyboard pieces by Johann Sebastian Bach, containing the 48 two-voice to five-voice fugues from books I and II of *Das Wohltemperirte Clavier*, the 15 two-voice inventions (BWV 772–786), and the 15 three-voice inventions (BWV 787–801). The smallest configurations consist of a single piece; the largest consists of all 78 pieces. VoiSe, the system by Kirlin and Utgoff, is trained and evaluated on four excerpts from yet another Bach composition: the chaconne from the second violin partita (BWV 1004). These excerpts are specifically selected for their study, where texture and style form the main selection criteria. In all, two data formats are used: MIDI and `**kern`.

Three main problems now arise, the first of which concerns the provenance (and availability) of the data. The documentation shows that the respective datasets are retrieved from at least four resources, but since not all authors specify this, there might be more.[11] The second problem concerns possible inconsistencies due to any adaptations of the data. Only Chew and Wu address adaptations in some detail; they discuss quantisation as well as the treatment of chords containing more notes than there are voices in a piece (cf. Section 4.2.1 on in-voice chords and temporarily added extra voices). The third problem concerns possible inconsistencies deriving from manual annotation of the data, which in the case of unstructured datasets (more on this below) is necessary.

As discussed in Section 4.2.1, the Bach dataset used throughout this thesis, containing all three-voice and four-voice fugues, is retrieved from the MuseData repository. All data that is lacking for the current experiment— the inventions and the two-voice and five-voice fugues—is retrieved from there

---

[11]These resources are the MuseData repository at `http://www.musedata.org` (Chew and Wu, fugues; Madsen and Widmer, fugues); The Midi Archive at `http://archive.cs.uu.nl/pub/MIDI` (Chew and Wu, inventions); the Johann Sebastian Bach Midi Page at `http://www.bachcentral.com` (Madsen and Widmer, inventions); and KernScores at `http://kern.humdrum.org` (Rafailidis et al.; Jordanous).

as well. As already described in Section 4.2.1, three kinds of adaptation are made to the three-voice and four-voice fugues: the separation of the original MIDI files into sets of MIDI files, the reduction of in-voice chords, and the removal of temporarily added extra voices. These adaptations apply to the remaining data as well, where it must be noted that in the case of the three-voice inventions, the separation of the original MIDI files into sets of MIDI files is slightly more involved, as in this data all information is always stored in a single channel. The separation is therefore carried out manually, where the interpretation as given in the Neue Bach-Ausgabe *Urtext* edition (von Dadelsen, 1970) is followed.[12] Furthermore, the original data for both the two-voice and the three-voice inventions occasionally requires some quantisation: because of liberties in performance or rounding errors, sometimes overlapping MIDI notes occur within a voice. In such cases, the left note is simply shortened so that its offset time is equal to the onset time of the right note.

The MuseData repository does not contain the second violin partita as used by Kirlin and Utgoff. Although this work is available in MIDI format elsewhere, no attempts are made to replicate their highly customised dataset. A comparison with Kirlin and Utgoff's system, which is further complicated by their detailed evaluation method, is therefore omitted.[13]

5.6.2   Evaluation metrics

In the evaluation of the systems described above, three evaluation metrics that have not yet been introduced are used: *average voice consistency*, *precision*, and *recall*. They are defined as follows:

Average voice consistency Average voice consistency (AVC), a metric devised by Chew and Wu (2005), measures, "on average, the proportion of notes from the same voice that have been assigned . . . to the same voice" (p. 15). First, the *voice consistency* (VC) for voice v is calculated as follows:

$$\mathrm{VC}(v) = \frac{100}{|S(v)|} \max_{u \in V}\{|n \in S(v) : vN(n) = u|\}, \qquad (5.2)$$

---

[12] An issue arising during this separation process is that, unlike in the two-voice inventions data and the fugues data, in the three-voice inventions data unisons are not always represented as two MIDI notes, but sometimes as only one. This issue, which leads to small voice interruptions after the separation, is left unresolved.

[13] All datasets used for Experiment 4 can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

where $S(v)$ is the set of notes assigned to v, V is the set of all voice indices, and $vN(n)$ is the correct voice for note n. The AVC, then, is the average voice consistency over all voices:

$$\text{AVC} = \frac{1}{|V|} \sum_{v \in V} \text{VC}(v). \tag{5.3}$$

Precision Precision (prc) is taken to be the percentage of notes that have been assigned to a voice and that indeed belong to that voice:

$$\text{prc} = \frac{|C|}{|C| + |I|} \cdot 100, \tag{5.4}$$

where C and I are the sets of notes assigned correctly and incorrectly to the voice, respectively.

Recall Recall (rcl), a metric complementary to precision, reflects the percentage of notes in a voice that have been successfully assigned to it:

$$\text{rcl} = \frac{|C|}{|C| + |R|} \cdot 100, \tag{5.5}$$

where C again is the set of notes assigned correctly to the voice, and R the set of remaining notes that also belong to it, but have not been assigned to it.

Two problems now arise. First, the definitions of precision and recall as given above are derived from the textual descriptions provided by Jordanous. These in turn agree with the definitions for these metrics as commonly used in the field of information retrieval—where precision is defined as the proportion of retrieved instances that are relevant, and recall as the proportion of relevant instances that are retrieved. Karydis et al. also use a precision metric, but—and this is the actual problem—in neither of their studies a definition or description is provided. It is therefore assumed that in their studies the term is also used in its conventional meaning.

Second, the definitions as given in Equations 5.4 and 5.5 are per-voice definitions. The same applies to the definitions of soundness and completeness as given in Kirlin and Utgoff, metrics also used by Madsen and Widmer. An extension of these definitions to all voices can be achieved in two manners: the per-voice numbers can either be averaged, or they can be summed and then divided by the total number of notes (or, in the case of soundness and completeness, note pairs) in all voices. For this thesis, the latter approach is adopted (see Section 3.2.2).[14] The problem now is that only in the

---

[14]C, I, and R in Equations 5.4 and 5.5 thus simply denote the sets over all voices. Note that precision and recall will now always be equal not only to each other, but also to

work of Jordanous it is clarified how the definitions are extended to all voices (she opts for the averaging approach); Karydis et al., Kirlin and Utgoff, and Madsen and Widmer do not address this. As a consequence, strictly speaking Jordanous's implementations of precision and recall are incompatible with the implementations used in this thesis; the same possibly also applies to Karydis et al.'s implementation of precision, and Madsen and Widmer's implementations of soundness and completeness. However, because the different implementations produce outcomes that are generally only marginally different, this incompatibility is taken for granted.

5.6.3   Results and conclusion

The N and B models are trained and evaluated separately on six subsets of the 78 Bach keyboard pieces, based both on composition type and number of voices: the two-voice and three-voice inventions, and the two-voice, three-voice, four-voice, and five-voice fugues. In the case of the N model, this is done both in forward and in backward processing mode. On the two-voice fugues subset, which only consists of a single piece, the piecewise cross-validation procedure that is adopted (see Section 3.2.1) poses a problem. This is solved by splitting this fugue into two, effectively creating two pieces out of it and thus enabling two-fold cross-validation. The number of pieces in the five-voice fugues subset (two) is doubled in the same manner, which in the case of this more challenging subset is done purely to have more folds in cross-validation.[15] The polyphonic information generated by the best-performing unidirectional model is used for the data annotation necessary for training and evaluating the bidirectional model. Table 5.20 gives an overview of the results on all six subsets.

    The performance of all three models in *application* mode on the 12 different datasets used is then compared with the performance of the other systems on these datasets. Each of the datasets used is either (i) equal to one of the six voice-based and composition-based subsets (datasets (1), (2), (11), and (12)), or (ii) a combination of two or more of these subsets (datasets (3) and (4)), or (iii) a single piece from one of these subsets (datasets (5)–(10)). In

---

accuracy as defined in Section 3.2.2. This is because each note that belongs to I has an equivalent in R; therefore, $|C| + |I| = |C| + |R| = |N|$ (where N is the set of all notes).

[15]The two-voice Fugue 10 in e minor (BWV 855) and the five-voice Fugue 4 in c♯ minor (BWV 849) and Fugue 22 in b♭ minor (BWV 867) (all from book I) consist of 42, 115, and 75 bars, respectively, and are split at the beginning of bars 22, 58, and 38. Note that an undesired side effect of creating two pieces out of one is that it increases the risk of identical samples ending up in both the training and the test set (see Section 3.2.1). This is tolerated in these particular cases.

Table 5.20 Experiment 4: comparison with existing voice separation systems, model performance on the six inventions and fugues subsets.

Subset	Model	Training				Test				Application			
		acc	snd	cmp	AVC	acc	snd	cmp	AVC	acc	snd	cmp	AVC
Inventions (3vv)	N (fwd)	98.89 (0.10)	97.95	97.99	98.82	98.34 (1.16)	97.40	97.37	98.23	93.28 (2.97)	96.16	96.15	92.97
	N (bwd)	99.10 (0.08)	98.34	98.37	99.05	98.20 (1.01)	97.26	97.24	98.09	95.28 (4.25)	96.88	96.73	95.07
	B	97.13 (0.29)	97.34	96.72	96.99					94.78 (4.61)	97.20	96.44	94.52
Inventions (2vv)	N (fwd)	99.85 (0.05)	99.72	99.71	99.85	99.63 (0.47)	99.32	99.32	99.62	98.76 (1.49)	99.15	99.17	98.76
	N (bwd)	99.88 (0.04)	99.77	99.77	99.88	99.51 (0.50)	99.27	99.16	99.50	99.24 (1.01)	99.18	99.12	99.23
	B	99.62 (0.07)	99.51	99.49	99.62					98.90 (1.03)	98.96	98.94	98.89
Fugues (5vv)	N (fwd)	99.69 (0.19)	99.40	99.40	99.69	90.12 (7.15)	88.57	88.61	90.31	63.63 (9.92)	87.11	86.33	69.48
	N (bwd)	99.91 (0.08)	99.83	99.83	99.92	90.75 (2.84)	87.93	89.73	90.61	68.25 (10.07)	85.21	83.56	70.73
	B	95.15 (1.11)	92.78	92.47	95.19					62.09 (4.81)	85.40	79.03	64.09
Fugues (4vv)	N (fwd)	98.04 (0.06)	96.40	96.56	98.04	97.57 (0.85)	95.86	96.02	97.54	80.70 (7.50)	93.91	93.81	81.20
	N (bwd)	97.90 (0.10)	96.43	96.44	97.91	97.00 (2.06)	95.48	95.29	96.93	81.32 (7.32)	94.12	93.46	81.39
	B	87.82 (0.41)	92.40	89.81	87.78					81.94 (5.94)	93.95	90.50	81.78
Fugues (3vv)	N (fwd)	98.86 (0.06)	97.95	98.06	98.83	98.62 (0.95)	97.68	97.91	98.55	92.49 (5.03)	97.19	97.13	92.30
	N (bwd)	98.81 (0.08)	97.93	98.02	98.78	98.48 (1.23)	97.64	97.68	98.44	93.33 (4.20)	97.01	96.85	93.21
	B	95.22 (0.15)	96.82	95.72	95.13					94.33 (3.06)	97.36	96.03	94.15
Fugues (2vv)	N (fwd)	100.00 (0.00)	100.00	100.00	100.00	99.38 (0.17)	99.13	99.13	99.36	96.54 (4.19)	99.13	99.13	96.76
	N (bwd)	100.00 (0.00)	100.00	100.00	100.00	99.38 (0.17)	98.88	98.75	99.37	99.38 (0.17)	98.88	98.75	99.37
	B	100.00 (0.00)	100.00	100.00	100.00					99.50 (0.00)	99.25	99.25	99.49

case (ii), the results given for the N and B models are the results averaged over the individual subsets as described in Section 3.2.1. In case (iii), the results given for the N and B models are the results for the respective folds where the piece in question serves as test set (and where the models are thus trained on the remaining pieces in the subset). In those cases where multiple experiments yielding different outcomes are conducted (Ishigaki et al.; Madsen and Widmer), the best results reported are used for the comparison.

It must be noted, lastly, that the comparison with the system by Madsen and Widmer is complicated by two factors. First, since Madsen and Widmer (2006) only state to have evaluated their system "mainly on the 15 two and 15 three part inventions . . . as well as the 48 fugues from the Well Tempered Clavier" (p. 58), it does not become clear whether their system is evaluated on the full dataset, on parts of it, or both. Second, they state to use soundness and completeness "as suggested by Kirlin [and Utgoff]" (p. 58) as evaluation metrics, but following the textual definitions they give, compared to Kirlin and Utgoff they seem to have switched the meanings of these metrics. The fact that the completeness values reported are consistently much lower than the soundness values further indicates implementation differences.

The results for this experiment are shown in Table 5.21. The table shows that the models generally perform as well as, and often better than, the systems. Apart from two exceptions, each system is always outperformed by at least two of the three models on at least half of the datasets it is evaluated on. The exceptions are the system by Ishigaki et al. and Madsen and Widmer: the former is only outperformed (by all three models) on one of the three datasets it is evaluated on; the latter is only outperformed (again by all three models) in terms of completeness. Where a model does not outperform a system on a dataset, its performance is generally close, or at least comparable, to that of the system. It is worth noting, finally, that the system by Jordanous, which is the only machine learning system in the comparison, is outperformed by all models on all datasets.

5.7   Overarching issues

There are three overarching issues, encountered throughout the experiments conducted in this chapter, that require further discussion: (i) conflicts and conflict resolution, (ii) error propagation, and (iii) the handling of three complex musical phenomena: single-note unisons, voice crossing, and imitation. The issue of conflicts and conflict resolution is specific to MA1, while the issue of error propagation applies to both MA1 (unidirectional model only) and MA2, approaches in which voice class or mapping decisions are made succes-

Table 5.21 Experiment 4: comparison with existing voice separation systems, pairwise comparisons. Results for the N and B models are results in application mode. EM = evaluation metric, CW05 = Chew and Wu (2005), I11 = Ishigaki et al. (2011), MW06 = Madsen and Widmer (2006), K07 = Karydis et al. (2007a,b), R09 = Rafailidis et al. (2009), J08 = Jordanous (2008).

Dataset	EM	Model			System					
		N (fwd)	N (bwd)	B	CW05	I11	MW06	K07	R09	J08
(1) Inventions (3vv)	AVC	92.97	95.07	94.52	93.35	**95.27**				
(2) Inventions (2vv)	AVC	98.76	99.23	98.89	**99.29**	98.73				
(3) Fugues (2–5vv)	AVC	86.91	87.57	87.97	84.39	**89.21**				
(4) Inventions (2–3vv) and	snd	96.08	96.11	96.22			97.58			
fugues (2–5vv)	cmp	**96.00**	95.78	94.39			71.58			
(5) Invention 1 in C major	prc	**100.00**	99.58	99.37				99.34		
(BWV 772) (2vv)	acc	**100.00**	99.58	99.37					99.00	
(6) Invention 13 in a minor	prc	**99.64**	99.29	98.58				96.45		
(BWV 784) (2vv)	acc	**99.64**	99.29	98.58					96.00	
(7) Fugue 1 in C major	prc	84.51	89.53	85.05			**92.38**			
(BWV 846) (4vv)	acc	84.51	89.53	85.05				**92.00**		
(8) Fugue 14 in f♯ minor	prc	85.93	87.44	77.09			**95.56**			
(BWV 859) (4vv)	acc	85.93	87.44	77.09				**93.00**		
(9) Fugue 7 in E♭ major	prc	89.29	94.12	**97.58**				97.52		
(BWV 852) (3vv)	acc	89.29	94.12	**97.58**					91.00	
(10) Fugue 11 in F major	prc	94.94	95.05	**96.00**				87.31		
(BWV 856) (3vv)	acc	94.94	95.05	**96.00**					94.00	
(11) Fugues (4vv)	prc	80.70	81.32	**81.94**						75.07
	rcl	80.70	81.32	**81.94**						75.23
(12) Fugues (3vv)	prc	92.49	93.33	**94.33**						88.63
	rcl	92.49	93.33	**94.33**						88.34

Table 5.22 Conflict numbers and conflict percentage c for the N, N′, B, and B′ models. Tablature and Bach datasets, four-voice pieces. PM = processing mode, TC = total number of conflicts, CC = number of conflicts resolved correctly.

| Model | PM | Exper- | Tablature dataset | | | Bach dataset | | |
		iment	TC	CC	c	TC	CC	c
N	fwd	1	15	4	0.16	85	1	0.39
	bwd	3.1	7	1	0.07	166	44	0.76
N′	fwd	3.2	41	8	0.46			
	bwd	3.2	18	8.5	0.20			
B	(fwd)	3.3	53	41	0.59	777	347	3.56
B′	(fwd)	3.3	65	39.5	0.73			

sively (rather than simultaneously), using previously generated information. The third issue, the handling of complex musical phenomena, applies to all three modelling approaches, and has to do with the amount of contextual (and precursory) information that is available to a model.

5.7.1 Conflicts and conflict resolution

An issue that concerns only MA1 is conflicting voice class decisions in application mode. As described in Section 3.3, conflicts may occur either during (unidirectional model) or after (bidirectional model) the voice assignment process. In both cases they must be resolved. To illustrate how often conflicts occur and to which extent they can thus affect model performance, the following is done. For each model used in MA1 all conflicts encountered when the model is applied to the four-voice tablature subdataset and to the four-voice Bach subdataset are counted.[16] (The three-voice subdatasets are not investigated separately, as results are expected to be similar.) The number found is then compared with the total number of voice class decisions for the subdataset (which is equal to its total number of notes), and the *conflict percentage*, c, is calculated. Furthermore, to be able to assess the success of the two conflict resolution approaches—the online approach used for the unidirectional model and the postprocessing approach used for the bidirectional model (see Sections 3.3.1 and 3.3.2, respectively)—for each conflict it is verified whether it is resolved correctly or incorrectly. The results are

---

[16]In the case of the N′ model, this is done only for the conflicts where a note's *voice class* is adapted; conflicts where its duration class is adapted are not taken into consideration as they do not affect model performance in terms of voice assignment. The latter occur far more often than the former: in forward processing mode, a total of 759 of such conflicts are counted, and in backward processing mode a total of 1339.

Table 5.23 Error propagation percentage m for the N, N$'$, and C models. Tablature and Bach datasets, four-voice pieces. PM = processing mode.

Model	PM	Exper-iment	Tablature dataset			Bach dataset		
			Test	Application		Test	Application	
			acc	acc	m	acc	acc	m
N	fwd	1	92.04	79.63	60.95	97.57	80.70	87.44
	bwd	3.1	92.50	78.62	64.93	97.00	81.32	83.95
N$'$	fwd	3.2	96.61	70.47	88.53			
	bwd	3.2	91.65	77.71	62.55			
C	(fwd)	1	80.85	75.26	22.60	96.86	79.56	84.63

shown in Table 5.22; references to the relevant experiments are also given. The presence of the half values in the table can be accounted for by conflicts involving single-note unisons, which may be resolved only half correctly (cf. the evaluation of single-note unisons as described in Section 3.2.3).

With respect to the occurrence of conflicts it is observed, first, that the conflict percentage is always very low: only in one out of nine cases the 1% boundary is exceeded. Second, it is observed that for the N and B models, the conflict percentage is always higher on the Bach dataset. This can be explained by the fact that on this dataset, in addition to conflicts with decisions for lower chord notes, conflicts with decisions for sustained previous notes or interrupting next notes can now occur (on the tablature dataset this is not so when modelling only voice). Third, it is observed that the unidirectional models yield lower conflict percentages than the bidirectional models. With respect to the success of the two conflict resolution approaches, the table shows that this is generally low for the unidirectional models, and acceptable to good for the bidirectional models. In the end, however, the absolute number of conflicts resolved incorrectly is generally lower for the unidirectional models.

5.7.2   Error propagation

A recurring issue in MA1 and MA2 is error propagation, the phenomenon encountered in application mode in which an incorrect voice or duration assignment influences the voice class or mapping decision for the following notes negatively (see Section 3.2.4.1). Error propagation can occur using the N model, the N$'$ model, as well as the C model, but not using the B model (see Section 3.2.4.2). As explained in Section 3.2.4.1, the percentage of misassignments due to error propagation, m, can be calculated. To illustrate to which extent error propagation affects model performance in the current

```
(1)  19  20   22  24   39   40   41   42   43   44   45   47   53   65
71   73   92  93   94  95   96   124  130  150  151  153  155  170  171
172  173  174  175  176  177  178  179  181  187  199  205  207  212  214
215  216  217  218  219  228  229  230  237  299  300  307
```

Figure 5.1 Error propagation, N (fwd) model: indices of notes assigned to an incorrect voice. Ochsenkun (1558_5), 'Herr Gott laß dich erbarmen'. Indices returned both in test and application mode are printed in regular type, indices returned exclusively in application mode are printed in bold, and indices returned exclusively in test mode are placed in parentheses.

implementations, the percentages are calculated for the N and N′ models, both in forward and in backward processing mode, as well as for the C model. This is done on the four-voice subdatasets only (the results on the three-voice subdatasets are again expected to be similar). The results are shown in Table 5.23; the corresponding accuracies in test and application mode, as well as references to the relevant experiments, are also given.

As the table shows, error propagation generally accounts for 60–65% of the misassignments on the tablature dataset, and even for approximately 85% of the misassignments on the Bach dataset. The two outliers on the tablature dataset, the value for the N′ model in forward processing mode and the value for the C model, are due to the relatively low accuracies in application and test mode, respectively, that these models yield. An explanation for the forward-processing N′ model's low accuracy in application mode is given in Section 5.5.2; an explanation for the C model's low accuracy in test mode is that it simply does not generalise very well—even when provided the correct polyphonic information (cf. the high test accuracies for all the other models on both datasets).

A concrete example can give some further insights. Figure 5.1 lists the indices of the notes assigned to an incorrect voice when the trained forward-processing N model is applied to a four-voice intabulation containing 371 notes (the example is taken from Experiment 1). Indices returned both in test and in application mode are printed in regular type, and indices returned exclusively in application mode are printed in bold. Indices returned exclusively in test mode, lastly, are placed in parentheses. (The absence in application mode of an index returned in test mode, which seems counter-intuitive, is due either to a correctly resolved conflict—the note has initially been assigned incorrectly as well, but because of a conflict it has been reassigned correctly—or to the use of different polyphonic information now resulting in the correct voice class decision.)
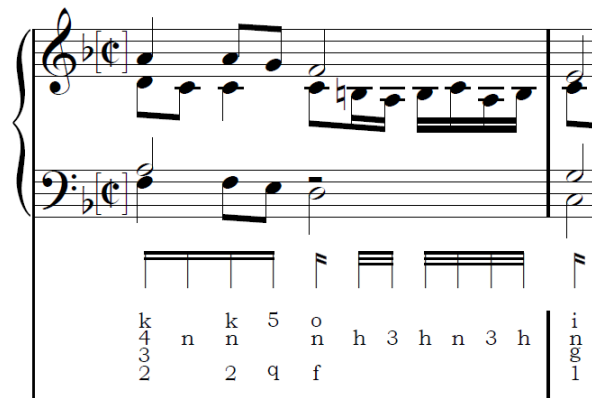
```
[S]  204 --- 208 210 213 --- --- --- --- --- --- | 223
[A]  203 205 207 --- 212 214 215 216 217 218 219 | 222
[T]  202 --- --- --- --- --- --- --- --- --- --- | 221
[B]  201 --- 206 209 211 --- --- --- --- --- --- | 220
```

(a) Correct transcription and voice assignments.

```
[S]  204 --- 208 210 213 --- --- --- --- --- --- | 223
[A]  203 --- --- --- --- --- --- --- --- --- --- | 222
[T]  202 205 207 --- 212 214 215 216 217 218 219 | 221
[B]  201 --- 206 209 211 --- --- --- --- --- --- | 220
```

(b) Voice assignments given by the model.

Figure 5.2 Error propagation group, N (fwd) model. Ochsenkun (1558_5), 'Herr Gott laß dich erbarmen', bars 14–15 (note indices 201–223).

The figure shows how single incorrect assignments shattered across the piece propagate, thus forming small groups—in this example varying in size from two to nine—of incorrect assignments. What the figure does not show is that the notes in these groups tend to be misassigned to the *same* incorrect voice. This often yields musically acceptable results. The last group in Figure 5.1, starting at index 205 and ending at index 219, is a prototypical example. Figure 5.2 shows both the correct voice assignments for the notes at indices 201–223 (which correspond to bar 14 and the first chord in bar 15), as well as the voice assignments given by the model.

As Figure 5.2 shows, the notes at indices 205, 207, 212, and 214–219 all belong to the altus, but have been misassigned by the model to the tenor.

Although strictly speaking incorrect, this assignment is musically acceptable. The latter is reflected by the evaluation metrics measuring transitions between note pairs, soundness and completeness: while the accuracy of this complete fragment is only $\frac{23-9}{23} \cdot 100 = 60.86\%$, soundness and completeness both measure $\frac{17}{19} \cdot 100 = 89.47\%$.

5.7.3   Complex musical phenomena

A recurring issue in all three modelling approaches is the handling of complex musical phenomena, three of which are discussed here: single-note unisons, voice crossing, and imitation.

5.7.3.1   Single-note unisons

Single-note unisons, that is, single notes belonging to two voices (see Section 2.2.2), are fairly common in the tablature dataset. The four-voice tablature subdataset, which contains a total of 8892 notes, counts 129 single-note unisons; the three-voice subdataset, which contains 2749 notes, counts 29 of them. Single-note unisons thus constitute approximately 1.5% of the tablature data.[17] As discussed in Section 3.2.3, in addition to the voice assignment categories *correct* and *incorrect*, for each note that is a single-note unison there are two further voice assignment categories: the *overlooked* category (the note has been assigned to only one voice and that voice is a correct voice) and the *half* category (the note has been assigned to two voices but only one of these is a correct voice). Table 5.24 shows, for all models in all modelling approaches, the spread of single-note unison assignments over the various voice assignment categories when the models are applied to the four-voice tablature subdataset in application mode; references to the relevant experiments are given.

The table shows that in all modelling approaches, single-note unisons are rarely assigned fully correctly. A positive outlier is the H model used in MA3, but even in this case the number of fully correct assignments remains low. On the other hand, single-note unisons are also only relatively rarely assigned fully incorrectly. Rather, in all modelling approaches, the overwhelming majority of assignments fall in the *overlooked* category.

The table also brings to light an undesired side effect of the deviation threshold heuristic as used in MA1, which enables a classification into two voice classes (see Section 3.1.2.1). Not only is this heuristic alone not suf-

---

[17] *Actual* unisons, that is, two notes with the same pitch and the same onset time, on the other hand, are much more rare: only six of these are counted, all in the four-voice subdataset.

Table 5.24 Spread of single-note unison assignments for all models in application mode. Tablature dataset, four-voice pieces. PM = processing mode, C = *correct*, I = *incorrect*, O = *overlooked*, H = *half*, S = *superfluous* voice assignment category.

Model	PM	Exper-iment	Voice assignment category				
			C	I	O	H	S
N	fwd	1	4	14	110	1	52
	bwd	3.1	5	10	114	0	41
N′	fwd	3.2	2	17	110	0	35
	bwd	3.2	0	8	120	1	32
B	(fwd)	3.3	6	11	112	0	104
B′	(fwd)	3.3	2	9	117	1	103
C	(fwd)	1	0	14	115	0	3
H	(fwd)	1	16	18	95	0	27

ficient to model single-note unisons, but it can also result in notes that are *not* single-note unisons being assigned to two voices. If in such a case neither of these voices is the note's correct voice, the assignment falls in the *incorrect* category, but if one of them is, it falls in the *superfluous* category. As the table shows, superfluous voice assignments are especially prevalent in the bidirectional model (in fact, they occur almost as often as overlooked voice assignments). A possible explanation for this is the fact that in this model the activation values for the individual output neurons tend to vary less. In the other two modelling approaches, especially in MA2, superfluous voice assignments are witnessed far less often.

5.7.3.2   Voice crossing and imitation

A quantification of model performance with respect to instances of voice crossing and motivic imitation, which is not as straightforward as a quantification of single-note unison assignments, is not carried out. An informal inspection of the models' output, however, shows that in the current model implementations both these musical phenomena are generally highly problematic. Figure 5.3 shows an example of a fragment containing both imitation (the tenor motif starting on E_4 in bar 1 is imitated a fifth higher by the superius in bar 2) and voice crossing (bar 3; note that the voice crossing serves to preserve the tenor motif.) That these phenomena are problematic is corroborated by the performance of all models in all modelling approaches on this fragment: firstly, the voice crossing is always reversed, and secondly, the tenor motif up to bar 3 is never kept intact (that is, is never recognised as a single entity or motif).

One reason for the models' incapacity to handle voice crossing—and, for

Figure 5.3 Imitation necessitating voice crossing. Phalèse (publ.) (1563_{12}), 'Las on peult', opening bars.

that matter, single-note unisons—correctly is hypothesised to be the limited (contextual) information that is available when the voice class or mapping decision is made. Two model adaptations are therefore expected to be helpful here: (i) the implementation of a larger decision context, and (ii) the inclusion of information about melodic trajectory. In addition to these two adaptations, an educated approach to recognition of imitative entries in a piece, lastly, requires stored knowledge of its melodic content (motifs etc.) to be available to a model.

5.7.4   Conclusion

Three overarching issues, all of them originating from limitations of the modelling approaches, are discussed: conflicts and conflict resolution, error propagation, and the handling of complex musical phenomena. The following conclusions can be drawn.

Given the overall low conflict percentages, the effect of conflicts on model performance in MA1 is marginal at best. The fact itself that relatively few conflicts occur corroborates earlier observations that the models in MA1 learn well. Furthermore, despite its low success rate, an online conflict resolution approach as used for the unidirectional models still seems to be preferable to a postprocessing conflict resolution approach as used for the bidirectional models.

Large differences in accuracy between results in test and application mode in MA1 and MA2 can be explained by error propagation occurring in application mode. A representative example shows, however, that the notes in error propagation groups tend to be misassigned to the *same* incorrect voice. This is indeed reflected by the soundness and completeness values in application mode, which generally remain much closer to the soundness and completeness values in test mode. It can therefore be concluded that despite the lower accuracy values encountered in application mode, with soundness and completeness values generally ranging between 85–95%, the voice separation performed by the various models proposed is in most cases musically acceptable.

With respect to complex musical phenomena, lastly, it is concluded, first, that in all modelling approaches, single-note unisons tend to be assigned to only one voice, where that voice is a correct voice (overlooked voice assignments). In all modelling approaches single-note unisons are thus hard to learn. Furthermore, as a results of the deviation threshold heuristic, in MA1 notes that are not single-note unisons are frequently incorrectly assigned to a second voice (superfluous voice assignments). The handling of voice crossing and imitation, second, is highly problematic in the current model implementations in all modelling approaches. This is hypothesised to be due to the lack of contextual information available to the models. Two model adaptations, which are also expected to aid in the handling of single-note unisons, are proposed: to implement a larger decision context, and to include information about melodic trajectory. In addition to these adaptations, making stored knowledge of the melodic content of a piece available to the model is expected to aid in the recognition of imitative entries.

6

Summary and conclusions

The main aim of this thesis is defined in Chapter 1 as the design, implementation, and evaluation of supervised machine learning models for voice separation in polyphonic music written in lute tablature and polyphonic music in MIDI format. This aim follows from the two-fold research question formulated in the same chapter, namely how supervised machine learning models can be used for voice separation in polyphonic music in symbolic representations, and which modelling approaches are the most effective. To meet the aim, three main objectives, each consisting of a number of sub-objectives, are formulated: Objective 1, to design and implement models; Objective 2, to create datasets; and Objective 3, to evaluate the models. In this final chapter, the research question is answered, and conclusions are drawn by readdressing individual objectives. This is done in Section 6.2. First, in Section 6.1, a summary of the research described in the preceding chapters is given. In Section 6.3, lastly, perspectives for future work are discussed.

6.1   Summary

Chapter 2 sets out the background against which the research is carried out. After three important **key terms**—*voice*, *polyphonic music*, and *voice separation*—are defined unambiguously, first the **musicological background** is outlined. The sixteenth-century lute, its music, and the notational format of this music are introduced. The four main sixteenth-century lute tablature systems are described briefly, where special attention is paid to the limitations in terms of visualising the music's polyphonic structure that this notation entails. Furthermore, the principal reason for including only intabulations—instrumental arrangements of polyphonic vocal works—in the dataset is given: they facilitate the process of labelling the data.

In the second part of Chapter 2, the **computational background** to the research is sketched. Two phases of research into voice separation are discerned. In the first phase (1980s–1990s), the focus was on the modelling of perceptual phenomena relating to polyphonic structure and auditory stream segregation. In the second phase (2000s–2010s), the development of systems for voice separation is witnessed. A total of 10 state-of-the-art systems are described in detail. The majority of these is rule-based, while two use machine learning techniques; in all systems the task of voice separation is approached differently. A common factor that links all systems together is that they are all based on at least one of two fundamental principles associated with auditory streaming: the Pitch Proximity Principle and the Principle of Temporal Continuity. Chapter 2 ends with the description of two early applications of rule-based voice separation systems intended for automatic transcription of German lute tablature. These systems, which were developed in the course of the 1980s, represent the sole efforts of research into automated analysis and transcription of lute tablature hitherto.

In Chapter 3, the methodology followed is described. A large part of this chapter deals with Objective 1, to design and implement models. First, the three main **modelling approaches** are introduced (Objective 1.1). In MA1 and MA2, standard three-layer feed-forward neural networks with resilient backpropagation as the learning algorithm and the sigmoid function as the activation function are used. MA1, which yields the N model, first, is a note-level classification approach. The task of voice separation is modelled as a multi-class classification problem, where the classes are the voices the notes can be classified into. The model learns the decision per note, where a greedy approach is followed within which no combinations of decisions are considered. This leads to a small state space when the decision must be made, which makes the approach computationally cheap. A disadvantage of the approach is that when the model is applied to unseen data, conflicting decisions—which require a conflict resolution tactic—can occur. In addition to modelling only voice, the N model can be extended to modelling voice and duration simultaneously. This is the first model extension (X1, proposed under Objective 1.2), and yields the N′ model. MA2, which yields the C model, second, is a chord-level regression approach. The task of voice separation is modelled as a regression problem, where mappings of chord notes to voices are rated. For each chord, all mappings of notes to voices are enumerated, and, using a relative training technique, the model is trained to rate the correct mapping the highest. This modelling approach, in which all possible combinations of note-level decisions are considered, leads to a large state space when the decision must be made and is therefore computationally ex-

pensive. An advantage of the approach, however, is that conflicting decisions when the model is applied to unseen data—as encountered in MA1—cannot occur. MA3, which yields the H model, lastly, is a chord-level probabilistic approach. The task of voice separation is modelled as a probability problem, where, given an observation sequence (a sequence of chords), a discrete hidden Markov model is used to estimate hidden states (mappings of notes to voices). Compared to MA1 and MA2, this approach is more straightforward in the sense that only pitch information is used in the feature vectors that represent the chords.

The **processing mode** determines the direction in which the music is processed. Two modes are discerned: forward and backward processing. The latter is only implemented for MA1, and constitutes the second model extension (X2, proposed under Objective 1.2).

The **decision context** is the polyphonic context within which the feature vectors are calculated. Two types are discerned: a unidirectional decision context, which extends to only one direction of a chord, and a bidirectional decision context, which extends to both directions. Using a bidirectional decision context is only implemented for MA1; it constitutes the third model extension (X3, proposed under Objective 1.2), and yields the B model. In its feature calculation, a bidirectional model requires the availability of polyphonic information in both directions of the note the voice class decision is made for. It can therefore only be applied in a second pass through the data, that is, after the data has been annotated with polyphonic information generated in a first pass.

The **features** that make up the feature vectors used in MA1 and MA2, respectively, stem from the same feature superset. This superset is designed for this thesis (Objective 1.3), and contains features belonging to four different categories. Note-level features capture individual properties of a note, note-chord features capture aspects of a note's position within a chord, chord-level features capture properties shared by all notes in a chord, and polyphonic embedding features capture aspects of the polyphonic relation of a note to other another note.

Evaluation is carried out using k-fold cross-validation, where k is set equal to the number of pieces in a dataset, and each complete piece serves as test set once. Three evaluation metrics are proposed: accuracy, measuring the percentage of notes that have been assigned to the correct voice, and soundness and completeness, both measuring correct transitions between notes. On unseen data, unidirectional models are evaluated in two different modes: test mode, where the feature vectors are calculated using the correct polyphonic information, and application mode, where they are calculated using the polyphonic information generated by the model. Results in test mode

are indicative of a model's optimal performance on unseen data; results in application mode reflect its actual, real-world performance on unseen data. Unlike in test mode, in application mode errors can propagate; the error propagation percentage can be quantified from the results in both modes. Bidirectional models are only evaluated in application mode; because pre-existing polyphonic information is used in the feature calculation, incorrect assignments cannot propagate.

A problem that is specific to MA1 is the occurrence of voice class decision **conflicts** when a trained model is applied to unseen data. A voice class decision conflicts with a previous voice class decision when it leads to two simultaneous or overlapping notes being assigned to the same voice. This is to be avoided, as voices are assumed to be monophonic, and the feature calculation becomes problematic if this is not so. Conflicts are only resolved in application mode; this is always done in a fixed sequence. In unidirectional application mode, where the polyphonic information generated by the model is used for the feature generation, conflicts must be resolved during the voice assignment process. In bidirectional application mode, where this is not the case, they are resolved in a postprocessing conflict resolution heuristic.

The models, the model extensions, the feature extraction algorithms, and the framework for training and evaluating the models are **implemented** in Java (Objective 1.4). For the sake of reproducible research, the code is made available as open source software.[1]

Chapter 4 is dedicated to Objective 2, to create datasets. Two datasets are created: a tablature dataset (Objective 2.1), containing six three-voice and nine four-voice intabulations, and a Bach dataset (Objective 2.2), containing the 26 three-voice and the 19 four-voice fugues from *Das wohltemperirte Clavier*. The **tablature dataset** is created from scratch. Each intabulation is represented as a machine-readable encoding of the tablature and a set of monophonic MIDI files containing the correct voice information; each fugue only as a set of monophonic MIDI files. The encodings and MIDI file sets that together constitute the tablature dataset are created manually. In the case of the encodings, this is done by transcribing facsimile reproductions of the tablature. The encoding format used, `tab+`, is a customised format developed for this thesis that supports all tablature systems. In the case of the MIDI file sets, it is done by first devising polyphonic transcriptions in modern music notation using professional music notation software, and then exporting the individual voices as separate MIDI files. The transcriptions are

---

[1]The code can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

devised by polyphonically aligning the intabulations and their vocal models, whose polyphonic structure is always unambiguous.

The **Bach dataset** is an adaptation of an existing MIDI dataset. The adaptations made, apart from the separation of the original files into sets of MIDI files, concern the removal of in-voice chords (as to ensure that each voice is monophonic) and temporarily added extra voices (as not to exceed the nominal number of voices).[2]

Chapter 5, lastly, is dedicated to Objective 3, to evaluate the models. A total of six experiments are conducted. The first two are **preliminary experiments** concerning model optimisation. In the first preliminary experiment, which applies only to the neural network models used in MA1 and MA2, three hyperparameters are optimised: the hidden layer size, the regularisation parameter λ, and the margin ε. In the second preliminary experiment, which applies only to the hidden Markov model used in MA3, the configuration of the three model matrices is optimised.

In **Experiment 1**, the models used in the three modelling approaches are evaluated (Objective 3.1). MA1 is found to be both the most effective and the most efficient; therefore, only the N model is used for the remaining experiments.

In **Experiment 2**, the relevance of the features is evaluated (Objective 3.2). Two sub-experiments are conducted, the first of which concerns the effect of the amount of context information encoded in the feature vector on model performance (Experiment 2.1), and the second the effect of tablature information being included (Experiment 2.2). It is found, first, that more context information leads to increasingly better model performance (where especially the inclusion of information on polyphonic embedding is shown to be effective), and second, that the effect of including tablature information in the feature vector is positive but small.

In **Experiment 3**, the three model extensions are evaluated (Objective 3.3) in three sub-experiments, where the effect of backward processing (Experiment 3.1), the effect of simultaneously modelling voice and duration (Experiment 3.2), and the effect of using a bidirectional decision context in a second pass through the data (Experiment 3.3) on model performance are investigated. The effect of backward processing is found to be negligible, and the effect of simultaneously modelling voice and duration is found to be positive only in forward processing mode, and only in combination with a high

---

[2]The encodings and MIDI file sets that constitute the tablature dataset, as well as the polyphonic transcriptions, and the MIDI file sets that constitute the Bach dataset, as well as a list of all notes that are removed from the original data, can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

duration assignment performance. The effect of using a bidirectional decision context, lastly, is found to be positive but small, and strongly dependent on the correctness of the polyphonic information the data is annotated with in the second pass.

In **Experiment 4**, lastly, the performance in application mode of the two unidirectional N models as well as the bidirectional N model is compared with the performance of a number of existing voice separation systems (Objective 3.4). The experiment shows that most of the systems are outperformed by at least two of the three models on at least half of the datasets they are evaluated on.[3]

Chapter 5 is concluded with a discussion of three **overarching issues**, all originating from limitations of the modelling approaches, encountered throughout the experiments. The issue of conflicts and conflict resolution, firstly, applies only to MA1, and is a direct result of the greedy approach followed. The issue of error propagation, secondly, applies to both MA1 and MA2, and has to do with the fact that voice class or mapping decisions are made successively, using previously generated information. The handling of three complex musical phenomena—single-note unisons, voice crossing, and imitation—, thirdly, applies to all modelling approaches, and has to do with the amount of contextual and precursory information that is available to the models.

6.2   Conclusions

In this thesis it is shown that machine learning models can be used for voice separation in polyphonic music in symbolic representations by:

▶   Designing different extensible modelling approaches, in each of which the problem is cast differently and in which different learning models are used.

▶   Defining a set of features relevant to the task.

▶   Implementing the models, the model extensions, the feature extraction algorithms, and the framework for training and evaluating the models.

▶   Creating datasets to evaluate the models on.

The conclusions drawn are given below.

---

[3]All datasets used for this experiment can be retrieved from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis`.

With regard to the **modelling approaches**, first, the following is concluded. MA1, the note-level classification approach, is both the most effective and the most efficient modelling approach. It is shown that the N model (as used in MA1) always performs better than the C model (as used in MA2) in both training and test mode, and that it always performs better than both the C model and the H model (as used in MA3) in application mode. The performance is found to be always statistically significantly better in training mode, and in most cases in test and application mode.[4] Although in application mode the N model is susceptible to conflicting voice class decisions, it is shown that the conflict percentages are very low, and that the effect of conflicts on model performance is marginal. Furthermore, in MA1 a good trade-off between computational cost (which is always low) and overall model performance is achieved.

The latter is not the case for MA2, the chord-level regression approach. Although this approach is conceptually more sound because a larger state space is explored when the mapping decision is made, this comes at a computational cost that is not balanced by a better performance. Unless significant performance improvement can be achieved, this renders MA2 less attractive.

In application mode, the N and C models both suffer from error propagation, where the error propagation percentage can increase up to approximately 90%. It is shown that while error propagation can affect model accuracy significantly, it has much less influence on model performance in terms of soundness and completeness. The overall high soundness and completeness values in application mode indicate that the voice separation performed, even if less accurate, is always at least musically acceptable.

MA3, the chord-level probabilistic approach, lastly, yields a fairly straightforward model that performs similarly to the C model at best. Although the H model has the advantage that it is not affected by error propagation, in its current implementation it does not meet the performance standards set by the N and C models.

With regard to the **feature set**, second, the following conclusions are drawn. The results yielded in both MA1 and MA2 demonstrate that the feature set defined is effective. The importance of the amount and type of context information available when the voice class decision is made is shown, where the inclusion of the polyphonic embedding features is found to be particularly effective for the task of correct voice separation. Furthermore, it is shown

---

[4]A model performance is said to be *statistically significantly better* than another here if all three evaluation metrics—accuracy, soundness, and completeness—are higher, and if the difference in accuracy is statistically significant.

that the tablature information currently encoded in the feature vector does not convey important information about polyphonic structure.

Third, with regard to the three **model extensions** implemented for MA1, the following is concluded. The effect of using a backward processing mode on model performance is shown to be statistically significantly positive only on the tablature dataset in training mode. It is found to be statistically significantly negative on the Bach dataset in training mode, and not statistically significant on both datasets in test and application mode. Modelling backward thus cannot be said to be beneficial to voice separation, as hypothesised.

In forward processing mode, the effect of additionally modelling duration on model performance is shown to be statistically significantly positive in both training and test mode, but statistically significantly negative in application mode. In backward processing mode, conversely, the effect is found to be statistically significantly negative in training and test mode, and negligible in application mode. As hypothesised, additionally modelling duration thus can be said to be beneficial to voice separation—but only in forward processing mode, and only if the duration assignment performance is of a certain quality. The duration assignment performance itself is shown to be statistically significantly better (in all modes) in backward processing mode.

The effect of using a bidirectional decision context on model performance is shown to be statistically significantly negative on both the tablature and the Bach dataset in training mode, but positive—although not in all cases statistically significantly—on both datasets in application mode. When the data is annotated with the correct polyphonic information (rather than the information generated in a first pass through it), the effect is shown to be statistically significantly positive on both datasets in both training and in *test* mode. Annotating the tablature dataset with both voice and duration information is shown always to have a positive effect on model performance compared to annotating it with only voice information, but only when the correct information is used the effect is shown to be always statistically significant. As hypothesised, a bidirectional model can thus be said to be better equipped for the task of voice separation than a unidirectional model; however, its real-world performance depends strongly on the correctness of the polyphonic information the data is annotated with.

Fourth, in the **evaluation** of the models on the different datasets it is shown that all three models perform better (in training, test, and application mode) on the three-voice subdatasets, and that the N and C models perform better (in all modes) on the Bach dataset, while the H model performs better on the

tablature dataset. Furthermore, it is established that all three models have difficulty handling complex musical phenomena such as single-note unisons, voice crossing, and imitation. A comparison of the performance in application mode of the two unidirectional N models as well as the bidirectional N model with that of a number of existing systems for voice separation, lastly, shows that these models can compete seriously with state-of-the-art systems.

6.3   Future work

The above conclusions show that machine learning models have good potential for performing the task of voice separation in symbolic music representations. The models presented perform slightly better than or similar to existing state-of-the-art systems for voice separation. On two-voice and three-voice datasets, accuracy values higher than 85% are always achieved, and accuracy values above 90% are common. When the number of voices exceeds three, however, accuracy values tend to stagnate around 80%. Comparable results are reported for the existing systems. As in many MIR tasks, a difficult-to-surpass glass ceiling—an optimal performance that is lower than that of a human performing the same task—thus seems to have been reached (Sturm, 2014). The advantage of a machine learning approach (as opposed to a rule-based approach) as presented in this thesis, however, is that it is flexible and easily extensible. The following perspectives for future work may lead the way into passing the glass ceiling.

Both the classification approach MA1 and the regression approach MA2 may benefit, first, from the inclusion of different features, encoding, for example, more advanced instrumental (tablature dataset only), structural, harmonic, or melodic information. Additionally, in MA2, a more optimal solution may have to be found for the default features that must be included in the feature vector to ensure an uniform feature vector dimension. Another angle is to try automatic feature extraction, that is, to learn features using deep networks instead of designing them. Second, increasing the decision context further—that is, considering more than one previous or next note in each voice—is expected to improve model performance, especially where complex musical phenomena such as single-note unisons and voice crossing are concerned. An interesting question here is how to model this in terms of feature design. Third, it would be interesting to experiment also in MA1 with a larger window within which possible combinations of individual voice class decisions are evaluated. This decision window is larger by default in MA2; here, however, a further extension is more problematic, as the size of the state space when the decision must be made grows exponentially with

the length of the decision window. Approaches that explore the state space in a dynamic and adaptive way may thus be preferred over exhaustive search here. Fourth, with regard to the bidirectional model used in MA1, it may be worthwhile experimenting with additional passes through the data, where in each new pass the data is annotated with the polyphonic information generated in the most recent pass. Such a bootstrapping approach might progressively improve model performance. Fifth, the inclusion of melodic trajectory information in the models is expected to be beneficial to model performance, especially with respect to the handling of complex musical phenomena such as single-note unisons, voice crossing, and imitation. A suitable starting point would be to include trained melody prediction models as features. Such models could, but need not necessarily, be trained on the same dataset. The rate of correct identification of imitations and voice crossing—which often go hand in hand—is expected to increase further when (stored) knowledge of the themes and motifs used in a dataset is available to a model.

The probabilistic approach MA3, lastly, in its current implementation is fairly straightforward. Worthwhile experimenting with are, for example, hidden Markov models that use more appropriate or more extended vectorial representations of the observations, as well as (or possibly in combination with) higher-order hidden Markov models. Another possibility is to look into factorial hidden Markov models, where the observations are again the chords, represented as some feature vector, but where the state space consists of *multiple* chains, each of which represents a voice.

The work presented in this thesis shows how MIR research can provide a solution for a long-existing musicological problem. With the implementation of the different models for voice separation, first steps towards a system for automatic transcription of lute tablature into modern music notation have been achieved. Such a system is hoped to spark new research into lute music. From a larger perspective, it is hoped that this thesis will revive interest in research into voice separation in symbolic music representations. There are still many possible paths to pursue.

Bibliography

Allaire, G., and Cazeaux, I. (eds.). (1974). *Claudin de Sermisy: Opera omnia*, vol. IV. Corpus Mensurabilis Musicae, 52. [Rome]: American Institute of Musicology.

Baum, E. B., and Haussler, D. (1989). What size net gives valid generalization? *Neural Computation, 1*(1), 151–160.

Braun, H., Feulner, J., and Ullrich, V. (1991). Learning strategies for solving the problem of planning using backpropagation. In *Proceedings of the 4th International Conference on Neural Networks and Their Applications, Nîmes, France* (n.p.).

Bregman, A. S. (1990). *Auditory scene analysis: The perceptual organization of sound.* Cambridge, MA: MIT Press.

Bregman, A. S., and Campbell, J. (1971). Primary auditory stream segregation and perception of order in rapid sequences of tones. *Journal of Experimental Psychology, 89*(2), 244–249.

Brown, H. M. (1965). *Instrumental music printed before 1600: A bibliography.* Cambridge, MA: Harvard University Press.

Brown, H. M. (1973–1974). Embellishment in early sixteenth-century Italian intabulations. *Proceedings of the Royal Musical Association, 100,* 49–83.

Brown, H. M. (1976). Accidentals and ornamentation in sixteenth-century intabulations of Josquin's motets. In E. E. Lowinsky and B. J. Blackburn (eds.), *Josquin des Prez: Proceedings of the International Josquin Festival-Conference* (pp. 475–522). London: Oxford University Press.

Byrd, D., and Crawford, T. (2002). Problems of music information retrieval in the real world. *Information Processing and Management, 38*(2), 249–272.

Cambouropoulos, E. (2000). From MIDI to traditional musical notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music, Austin, TX, USA* (n.p.).

Cambouropoulos, E. (2008). Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception, 26*(1), 75–94.

Canguilhem, P. (2001). Fronimo *de Vincenzo Galilei.* Paris: Minerve.

Charnassé, H. (ed.). (1988). *Informatique et musique: Session musicologique de l'International Computer Music Conference.* Ivry-sur-Seine: ELMER-ATTO.

Charnassé, H., and Ducasse, H. (1971). De l'emploi de l'ordinateur pour la transcription des tablatures (notes sur un programme expérimental de transcription automatique). *Revue de Musicologie, 57*(2), 107–133.

Charnassé, H., and Ducasse, H. (eds.). (1973). *Informatique musicale: Journées d'étude 1973, textes des conférences E.R.A.T.T.O.* Paris: Centre de Documentation Sciences Humaines.

Charnassé, H., and Stepien, B. (1986). Automatic transcription of sixteenth century musical notations. *Computers and the Humanities, 20*(3), 179–190.

Charnassé, H., and Stepien, B. (1991a). Computer and musicology: Automatic transcription of 16th century music: German lute tablatures. In H. Best, E. Mochmann, and M. Thaller (eds.), *Computers in the humanities and the social sciences: Achievements of the 1980s, prospects for the 1990s* (pp. 75–83). Munich: Saur.

Charnassé, H., and Stepien, B. (1991b). ERATTO software for German lute tablatures. *Computing in Musicology, 7*, 60–62.

Charnassé, H., and Stepien, B. (1992). Automatic transcription of German lute tablatures: An artificial intelligence application. In A. Marsden and A. Pople (eds.), *Computer representations and models in music* (pp. 143–170). London: Academic Press.

Chew, E., and Wu, X. (2005). Separating voices in polyphonic music: A contig mapping approach. In U. K. Wiil (ed.), *Computer Music Modeling*

and Retrieval: Second international symposium, CMMR 2004 (pp. 1–20). Berlin: Springer.

Crawford, T. (1991). *TabCode* for lute repertories. *Computing in Musicology*, *7*, 57–59.

von Dadelsen, G. (ed.). (1970). *Johann Sebastian Bach: Inventionen und Sinfonien.* Neue Ausgabe sämtlicher Werke, V(3). Kassel: Bärenreiter.

Dalitz, C., and Crawford, T. (2013). From facsimile to content based retrieval: The Electronic Corpus of Lute Music. *Phoibos, 2013*(2), 167–185.

Dalitz, C., and Karsten, T. (2005). Using the Gamera framework for building a lute tablature recognition system. In *Proceedings of the 6th International Conference on Music Information Retrieval, London, UK* (pp. 478–481).

Dart, T., Morehen, J., and Rastall, R. (2001). Tablature. In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 24 (pp. 905–914). London: Macmillan.

Deutsch, D. (1975). Two-channel listening to musical scales. *Journal of the Acoustical Society of America, 57*(5), 1156–1160.

DeWitt, L. A., and Crowder, R. G. (1987). Tonal fusion of consonant musical intervals: The oomph in Stumpf. *Perception & Psychophysics, 41*(1), 73–84.

Dorfmüller, K. (1967). *Studien zur Lautenmusik in der ersten Hälfte des 16. Jahrhunderts.* Tutzing: Schneider.

Dorfmüller, K. (2001). Ochsenkun [Ochsenkhun], Sebastian. In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 18 (p. 312). London: Macmillan.

Downie, J. S. (2003). Music information retrieval. *Annual Review of Information Science and Technology, 37*, 295–340.

Drabkin, W. (2001). Part (ii). In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 19 (p. 164). London: Macmillan.

Elisseeff, A., and Paugam-Moisy, H. (1997). Size of multilayer networks for exact learning: Analytic approach. In M. C. Mozer, M. I. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems 9* (pp. 162–168). Cambridge, MA: MIT Press.

Fahlman, S. E. (1989). Faster-learning variations on back-propagation: An empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 38–51). San Mateo, CA: Morgan Kaufmann.

Finscher, L. (ed.). (1972). *Loyset Compère: Opera omnia*, vol. V. Corpus Mensurabilis Musicae, 15. [Rome]: American Institute of Musicology.

Forney, G. D., Jr. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268–278.

Frobenius, W., Cooke, P. R., Bithell, C., and Zemtsovsky, I. (2001). Polyphony. In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 20 (pp. 74–83). London: Macmillan.

Geering, A., and Altwegg, W. (eds.). (1962). *Ludwig Senfl: Deutsche Lieder II*. Sämtliche Werke, IV. Wolfenbüttel: Möseler.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*(1), 1–58.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *15*(1), 9–42.

Gjerdingen, R. O. (1994). Apparent motion in music? *Music Perception*, *11*(4), 335–370.

Göllner, M. L. (1984). On the process of lute intabulation in the sixteenth century. In H. Leuchtmann and R. Münster (eds.), *Ars iocundissima: Festschrift für Kurt Dorfmüller zum 60. Geburtstag* (pp. 83–96). Tutzing: Schneider.

Griffiths, J. (2002). The lute and the polyphonist. *Studi Musicali*, *31*(1), 89–108.

Grossberg, S., and Rudd, M. E. (1989). A neural architecture for visual motion perception: Group and element apparent motion. *Neural Networks*, *2*(6), 421–450.

Hong, C. M. (1984). *Sebastian Ochsenkun's Tabulaturbuch auff die Lauten (1558): Transcription and study*. PhD thesis, Michigan State University.

Honingh, A., Burgoyne, J. A., van Kranenburg, P., and Volk, A. (2014). *Strengthening interdisciplinarity in MIR: Four examples of using MIR tools for musicology* (technical report no. PP-2014-18). Amsterdam: Institute for Logic, Language and Computation.

Hörnel, D. (2004). ChordNet: Learning and producing voice leading with neural networks and dynamic programming. *Journal of New Music Research, 33*(4), 387–397.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251–257.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366.

Huang, S.-C., and Huang, Y.-F. (1991). Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks, 2*(1), 47–55.

Hudson, B. (ed.). (1995). *Josquin des Prez: Masses based on secular polyphonic songs 2.* The Collected Works of Josquin des Prez, 8. Utrecht: Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis.

Huron, D. (1989a). Voice denumerability in polyphonic music of homogeneous timbres. *Music Perception, 6*(4), 361–382.

Huron, D. (1989b). *Voice segregation in selected polyphonic keyboard works by Johann Sebastian Bach.* PhD thesis, University of Nottingham.

Huron, D. (1991). Tonal consonance versus tonal fusion in polyphonic sonorities. *Music Perception, 9*(2), 135–154.

Huron, D. (2001). Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception, 19*(1), 1–64.

Igel, C., and Hüsken, M. (2000). Improving the Rprop learning algorithm. In *Proceedings of the 2nd International Symposium on Neural Computation, Berlin, Germany* (pp. 115–121).

Igel, C., and Hüsken, M. (2003). Empirical evaluation of the improved Rprop learning algorithms. *Neurocomputing, 50*, 105–123.

Ishigaki, A., Matsubara, M., and Saito, H. (2011). Prioritized contig combining to segregate voices in polyphonic music. In *Proceedings of the 8th Sound and Music Computing Conference, Padua, Italy* (n.p.).

Jas, E. (ed.). (2008). *Josquin des Prez: Motets on texts from the Old Testament 4.* The Collected Works of Josquin des Prez, 17. Utrecht: Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis.

Jordanous, A. (2008). Voice separation in polyphonic music: A data-driven approach. In *Proceedings of the International Computer Music Conference, Belfast, Ireland* (n.p.).

Karydis, I., Nanopoulos, A., Papadopoulos, A., Cambouropoulos, E., and Manolopoulos, Y. (2007a). Horizontal and vertical integration/segregation in auditory streaming: A voice separation algorithm for symbolic musical data. In *Proceedings of the 4th Sound and Music Computing Conference, Lefkada, Greece* (pp. 299–306).

Karydis, I., Nanopoulos, A., Papadopoulos, A. N., and Cambouropoulos, E. (2007b). VISA: The voice integration/segregation algorithm. In *Proceedings of the 8th International Conference on Music Information Retrieval, Vienna, Austria* (pp. 445–448).

Kilian, J., and Hoos, H. H. (2002). Voice separation—A local optimization approach. In *Proceedings of the 3rd International Conference on Music Information Retrieval, Paris, France* (pp. 39–46).

Kirlin, P. B., and Utgoff, P. E. (2005). VoiSe: Learning to segregate voices in explicit and implicit polyphony. In *Proceedings of the 6th International Conference on Music Information Retrieval, London, UK* (pp. 552–557).

Lawrence, S., Giles, C. L., and Tsoi, A. C. (1996). *What size neural network gives optimal generalization? Convergence properties of backpropagation* (technical report no. UMIACS-TR-96-22 and CS-TR-3617). College Park, MD: Institute for Advanced Computer Studies.

Lewis, D., Crawford, T., and Gale, M. (2004). An Electronic Corpus of Lute Music (ECOLM): Technological challenges and musicological possibilities. In *Proceedings of the 1st Conference on Interdisciplinary Musicology, Graz, Austria* (n.p.).

Madsen, S. T., and Widmer, G. (2006). Separating voices in MIDI. In *Proceedings of the 7th International Conference on Music Information Retrieval, Victoria, BC, Canada* (pp. 57–60).

Marsden, A. (1992). Modelling the perception of musical voices: A case study in rule-based systems. In A. Marsden and A. Pople (eds.), *Computer*

representations and models in music (pp. 239–263). London: Academic Press.

McCabe, S. L., and Denham, M. J. (1997). A model of auditory streaming. *Journal of the Acoustical Society of America*, *101*(3), 1611–1621.

Merritt, A. T., and Lesure, F. (eds.). (1965–1971). *Clément Janequin: Chansons polyphoniques*, vols. I–VI. Monaco: L'Oiseau-Lyre.

Minamino, H. (1988). *Sixteenth-century lute treatises with emphasis on process and techniques of intabulation*. PhD thesis, University of Chicago.

Ness, A. J., and Kolczynski, C. A. (2001). Sources of lute music. In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 24 (pp. 39–63). London: Macmillan.

Neubarth, K., Bergeron, M., and Conklin, D. (2011). Associations between musicology and music information retrieval. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, Miami, FL, USA* (pp. 429–434).

van Noorden, L. P. A. S. (1975). *Temporal coherence in the perception of tone sequences*. PhD thesis, Technische Hogeschool Eindhoven.

Ochsenkun, S. (1558). *Tabulaturbuch auff die Lauten*. Heidelberg: Kholen.

Orio, N. (2006). Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, *1*(1), 1–90.

Perkins, L. L. (ed.). (2011). *Josquin des Prez: Motets on texts from the Old Testament 5*. The Collected Works of Josquin des Prez, 18. Utrecht: Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis.

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

Radke, H. (1963). Beiträge zur Erforschung der Lautentabulaturen des 16.–18. Jahrhunderts. *Die Musikforschung*, *16*(1), 34–51.

Rafailidis, D., Cambouropoulos, E., and Manolopoulos, Y. (2009). Musical voice integration/segregation: VISA revisited. In *Proceedings of the 6th Sound and Music Computing Conference, Porto, Portugal* (pp. 42–47).

Rhodes, C., and Lewis, D. (2006). An editor for lute tablature. In R. Kronland-Martinet, T. Voinier, and S. Ystad (eds.), *Computer Music Modeling and Retrieval: Third international symposium, CMMR 2005* (pp. 259–264). Berlin: Springer.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons—From backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, *16*(3), 265–278.

Riedmiller, M., and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA* (pp. 586–591).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Schiffmann, W., Joost, M., and Werner, R. (1993). Comparison of optimized backpropagation algorithms. In *Proceedings of the 1st European Symposium on Artificial Neural Networks, Brussels, Belgium* (pp. 97–104).

Seay, A. (ed.). (1978). *Constanzo Festa: Opera omnia*, vol. VIII. Corpus Mensurabilis Musicae, 25. [Rome]: American Institute of Musicology.

Sherr, R. (ed.). (2002). *Josquin des Prez: Motets on texts from the Old Testament 1*. The Collected Works of Josquin des Prez, 14. Utrecht: Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis.

Smith, D. A. (2002). *A history of the lute from antiquity to the Renaissance*. [Lexington, VA]: The Lute Society of America.

Student (1908). The probable error of a mean. *Biometrika*, *6*(1), 1–25.

Sturm, B. L. (2014). Making explicit the formalism underlying evaluation in music information retrieval research: A look at the MIREX automatic mood classification task. In M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad (eds.), *Sound, music, and motion: 10th international symposium, CMMR 2013* (pp. 89–104). Cham: Springer.

Szeto, W. M., and Wong, M. H. (2006). Stream segregation algorithm for pattern matching in polyphonic music databases. *Multimedia Tools and Applications*, *30*(1), 109–127.

Temperley, D. (2001). *The cognition of basic musical structures*. Cambridge, MA: MIT Press.

Temperley, D. (2009). A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research, 38*(1), 3–18.

Thibault, G. (1976). Instrumental transcriptions of Josquin's French chansons. In E. E. Lowinsky and B. J. Blackburn (eds.), *Josquin des Prez: Proceedings of the International Josquin Festival-Conference* (pp. 455–474). London: Oxford University Press.

de Valk, R., and Weyde, T. (2015). Bringing 'Musicque into the tableture': Machine-learning models for polyphonic transcription of 16th-century lute tablature. *Early Music, 43*(4), 563–576.

de Valk, R., Weyde, T., and Benetos, E. (2013). A machine learning approach to voice separation in lute tablature. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, Curitiba, Brazil* (pp. 555–560).

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2), 260–269.

Volk, A., Wiering, F., and van Kranenburg, P. (2011). Unfolding the potential of computational musicology. In *Proceedings of the 13th International Conference on Informatics and Semiotics in Organisations, Leeuwarden, the Netherlands* (pp. 137–144).

Wachsmann, K., McKinnon, J. W., Anderson, R., Harwood, I., Poulton, D., Van Edwards, D., Sayce, L., and Crawford, T. (2001). Lute. In S. Sadie (ed.), *The new Grove dictionary of music and musicians*, 2nd ed., vol. 15 (pp. 329–363). London: Macmillan.

Ward, J. (1952). The use of borrowed material in 16th-century instrumental music. *Journal of the American Musicological Society, 5*(2), 88–98.

Weyde, T. (2005). Modelling cognitive and analytic musical structures in the MUSITECH framework. In *Proceedings of the 5th Conference Understanding and Creating Music, Caserta, Italy* (n.p.).

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

Weyde, T., and de Valk, R. (2016). Chord- and note-based approaches to voice separation. In D. Meredith (ed.), *Computational music analysis* (pp. 137–154). Cham: Springer.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83.

Wolf, J. (ed.). (1907). *Heinrich Isaac: Weltliche Werke.* Denkmäler der Tonkunst in Österreich, 28. Vienna: Artaria.

Zheng, Z., Zha, H., Chen, K., and Sun, G. (2007). A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th Annual International ACM Special Interest Group on Information Retrieval Conference, Amsterdam, the Netherlands* (pp. 287–294).

Appendix

Table A.1 Hyperparameter optimisation, N model: hidden layer size and λ, accuracies for all combinations in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	86.63	89.89	91.44	92.20	92.31	91.78	92.42	92.29	92.25	92.37
$1/4 \cdot$ IL	86.70	89.77	91.22	92.39	92.86	92.96	92.94	92.95	93.01	93.00
$1/2 \cdot$ IL	86.32	89.41	91.23	92.46	92.88	93.17	93.19	93.23	93.28	93.19
$1 \cdot$ IL	86.07	89.30	91.20	92.49	92.97	93.22	93.39	93.36	**93.40**	93.32
$2 \cdot$ IL	86.24	89.31	91.24	92.38	93.15	93.29	93.38	93.38	93.19	93.35
$4 \cdot$ IL	86.24	89.47	91.17	92.44	93.00	93.17	93.32	93.24	93.29	93.34
$1/8 \cdot$ IL	85.85	89.13	90.58	91.12	91.24	90.78	91.27	91.12	90.65	91.22
$1/4 \cdot$ IL	85.82	89.22	90.28	91.08	91.79	91.38	91.69	91.62	91.48	91.73
$1/2 \cdot$ IL	85.63	88.88	90.51	91.34	91.54	91.91	**92.07**	91.95	91.69	91.87
$1 \cdot$ IL	85.29	88.39	90.54	91.21	92.04	91.45	91.69	92.04	91.94	91.50
$2 \cdot$ IL	85.23	88.87	90.79	91.14	91.59	91.54	91.82	91.70	91.90	91.97
$4 \cdot$ IL	85.19	88.48	90.35	90.99	91.49	91.33	91.93	91.67	91.61	91.59
$1/8 \cdot$ IL	55.51	68.76	76.84	74.49	76.75	77.17	77.72	77.29	76.68	76.06
$1/4 \cdot$ IL	54.70	69.13	75.85	76.89	76.60	76.69	77.78	75.74	77.27	76.28
$1/2 \cdot$ IL	56.57	68.71	73.77	76.64	77.80	78.68	76.35	78.15	78.36	78.89
$1 \cdot$ IL	52.78	68.40	75.72	77.74	**79.63**	78.25	79.18	78.46	79.42	78.74
$2 \cdot$ IL	60.21	68.28	75.47	76.92	78.04	78.49	77.86	78.66	78.63	77.89
$4 \cdot$ IL	54.66	69.19	74.69	77.77	78.55	78.00	79.39	78.12	78.00	78.67

Table A.2 Hyperparameter optimisation, N model: hidden layer size and λ, p-values for pairwise comparison of the highest accuracy value with all other values in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
$1/4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0195*	*0.0039*	*0.0039*	*0.0039*
$1/2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	0.0976	*0.0195*	0.0742	0.1640	0.0546
$1 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0078*	0.0976	0.9101	0.3593		0.4960
$2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0390*	0.4257	0.9101	1.0000	*0.0195*	0.6523
$4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0117*	0.0742	0.7343	0.1640	0.2031	0.3593
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0078*	*0.0039*	*0.0039*	*0.0117*
$1/4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	0.1289	*0.0117*	*0.0195*	0.3007	*0.0390*	0.2031
$1/2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0273*	0.0546	0.4257		0.4257	0.1640	0.5703
$1 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	0.6523	0.2031	0.6523	0.6523	0.6523	0.0976
$2 \cdot$ IL	*0.0039*	*0.0039*	*0.0078*	*0.0078*	0.0546	*0.0390*	*0.0195*	0.4257	0.2500	1.0000
$4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0117*	0.0546	0.8203	0.1640	0.0976	0.1640
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0273*	0.0742	0.0742	0.1640	0.4257	*0.0078*	0.0976
$1/4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0195*	0.4960	*0.0390*	0.2031	0.3593	0.3007	0.4257
$1/2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	0.0546	0.3007	0.7343	0.4960	0.1289	0.9101	1.0000
$1 \cdot$ IL	*0.0039*	*0.0039*	0.0742	0.1289			0.2031	1.0000	1.0000	1.0000	1.0000
$2 \cdot$ IL	*0.0039*	*0.0039*	0.0976	*0.0039*	0.2031	0.3593	0.3007	0.4960	0.3007	0.8203
$4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	0.0546	*0.0273*	0.1640	0.9101	0.0976	0.4257	0.2031

Table A.3 Hyperparameter optimisation, N model: hidden layer size and λ, accuracies for all combinations in training (top), test (middle), and application (bottom) mode. Bach dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	93.33	95.98	96.66	97.00	96.62	96.50	96.77	96.81	96.79	96.42
$1/4 \cdot$ IL	93.78	95.94	96.90	97.56	97.77	97.81	97.82	97.75	97.82	97.78
$1/2 \cdot$ IL	93.60	95.78	96.96	97.67	97.84	97.92	97.94	97.91	97.94	97.95
$1 \cdot$ IL	93.61	95.76	96.98	97.68	97.89	97.96	98.03	98.02	98.04	98.01
$2 \cdot$ IL	92.90	95.69	96.98	97.71	97.94	97.99	98.03	98.04	98.04	**98.06**
$4 \cdot$ IL	93.20	95.77	96.97	97.73	97.94	97.99	98.03	98.06	98.04	98.06
$1/8 \cdot$ IL	93.18	95.75	96.29	96.61	96.20	96.03	96.42	96.81	96.28	96.38
$1/4 \cdot$ IL	93.67	95.85	96.70	97.26	97.44	97.56	97.54	97.41	97.48	97.31
$1/2 \cdot$ IL	93.46	95.53	96.69	97.42	97.44	97.53	97.58	97.48	97.51	97.57
$1 \cdot$ IL	93.25	95.51	96.84	97.37	97.59	97.56	97.64	97.60	97.57	**97.70**
$2 \cdot$ IL	92.39	95.47	96.70	97.42	97.55	97.63	97.53	97.62	97.66	97.62
$4 \cdot$ IL	92.86	95.68	96.72	97.42	97.58	97.52	97.62	97.60	97.53	97.59
$1/8 \cdot$ IL	69.89	71.58	74.96	75.96	73.22	76.01	75.30	76.54	74.27	74.23
$1/4 \cdot$ IL	70.33	70.86	77.17	77.41	77.36	78.70	77.67	79.23	77.18	77.61
$1/2 \cdot$ IL	70.05	68.81	74.50	78.40	77.13	79.27	77.88	78.62	78.07	78.60
$1 \cdot$ IL	67.70	69.23	74.40	78.37	78.74	78.38	78.55	79.76	**80.70**	79.63
$2 \cdot$ IL	68.39	70.68	75.79	80.42	78.83	80.16	79.70	78.70	78.42	79.74
$4 \cdot$ IL	65.82	68.66	74.73	78.28	79.77	79.68	78.22	79.09	80.42	79.95

Table A.4 Hyperparameter optimisation, N model: hidden layer size and λ, p-values for pairwise comparison of the highest accuracy value with all other values in training (top), test (middle), and application (bottom) mode. Bach dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*
$1/4 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*
$1/2 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0003*	*0.0001*	*0.0001*	*0.0001*
$1 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0004*	0.1956	0.1133	0.2412	0.0728
$2 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0123*	0.2579	0.1818	0.4653	
$4 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0001*	*0.0028*	0.4180	0.8287	0.3524	0.8905
$1/8 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0005*	*0.0001*	*0.0001*	*0.0001*	*0.0009*	*0.0001*	*0.0001*
$1/4 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0002*	*0.0024*	0.2412	0.0728	*0.0053*	0.0798	*0.0001*
$1/2 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0159*	*0.0053*	0.1446	0.1956	*0.0028*	0.3954	*0.0159*
$1 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0004*	0.3954	0.1687	0.3124	0.4413	0.0545	
$2 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0289*	0.2412	*0.0493*	0.1564	0.8905	0.4899	0.2253
$4 \cdot$ IL	*0.0001*	*0.0001*	*0.0001*	*0.0039*	0.4413	0.1041	0.7982	0.6507	*0.0323*	0.5152
$1/8 \cdot$ IL	*0.0001*	*0.0003*	*0.0094*	*0.0180*	*0.0016*	0.0545	*0.0289*	0.0545	*0.0071*	*0.0033*
$1/4 \cdot$ IL	*0.0001*	*0.0003*	0.0602	0.3320	0.0874	0.4899	0.0798	0.6225	0.1564	*0.0323*
$1/2 \cdot$ IL	*0.0001*	*0.0001*	*0.0006*	0.1687	*0.0360*	0.3736	0.0602	0.2412	*0.0323*	0.2935
$1 \cdot$ IL	*0.0001*	*0.0001*	*0.0108*	0.2935	0.2935	*0.0493*	0.0545	0.8287		0.5152
$2 \cdot$ IL	*0.0001*	*0.0003*	*0.0014*	0.5948	0.2935	0.6225	0.6507	0.2935	0.0955	0.3736
$4 \cdot$ IL	*0.0001*	*0.0001*	*0.0014*	0.0798	0.9843	0.5412	0.2935	0.6507	1.0000	0.7680

Table A.5 Hyperparameter optimisation, N$'$ model: hidden layer size and λ, accuracies for all combinations in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	91.68	95.33	94.05	91.94	91.89	91.69	89.27	91.79	92.90	91.66
$1/4 \cdot$ IL	92.75	95.46	96.64	96.82	96.09	96.38	96.24	95.73	96.72	96.44
$1/2 \cdot$ IL	92.46	95.34	96.73	97.18	97.32	97.33	97.33	97.36	97.37	97.33
$1 \cdot$ IL	91.90	94.94	96.64	97.10	97.30	97.26	97.28	97.27	**97.38**	97.32
$2 \cdot$ IL	92.14	94.86	96.42	96.94	97.03	97.13	97.12	97.22	97.10	97.06
$4 \cdot$ IL	92.58	94.85	96.29	96.81	96.96	97.02	96.98	97.02	97.02	96.99
$1/8 \cdot$ IL	90.68	94.71	93.56	90.56	91.93	90.25	89.86	90.50	90.18	90.66
$1/4 \cdot$ IL	91.54	94.34	96.11	95.96	95.15	95.00	95.37	94.64	95.90	95.37
$1/2 \cdot$ IL	91.77	94.80	96.13	96.53	96.42	**96.68**	96.55	96.61	96.59	96.54
$1 \cdot$ IL	90.26	94.33	96.09	96.46	96.54	96.50	96.41	96.67	96.46	96.24
$2 \cdot$ IL	91.76	94.45	95.74	96.28	96.33	96.44	96.33	96.30	96.13	96.00
$4 \cdot$ IL	91.76	94.23	95.65	96.15	96.13	95.93	96.16	96.24	96.09	96.06
$1/8 \cdot$ IL	62.68	65.71	64.81	63.34	62.25	62.79	57.18	61.05	63.34	63.72
$1/4 \cdot$ IL	63.39	66.52	67.82	68.40	66.41	67.00	66.58	66.25	68.83	65.63
$1/2 \cdot$ IL	62.91	68.33	68.42	68.57	69.03	69.48	69.57	**70.47**	69.77	69.88
$1 \cdot$ IL	61.80	67.82	68.09	68.07	68.49	67.64	68.99	69.18	69.41	68.56
$2 \cdot$ IL	62.57	64.94	67.45	69.91	67.93	67.36	68.84	65.94	67.48	67.54
$4 \cdot$ IL	64.17	66.48	68.02	67.01	67.49	67.08	68.20	66.62	68.34	66.97

Table A.6 Hyperparameter optimisation, N′ model: hidden layer size and λ, p-values for pairwise comparison of the highest accuracy value with all other values in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
$1/4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
$1/2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	0.3007	0.3593	0.8203	0.6523	0.9101	0.3593
$1 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0195*	0.0742	*0.0390*	0.0976		0.3007
$2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0117*	*0.0078*	0.0546	*0.0039*	*0.0039*
$4 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0039*
$1/4 \cdot$ IL	*0.0039*	*0.0039*	*0.0117*	*0.0117*	*0.0039*	*0.0039*	*0.0078*	*0.0039*	*0.0039*	*0.0039*
$1/2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	0.1640	*0.0195*		0.2500	0.1640	0.2500	0.1640
$1 \cdot$ IL	*0.0039*	*0.0039*	*0.0078*	*0.0273*	0.2031	0.0742	0.1289	0.2500	0.0976	*0.0039*
$2 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	0.0976	0.0546	0.1640	*0.0195*	*0.0390*	*0.0078*	*0.0039*
$4 \cdot$ IL	*0.0039*	*0.0039*	*0.0078*	*0.0039*	*0.0078*	*0.0039*	*0.0039*	0.0390	*0.0117*	*0.0117*
$1/8 \cdot$ IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0078*	0.0546	*0.0078*	0.0546	*0.0039*	*0.0078*
$1/4 \cdot$ IL	*0.0039*	*0.0195*	0.1640	0.2500	0.0742	0.1289	0.0976	*0.0390*	0.1640	*0.0390*
$1/2 \cdot$ IL	*0.0039*	0.0742	0.4960	0.5703	0.5703	0.7343	0.4960		1.0000	0.5703
$1 \cdot$ IL	*0.0039*	0.1289	0.2031	0.2031	0.2500	0.1289	0.6523	0.3593	0.4257	0.1640
$2 \cdot$ IL	*0.0039*	*0.0078*	0.2500	0.7343	0.1640	*0.0273*	0.5703	*0.0390*	0.0976	0.1640
$4 \cdot$ IL	*0.0078*	*0.0078*	0.2500	*0.0390*	0.0546	*0.0195*	0.2031	0.4960	0.0976	0.0742

Table A.7 Hyperparameter optimisation, C model: hidden layer size and λ, accuracies for all combinations in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	61.50	70.88	78.38	82.30	83.79	**85.74**	83.01	84.45	83.37	84.82
$1/4 \cdot$ IL	62.33	73.31	78.48	82.36	84.02	83.32	82.05	84.19	85.41	85.39
$1/2 \cdot$ IL	64.02	71.76	78.38	81.50	82.79	84.62	83.01	85.04	83.75	83.89
$1 \cdot$ IL	64.41	72.63	76.56	81.30	82.70	83.09	82.34	84.19	84.18	83.05
$2 \cdot$ IL	64.85	71.62	77.63	88.19	81.35	92.03	92.76	88.06	92.38	90.83
$1/8 \cdot$ IL	61.98	67.33	76.14	81.08	82.20	84.32	83.94	81.48	80.43	82.68
$1/4 \cdot$ IL	62.76	72.08	78.40	82.32	80.85	82.74	81.90	83.45	**84.94**	83.50
$1/2 \cdot$ IL	63.55	72.05	76.83	80.92	82.42	82.50	81.91	82.91	83.44	82.11
$1 \cdot$ IL	64.94	68.40	76.72	78.01	80.60	81.08	82.18	81.86	79.72	81.66
$2 \cdot$ IL	63.08	71.24	75.23	91.80	83.56	92.48	88.90	88.72	91.58	93.49
$1/8 \cdot$ IL	63.28	67.62	70.46	72.98	74.26	73.88	72.94	75.10	74.35	74.97
$1/4 \cdot$ IL	64.65	68.66	70.77	70.72	**75.26**	73.10	75.09	72.91	74.41	73.22
$1/2 \cdot$ IL	65.90	67.98	70.04	71.91	72.91	71.89	73.71	71.85	73.11	72.85
$1 \cdot$ IL	65.22	67.39	68.65	67.60	71.80	70.32	71.70	71.32	67.76	68.56
$2 \cdot$ IL	64.36	66.00	61.71	42.32	59.39	34.58	33.96	48.86	36.43	30.20

Table A.8 Hyperparameter optimisation, C model: hidden layer size and λ, p-values for pairwise comparison of the highest accuracy value with all other values in training (top), test (middle), and application (bottom) mode. Tablature dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
1/8 · IL	*0.0039*	*0.0039*	*0.0078*	*0.0390*	0.0742		0.2500	0.1640	0.0976	0.2031
1/4 · IL	*0.0039*	*0.0039*	*0.0039*	*0.0117*	0.2500	0.0976	*0.0039*	*0.0390*	0.6523	0.4257
1/2 · IL	*0.0039*	*0.0039*	*0.0039*	*0.0039*	*0.0195*	0.2031	*0.0078*	0.4257	0.2500	0.1289
1 · IL	*0.0039*	*0.0039*	*0.0039*	*0.0078*	0.0546	*0.0390*	0.0273	0.1289	0.3007	*0.0390*
2 · IL	*0.0039*	*0.0039*	0.0742	0.5703	0.0546	0.0546	*0.0117*	0.2500	*0.0390*	*0.0195*
1/8 · IL	*0.0078*	*0.0195*	0.0546	*0.0390*	0.4960	0.9101	0.1289	1.0000	0.3593	0.4257
1/4 · IL	*0.0078*	*0.0039*	*0.0273*	0.0742	0.0976	0.4257	*0.0390*	0.0546		0.9101
1/2 · IL	*0.0078*	*0.0078*	*0.0078*	*0.0117*	0.1289	0.0546	*0.0117*	0.4257	0.5703	0.0546
1 · IL	*0.0039*	*0.0039*	*0.0195*	0.0742	*0.0390*	*0.0039*	*0.0390*	0.0976	*0.0117*	*0.0273*
2 · IL	*0.0078*	*0.0039*	0.0976	0.2500	0.6523	0.0742	*0.0078*	0.3007	0.0546	*0.0273*
1/8 · IL	*0.0195*	0.0546	0.3007	0.4257	1.0000	0.9101	0.2500	0.7343	0.9101	0.6523
1/4 · IL	*0.0195*	*0.0078*	0.3007	0.1289		0.5703	0.4257	0.1289	0.3007	0.3007
1/2 · IL	*0.0390*	*0.0273*	0.0546	*0.0195*	0.4257	*0.0390*	0.3007	0.2500	1.0000	0.1289
1 · IL	*0.0078*	*0.0117*	*0.0039*	0.0546	0.2031	*0.0039*	*0.0390*	0.0546	*0.0117*	0.0742
2 · IL	*0.0078*	*0.0117*	*0.0117*	*0.0195*	*0.0195*	*0.0078*	*0.0039*	*0.0195*	*0.0039*	*0.0039*

Table A.9 Hyperparameter optimisation, C model: hidden layer size and λ, accuracies for all combinations in training (top), test (middle), and application (bottom) mode. Bach dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	95.90	96.69	96.78	96.86	96.82	96.82	96.92	96.96	96.91	96.87
$1/4 \cdot$ IL	95.64	96.66	96.73	96.67	96.76	96.71	96.65	96.95	96.66	96.80
$1/2 \cdot$ IL	95.49	96.29	96.45	96.48	96.56	96.64	96.54	96.54	96.63	96.60
$1 \cdot$ IL	95.37	96.34	96.33	96.35	96.48	96.51	96.38	96.53	96.46	96.49
$2 \cdot$ IL	95.01	96.00	96.28	96.67	**97.10**	96.21	96.44	96.76	96.78	96.74
$1/8 \cdot$ IL	95.84	96.63	96.52	96.72	96.72	96.81	96.83	96.76	96.63	96.70
$1/4 \cdot$ IL	95.59	96.19	96.54	96.66	96.67	96.65	96.41	96.86	96.55	96.69
$1/2 \cdot$ IL	95.23	96.52	96.23	96.24	96.18	96.44	96.27	96.52	96.37	96.41
$1 \cdot$ IL	95.18	96.07	96.29	96.25	96.16	96.24	96.23	96.40	96.27	96.14
$2 \cdot$ IL	94.56	96.06	96.16	96.47	96.91	95.88	96.15	**97.22**	96.20	96.37
$1/8 \cdot$ IL	73.13	76.88	75.31	75.81	75.37	74.65	78.27	77.33	76.31	77.16
$1/4 \cdot$ IL	74.12	74.37	75.93	76.20	79.21	76.22	75.38	**79.56**	77.39	76.71
$1/2 \cdot$ IL	73.89	77.08	77.16	76.87	75.36	78.61	75.44	79.48	76.41	76.99
$1 \cdot$ IL	73.79	74.87	73.44	76.15	77.47	77.60	77.38	78.22	75.55	78.57
$2 \cdot$ IL	74.81	72.23	75.06	63.79	65.62	79.47	68.21	60.22	67.19	68.48

Table A.10 Hyperparameter optimisation, C model: hidden layer size and λ, p-values for pairwise comparison of the highest accuracy value with all other values in training (top), test (middle), and application (bottom) mode. Bach dataset, four-voice pieces. HL = hidden layer size, IL = input layer size.

HL	λ									
	0.1	0.03	0.01	0.003	0.001	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	0.0
$1/8 \cdot$ IL	*0.0033*	0.4899	0.7380	0.9529	0.7085	0.8595	0.6507	0.9529	0.8595	0.7085
$1/4 \cdot$ IL	*0.0001*	0.4899	0.4653	0.3736	0.7380	0.4899	0.4653	1.0000	0.2935	0.9843
$1/2 \cdot$ IL	*0.0001*	*0.0493*	0.1446	0.1231	0.2101	0.3320	0.1231	0.1818	0.4413	0.3954
$1 \cdot$ IL	*0.0001*	0.0602	0.0602	*0.0204*	0.0798	0.1231	0.0874	0.2253	0.1818	0.2935
$2 \cdot$ IL	*0.0001*	*0.0082*	0.0545	0.3954		*0.0323*	*0.0493*	0.9217	0.7085	0.3524
$1/8 \cdot$ IL	*0.0033*	0.2253	0.2753	0.6794	0.6794	0.8595	0.9843	0.5948	0.4653	0.5412
$1/4 \cdot$ IL	*0.0053*	0.3124	0.3736	0.2412	0.5152	0.6794	0.0728	0.8595	0.2412	0.4180
$1/2 \cdot$ IL	*0.0014*	0.4180	*0.0401*	0.0545	0.1133	0.1133	0.1133	0.2253	0.2412	0.0545
$1 \cdot$ IL	*0.0004*	0.0662	0.0728	*0.0123*	*0.0108*	0.0602	*0.0445*	0.3736	0.1133	*0.0071*
$2 \cdot$ IL	*0.0001*	*0.0108*	0.0798	0.1564	0.9217	*0.0020*	0.1818		*0.0493*	0.2253
$1/8 \cdot$ IL	*0.0053*	0.0602	*0.0258*	0.1818	0.1446	*0.0071*	0.7380	0.2579	*0.0289*	0.1041
$1/4 \cdot$ IL	*0.0016*	*0.0401*	0.2753	0.0545	0.7380	0.1956	0.0662		0.5152	0.1446
$1/2 \cdot$ IL	*0.0006*	0.8595	0.5412	0.1687	*0.0159*	0.4180	0.1336	0.7085	0.2579	0.1041
$1 \cdot$ IL	*0.0204*	0.0955	*0.0360*	*0.0094*	0.1133	0.5152	0.3524	0.3736	0.1041	0.2253
$2 \cdot$ IL	*0.0009*	*0.0053*	0.2935	*0.0123*	*0.0159*	0.6507	*0.0258*	*0.0053*	*0.0053*	*0.0002*