# Automatic Roman Numeral Analysis in Symbolic Music Representations

Néstor Nápoles López

Music Technology Area

Department of Music Research

Schulich School of Music

McGill University, Montréal

December 2022

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

# Contents

# Abstract

One of the most common ways to analyze a piece of tonal music is through Roman numeral analysis. This requires the inspection of several attributes related to chords and keys. Chords can be inspected in terms of their properties: root, quality, inversion, and function. Keys can be inspected in terms of their temporal scope as modulations or tonicizations. Each of these attributes (or tasks) of Roman numeral analysis can be modeled in isolation. However, recent research has found that analyzing several tonal tasks simultaneously leads to more robust Music Information Retrieval models. This has motivated the research of multitask neural networks for Roman numeral analysis. In this dissertation, I extend this line of research by developing a new Roman numeral analysis machine learning model. The model is based on a new convolutional recurrent neural network, which is trained with a large dataset of publicly available Roman numeral analysis annotations. The model is assisted by a new data-augmentation technique and multitask learning layout to learn the relevant attributes of chords and keys. Combining these ideas, the resulting model seems to achieve an improved performance over rare chords compared to previous automatic Roman numeral analysis methods. Among other applications, this model will facilitate advanced searching in music collections. For example, searching by chord progressions or by modulation trajectories.

# Résumé

Pour analyser un morceau de musique tonale, on en étudie généralement l'harmonie. Cela consiste en l'analyse de divers attributs liés aux accords et aux tonalités. Un accord est caractérisé par sa fondamentale, sa nature, son renversement et sa fonction. La tonalité est caractérisée par son étendue temporelle qui la définit comme une modulation ou une tonicisation. Chacun de ces attributs de l'harmonie peut être modélisé séparément. Cependant, des recherches récentes montrent que l'analyse simultanée de plusieurs tâches tonales produit des modèles d'extraction d'informations musicales plus robustes, ce qui incite à l'étude de réseaux neuronaux multitâches pour l'analyse harmonique. Dans ce mémoire, j'approfondis cet axe de recherche en développant un nouveau modèle d'apprentissage automatique pour l'analyse harmonique. Le modèle est basé sur un nouveau réseau neuronal convolutif récurrent, entraîné sur un grand ensemble de données d'annotations d'analyse harmonique disponibles publiquement. Le modèle est assisté par une nouvelle technique d'augmentation des données ainsi qu'une configuration d'apprentissage multitâches inédite pour l'apprentissage des attributs pertinents des accords et des tonalités. En combinant ces idées, le modèle résultant semble obtenir de meilleures performances sur la caractérisation des accords rares par rapport aux méthodes précédentes d'analyse harmonique automatique. Entre autres applications, ce modèle facilitera la recherche plus spécifique dans les corpus musicaux, par progressions harmoniques ou par parcours tonal, par exemple.

# Acknowledgements

I will first and foremost thank my PhD advisor, Ichiro Fujinaga. For the most part of my PhD, Ichiro scheduled weekly meetings with me that were of the utmost help. Together, we walked through machine-learning prototypes, presentations, code, music theories, and papers. Ichiro and Masako also made me feel supported since I set foot in Montréal,[1] making it a pleasant experience for me and my partner to live and work in the city all these years. Thank you.

Secondly, I would like to thank my family, who have supported me all along: my spouse and best friend, Kinga; my parents, Martha and Eduardo; my siblings, Grecia, Eric, and sister-in-law Fabiola; and my parents-in-law, Grażyna and Mirosław.

After my advisor, my co-authors contributed ideas and feedback related to the research presented here. Thanks to all of you: Gabriel Vigliensoni, Claire Arthur, Laurent Feisthauer, Florence Levé, Mark Gotham, and the **Computational Tonal Studies (CTS)** group.[2]

Thanks to all the members of the **Distributed Digital Music Archives and Libraries (DDMAL)** and **Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)** communities, but particularly, thanks to Gabriel Vigliensoni, Martha Thomae, Matan Gover, Yulia Draginda, Yuval Adler, Alex Daigle, Junhao Wang, Emily Hopkins, Yaolong Ju, Timothy de Reuse, Dylan Hillerbrand, Sevag Hanssian, Sylvain Margot, Jacob Sanz-Robinson, Erica Huynh, Iza Korsmit, and Ké Zhang.

1. To be precise, since I set foot in Canada for the first time in my life, at the Trudeau airport in Montreal.
2. Jacob deGroot-Maggetti, Laurent Feisthauer, Ichiro Fujinaga, Sam Howes, Yaolong Ju, Suzuka Kokubu, Sylvain Margot, Tim de Reuse, and Finn Upham.

Thanks to McGill professors Julie Cumming, Peter Schubert, William Caplin, Jon Wild, Philippe Depalle, Marcelo Wanderley, Gary Scavone, and Stephen McAdams from whom I learned different aspects of music and technology that contributed explicitly or implicitly to the work presented in this dissertation.

Thanks to Joe Plazak, Edith Gosselin Picard, Rob Gonsalves, and everyone in the Avid, Sibelius, and RAD Lab families, who have welcomed me as one of their own, and kindly mentored me in the art of commercial-grade music software development and applied research.

Thanks to other members of the global music information retrieval and computational musicology communities, particularly, Craig Sapp, Helena Cuesta, Fabian Moss, Emilia Parada-Cabaleiro, Michael Scott Cuthbert, Jinny Park, Jacob Tyler Walls, Reinier de Valk, Rebecca Salganik, Johannes Hentschel, and others with whom I shared code, papers, and friendship.

The research in this dissertation benefitted from the generous support of different institutions. The **Fonds de recherche — Société et culture (FRQSC)**, from which I obtained a doctoral scholarship from 2019–2022 (dossier number 271479); **CIRMMT**, which awarded me with travel funding and a student project award; the **Single Interface for Music Score Search and Analysis (SIMSSA)** project, which provided a generous stipend during the first two years of my PhD (2017–2018) and a role as a casual research assistant for the remainder of it; the Schulich School of Music, which provided additional travel and conference registration awards; the Digital Research Alliance of Canada, which facilitated computing resources to run experiments; and Mitacs, which funded part of an internship project at Avid Technology. Thank you to all of these wonderful organizations and their commitment to music research.

Special thanks to Laurent Feisthauer for translating the English abstract of this dissertation into French, and to Geneviève Gates-Panneton for proofreading the final version of it.

Thanks to people who had an early influence in me and set me in my current path: Luis Alberto Casillas Santillán, Abelardo Gutiérrez, Juan José López Sandoval, Nancy Arana Daniel, Sergio Bolaños, Leo Hendrik Reyes, and Juan Rodrigo Pérez Cárdenas. Lastly, my friends, for always being there: Jhonnatan Razo, Dulce Alcalá, and Elí Lezama. ¡Gracias, amigos!

# Contributions to Original Knowledge

This dissertation is a multidisciplinary research effort at the intersection of music theory and **Music Information Retrieval (MIR)**.

**Chapter 2** This chapter contributes a historical review of the **Roman Numeral Analysis (RNA)** notation throughout multiple harmony books. This is a contribution to the existing knowledge of this subject because some of these relationships have not been explored before. For example, the adoption of a certain **RNA** symbol to denote a specific musical chord.

**Chapter 3** This chapter contributes a survey of **MIR** approaches for **Automatic Roman Numeral Analysis (ARNA)**, **Global-Key Estimation (GKE)**, **Local-Key Estimation (LKE)**, and pitch spelling.

**Chapter 4** This chapter describes the, to the best of my knowledge, largest publicly available **ARNA** dataset used for supervised learning. The dataset is aggregated from publicly available **RNA** annotations, and presents a short survey of its underlying collections. This chapter also contributes the description of a new data-augmentation technique proposed in this dissertation.

**Chapter 5** This chapter describes the main **ARNA** machine learning model contributed in this dissertation. The chapter also introduces an original **Multitask Learning (MTL)** configuration of tonal tasks, which is distinct to existing **ARNA** approaches.

**Chapter 6** This chapter contributes the experimental evaluation of the machine learning model proposed in the dissertation. The evaluation considers ablation studies to explore the underlying components of the proposed **Convolutional Recurrent Neural Network (CRNN)** model. Furthermore, it presents an original experiment illustrating the effects of two data-augmentation techniques. Lastly, it compares five **ARNA** models using the same evaluation framework. Both the number of methods compared (5) and the common evaluation framework are a contribution to existing research.

# Contributions of Authors

**Chapter 1** All contributions in this chapter are by me.[3]

**Chapter 2** All contributions in this chapter are by me. However, I benefitted from a fruitful discussion in *SMTDiscuss*,[4] which lead to new references and ideas. Particularly, thanks to Nicolas Meeùs and Phil Duker for participating in this conversation.

**Chapter 3** Section 3.3.1 takes several ideas and findings from a published conference paper (Nápoles López et al. 2020). Particularly, the reviews of **GKE** and **LKE** models was conducted by Laurent Feisthauer and myself, in equal contribution. The review of **GKE** had minor contributions by Florence Levé.

**Chapter 4** During the course of this research, Mark Gotham and myself, independently (and in collaboration, during the writing of a co-authored conference paper: Nápoles López, Gotham, and Fujinaga 2021) contributed conversions, revisions, and corrections to the listed Roman numeral analysis datasets. Workflows for data "hygiene" and correcting mistranscriptions were also independently developed by both. The data-curation workflows described in Section 4.1, however, are the ones I developed myself. The data-augmentation technique introduced in Section 4.5.1, is a contribution of mine. However, co-author Mark Gotham first proposed the idea of artificial texturization for this approach. The patterns chosen for texturization and the software implementation of those patterns were contributed by me.

3. Néstor Nápoles López.
4. https://discuss.societymusictheory.org/discussion/553/history-of-roman-numeral-analysis

**Chapter 5** This chapter is based on (and extends) a peer-reviewed conference paper (Nápoles López, Gotham, and Fujinaga 2021). The proposed neural network architecture is my contribution.

**Chapter 6** All the experiments reported in this chapter were conducted by me. Likewise for all the tables and comparisons.

**Chapter 7** All contributions in this chapter are by me.

**Appendix A** The methods described in Appendix A are a contribution of mine.

# List of Figures

# List of Tables

# Glossary

****harm** A syntax for Roman numeral analysis from the family of *Humdrum* representations. 39, 40, 41, 42, 43, 46, 47, 76, 91, 98

****kern** A syntax for musical notation from the family of *Humdrum* representations. 40, 45, 46, 47, 49, 76, 89, 98, 111, 154, 156, 159

****text** A syntax for lyrics from the family of *Humdrum* representations. 46

**Alto35** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.3 for further details. xxv, xxxiv, 124, 125, 127, 128, 139, 146, 153, 183

**AugmentedNet** An Automatic Roman Numeral Analysis model, originally introduced in Nápoles López, Gotham, and Fujinaga (2021), and extended in this dissertation. The main model described throughout the dissertation. xxvii, 14, 81, 106, 110, 114, 132, 153, 157, 160, 161, 163, 165, 166, 167, 168, 169, 172, 173, 179, 182, 183, 188, 197

**augmented sixth** A family of chords in Western tonal music characterized by an "augmented sixth" interval, which comprises at least three types of chords: the Italian (**It**), French (**Fr**$^{7}$), and German (**Ger**$^{7}$) augmented sixth chords. xx, 5, 26, 29, 30, 157, 161, 163, 168, 172, 173, 187, 196

**Bass35** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.1 for further details. xxiv, xxxiv, 124, 125, 126, 127, 128, 132, 138, 146, 153, 183

**cadential six-four** The *cadential six-four* is a tonic triad in second inversion. When the chord is used in a cadential context, it receives this special name. The "six-four" refers to the Arabic numerals written next to the Roman numeral. This chord is often a source of disagreement in music theory, because it lies at the boundary of verticality (a tonic chord root, arranged in second inversion) and horizontality (a dominant chord with a double suspension) in harmonic analysis. See Section 2.3 for a longer discussion of this chord in Roman numeral analysis. 25, 26, 27, 38, 86, 187

**closed-position form** In the context of chord realizations, a closed-position chord is an arrangement of the chord, usually in four parts, such that the three upper voices span a maximum interval of an octave. xxiv, 125, 127, 128, 177

**common-practice period** In Western art music, this refers to the period of music roughly between 1650–1900, which is characterized by the use of the tonal system. 85, 185

**dot product** Given two equal-length sequences of numbers, the dot product is given by $a\dot{b} = \sum_{i=1}^{n} a_i b_i$. 60

**HarmonicRhythm7** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.9 for further details. 116, 124, 130, 131, 132, 139, 146, 153, 177, 179, 181, 183

**key profile** A key profile is a pitch-class distribution used to correlate pitch histograms with a certain key or scale. They were arguably introduced for the first time in Krumhansl and Shepard (1979) and Krumhansl and Kessler (1982), and applied on global-key estimation problems since Krumhansl (1990). 4, 59, 60, 61, 62, 64, 66, 67, 68, 69, 70

**line of fifths** A line of fifths is an extension of the more commonly known "circle of fifths". A line of fifths occurs when enharmonic nonequivalence is assumed in the circle of fifths.

In the circle of fifths, the pitches in the circle repeat because it is assumed that two enharmonic pitches have the same location in the circle (e.g., G♭ and F♯). In the line of fifths, however, the "sharper" and "flatter" pitches are, the further away from the origin of the line of fifths they are. See Moss, Neuwirth, and Rohrmeier (2022) and Temperley (2000) for an extensive discussion on the topic. xxix, 73, 87, 89, 92, 97, 100, 101, 198, 204

**LocalKey38** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.5 for further details. 124, 128, 129, 130, 132, 145, 147, 153, 181, 183

**LowestNote19** One of the three input representations in the neural network proposed in this dissertation. See Section 5.1.4.1 for further details on this representation. xxiv, xxv, 116, 117, 119, 120, 121, 122, 125, 126, 136, 137, 138, 139, 140, 141, 146, 183

**Melisma** The *Melisma Music Analyzer (version 1)* is a suite of programs for music analysis. One of the analytical features of the suite is automatic Roman numeral analysis. After the contributions in Sapp (2009), the *Melisma* Roman numeral analysis module probably became the first end-to-end Roman numeral analysis system. See Nápoles López (2017) for a discussion of the Roman numeral analysis component of *Melisma*, and see Temperley (2004) for the underlying research behind the algorithms of the system. xxvi, 65, 73, 76, 153, 154, 156, 157, 159, 160, 161, 162, 163, 168, 169, 170

**MusicXML** A symbolic music format based on **Extensible Markup Language (XML)**. 45, 49, 50, 71, 82, 84, 88, 95, 111, 154, 156, 159, 160, 181, 182, 183

**NaTEM** The Numerator and Tonicization Estimation Method (NaTEM) is an algorithm to retrieve a Roman numeral numerator $\eta$ and tonicization $\tau$ when only a **pcset** $\rho$ and key $\kappa$ are known. See Section A.7 for more details on the method. xxix, 133, 134, 157, 158, 159, 161, 163, 170, 200, 201, 202

**Neapolitan** A special type of chord in Western tonal music, analogous to a ♭**II** Roman numeral. xix, xx, xxi, 3, 4, 5, 26, 29, 30, 32, 37, 38, 39, 94, 96, 157, 161, 163, 168, 172, 173, 196

**nonchord tone** A nonchord tone is an ornamental note that is not a member of the tones of a given chord, but that sounds in combination or proximity with them for aesthetic motivations. 6, 76, 83, 88, 109, 184, 185

**Notes19** One of the three input representations in the neural network proposed in this dissertation. See Section 5.1.4.2 for further details on this representation. xxiv, xxv, 117, 118, 119, 120, 121, 122, 125, 136, 137, 138, 139, 140, 141, 146, 183

**Numerator31** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.8 for further details. 124, 130, 132, 139, 146, 147, 153, 177, 183

**Onsets14** One of the three input representations in the neural network proposed in this dissertation. See Section 5.1.4.3 for further details on this representation. xxiv, xxv, 116, 117, 119, 120, 121, 122, 131, 136, 137, 139, 140, 146, 177, 178, 181, 183

**overfit** Overfitting is a phenomenon where a machine learning model excessively fits to the training data, compromising its capability to generalize on new—distinct—data. It is generally considered an undesirable effect when training a machine learning model. 103

**pcset** A pitch-class set. More precisely, a subset $\rho$ of the 12 "pitch classes" in the Western chromatic scale $\rho \subset [0, 11]$. In the context of this dissertation, a pitch-class set always refers to a set $|\rho| = 3$ or $|\rho| = 4$ that contains the pitch classes of a triad or seventh chord. See Section A.6 for further discussion on pitch-class sets. xxix, xxxii, 77, 82, 91, 96, 124, 130, 132, 133, 157, 158, 159, 160, 161, 168, 185, 195, 196, 197, 199, 200, 201, 202, 204

**PitchClassSet121** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.7 for further details. 124, 130, 139, 146, 153, 168, 177, 179, 183

**realize** In the context of chords, a chord realization is an arrangement of a chord label into notes. For example, if a chord is indicated as **F**:**V**$^{7}$, one realization consists of the tuple of notes (C4, E4, G4, B♭4), whereas a different realization consists of the notes (C4, G4, E5, B♭5). xxiii, 105, 106

**RomanText** A digital standard for Roman numeral analysis introduced in Gotham, Tymoczko, and Cuthbert (2019). xxix, 40, 41, 42, 82, 84, 100, 101, 106, 155, 156, 158, 161, 173, 181, 183

**SATB35** A short name for the four classification tasks: **Soprano35**, **Alto35**, **Tenor35**, and **Bass35**. xxiv, 125, 126, 130, 133, 139, 146, 177, 179

**Soprano35** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.4 for further details. xxv, xxxiv, 124, 125, 128, 129, 139, 146, 153, 183

**spine** In the context of the *Humdrum* family of formats, a spine is a column of data. 93

**Tenor35** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.2 for further details. xxiv, xxxiv, 124, 125, 127, 139, 146, 153, 183

**Tonicization38** One of the nine classification tasks in the neural network proposed in this dissertation. See Section 5.4.6 for further details. 124, 128, 129, 130, 132, 145, 147, 153, 181, 183

**vanishing gradients** Vanishing and exploding gradients are undesirable effects during the training of an artificial neural network. They consist of either negligible (vanishing) or

very large (exploding) updates to the trainable parameters. In the first instance, it leads to no learning, in the second instance, it leads to numerical instability. This effect was noticeable in Recurrent Neural Networks, and it made them difficult to train. See Bengio, Simard, and Frasconi (1994) for further information on this topic. 56

# Abbreviations

♩ Quarter note. xxiv, 86, 112, 113

𝅘𝅥𝅰 Thirty-second note. xxiv, 112, 113, 122, 123, 142, 145

𝅘𝅥𝅯 Sixteenth note. 88, 112

𝐂 Common time. 113

♯ A “sharp” accidental. 114, 115

♭ A “flat” accidental. 114, 115

𝄪 A “double-sharp” accidental. 114, 115

𝄫 A “double-flat” accidental. 114, 115

𝅝 Whole note. 122, 142, 145

𝅗𝅥. Dotted half note. 181

♩. Dotted quarter note. 181

**12-TET** Twelve-Tone Equal Temperament. 115

**ABC** Annotated Beethoven Corpus. xxii, 41, 42, 85, 86, 87, 95, 99, 100, 101, 148, 149, 150, 151, 152, 153

**ACR** Automatic Chord Recognition. 4, 45, 71, 77, 105

**ANN** Artificial Neural Network. 50, 51, 62

**API** Application Programming Interface. 184

**ARNA** Automatic Roman Numeral Analysis. xv, xvi, xxvii, 4, 5, 6, 8, 9, 10, 12, 13, 14, 44, 45, 46, 47, 49, 50, 52, 57, 58, 70, 71, 72, 74, 76, 81, 88, 91, 105, 110, 111, 115, 116, 126, 129, 131, 132, 133, 135, 152, 153, 154, 155, 156, 159, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 187, 188, 189

**BLSTM** Bidirectional Long Short-Term Memory. 57

**BPS** Beethoven Piano Sonatas. xxii, 77, 85, 88, 89, 90, 104, 148, 149, 150, 151, 152, 153

**BPTT** Backpropagation Through Time. 54

**CBLSTM** Convolutional Bidirectional Long Short-Term Memory. 55

**CIRMMT** Centre for Interdisciplinary Research in Music Media and Technology. xiii, xiv

**CNN** Convolutional Neural Network. 52, 53, 54, 66, 77, 121, 122, 143

**CRNN** Convolutional Recurrent Neural Network. xvi, xxv, 12, 13, 14, 55, 110, 120, 124, 125, 130, 131, 132, 136, 139, 141, 144, 153, 177

**CSV** Comma-Separated Values. 88

**CTS** Computational Tonal Studies. xiii

**DAW** Digital Audio Workstation. 49

**DCML** Digital and Cognitive Musicology Lab. 41, 42, 86, 100, 156

**DDMAL** Distributed Digital Music Archives and Libraries. xiii

**FFT** Fast Fourier Transform. 63

**FRQSC** Fonds de recherche — Société et culture. xiv

**GKE** Global-Key Estimation. xv, xvii, 58, 59, 61, 63, 66, 67, 68, 69, 70, 105

**GPU** Graphics Processing Unit. 159

**GRU** Gated Recurrent Unit. 56, 57, 77, 78, 123, 124, 136, 144

**HaydnSun** Haydn "Sun" String Quartets, Op. 20. xxii, xxvii, 85, 89, 90, 91, 92, 100, 148, 149, 150, 151, 152, 153, 171, 181

**HDF5** Hierarchical Data Format (v5). 182

**HLSD** Hooktheory Lead Sheet Dataset. 85

**HMM** Hidden Markov Model. 4, 61, 62, 63, 64, 65, 68, 69, 70, 77

**HPCP** Harmonic Pitch Class Profile. 62

**ISMIR** International Society for Music Information Retrieval. 178, 179

**KMT** Key Modulations and Tonicizations. xxii, xxix, 85, 87, 93, 94, 95, 96, 101, 102, 148, 149, 150, 151, 152, 153, 155, 158, 182

**KNN** K-Nearest Neighbours. 65

**LISP** LISt Processor. 75

**LKE** Local-Key Estimation. xv, xvii, xix, 4, 5, 58, 61, 66, 67, 68, 69, 70, 74

**LSTM** Long Short-Term Memory. 55, 56, 57, 77

**MEI** Music Encoding Initiative. 45, 46, 47, 48, 49, 111

**MIDI** Musical Instrument Digital Interface. 5, 45, 46, 48, 49, 65, 71, 72, 73, 74, 88, 111

**MIR** Music Information Retrieval. xv, 4, 8, 9, 14, 18, 44, 45, 46, 53, 55, 57, 58, 67, 69, 70, 71, 74, 75, 150, 174, 186, 191

**MIREX** Music Information Retrieval Evaluation eXchange. xxviii, 63, 64, 65, 66, 67

**MLOps** Machine Learning Operations. 13

**MLP** Multilayer Perceptron. 52

**MNIST** Modified National Institute of Standards and Technology. xix, 6, 7

**MPC** Music Perception and Cognition. 186

**MPS** Mozart Piano Sonatas. xxii, xxiii, 42, 85, 95, 96, 97, 100, 148, 149, 150, 151, 152, 153, 155, 181

**MTG** Music Technology Group. 66

**MTL** Multitask Learning. xv, xxv, 5, 13, 14, 77, 78, 115, 124, 125, 130, 131, 132, 136, 139, 156, 160, 175, 176

**NADE** Neural Autoregressive Density Estimator. 78, 131, 155

**OMR** Optical Music Recognition. 45, 53, 55

**ReLU** Rectified Linear Unit. 143, 144

**RNA** Roman Numeral Analysis. xv, xix, xx, xxii, xxiii, xxviii, xxix, 1, 2, 3, 6, 7, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23, 24, 25, 29, 33, 39, 40, 42, 43, 46, 47, 48, 49, 50, 57, 70, 71, 74, 76, 77, 78, 79, 80, 81, 82, 84, 85, 86, 87, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 106, 116, 125, 128, 129, 130, 131, 132, 133, 148, 149, 154, 156, 157, 158, 159, 168, 173, 174, 175, 176, 177, 179, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 196, 197, 198, 199, 200, 202, 203, 204

**RNN** Recurrent Neural Network. 54, 55, 56, 57, 112, 122, 136

**SIMSSA** Single Interface for Music Score Search and Analysis. xiv

**SOM** Self-Organizing Maps. 61

**SVM** Support Vector Machine. 62

**TAVERN** Theme and Variation Encodings with Roman Numerals. xxiii, 85, 86, 91, 98, 99, 100, 148, 149, 150, 151, 152, 153

**TEI** Text Encoding Initiative. 48

**TPC** Tonal Pitch Class. 73

**TSD** Tonic, Subdominant, and Dominant. 98

**TSV** Tab-Separated Values. 156, 183

**WiR** When in Rome. xxiii, 82, 85, 86, 87, 99, 100, 101, 102, 148, 149, 150, 151, 152, 153

**WTC** [The] Well-Tempered Clavier. 101

**XML** Extensible Markup Language. xxxii, 48, 49

# Chapter 1

# Introduction

This dissertation deals with the methods for automatically annotating a musical score with **Roman Numeral Analysis (RNA)** labels. Among other applications, this facilitates advanced searching workflows in music collections. For example, searching by chord progressions or by modulation trajectories.

In Section 1.1, I present my personal motivation to contribute to this problem. Section 1.2 introduces the main challenges of this problem, as well as existing progress. Section 1.3 presents the contributions of this dissertation. Finally, Section 1.4 presents the structure of the remaining chapters.

## 1.1 Motivation for Roman Numeral Analysis

One of the most common ways to analyze a piece of tonal music is through **RNA**. This is a notation that is compact enough to be annotated within a regular musical score, yet encodes explanations to advanced musical concepts that would require much more words and effort to describe otherwise. In order to illustrate this more visually, consider the musical example in Figure 1.1. In this example, Chopin's Prelude Op. 28 No. 20, two successions of chords can be seen in measures 2 and 12 with their corresponding **RNA** annotations (below the staff) and chord labels (above the staff). Although the note arrangement and chord labels of the first two

chords (A♭ major and D♭ major, highlighted in green in the figure) are identical, their **RNA** labels are not. This is because in **RNA**, not only the chords are important, but also the context of the musical key.

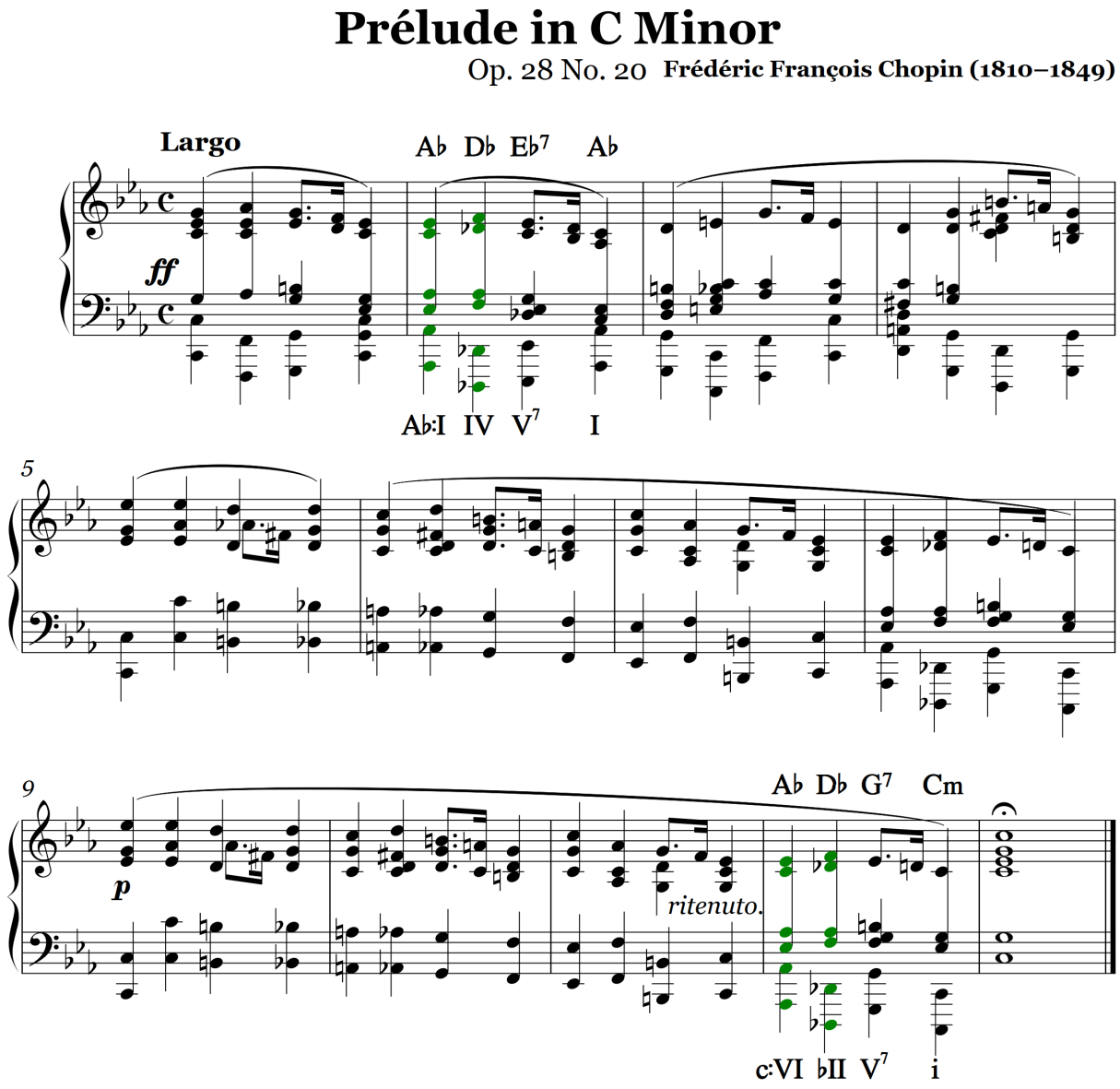


Figure 1.1: *Musical analysis of two successions of chords within Chopin's Op. 28 No. 20. The first succession occurs in measure 2. The second one in measure 12. Both successions begin with the same chords, however, their **RNA** labels are different.*

In the first instance (i.e., m. 2), the chords can be analyzed in the context of the **A**♭ major key. The first and second chord, A♭ major and D♭ major, are a tonic and subdominant chords,

respectively. These are followed by the third chord, a dominant seventh chord (E♭$^7$ dominant seventh), and the fourth chord, a tonic triad again (A♭ major). In the second succession of chords (i.e., m. 12), these can be analyzed in the context of the **c** minor key, which is the main key of the piece. Here, the first chord (A♭ major) is a submediant chord and the second chord (D♭ major) is a flattened second degree, commonly known as a **Neapolitan**.[1] The third chord is a dominant seventh chord (G$^7$ dominant seventh), and the last chord is the tonic of the piece, C minor.

Thus, annotating a musical score with **RNA** requires the analysis of several attributes related to chords and keys. Regarding chords, it is important to analyze their root, quality, inversion, and function. Regarding keys, it is suitable to analyze their temporal scope as modulations or tonicizations. All of this information is accomodated in a compact text representation, as shown in Figure 1.1. This compactness is maybe the reason why a number of musicians have adopted the notation throughout the years. The Roman numeral notation can describe complex—sometimes exotic, like the **Neapolitan**—chords, fluctuations of musical key, and other tonal situations using a few symbols. This is also motivating from the computational perspective, as it is possible to encode and retrieve such tonal information from the annotations. Clearly, if an **RNA** label is computed automatically, these tonal attributes become automatically available too for musical applications. An important caveat, of course, is that because each of these labels requires various layers of musical analysis, it is a challenging problem to retrieve them automatically via a computational model.

## 1.2 Challenges

In this section, I describe the main challenges in automatic **RNA** research, the progress made, remaining challenges, and the goals that I foresee for this problem in the future.

1. Although this chord is expressed here as ♭**II**, it is more commonly represented as **N** in **RNA**.

### 1.2.1 Main Challenges

Among the challenges that could come up when tackling **Automatic Roman Numeral Analysis (ARNA)** as an **Music Information Retrieval (MIR)** problem, I identify three that are impactful in most **ARNA** approaches: (1) it is difficult to draw a clear line separating different tonal problems (i.e., tonal problems like **Automatic Chord Recognition (ACR)** and key estimation often overlap), (2) there's a scarcity of high-quality data, and (3) the annotations that do exist are ambiguous and unregulated.

#### 1.2.1.1 Tonal Tasks Overlap

Tonal music analysis is abstract and often not well defined, in terms of what constitutes one task or the other.

**The Boundary between Chords and Keys.** In his work with *keyscapes*, Sapp (2011) presented examples of key analyses with different window lengths. In some of these analyses, with sufficiently short windows, an overlap between key analysis and chord analysis occurred. For example, a short window captured a change in harmony instead of a change of key. In our work designing a **Local-Key Estimation (LKE)** algorithm (Nápoles López, Arthur, and Fujinaga 2019), we found something similar with key-profiles and sequences of pitch classes analyzed with a **Hidden Markov Model (HMM)**. A **key profile** that heavily penalizes non-diatonic degrees finds a "change of key" in Chopin's Op.28 No.20. The location of this presumed change of key coincides with a **Neapolitan** chord, as shown in Figure 1.2. It is unclear whether this type of chromatic chord could be considered a deviation of the musical key. Arguments can be made for both positions on this topic. Thus, the line between *key analysis* and *chord analysis* can be blurry.

**Pitch Spelling.** In their work on a pitch spelling algorithm, Teodoru and Raphael (2007) described the dependency of "pitch spelling" task to **LKE** and **ACR**, noting that

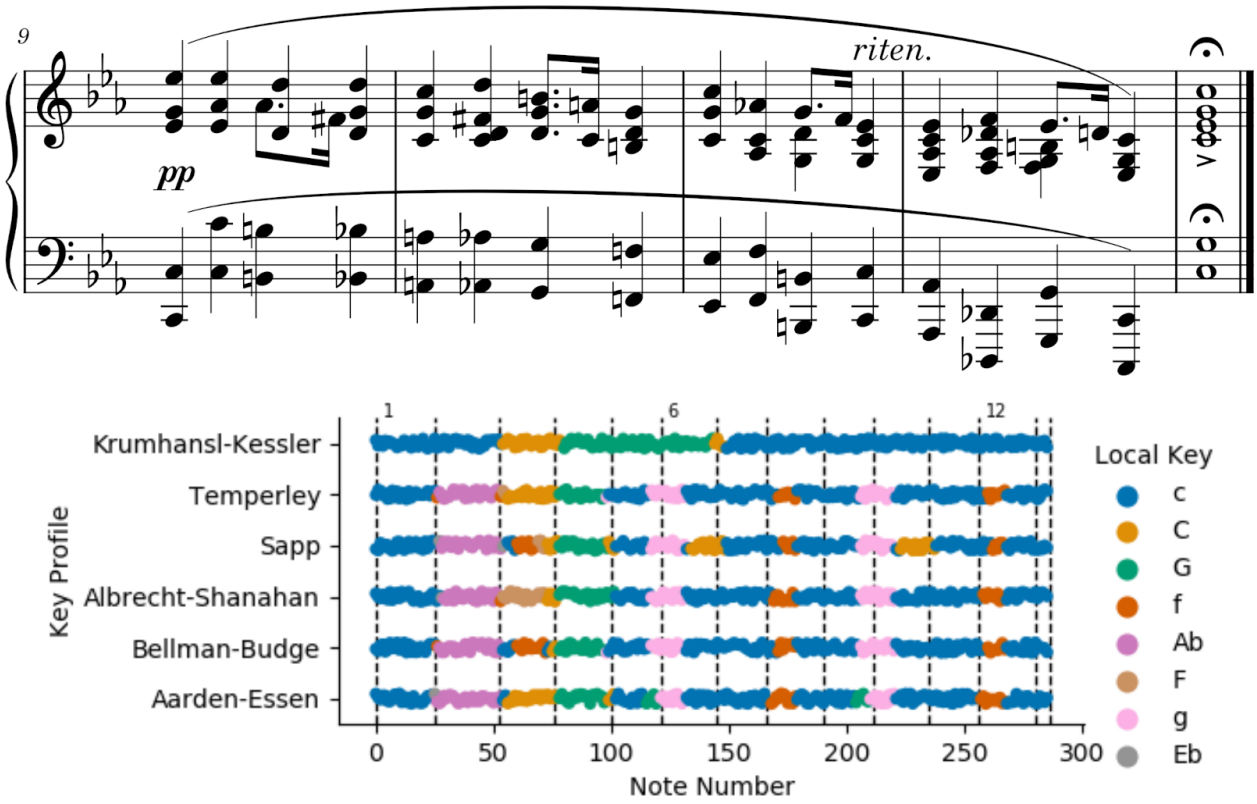


Figure 1.2: *An excerpt from Chopin's Op. 28 No. 20 (mm. 9–13). In this excerpt, measure 12 features a* ***Neapolitan*** *chord (the D♭ triad on beat 2). On the bottom, a* ***LKE*** *algorithm shows a "change of key" to* ***f*** *around the location of the* ***Neapolitan****. Figure taken from Nápoles López, Arthur, and Fujinaga (2019).*

> *[some tonal] situations require a deeper notion of the harmonic state than provided by the local key, as in the German* ***augmented sixth*** *chord, which seems nearly impossible to spell correctly without recognizing it as such.*

This poses the question of whether it is possible to design an algorithm to spell the pitches in a **Musical Instrument Digital Interface (MIDI)** file, without also developing a key estimation and chord recognition algorithms. **ARNA**, naturally sitting at the intersection of various tonal tasks, is constantly involved in similar "chicken and egg" problems. This may be also one of the motivations for **Multitask Learning (MTL)** approaches, trying to leverage the need to tackle several tonal problems at once.

#### 1.2.1.2 Ambiguous Annotations

In his dissertation, Ju (2021) describes ambiguity in music analysis and, particularly, chord labels. Ju argues that one possible reason for the ambiguity in musical analysis is *under-specification*. That is, the fact that based on the information provided, multiple answers can respond the same question. A typical example in chord labels would be whether one analyst considers a note as a chord tone, or a **nonchord tone**. Figure 1.3 shows an example of this situation. A *passing tone* is an ornamental note that connects two notes that are an interval of a third apart from each other. In the example, the ornamental note also happens to form an Emin7 chord ($\mathbf{vi}^{6}_{5}$ of G major in Roman numeral notation) with the other notes. Thus, two answers are arguably correct for this passage.

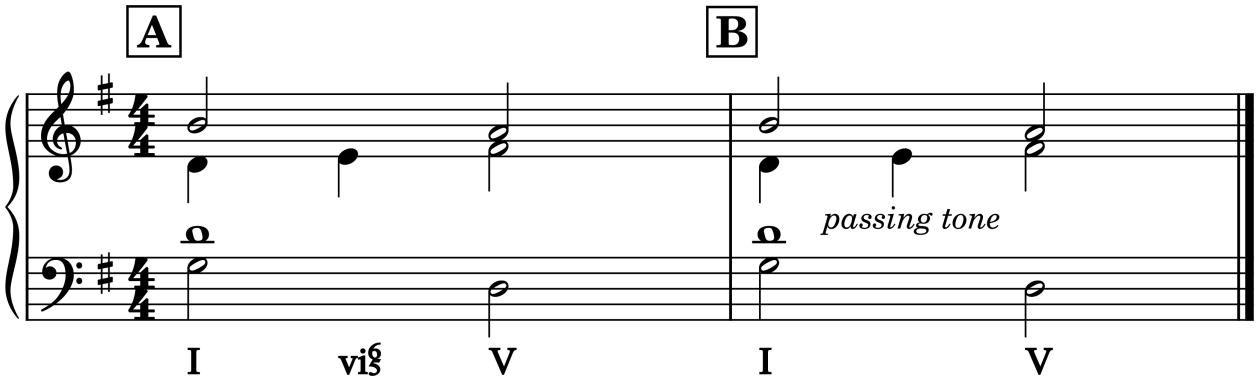


Figure 1.3: *Example of ambiguity in chord analysis. Two analyses are offered. In **[A]**, the second beat is considered an Emin7 chord. In **[B]**, the note in the second beat is considered a passing tone (i.e., there is no chord in the second beat). Reasonable arguments can be made for either analysis. Thus, both are correct.*

#### 1.2.1.3 Time-Consuming Annotations

Compared to other machine learning tasks, there is not as much high-quality data available for training **ARNA** models. In order to illustrate some of the reasons why this is the case, I contrast the process of annotating data for the well-known **Modified National Institute of Standards and Technology (MNIST)** dataset (LeCun et al. 1989) against **RNA**.

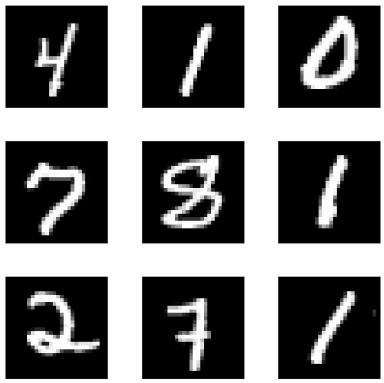

Figure 1.4: *Handwritten digits to be annotated, as in the **MNIST** dataset.*

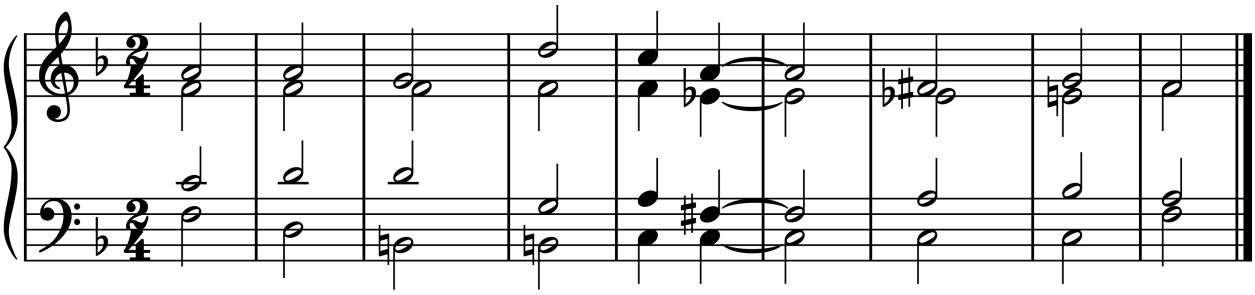

Figure 1.5: *Succession of chords to be annotated with Roman numerals, as in a **RNAs** dataset.*

Figure 1.4 shows a grid of nine handwritten digits, which could be labeled to create training examples for the **MNIST** dataset. Figure 1.5 shows a chord progression with nine chords. Each of the chords can be labeled with a Roman numeral annotation to create training examples for an **RNA** dataset. It can be noted that:

1. The handwritten digits can be labeled by a wider range of annotators, whereas the chords require an annotator with expertise on tonal harmony.

2. Even for an annotator with expertise on tonal harmony, the handwritten digits take much less time to annotate than the chords.[2]

In conjunction, these problems, make high-quality **RNA** data expensive and scarce.

2. As an informal test, I timed myself annotating the nine labels of each figure the first time I saw them. I spent, roughly, 6 seconds annotating the handwritten digits and 75 seconds annotating the Roman numeral labels. That is, annotating the chords was 12.5x more time consuming for me than annotating the handwritten digits.

### 1.2.2 Progress

Recent years have seen significant progress and interest from the **MIR** community in **ARNA**. For example, whereas pioneering approaches were spaced for decades (Winograd 1968; Maxwell 1984; Temperley 1997), the last year saw at least four new **ARNA** approaches (Chen and Su 2021; Micchi et al. 2021; McLeod and Rohrmeier 2021; Nápoles López, Gotham, and Fujinaga 2021). These approaches have introduced new ideas and methods to tackle **ARNA**.

#### 1.2.2.1 Predict Tasks Simultaneously

For the first time in **ARNA**, Chen and Su (2018) approached the problem using deep neural networks. An important finding in their experiments was that multitask learning[3] provided better results when predicting Roman numeral labels. That is, the intrinsic tasks of **ARNA**, key estimation and chord recognition, are best modeled together, sharing the same neural network weights. This has shown others a new paradigm to approach this problem. Nevertheless, this approach is continued to be challenged, for example, with modular approaches that separate the tasks and tackle them in a dedicated way (McLeod and Rohrmeier 2021).

#### 1.2.2.2 Provide Better Representations

After Chen and Su (2018), subsequent work by Micchi, Gotham, and Giraud (2020) considered different input representations of the musical input delivered to the **ARNA** model. What they found is that providing information about the lowest sounding note and the pitch classes is generally better than a full *pianoroll* input, and results in a smaller neural network too. Subsequent research (e.g., the one presented in this dissertation) confirms that this representation results in more accurate predictions. The support for pitch-spelling representations has continued in McLeod and Rohrmeier (2021), and the work of this dissertation.

3. A good summary of multitask learning is presented by Ruder (2017).

#### 1.2.2.3 Applying Music-Theory Domain Knowledge

In their work, Micchi, Gotham, and Giraud (2020) also demonstrated how the use of music-theory domain knowledge benefits the design of **ARNA** models. Particularly, two ideas inspired by music theory included the consideration of pitch spelling in the model[4] and taking modulations into account when transposing music for data augmentation.

### 1.2.3 Remaining Challenges

Although recent methods have achieved significant progress, some aspects of **ARNA** remain elusive. For example, the segmentation of chords and the scarcity of data.

#### 1.2.3.1 Segmentation

Predicting the location of chords (i.e., the harmonic rhythm) is a difficult problem. In their subsequent work on **ARNA**, Chen and Su (2019, 2021) introduced the use of Transformer-based networks (Vaswani et al. 2017) to improve the segmentation of the annotations. The segmentation refers to the location where a Roman numeral label ends and the new one begins. This has also become a priority in the work of Micchi et al. (2021) and remains a relevant challenge. This problem not only affects analytical models but generative ones too, requiring Wu et al. (2021) to develop an independent model of harmonic rhythm for melody harmonization. The work in this dissertation also considers ways to tackle this problem (see Section 5.4.9).

#### 1.2.3.2 Data Scarcity

For the same reasons discussed in Section 1.2.1.3, the scarcity of high-quality data remains an important challenge in **ARNA**. In this dissertation, a large **RNA** dataset has been aggregated from publicly available annotations (see Section 4.2). Although this comprises an ex-

4. Taking pitch spelling into account means that two enharmonic pitch classes are different classes, such that $C\sharp \neq D\flat$, when the convention in most **MIR** research is that $C\sharp = D\flat$.

tensive amount of RNA annotations, it remains relatively small for the scope of deep learning projects.

### 1.2.4 Short-Term Goal: Better Models

In the past few years, deep learning models[5] have achieved significant improvements. This has been possible because new, more powerful, deep learning techniques have been introduced. Similarly, the libraries for prototyping experiments, such as *Tensorflow* (Abadi et al. 2016) or *Pytorch* (Paszke et al. 2019), have become more powerful and easier to use. Thus, it is not surprising to think that in the next years, the performance of ARNA models will only improve from the models we have today. Although expensive, new datasets are created at an increasing pace. The short-term goal around this problem is, necessarily, to increase the accuracy and performance that these models achieve. Although the ambiguity of tonal problems and annotations is an important issue (Ju 2021), I am convinced that more and better data combined with larger deep learning models will lead to more accurate ARNA systems.

### 1.2.5 Long-Term Goal: Interpretability

Perhaps a more interesting question is, what do we do if the models become sufficiently accurate? One important aspect that may be useful in music theory is that computers quickly evaluate algorithms that would take a long time for a human being to internalize. A simple example would be voice-leading rules, which take many hours of practice for a human to learn, but can be implemented and fine-tuned for a computer within hours. Figure 1.6 shows an example of a rule-based voice-leading algorithm, which I implemented based on the rules in Huron (2016, 10).[6] The model arranges each chord based on the input RNA annotations provided, respecting the encoded voice-leading rules while connecting each pair of chords.

5. "Deep learning" is an umbrella term referring to machine learning algorithms—mainly artificial neural networks—that feature a large (i.e., deep) number of hidden layers. See Section 3.2 for a broader discussion of this topic.

6. https://github.com/napulen/romanyh

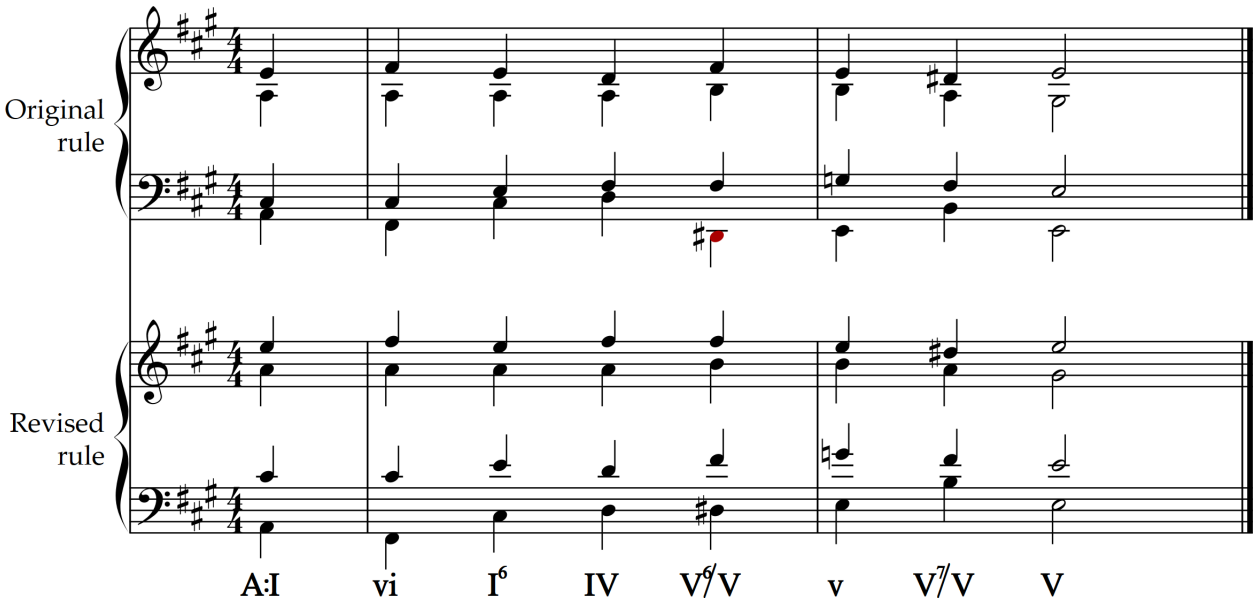


Figure 1.6: *Output of a voice-leading algorithm. The input to the algorithm were* ***RNA*** *annotations from a J. S. Bach chorale (BWV 347). Two versions of the algorithm are displayed, one that implements the voice-leading rule "avoid augmented melodic intervals" literally (above), and a revised version, which accomodates for augmented unisons (below).*

One challenge in the development of this algorithm was that some rules seemed to be detrimental to good voice leading when they were implemented textually. For example, the rule about augmented intervals (Huron 2016, p. 12):

> *Rule 16. Augmented intervals rule: Avoid augmented melodic intervals.*

Implementing this rule unchanged results in a model that avoids augmented unisons entirely. This approach is detrimental in chord progressions where the bass ascends chromatically, as shown in Figure 1.6. Interpreting the outputs of the algorithm was useful to become aware of this exception, where the rule did not work properly. One could argue that computational models provide a rigorous empirical platform to "test" the robustness of a music theory, a rigorous platform that is difficult to come by in traditional human pedagogy. Although the internal "algorithm" in a deep learning model is not as easy to interpret as the one in this voice-leading example, it is still possible to review the features learned by a deep learning model in its internal representations. This process has been studied in computer vision, and it would be worth to be studied in music models as these get increasingly accurate. Perhaps this goal,

interpretability, should be one that researchers working on computational models should take into consideration in the long term.

## 1.3 Outline of Thesis Contributions

This section outlines the contributions of this thesis, such as the data-augmentation technique developed, the workflow for aggregating the seven publicly available datasets, and the proposed improvements in **Convolutional Recurrent Neural Network (CRNN)** architectures for **ARNA**.

### 1.3.1 Additional Tonal Tasks

An important aspect of **RNA** is that it may span an incredibly large vocabulary of chords, with special symbols (e.g., **N** or **Ger**[7]) acting in special context, such as modulations and tonicizations. Because of this, chords in **RNA** are difficult to understand. However, they implicitly encode an extensive amount of information about the tonal context. An idea proposed in this dissertation is to create different tonal classification problems from a single **RNA** label. Although this has been done already, for example, in Chen and Su (2018), this dissertation presents a different set of these classification problems to be included in the neural network architecture. These are discussed in Section 5.4.

### 1.3.2 Artificial Training Examples

One way to overcome the scarcity of data is by synthesizing artificial training examples from existing data. This technique is popular in computer vision, where labeled images are transformed in various ways (e.g., rotation, mirroring, skewing, changing the brightness, etc.) I propose a new strategy to generate new training examples from existing Roman numeral annotations.

### 1.3.3 Data Hygiene

In recent years, machine learning researchers have become aware of the critical role that data plays in any experiment. For example, making sure that the quality or "hygiene" of the data employed in an experiment meets a certain quality. This effort frequently takes a considerable amount of time in a machine learning project. The idea of improving the data workflows has received the name of **Machine Learning Operations (MLOps)**. A description of some **MLOps** principles is provided in Renggli et al. (2021).

There are at least four different digital standards of Roman numeral annotations. The available data is encoded with these different formats. Thus, it is challenging to aggregate the data into a unified training dataset, often leading to noisy and inaccurate training examples. Throughout this dissertation, I make an effort to describe how to operate with the existing data. For example, describing the process of detecting errors in the annotations in Section 4.1.2, processing the outputs of different models in the evaluation (see Section 6.4.2), and, maybe more importantly, releasing all the curated data used to train the model presented here (see Section 7.3.3). This abstracts the data curation process for researchers interested in re-training this or another model, or wanting to reproduce the results.

### 1.3.4 Generating Roman Numeral Labels from Predictions

In the work by Chen and Su (2018), Roman numeral labels were divided into a subset of classification tasks, from which the full Roman numeral label was predicted. However, the process for turning the **MTL** predictions of the network into **RNA** labels was not described. This process is far from trivial. In this dissertation, I propose two methods to process the predictions of an **ARNA** model and turn them into **RNA** annotations in text form. One of the methods is tailored for the specific **CRNN** described in this dissertation (see Section 5.5.1). Yet, another method is more generic and can facilitate the process of retrieving **RNA** annotations from other models of the literature (see Section 5.5.2).

### 1.3.5 Original Input Representation of Pitch Spelling

Extending over the work of Micchi, Gotham, and Giraud (2020), in this dissertation, I present an alternative encoding method for pitch spelling. This method provides similar benefits than the one in Micchi, Gotham, and Giraud (2020), but it requires less parameters in the neural network architecture. Particularly, this new encoding method is used in the input representations of the network, described in Section 5.1.3.

### 1.3.6 Original Layout of Neural Network

Lastly, the **CRNN** layout presented in this dissertation is inspired by, but different, to existing architectures. The characteristics of the network are described in Chapter 5.

## 1.4 Thesis Structure

This dissertation is organized in seven chapters and an appendix. Chapter 1 described the motivations, challenges, and existing progress toward **ARNA**. Chapter 2 introduces **RNA** from a musical perspective, its history and digitization. Chapter 3 introduces the relevant research around music representation, deep learning, and **MIR** for tonal music analysis. Chapter 4 introduces the publicly available datasets, the data aggregation process, and the data-augmentation technique based on synthesis of training examples. Chapter 5 presents the design choices of the **CRNN** presented in this dissertation, **AugmentedNet**: its input and output representations, convolutional layers, recurrent layers, **MTL** configuration, and methods to generate **RNA** labels from the predictions of the model. Chapter 6 presents the evaluation of the model. This includes ablation studies over the design choices of the network, an exploration of the effects of data augmentation, a full evaluation on the aggregated dataset and test sets of each individual dataset, and a comparison against four existing **ARNA** methods. Chapter 7 summarizes the main findings and presents closing remarks on the current state of automatic tonal analysis and future directions in the field. In addition to this, this chapter introduces the resources

for reproducing, improving, and sharing the results of this research. Lastly, Appendix A is a special annex chapter that introduces several formal methods for systematic Roman numeral analysis, which are referenced throughout the remaining chapters.

# Chapter 2

# Introduction to Roman Numeral Analysis

**Roman Numeral Analysis (RNA)** is a common methodology in modern music theory textbooks dealing with tonal music. The name derives from the use of Roman numerals to denote scale degrees, which, in turn, indicate diatonic chords in a certain musical key. The musical key is usually prepended to the first Roman numeral of the piece, using a case-sensitive note letter[1] followed by a colon. When there are changes of key throughout the piece, these are indicated in a similar way, with a new key letter followed by a colon, prepended to a Roman numeral.

**RNA** is an alternative way to annotate chords than the, possibly more popular, chord label system. **RNA** facilitates the annotation of more information than chord labels. Namely, changes of key and special chords. Perhaps for this reason, **RNA** is common among tonal music theory textbooks,[2] as it is helpful to break down musical pieces analytically. Figure 2.1 shows a fragment of music annotated with both Roman numeral and chord label annotations.

1. In the case-sensitive notation, upper-case numerals are used to indicate major keys, and lower-case numerals are used to indicate minor keys.
2. For examples, see the adoption of the notation discussed in Section 2.2.

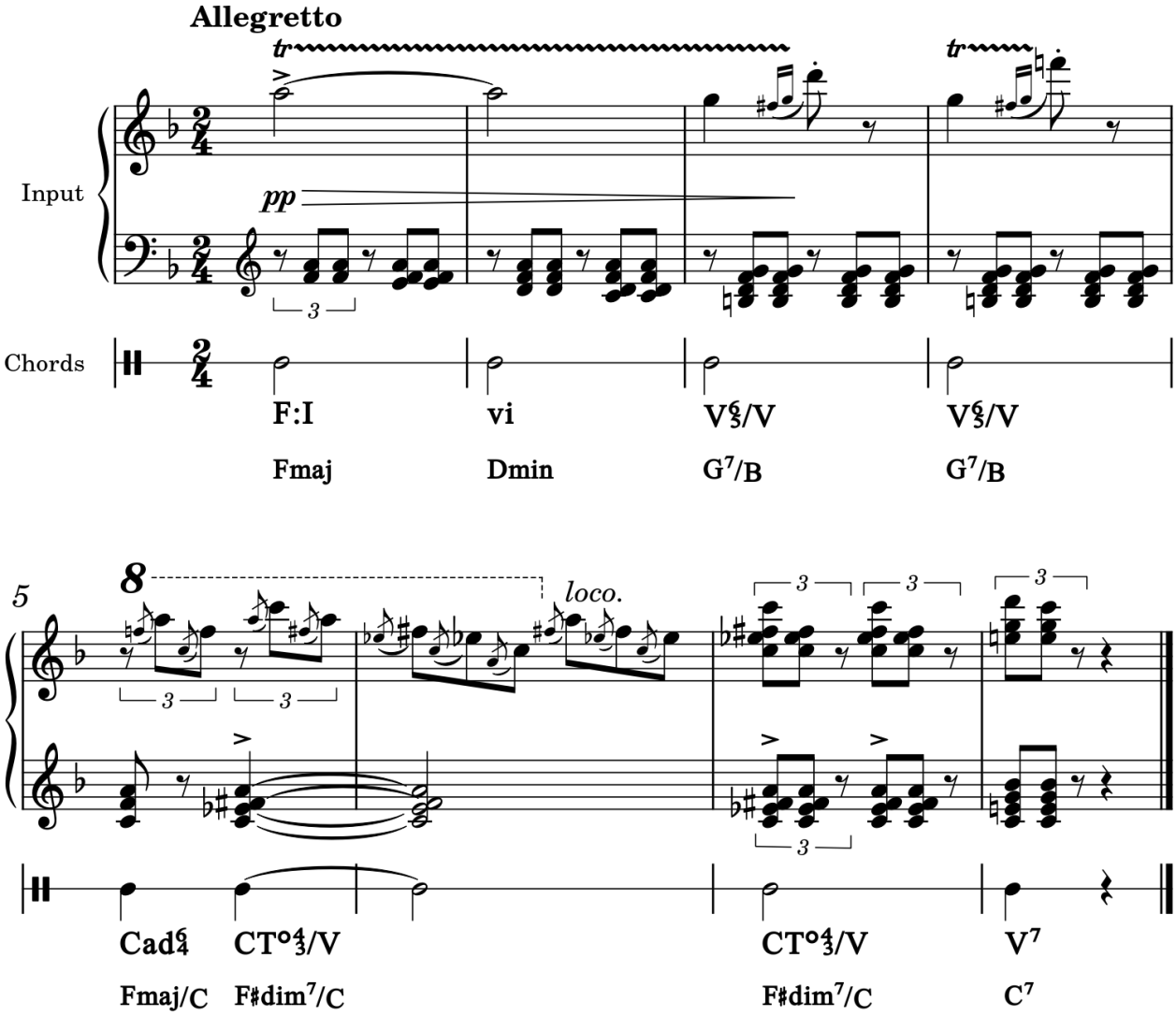


Figure 2.1: *The first eight measures of Josephine Lang's Op.8 No.1 "Schmetterling," annotated with **RNA** and chord labels underneath.*

## 2.1 Roman Numeral Analysis and Chord Labels

Chord labels, such as the ones shown in the lower annotations of Figure 2.1, generally indicate the root of a chord and its quality. Less frequently, they also indicate the bass (which is included in Figure 2.1 for completeness). A Roman numeral label provides this information as well, complemented with information about keys, inversions, and functional roles of the chords. Because Roman numerals are always relative to a key, the key must be indicated at all times to disambiguate the meaning of the Roman numeral. This is particularly helpful in

**Music Information Retrieval (MIR)**, where key-estimation and chord-recognition models can be developed using the same Roman numeral annotations. Roman numerals indicate inversions for triads and seventh chords. These are often indicated with $^{6}$ and $^{6}_{4}$ in triads and $^{6}_{5}$, $^{4}_{3}$, and $^{2}$ in seventh chords. In the modern syntax, it is also customary to indicate *applied chords* or *secondary dominants*. Once the concept of *tonicization* emerged,[3] in the twentieth century, this quickly became an important idea of tonal music, particularly relevant to analyze the chromatic music of the 19th century. Roman numerals facilitate the indication of tonicizations using a *slash* ("/") symbol. For example, the annotation $\mathbf{V^6_5/V}$ in measures 3 and 4 of Figure 2.1 indicates a *dominant of the dominant* (i.e., tonicization of the dominant key). With these syntactic conventions, a musical analysis can be more informative, indicating the analyst's point of view of the musical key at a particular moment of the piece.[4]

The next section explores the historical evolution of the syntax, from the early precedents in the late 18th century, to its modern notation.

## 2.2 A Brief History of Roman Numeral Analysis

Over the last 200 years, the **RNA** syntax has evolved in an unregulated way, with Western music theorists proposing new notations, and adopting the ones of previous theorists at discretion. Beyond the recent formalization efforts from researchers who have digitized **RNA** (Huron 1994; Nápoles López 2017; Neuwirth et al. 2018; Gotham, Tymoczko, and Cuthbert 2019; Nápoles López and Fujinaga 2020a; Hentschel, Neuwirth, and Rohrmeier 2021; Hentschel et al. 2022), there is no agreed "standard" way of writing Roman numerals. That is, the notation often varies with the analyst.[5] One way to investigate the different notational conventions in **RNA** could be to observe, chronologically, the practice of the notation over the years.

3. A tonicization is a brief deviation to a different key, usually with the intention of emphasizing a certain scale degree or harmony. For a broader discussion on this topic in the context of **MIR**, see Nápoles López et al. (2020).
4. See Section A.1 for a more formal take on the **RNA** syntax.
5. See Section 2.3 for a discussion of some aspects of the notation that often vary.

In this section, I present a summary of the RNA notation across nearly 150 primary sources between the late 18th and early 20th centuries. The primary sources were collected from three keyword searches in the WorldCat[6] library catalog: "harmony," "harmonie," and "harmonielehre." These keywords were used to search for books in the English, French, and German languages, respectively. For books with multiple editions, the earliest available edition was reviewed.[7] The full table of references is available on Table 2.1.

The objective of reviewing the primary sources was to observe the evolution of the RNA syntax across harmony textbooks. For each book, I inspected the annotated musical examples, occasionally observing annotations in the body of the textbook, such as the tables in Figure 2.2. Although the survey was centered around Roman numeral annotations, because many of the musical examples included figured bass annotations, the usage of figured bass notation was also documented.

### 2.2.1 Early Precedents and the Weber Syntax

Before the development of the RNA syntax, Roman numerals were used to indicate scale-degree relationships. For example, in two tables of *Die Kunst des reinen Satzes in der Musik* in Kirnberger (1774). One of these tables is shown in Figure 2.2.

Later, Vogler introduced Roman numeral symbols underneath a staff, which also referred to a scale degree (Vogler 1778, 1802). These are used a few times on both works. Figure 2.3 shows an example in Vogler (1778).

While it is difficult to credit someone with "inventing" RNA, the modern notation would probably not exist without the precedent of Weber (1818). Weber extended the notation to indicate not only scale degrees but their chord quality. A special notation separates diminished triads (i.e., the seventh degree) from major and minor chords; whereas major and minor are

6. https://www.worldcat.org/
7. Physically available at the McGill Marvin Duchow Library, or digitally available in e-book form.

Table 2.1: *All primary sources reviewed in terms of their RNA syntax.*

Kirnberger (1774)
Weber (1818)
Fétis (1844)
Sechter (1853)
Bazin (1857)
Reber (1862)
Tiersch (1868)
Bussler (1878)
Clarke (1880)
Mangold (1883)
Jadassohn (1883)
Riemann (1887)
Shepard (1889)
Vivier (1890)
Buwa (1893)
Chadwick (1897)
Boise (1898)
Bridge (1900)
Riemann (1902)
Schenker (1906)
Capellen (1908)
Klauser (1909)
York (1909)
Gardner (1912)
Molitor (1913)
Gilson (1914)
Leavitt (1916)
Fowles (1918)
Watt (1919)
Gilson (1919)
Kitson (1920)
Alchin (1921)
Schenker (1921)
Schoenberg (1922)
McConathy (1927)
Koechlin (1928)
Morris (1931)
Jones (1939)
Bairstow (1945)
Jacobs (1958)
Tischler (1964)
Siegmeister (1965)
Schoenberg (1967)
Mickelsen (1977)
Aldwell and Schachter (1978)
Tunley (1984)
Levy (1985)
Sarnecki (2010)

Vogler (1778)
Logier (1827)
Lobe (1850)
C. C. Spencer (1854)
Volckmar (1860)
Oettingen (1866)
Tiersch (1874)
Emery (1879)
Bowman (1881)
Coon (1883)
Oakey (1884)
Broekhoven (1889)
Jadassohn (1890)
Goetschius (1892)
Norris (1894)
Gladstone (1898)
Werker (1898)
Halm (1900)
Shinn (1904)
Heacox (1907)
Loewengard (1908)
Lavignac (1909)
White (1911)
Mokrejs (1913)
Riemann (1913)
S. R. Spencer (1915)
Orem (1916)
Robinson (1918)
Foote (1919)
Ham (1919)
Buck (1920)
Knorr (1921)
Klatte (1922)
Wedge (1924)
Hába (1927)
Wedge (1930)
Schenker (1935)
Piston (1941)
Morris (1946)
Ottman (1961b)
Goldman (1965)
Abraham (1965)
Schoenberg (1969)
Dreyer (1977)
Forte (1979)
S. M. Kostka and Payne (1984)
Carter (2002)
Roig-Francoli (2011)

Vogler (1802)
Hamilton (1840)
Meister (1852)
Southard (1855)
Richter (1860)
Ouseley (1868)
Tracy (1878)
Kistler (1879)
Durand (1881)
Riemann (1883)
Saint-Saëns (1885)
Prout (1889)
Riemann (1890)
Goodrich (1893)
Shepard (1896)
Clarke (1898)
Cutter (1899)
Cutter (1902)
Foote and Spalding (1905)
Louis and Thuille (1907)
Gladstone (1908)
Vinée (1909)
Eyken (1911)
Kallenberg (1913)
Lenormand (1913)
Hull (1915)
Heacox (1917)
Anger and Clough-Leighter (1919)
Deveaux (1919)
Macpherson (1920)
Koch (1920)
Dubois (1921)
Schenker (1922)
Scholes (1924)
Krehl (1928)
Campbell-Watson (1930)
Barnes (1937)
Hindemith (1943)
Murphy and Stringham (1951)
Ottman (1961a)
Mitchell (1965)
Ulehla (1966)
Scheidt (1975)
Motte (1978)
Lester (1982)
Toutant (1985)
Swain (2002)

distinguished by the size of the Roman numeral.[8] Weber introduced several other traits of

8. Nowadays, instead of Roman numerals of different sizes, it is more common to see upper- and lower-case Roman numerals to indicate major and minor triads, respectively.

**Tabelle der Intervalle**

**in den Tonarten der Alten.**

| | I | II | III | IV | V | VI | VII | VIII |
|---|---|---|---|---|---|---|---|---|
| C | 1 C | $\frac{8}{9}$ D | $\frac{4}{5}$ E | $\frac{3}{4}$ F | $\frac{2}{3}$ G | $\frac{3}{5}$ A | $\frac{8}{15}$ B | $\frac{1}{2}$ c |
| D | 1 D | $\frac{9}{10}$ E | $\frac{27}{32}$ F | $\frac{3}{4}$ G | $\frac{27}{40}$ A | $\frac{3}{5}$ B | $\frac{9}{16}$ c | $\frac{1}{2}$ d |
| E | 1 E | $\frac{15}{16}$ F | $\frac{5}{6}$ G | $\frac{3}{4}$ A | $\frac{2}{3}$ B | $\frac{5}{8}$ c | $\frac{5}{9}$ d | $\frac{1}{2}$ e |
| F | 1 F | $\frac{8}{9}$ G | $\frac{4}{5}$ A | $\frac{32}{45}$ B | $\frac{2}{3}$ c | $\frac{16}{27}$ d | $\frac{8}{15}$ e | $\frac{1}{2}$ f |
| G | 1 G | $\frac{9}{10}$ A | $\frac{4}{5}$ B | $\frac{3}{4}$ c | $\frac{2}{3}$ d | $\frac{3}{5}$ e | $\frac{9}{16}$ f | $\frac{1}{2}$ g |
| A | 1 A | $\frac{8}{9}$ B | $\frac{5}{6}$ c | $\frac{20}{27}$ d | $\frac{2}{3}$ e | $\frac{5}{8}$ f | $\frac{5}{9}$ g | $\frac{1}{2}$ a |
| B | 1 B | $\frac{15}{16}$ c | $\frac{5}{6}$ d | $\frac{3}{4}$ e | $\frac{45}{64}$ f | $\frac{5}{8}$ g | $\frac{9}{16}$ a | $\frac{1}{2}$ b |

Figure 2.2: *Roman numerals in Kirnberger (1774, 15).*

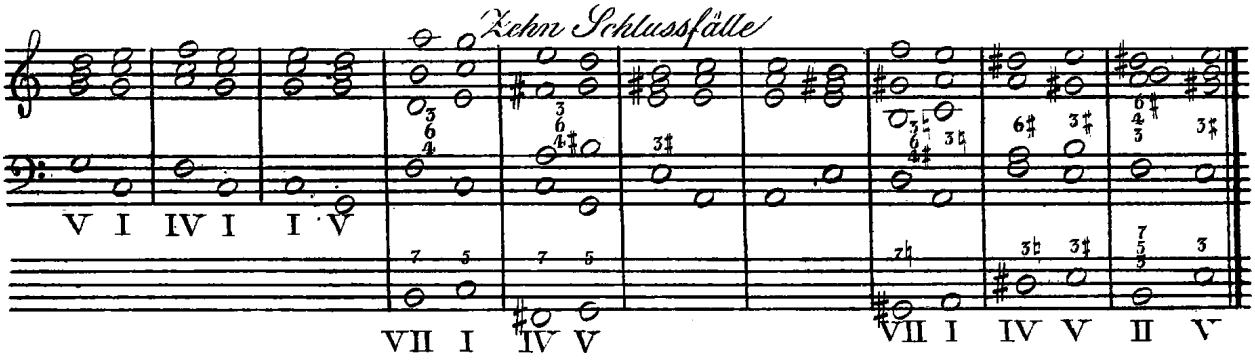


Figure 2.3: *Roman numerals in Vogler (1778, Tab XXI).*


Figure 2.4: *Roman numerals in Weber (1818, 37). Keys are indicated with colons (e.g., mm. 1 and mm.5). Dominant seventh chords are indicated as* $V^7$*. Smaller Roman numerals indicate minor triads.*

modern RNA. For example, changes of key are indicated using a colon ("**:**") symbol and analyses with several rows of keys and Roman numeral indications appear in modulating passages. Finally, whereas Kirnberger and Vogler used Roman numerals sporadically, Weber used the notation extensively in the *Versuch einer geordneten Theorie der Tonsetzkunst* (Weber 1818), as shown in Figure 2.4. Weber indicated dominant seventh chords as $\mathbf{V}^7$, which is a notation that is still in use today. Whereas others relied on figured bass to explain their examples, Weber relied mostly on Roman numerals.

### 2.2.2 Adoption of the Weber Syntax

The notational system introduced in Weber (1818) was not immediately adopted by other theorists. One of the first adopters was Hamilton, a British music professor. In the syntax of Hamilton (1840), all the scale degrees were indicated with a single Roman numeral style, regardless of the chord quality of the triad. That is, although Weber's notation was partially adopted, this did not include the notation for major and minor chord qualities. Another difference is that, in Hamilton's annotations, Roman numerals indicate melodic scale degrees, instead of chord roots, as shown in Figure 2.5.

Another early adoption of the Roman numeral notation, arguably the first among the German theorists, was in Meister (1852). Meister often accompanied the Roman numeral notation with chord labels and figured bass indications, as shown in Figure 2.6.

In order, the subsequent authors who adopted Roman numerals seem to be Sechter, Richter, Tiersch, and Tracy. Although Sechter (1853) was annotated with mainly chord labels, the third part of the book (starting on page 98) introduces Roman numerals in the musical examples, possibly to relate the scale degree of a chord root to different key contexts. An example is shown in Figure 2.7. The Weber syntax for major and minor chord qualities appears to be adopted in Richter (1860), where a single-quote symbol (') was also introduced to indicate augmented chords, shown in Figure 2.8. The use of Roman numerals is absent in the earlier treatise of Tiersch (1868). However, in Tiersch (1874), most of the annotations provided in the musical

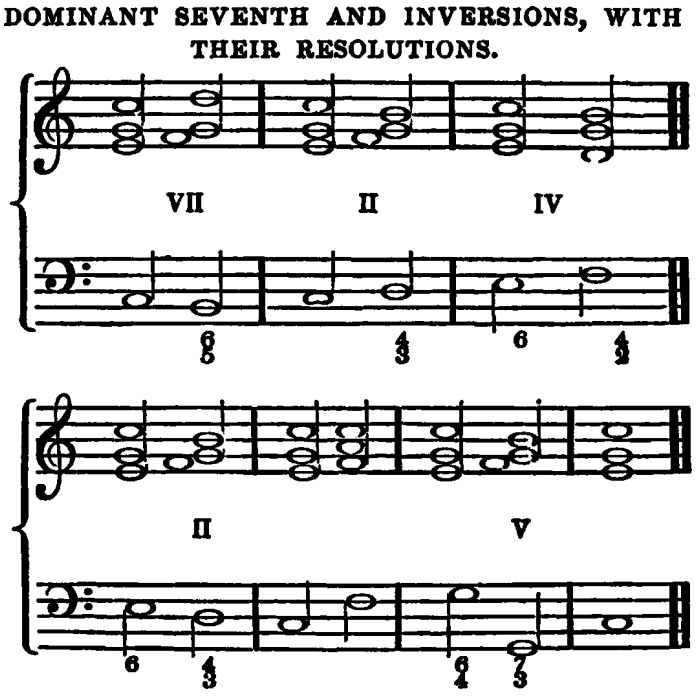


Figure 2.5: *Use of Roman numerals in Hamilton (1840, 44). The Roman numeral indicates the scale degree of the bass, regardless of the chord root. In modern Roman numeral notation, these annotations would be written as* $\mathbf{V}^6_5$, $\mathbf{V}^4_3$, $\mathbf{V}^2$, $\mathbf{V}^4_3$, *and* $\mathbf{V}^7$, *respectively.*

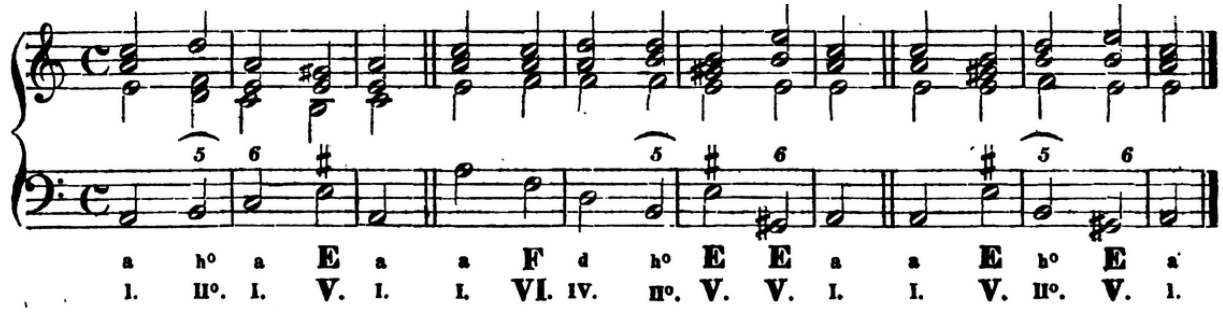

Figure 2.6: *Use of Roman numerals in Meister (1852, 32). The Roman numerals are accompanied by chord label and figured bass indications.*

examples were figured bass or Roman numeral annotations. Tracy (1878) included the notation with some peculiarities, for example, using a seven ("**7**") figure for all seventh chords except dominant seventh chords, which were simply written as **V**.

More authors followed in adopting Roman numerals during the late 19th century and, by the beginning of the 20th century, it was more common than not to observe **RNA** in emerging harmony textbooks. One exception was among textbooks in French, where figured bass nota-

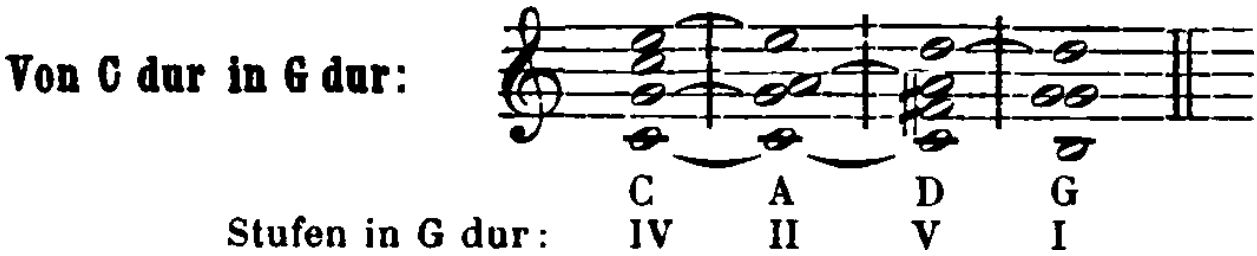


Figure 2.7: *Use of Roman numerals, underneath chord label annotations, in Sechter (1853, 103).*

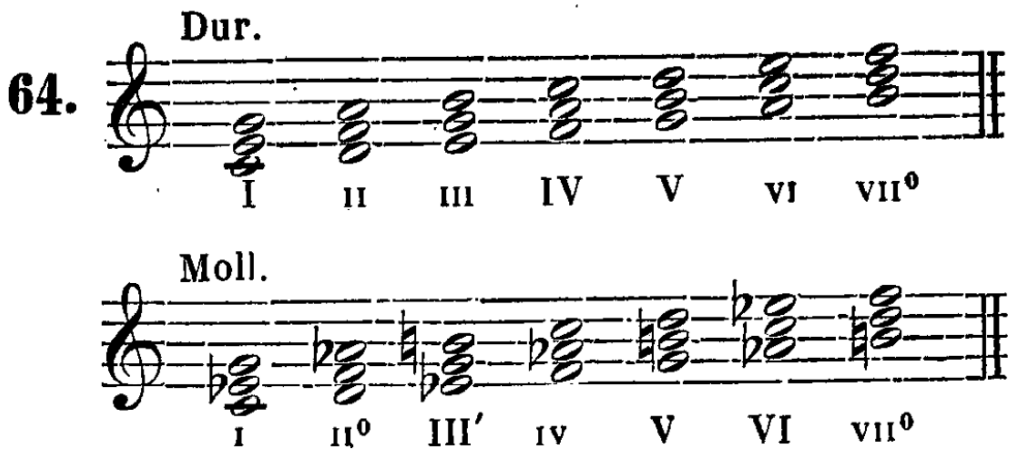


Figure 2.8: *Adoption of the Weber syntax in Richter (1860, 34), featuring a notation for augmented triads,* ***III'****, which appears for the first time among the sources surveyed.*

tion seemed more prominent. A textbook in French by Koechlin (1928) used Roman numerals sporadically to indicate scale degrees of the bass note, as shown in Figure 2.9. This practice, which was similar to the one in Hamilton (1840), was uncommon by the time of Koechlin (1928), as most English or German books then used Roman numerals to refer to the scale degree of the chord root, not the bass. This "archaic" **RNA** practice of Koechlin might suggest the importance that figured bass notation had in the French language. In fact, some English and German books referred to figured bass as the *French* system.[9] Beyond being the preferred system in French textbooks, figured bass also heavily influenced the evolution of **RNA** in other ways.

9. For example, Norris (1894).

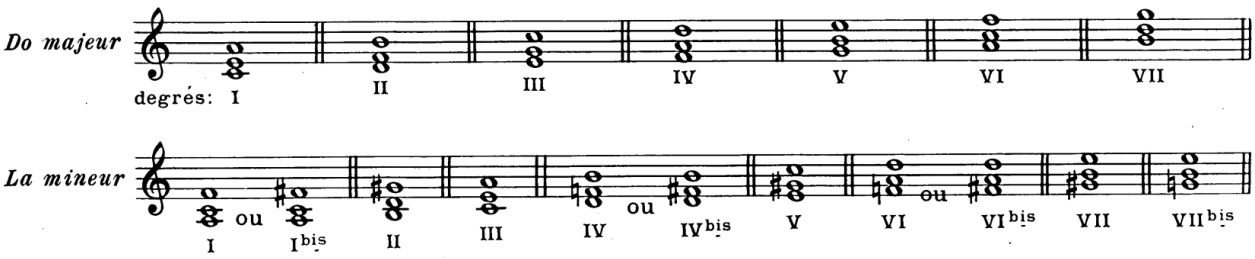


Figure 2.9: *The scale-degrees of Koechlin (1928, 26), which refer to the bass note and not the root of the chord.*

### 2.2.3 Chord Inversions and Figured Bass

During the **RNA** examples of the 19th century that followed Weber (1818), it was common to complement the notation with figured bass. The figured bass was often clearly separated from the Roman numerals, for example, by annotating one below the staff and the other one above, as seen in Figures 2.5, 2.6, or 2.11.

In Figure 2.10, however, a differing example in Bussler (1878) shows the Roman numerals and figured bass annotations in the same analytical layer. One could argue that this syntax is an early version of the modern chord inversion syntax, where inversions are denoted by Arabic numerals next to the Roman numeral. In principle, those Arabic numerals represent intervals, as in the figured bass notation. However, in **RNA**, it is common to see certain "stacks" of Arabic numerals appearing often, such as $^{7}$, $^{6}_{5}$, $^{6}_{4}$, $^{4}_{3}$, or $^{2}$. Nowadays, these stacks are generally understood as "chord inversions."

The practice of using Arabic numerals in **RNA** was further developed in Emery (1879), shown in Figure 2.11. In Emery's example, it is clearer that the stacks of Arabic numerals in the **RNA** layer have a special meaning, as figured bass annotations also appear above the staff. These stacks of Arabic numerals are indicators of special chords, which will be discussed in Section 2.2.4.

In addition to the special chords in Emery (1879), the so-called **cadential six-four** chord could have also encouraged the numeric notation for chord inversions.

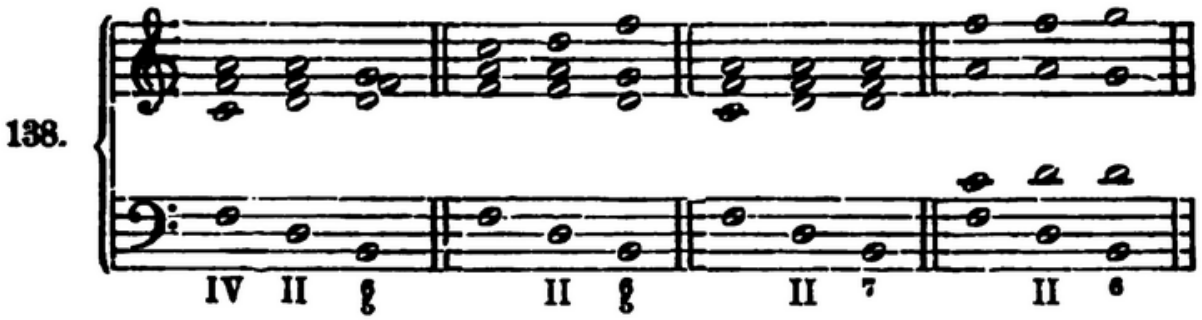


Figure 2.10: *A single layer of chord annotations underneath the bass staff in Bussler (1878, 63), where Roman numerals and figured bass labels are intertwined.*

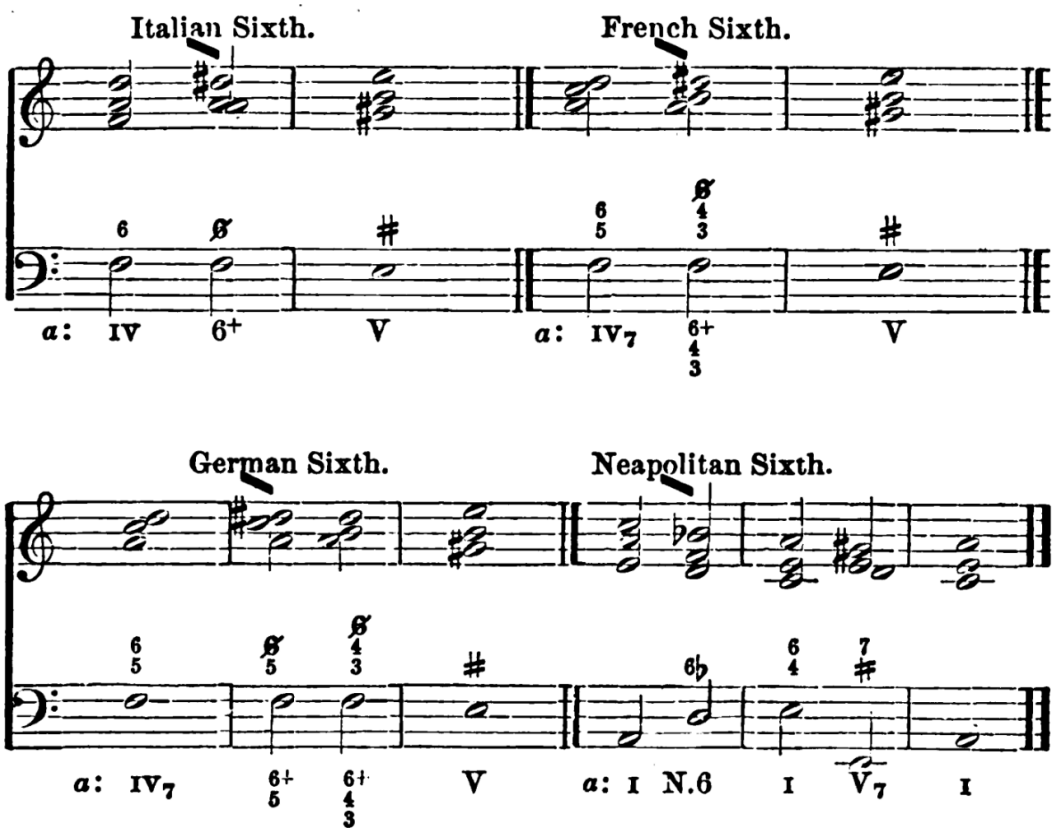


Figure 2.11: *Use of Roman numerals and figured bass in Emery (1879, 51). In the Roman numeral layer, stacks of Arabic numerals indicate* ***augmented sixth*** *chords. Additionally, a notation for* ***Neapolitan*** *chords is introduced. See Section 2.2.4 for further discussion of* ***Neapolitan*** *and* ***augmented sixth*** *chords.*

Shepard (1896) included figured bass annotations next to the Roman numeral to indicate certain chord inversions, notably, the **cadential six-four** chord, as shown in Figure 2.12. This notation was consistent even when there was no music notation involved. In such case, Shep-

ard would write the Arabic numerals next to the Roman numeral in plain-text form, as shown in Figure 2.13.

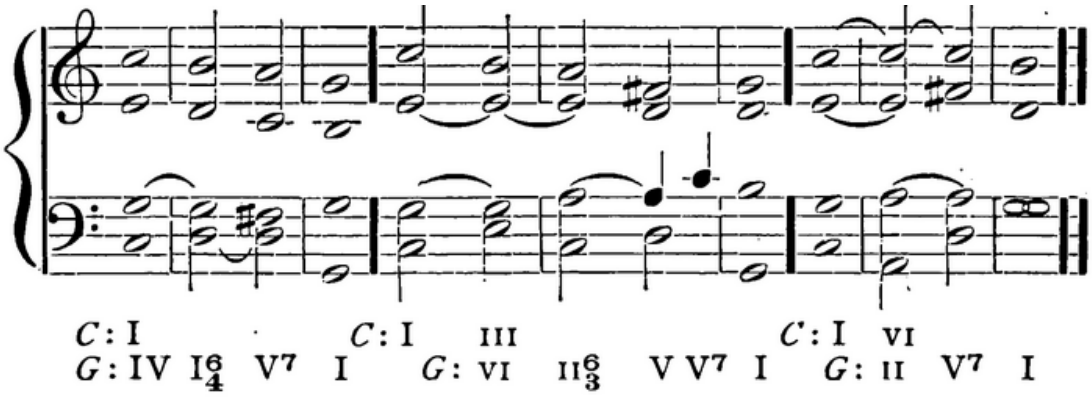


Figure 2.12: *Numeric inversions in Shepard (1896, 184).*

A few of the more common forms are —
(*a*.) IV, V7, I.
(*b*.) IV, I6/4, V7, I.
(*c*.) II, V7, I.
(*d*.) IV, II, V7, I. Play them.

Figure 2.13: *Numeric inversions in plain-text, without accompanying music notation (Shepard 1896, 117).*

Chadwick (1897) followed a similar practice, also indicating the numeric inversions in examples without music notation, as shown in Figure 2.14. Chadwick (1897) also provided an explanation of the numeric inversions, shown in Figure 2.15. This notation goes beyond triads, including, for example $\mathbf{V}^{4}_{3}$, as shown in Figure 2.16.

Finally, a more compelling example of the role of **cadential six-four** chords in the notation of chord inversions can be seen in Loewengard (1908), where the **cadential six-four** figure ($\mathbf{I}^{6}_{4}$) has the numeric inversion notation, but the $\mathbf{ii}^{6}$ chord in the same line does not (see Figure 2.17).

An alternative notation for chord inversions adopted by several theorists consists of the use of letters. In Cutter (1902), both notations are explained:

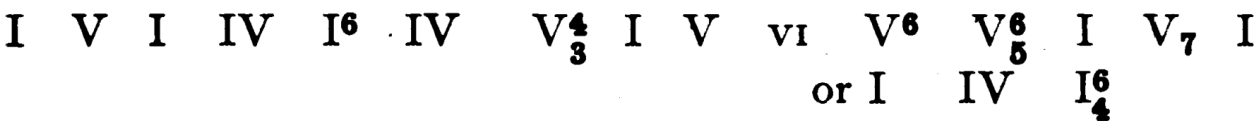


Figure 2.14: *Chadwick (1897, 38) describing numeric inversions even in examples without music notation in them.*

The first inversion of a triad is figured $\frac{8}{6}$ or $\frac{6}{3}$ or 6. These figures applied to the Roman numerals indicate both the fundamental harmony and its inversions, i.e.

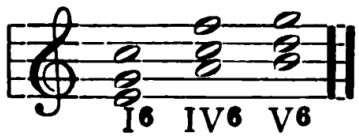


Figure 2.15: *Arabic-numeral inversions in Chadwick (1897, 12).*

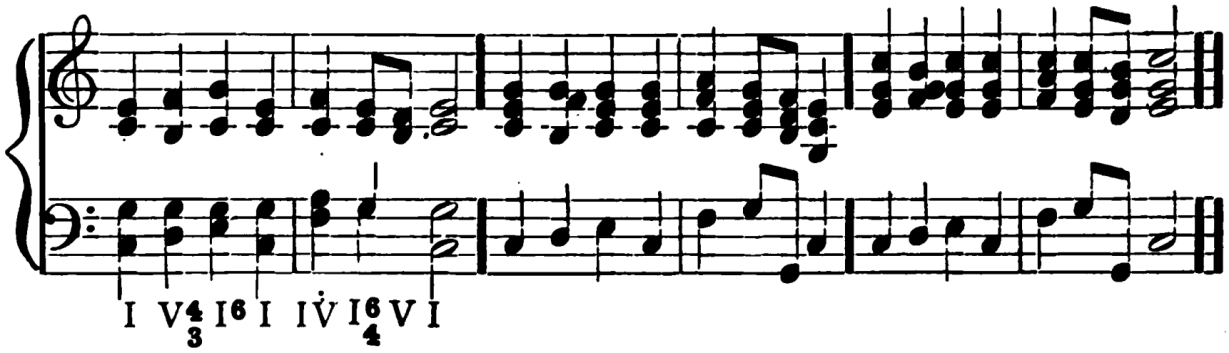


Figure 2.16: *A dominant seventh chord in second inversion in Chadwick (1897, 28).*

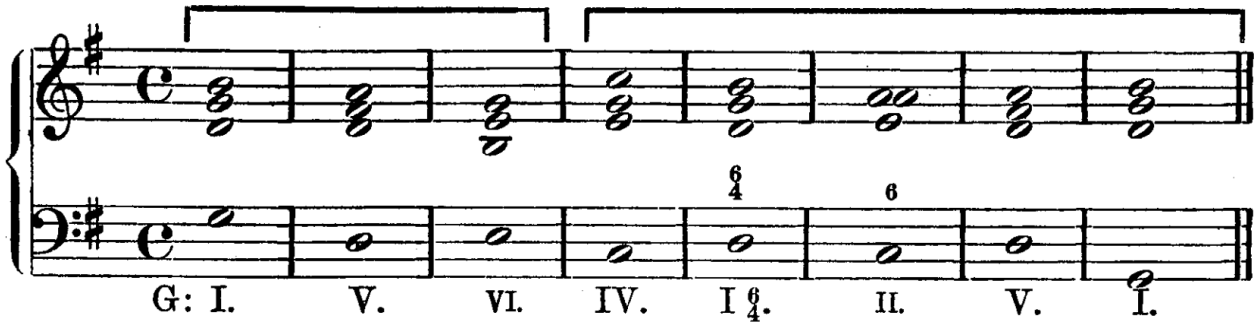


Figure 2.17: *Missing first inversion of the* $\boldsymbol{ii^6}$ *chord (measure 6) in Loewengard (1908, 45).*

*The inversions of triads and of seventh chords, both principal and secondary, will be indicated by the customary figurings:* $^{6}$, $^{6}_{4}$, $^{6}_{5}$, $^{4}_{3}$, $^{4}_{2}$, *attached to the respective Roman numerals. Or, the letters* ***a***, ***b***, ***c***, ***d***, *meaning root-form, first, second, and third inver-*

*sions, may be used with these same numerals. Thus:* $\boldsymbol{I_a}$, $\boldsymbol{I_b}$, $\boldsymbol{I_c}$, $\boldsymbol{ii^{o7}_a}$, $\boldsymbol{ii^{o7}_b}$, $\boldsymbol{ii^{o7}_c}$, $\boldsymbol{IV^{+}_a}$, $\boldsymbol{iii^{7}_c}$, *etc. The diminished seventh chord, in its various forms, will be marked:* $\boldsymbol{vii^{o}_{7o}}$, $\boldsymbol{vii^{o6}_{5}}$, $\boldsymbol{vii^{o4}_{3}}$, $\boldsymbol{vii^{o4}_{2}}$ — *or* $\boldsymbol{vii^{o7o}_a}$, $\boldsymbol{vii^{o7o}_b}$, $\boldsymbol{vii^{o7o}_c}$, $\boldsymbol{vii^{o7o}_d}$.

— Cutter (1902, 4)

The numeric inversion syntax was already used by previous theorists, however, the letters $a, b, c, d$ appear for the first time in Cutter (1902) among the books surveyed. Cutter also seems to be the only author acknowledging the existence of both notations, possibly because his treatise was on harmonic analysis, rather than a harmony textbook. Throughout the examples, however, only the numeric inversions are used.

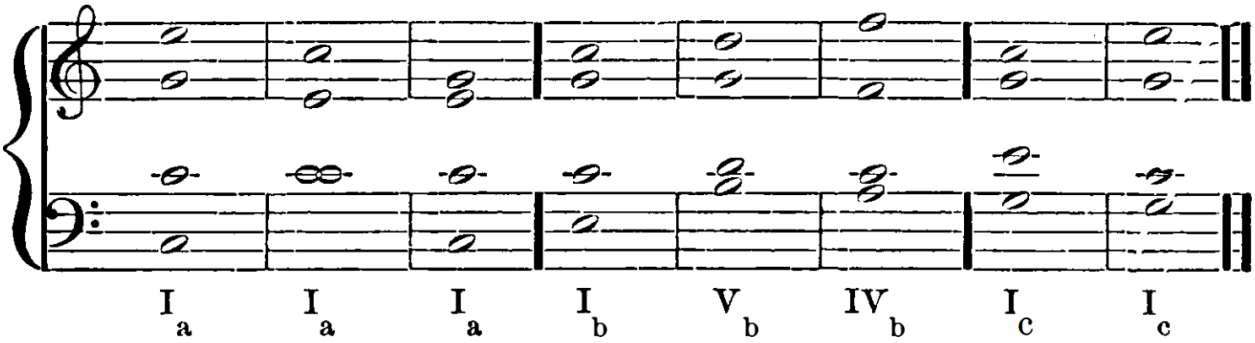


Figure 2.18: *York (1909, 19), where the inversion by letter notation is preferred.*

The notation for chord inversions based on letters is less common than the stacks of Arabic numerals. One place where the letter notation was preferred was York (1909), where it appears prominently. Figure 2.18 shows the introduction of the chord inversion notation used by York.

### 2.2.4 Neapolitans, Augmented Sixths, and other Special Chords

The stacks of Arabic numerals in Emery's example (Figure 2.11) describe **augmented sixth** chords. This is possibly the first time that the **RNA** notation is extended to describe chords that lie beyond the diatonic harmonies occurring in major and minor scales.[10] Shepard (1896)

10. Annotating special chords, such as a **Neapolitan** or **augmented sixth** chords is probably one of the most important characteristics of **RNA** to this day, as these chords are generally beyond the scope of traditional chord labels. Presumably, this is because identifying these chords requires an understanding of the key, an aspect that is generally neglected in traditional chord labels but intrinsically available in **RNA**.

adopted a similar notation for **augmented sixth** chords, as shown in Figure 2.19. Eventually, these chords would received their own symbols (**It**, **Fr**, and **Ger**) instead of Arabic numerals, although that syntax would only be established in the 20th century. Figure 2.20 shows a more modern example in Goldman (1965), which made use of the new symbols.

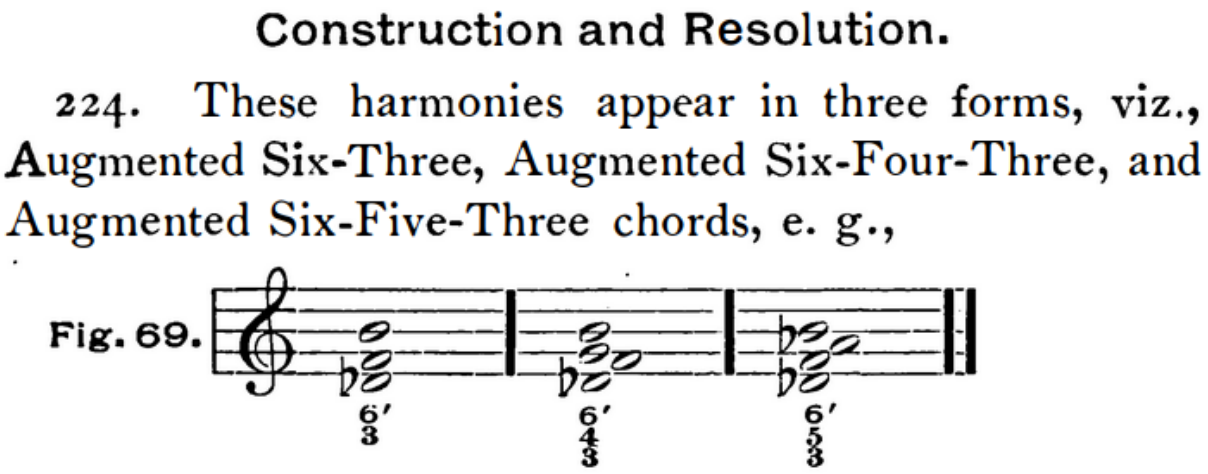


Figure 2.19: *Augmented sixth chords in Shepard (1896, 137).*

Emery also introduced a special symbol to annotate **Neapolitan** chords, **N.6**. Similarly to Emery (1879), Chadwick annotated **Neapolitan** chords using a special figure, **N**. Chadwick was explicit in displaying a **Neapolitan** as being in first inversion, which is its most common arrangement. Thus, the syntax, shown in Figure 2.21, featured the inversion figure "6" next to the letter "N", in the form of $\mathbf{N^6}$. Heacox (1907), Alchin (1921), and other sources adopted this syntax as well. Heacox (1907) notated different chord inversions of the **Neapolitan** using the corresponding stacks of Arabic numerals.

Another Roman numeral symbol that received attention in the late 19th century was the augmented triad. In the early syntax introduced in Weber (1818), there were no examples of augmented triads. Thus, the notation did not include a symbol for them. Richter (1860) was arguably the first source introducing the single-quote symbol to denote augmented triads (see Figure 2.8). This notation was adopted by several others, including Jadassohn (1883), Broekhoven (1889), Buwa (1893), and Shepard (1896). However, it is a fairly rare notation

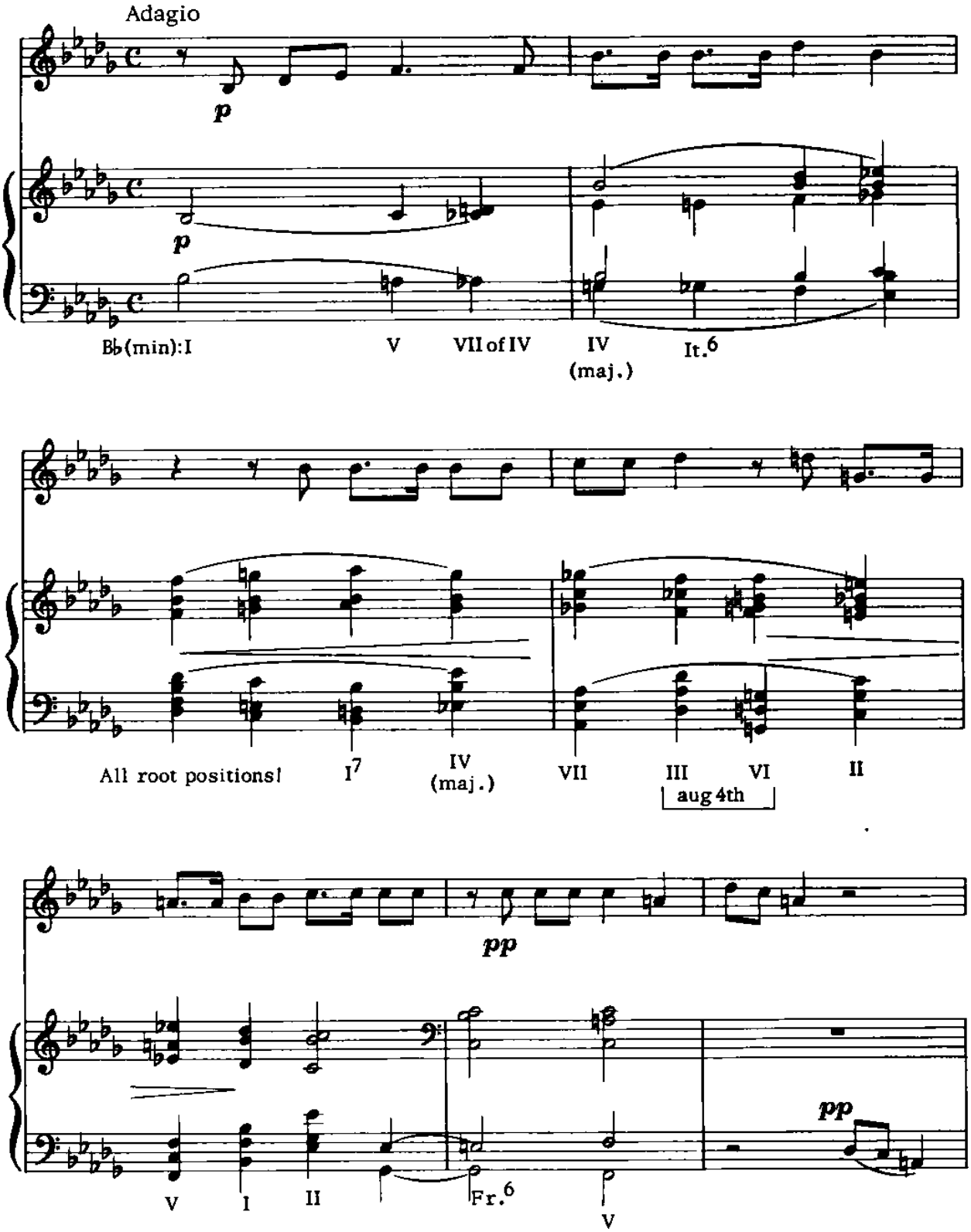


Figure 2.20: *Augmented sixth chords in Goldman (1965, 86) with the symbols **It.** and **Fr.***

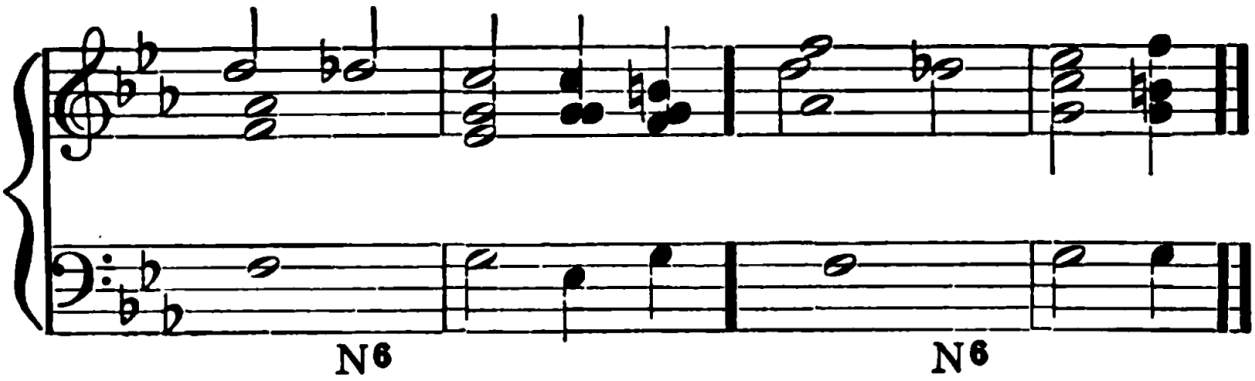


Figure 2.21: ***Neapolitan*** *in Chadwick (1897, 148).*

nowadays. Much more common is the use of **III+**, with a plus ("+") symbol in place of the single quote. This notation, shown in Figure 2.22, was perhaps popularized after Riemann (1890), as Riemann seems to be the first one introducing it. A few years later, Chadwick (1897) adopted it (see Figure 2.23), and others followed.

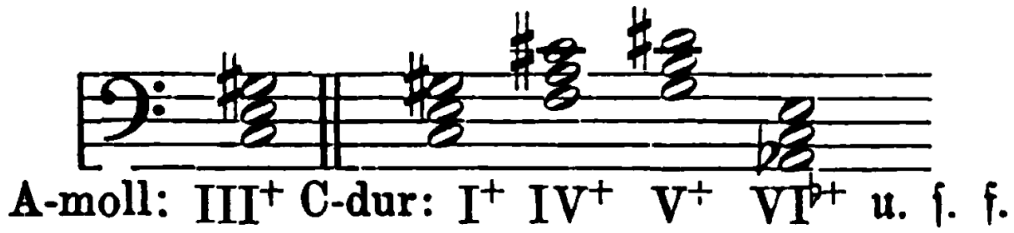


Figure 2.22: *Augmented triads in Riemann (1890, 64), indicated as* ***III+****.*

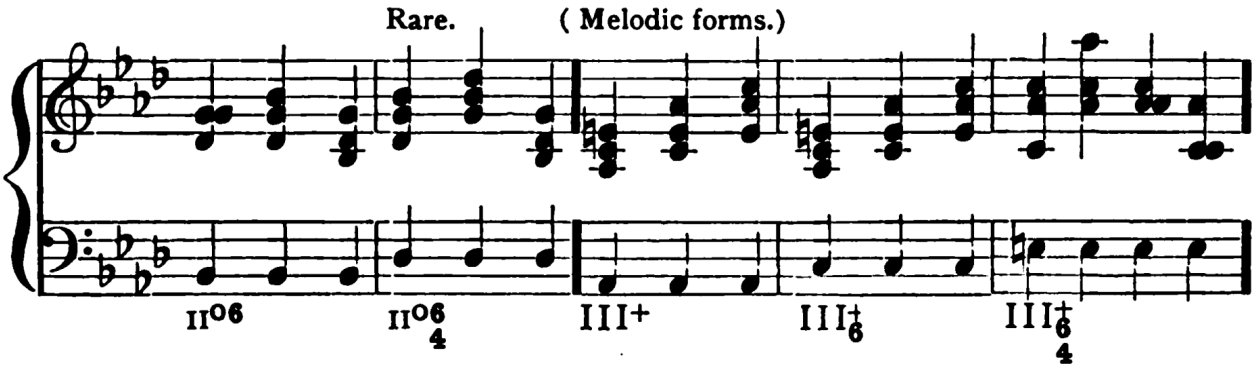


Figure 2.23: *The notation for augmented triads in Chadwick (1897, 53), which is similar to the one in Riemann (1890).*

### 2.2.5 Applied Chords and Tonicization

An important component of the modern RNA syntax is the one used for tonicizations. Tonicizations are slight deviations of key, which usually return to the original key, without "invoking" a modulation. Nowadays, tonicizations are generally written with a slash symbol, where the numerator indicates the scale degree and the denominator indicates the tonicized key. A common example is **V/V** or (dominant of the dominant). It is unclear where this notation originally emerged, however, a similar notation was introduced in Shepard (1889). Shepard's notation, which he calls "[A]ttendant chords," is presented as "**[A] of V**" in the notation, shown in Figure 2.24. This is analogous to the modern **V/V**. An advantage of the modern notation is that other degrees other than **V** (in the numerator) can be annotated using the same syntax.[11] In the explanations of his modulation method, Shepard also introduces a modulation "formula" that resembles the fraction-like notation we use today (see Figure 2.25).

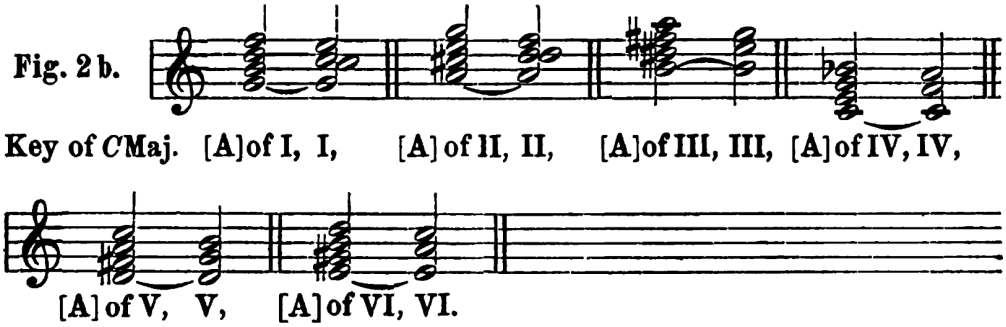


Figure 2.24: *Attendant chords in Shepard (1889, 5).*

Halm can be considered another pioneer of the **V/V** notation. He used a similar notation in Halm (1900), **V–V**, to refer to secondary dominant chords. However, the meaning of this notation is not very concise. For instance, it is used earlier in Halm (1900, II) as **I–V** and **I–IV** to denote chords that act as tonics in one key and as dominants in another. Yet, that notation would lead to a mistaken interpretation in the case of **V–V**, which should perhaps be written as ♯**II–V** using the same convention. Nevertheless, the notation is similar to the modern way of thinking about applied chords. An example of Halm's **V–V** notation is shown in Figure 2.26.

11. Piston (1941), in fact, used Shepard's notation in this way, using a syntax of the form "**V of V**".

**FORMULAE FOR MODULATION**

**From Any Key to Any Other.**

**By Means of**

**a) The Dominant Seventh Chord.** $\left\{\frac{\text{I}^{*)}}{\text{Old key}}, [\text{A}], \frac{\text{V}^7, \text{I}}{\text{New key}}.\right.$ **(See Chap. II.)**

**b) The Closing Formula.** $\left\{\frac{\text{I}}{\text{Old key}}, [\text{A}], \frac{\text{IV}, \overset{6}{\text{I}^4}, \text{V}^7, \text{I}}{\text{New key}}.\right.$ **(See Chap. III.)**

**c) The Diminished Seventh Chord.** $\left\{\frac{\text{I}}{\text{Old key}}, [\text{A}], \frac{\text{VII}^0{}_7, \text{I}}{\text{New key}}.\right.$ **(See Chap. IV.)**

Figure 2.25: *Modulation formula in Shepard (1889, 10).*

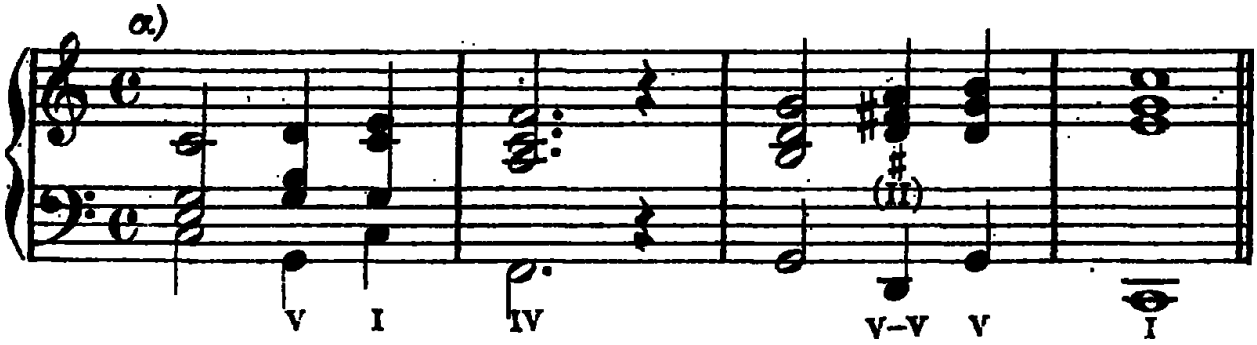


Figure 2.26: *Notation for secondary dominants in Halm (1900, III).*

Schenker displayed several notational conventions in *Neue Musikalische Theorien und Phantasien: Harmonielehre* (Schenker 1906). Throughout this book, pivot chords that have multiple meanings in different keys are usually expressed in Roman numerals. Schenker sometimes uses the notation **VI=IV**, and other times **VI/IV**. In both notations, the meaning seems to be "six becomes four." One of his specific examples can be seen in Figure 2.27.

An interesting notation for secondary dominants appears in White (1911), where it seems to be introduced particularly for diminished seventh chords. In this notation, secondary dominants are presented within parentheses, indicating the key they tonicize. It is clear that these changes of key have an effect within the scope of the parentheses, which typically affect a sin-

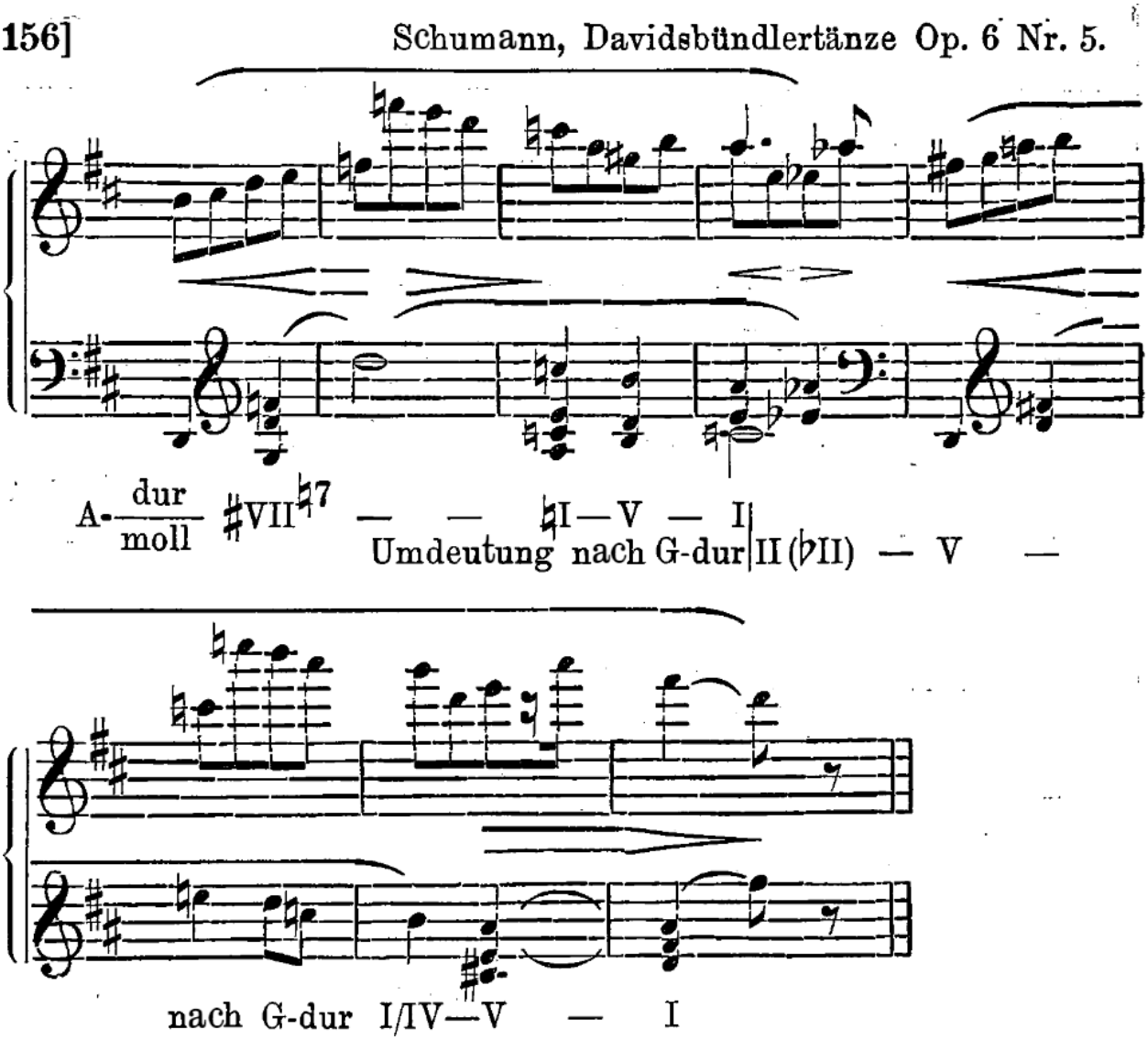


Figure 2.27: *Notation for applied chords in Schenker (1906, 190). Notice also what seems to be a typographical error in the book, where the second "nach G-dur" would make more sense as "nach D-dur".*

gle chord. This is consistent with a notation of *tonicizations* and *modulations*, as seen in recent works of computational music theory. An example of this notation is shown in Figure 2.28.

### 2.2.6 Summary of the Findings

In summary, the survey revealed the following findings:

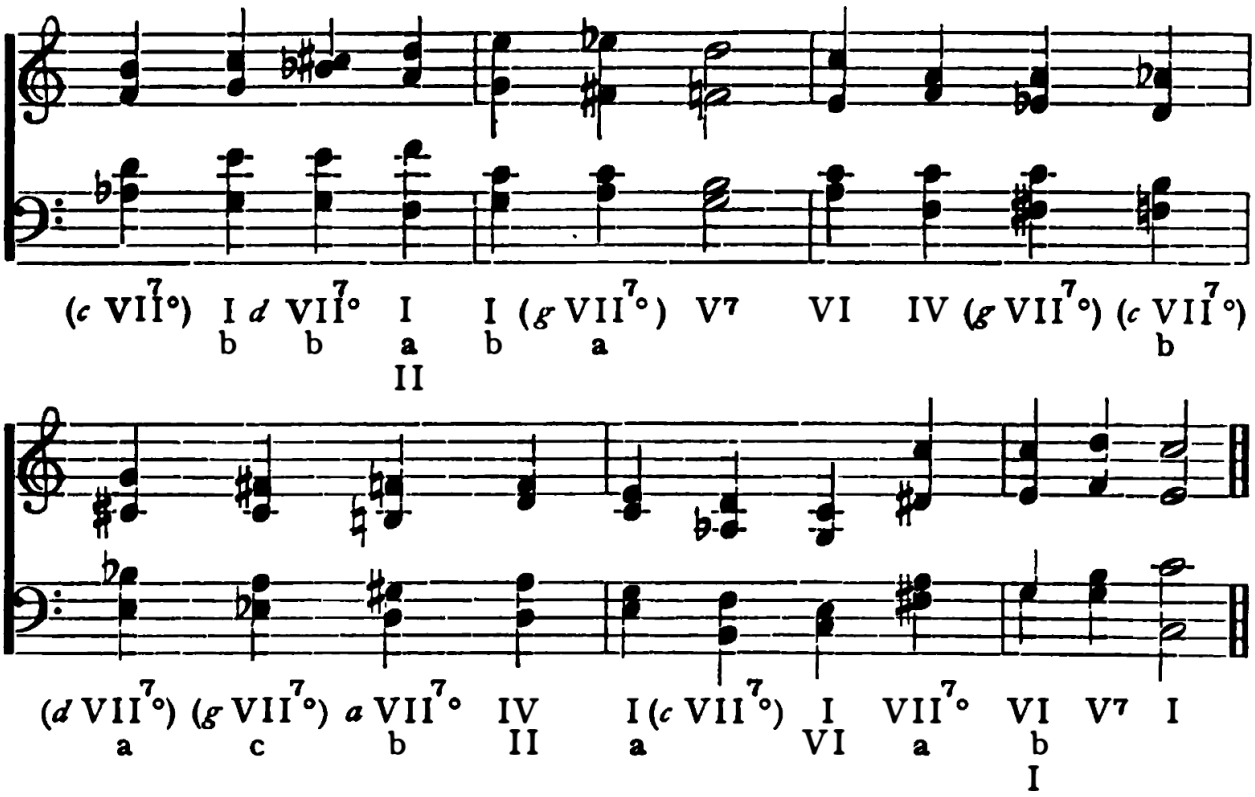


Figure 2.28: *Notation for key fluctuations, in parentheses, appearing in White (1911, 110). The keys within parentheses do not extend to any contiguous chords, affecting only the enclosed one. In this excerpt, the letters* a *and* b *represent a first or second inversion, respectively. Furthermore, some chords are indicated with two Roman numerals, these are interpretations in two different keys simultaneously* .

**The Weber Syntax.** The role of Weber (1818) in defining the basic characteristics of the notation was crucial.

**Adoption of the Syntax.** After Weber (1818), the Roman numeral annotation syntax slowly gained popularity among English and German theorists, achieving a more solid adoption by the end of the 19$^{th}$ century.

**Figured Bass and Roman Numeral Inversions.** The Roman numeral notation was often accompanied with figured bass. At first, Roman numerals did not indicate chord inversions. Later, the practice of interconnecting figured bass and Roman numeral annotations might have resulted in the notation used today to indicate chord inversions with Arabic numerals.

**Special Chords.** Special symbols were introduced for certain chords. Notably, Neapolitan chords received their own symbol ("**N**"). Sometimes "competing" syntaxes were used with the same meaning. For example, some authors preferred a single quote (') symbol to denote augmented triads, whereas others used a plus sign ("**+**").[12] An example of each of these notations is shown in Figures 2.8 and 2.22, respectively.

**The Syntax for Tonicizations.** The treatment of tonicizations was originally absent. Any change of key was notated with a colon preceded of a new reference key. Eventually, a few ways of notating common tonicizations (e.g., *dominant of the dominant*) emerged. These evolved into the slash syntax we use today, although it is unclear who should be credited for the slash notation.

**The Chord Vocabulary.** Recently, the vocabulary of chords has been extended, new additions consider, for example, the common-tone diminished seventh, a chord that is often written as $\mathbf{CT^{o7}}$. This also speaks of the continuing evolution of the syntax.

## 2.3 Standardization of Roman Numeral Annotations

Because the Roman numeral notation evolved in a rather unstructured way, there is no *standard* way of writing Roman numeral annotations. Some of the common variations include:

- Use of case-sensitive or case-insensitive numerals
- Different ways of denoting certain chords, notably, the so-called "cadential six-four" chord[13]
- Different notations for other chords (e.g., Neapolitan written as either $\flat\mathbf{II^6}$ or $\mathbf{N^6}$)
- Different ways of notating inversions (using the letters a–d or stacks of Arabic numerals)

12. Nowadays, the "plus" notation is more common.
13. Particularly, there is currently a heated debate about annotating these chords as $\mathbf{I^6_4}$ or $\mathbf{V^6_4}$.

A clear separation of chord qualities (major and minor) using case-sensitive Roman numerals has been proposed since Weber (1818). However, some analysts have preferred to indicate the scale degrees using an upper-case Roman numeral, regardless of the chord quality implied by the chord. This is often fine for human analysts, as the reader of the analysis will usually be able to complement the information. Computers will not be easily able to disambiguate this notation, however. In digital annotations, having explicit information about the chord quality is unequivocally better. Every digital representation considers Roman numerals to be case-sensitive for that reason.

A tonic triad in second inversion is a chord frequently used before a cadence. When it is used in this context (preceding a **V** chord) it receives the name of **cadential six-four**.[14] In order to indicate this chord using Roman numerals, it is common to write it as $\mathbf{I}^{6}_{4}$, $\mathbf{V}^{6}_{4}$, or $\mathbf{Cad}^{6}_{4}$. Out of these, the least ambiguous representation for a digital annotation is $\mathbf{Cad}^{6}_{4}$, as it disambiguates the chord as functioning in this specific way and not as a passing chord (e.g., a second-inversion tonic or dominant triads could be passing chords, which are neither considered a cadential six-four). Figure 2.29 exemplifies the suggested use of $\mathbf{I}^{6}_{4}$, $\mathbf{V}^{6}_{4}$, and $\mathbf{Cad}^{6}_{4}$ chords in machine-readable annotations, with all ambiguity removed.

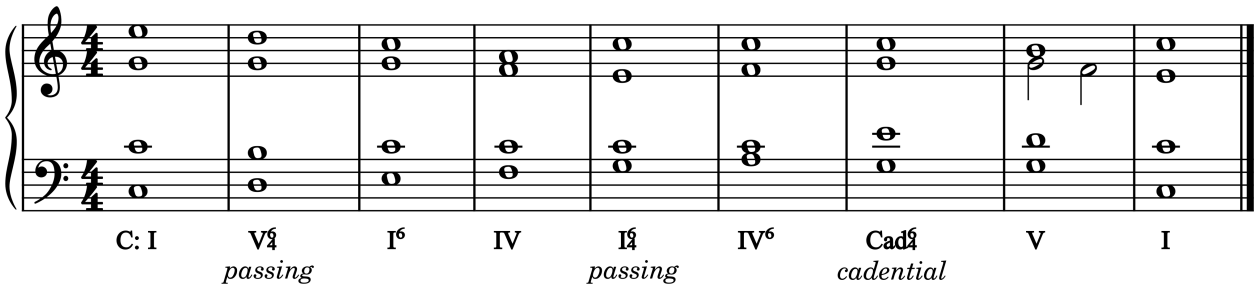


Figure 2.29: *Instances of the $\mathbf{V}^{6}_{4}$, $\mathbf{I}^{6}_{4}$, and $\mathbf{Cad}^{6}_{4}$ chords, as suggested for digital representations.*

**Neapolitan** chords are often considered a chromatic substitution of the subdominant chord. The root of the chord is the flattened second degree of the relevant major/minor key. For ex-

14. There is more to say about the role of **cadential six-four** chords in harmonic analysis and voice leading, which often puts them in the middle of heated music-theoretical debates. Two recent surveys on the topic include Mirka (2015) and Ninov (2016).

ample, the **Neapolitan** of **C major** (and **C minor**) is **D♭**. For this reason, it is either known as **♭II** or as **N**.

Inversions are generally notated using stacked Arabic numerals. The conventions for common triad and seventh chords are relatively standardized. These are indicated in Table A.4.

Table 2.2: *Notations for chord inversions using Arabic numerals or letters. Figures between square brackets are optional.*

| Chord type | Arabic numeral notation | Letter notation |
|---|---|---|
| Root position (triad) | | [a] |
| First inversion (triad) | 6 | b |
| Second inversion (triad) | 64 | c |
| Root position (seventh) | 7 | 7[a] |
| First inversion (seventh) | 65 | 7b |
| Second inversion (seventh) | 43 | 7c |
| Third inversion (seventh) | [4]2 | 7d |

Inversions have also been indicated with letters. This notation is common since the 19$^{th}$ century. It is also the notation of the first digital standard for **RNA**, ****harm**.

In the music theory classroom setting, the flexibility of the notation is perhaps a desirable goal. Students may be encouraged to develop their own "style" of tonal analysis, incorporating aspects of voice-leading, motivic, and key analysis, as they see fit. This is useful to extend the intrinsic limitations of **RNA** to summarize tonal music.

Excessive flexibility may be an issue, however, in computational work. Idiosyncratic and undocumented methods of analysis lead to incompatible annotations. It is unfeasible to assume that a single person will be able to annotate a sufficiently large number of Roman numeral analyses to be used for computational models. Thus, the compatibility between different annotations is a necessity, as well as the cooperation between expert analysts. This can only be achieved by standardizing the **RNA** notation. Several notational systems have emerged over the years to attempt to solve this problem.

In the rest of this section, I will review four efforts toward the standardization and digitization of **RNA** labels.

### 2.3.1 Digital Representation of Roman Numerals

There are several methods to encode **RNA** annotations in a digital representation.

#### 2.3.1.1 Humdrum(**harm)

The ****harm** syntax was proposed by Huron as part of the Humdrum(****kern**) format (Huron 1994). This syntax includes a comprehensive notation for triads, seventh chords, special chords, and descriptive (arbitrary) chords using Roman numeral notation. The notation is well documented, although several aspects of the notation are missing from the explanations.

The original ****harm** documentation is available online.[15] The notation summarizes the description of chords based on four attributes: (1) chord root, (2) chord type, (3) inversion, and (4) chord alterations. Chord roots are indicated with Roman numerals. Chord types (or qualities) are indicated with a case-sensitive notation of the Roman numeral plus the symbols '**+**' and '**o**' indicating augmented and diminished triads, respectively. All chord inversions are indicated with the letters *a*, *b*, *c*, and *d*. These letters indicate root position, first inversion, second inversion, and third inversion, respectively. The alterations, indicated with '**-**' and '**#**', alter the pitch expressed by the Roman numeral scale degree. This is useful, for example, when denoting a chord root that is non-diatonic to the current key.

#### 2.3.1.2 RomanText

This notation was first proposed in Gotham, Tymoczko, and Cuthbert (2019). An utilitarian aspect of this notation is that it is implemented as a module inside the popular *music21* Python library (Cuthbert and Ariza 2010).[16]

**RomanText** is intended as a stand-alone format for **RNA**, meaning it does not require the musical score. As part of the plain-text Roman numeral annotations, the location of the annotations (measure and beat) is a compulsory attribute of the annotation.

15. https://www.humdrum.org/Humdrum/representations/harm.rep.html
16. https://web.mit.edu/music21/doc/moduleReference/moduleRoman.html

Compared to the **harm notation, a clear advantage is that the software implementation facilitates a vast number of operations with the Roman numeral annotations. For example:

- A RomanText annotation file can be realized into a score with block chords
- The Roman numeral annotations can be transformed into chord labels, pitch class sets, or other representations relatively easily
- Easy access is provided to the secondary degrees, secondary keys, and other elements implicitly encoded in the annotation

All the inversions in RomanText are encoded through numbers, whereas the **harm syntax defines inversions by letter exactly to disambiguate these scenarios. Some disadvantages of the numeric notation for inversions preferred in RomanText is that it does not allow an intuitive way to encode inversions for extended $9^{th}$, $11^{th}$, and $13^{th}$ chords.

#### 2.3.1.3 The DCML Standard

This standard for harmonic annotations was developed at the Digital and Cognitive Musicology Lab (DCML) at the École Polytechnique Fédérale de Lausanne.

This standard was first used in the annotation of the Annotated Beethoven Corpus (ABC) dataset introduced by Neuwirth et al. (2018). Since then, it has been revised and documented in a dedicated repository[17] and a reference manual.[18] The introduction section of a tutorial for this annotation standard summarizes the philosophy behind the standard:[19]

> *The goal of this tutorial is to provide a systematic and condensed way of conveying the annotation philosophy and principles (aka "the guidelines") behind the annotation standard. What has been condensed are more than two years of discussions between*

17. https://github.com/DCMLab/standards
18. https://dcmlab.github.io/standards/build/html/reference/reference.html
19. https://dcmlab.github.io/standards/build/html/tutorial/index.html

*the music theory experts involved in the standard's creation and application, i.e. between annotators, reviewers, and users. Therefore, this tutorial is not about telling anybody "how harmonic analysis really works" or "how everyone should be using Roman numerals". Instead, it introduces a set of guidelines for analysts who want to use the DCML harmony annotation standard to encode a set of musical features in a consistent and machine-readable manner so that others can re-use and rely on the encoded information.*

In addition to the repository, reference manual, and tutorial mentioned above, the community behind this standard has also made available documentation regarding version control practices,[20] instructions for corpus creation (described by **DCML** as the corpus-creation pipeline),[21] and a page of questions and examples,[22] among other resources.

From the perspective of documentation and standardization, this is perhaps the most thoroughly described standard for **RNA** in existence today. For example, it offers specific examples to annotate voice-leading,[23] which is often a problematic type of annotation in digital representations. The revised version of the standard has been used in a subsequent iteration of the **ABC** dataset (version 2) as well as the **Mozart Piano Sonatas (MPS)** dataset (Hentschel, Neuwirth, and Rohrmeier 2021). Other collections exist in a private archive and are expected to be publicly available one day.

#### 2.3.1.4 Harmalysis

*Harmalysis* is an **RNA** grammar introduced in Nápoles López and Fujinaga (2020a).

The *harmalysis* grammar is based mainly on Huron's ****harm** syntax (see Section 2.3.1.1). The syntax is extended by borrowing elements from the **RomanText** format, such as Arabic numeral inversions (see Section 2.3.1.2), music notation fonts for **RNA**,[24] and conventions

20. https://dcmlab.github.io/standards/build/html/git/git.html
21. https://dcmlab.github.io/standards/build/html/pipeline/pipeline.html
22. https://dcmlab.github.io/standards/build/html/reference/examples.html
23. https://dcmlab.github.io/standards/build/html/tutorial/counterpoint.html?highlight=voice+leading
24. https://github.com/MarcSabatella/Campania

observed in existing datasets of **RNA**. As a result, the *harmalysis* syntax is a superset of the ****harm** syntax, which includes additional features and supports a wider range of customs of harmonic analysis.

The main advantage of this syntax is that it is formally defined as a context-free grammar. Thus, the exact mathematical validation of the syntax is available for review and discussion, which might not be the case for representations defined by their textual explanations.

# Chapter 3

# Background

In this chapter, I introduce the previous work on music technology, machine learning, and **Music Information Retrieval (MIR)**, that is relevant for **Automatic Roman Numeral Analysis (ARNA)**. Section 3.1 introduces the idea of music representation and symbolic music formats, which are the input to an **ARNA** system. Section 3.2 briefly introduces the concepts of supervised learning and common neural network architectures that are used in symbolic music analysis problems. Section 3.3 introduces the existing **MIR** techniques for **ARNA** and related problems: key estimation, automatic chord recognition, and pitch spelling.

## 3.1 Music Representation

The musical information of interest requires a digital representation before any **MIR** research can be done. According to Müller (2015, 1), there are generally three types of music representations: digital sheet music images, symbolic music representations, and audio representations.

Digital sheet music images consist of the digital version of printed musical scores. Image representations are useful to distribute musical scores among musicians, or to print them on paper. However, accessing the musical content of the scores (e.g., note names, durations, key signatures, or time signatures) is quite a difficult task, which usually involves the devel-

opment of complex **Optical Music Recognition (OMR)** systems (Calvo-Zaragoza, Hajič Jr., and Pacha 2020).

Symbolic music representations also often refer to digital representations of sheet music. The main difference is that symbolic representations are encoded in machine-readable formats, where the musical content is readily available for computational analysis. Examples of such representations include ****kern**, Lilypond, **Music Encoding Initiative (MEI)**, **Musical Instrument Digital Interface (MIDI)**, and **MusicXML**.

Audio representations refer to digital representations of acoustic sound waves. These representations are popular in digital media, because they more closely resemble the musical experience that most users want to consume. For example, as a musical performance streamed using a music-streaming service. Many **MIR** tasks operate on audio data because of the importance of audio representations in the daily experience of music.

An **ARNA** algorithm is closely related to a few "satellite" tasks: **Automatic Chord Recognition (ACR)**, automatic key estimation, and **MIDI** pitch spelling. Over the years, there have been numerous proposed chord and key estimation algorithms for symbolic and audio music representations. These representations encode music in different ways (music notation and acoustic signals, respectively) and do not contain the same information nor operate at the same semantic level. Usually, an algorithm focuses in a single type of digital music representation but algorithms that operate in both representations are possible.[1]

For the most part, **ARNA** models operate with symbolic music data. Thus, Section 3.1.1 presents the characteristics of some of the most common symbolic music formats.

### 3.1.1 Symbolic Music Formats

Müller (2015, 1) defines symbolic music representations as:

1. See, for example, our symbolic-and-audio key-estimation model in Nápoles López, Arthur, and Fujinaga (2019).

> *[Symbolic representations] refer to any machine-readable data format that explicitly represents musical entities. These musical entities may range from timed note events, as is the case of **MIDI** files, to graphical shapes with attached musical meaning, as is the case of music engraving systems.*

The musical entities often correspond to notes, rests, time signatures, beams, accidentals, slurs, and other musical symbols. For the same reason that "musical symbols" can be as diverse as timed note events or graphical shapes with musical meaning, there is also a wide range of symbolic music formats. Some formats focus more on the engraving aspect of music (e.g., Lilypond or **MEI**), whereas others focus more on the performative aspect of music (e.g., **MIDI**).

Depending on the **MIR** problem of interest, some of these symbols might be more useful than others. For example, most **MIR** systems for chord and key recognition require pitches and durations but not beams or slurs. Luckily, most symbolic music formats are able to encode the musical symbols that are useful in **ARNA** systems.[2] There are important differences, nevertheless.

#### 3.1.1.1 Humdrum(**kern)

**Origin of **kern.** The Humdrum toolkit is a family of forty-two data representations,[3] from which the ****kern** representation for musical scores is the most widely used. The toolkit was developed in the early 1990s and first thoroughly described by Huron (1994) in the *Humdrum Toolkit: Reference Manual*. As it often happens, the same word (in this case, "Humdrum"), can refer to multiple things:

- Humdrum is a meta-format or family of representations, which can encode diverse types of data, such as musical scores (****kern**), harmonic analysis annotations (****harm**), or lyrics (****text**).

2. See the list of digital **Roman Numeral Analysis (RNA)** standards in Section 2.3.1.
3. The full list is available in: https://www.humdrum.org/rep/

- Humdrum is the name of a collection of software tools, also known as the Humdrum toolkit. These tools are originally scripts in the *Perl* language, which have now been extended with other tools written in the *C/C++* language, *Humdrum Extra*[4] and *Humlib*.[5]

- Humdrum is also used to refer to a file containing data encoded in a Humdrum representation. For example, the Humdrum files in the *KernScores* library described by Sapp (2005).

**Applications of **kern.** The **kern symbolic music representation is very compact, and tailored toward music analysis. The content can be typed by a human encoder and analyzed with the toolkit scripts. An extensive list of applications is presented by Sapp (2011).

**Humdrum in Roman Numeral Analysis.** Humdrum is an extremely relevant format for RNA for two reasons:

1. It already provides support for RNA using the **harm syntax, which is also the first digital representation for RNA annotations.

2. Historically, the first end-to-end system for ARNA (see Section 3.3.4) was developed to operate with Humdrum(**kern) scores, generating automatic Humdrum(**harm) annotations for them.

#### 3.1.1.2 MEI

The MEI is described by Hankinson, Roland, and Fujinaga (2011, 293) as a "community-driven effort to define guidelines for encoding musical documents in a machine-readable structure." As part of those guidelines, a family of music formats have been developed over the years. In its flavor for common Western music notation, the MEI format describes musical

4. https://github.com/craigsapp/humextra
5. https://github.com/craigsapp/humlib

content using an **Extensible Markup Language (XML)**-based syntax, which emphasizes musical semantics.

**Origin of MEI.** The **MEI** community (and format) was first presented by Roland (2002), where it proposed to replicate the efforts of the **Text Encoding Initiative (TEI)** in the musical domain. Starting in 2013, an annual conference[6] organized by the **MEI** community has perpetuated the involvement of different people in the development of new guidelines for music encoding (Crawford and Lewis 2016).

**Applications of MEI.** The **MEI** format has gained popularity in digital collections and libraries. Thanks to tools like Verovio, the **MEI** music engraver by Pugin, Zitellini, and Roland (2014), several music libraries have utilized the **MEI** format to encode their collections.

**MEI in Roman Numeral Analysis.** Although there is no specific **RNA** recommendation for **MEI** files, the **MEI** standard supports a generic `<harm>`[7] element, which can be used to annotate Roman numeral annotations.

#### 3.1.1.3 MIDI

**Origin of MIDI.** The **MIDI** format is a specification to transmit musical information among hardware devices and software applications. It was originally presented as the *Universal Synthesizer Interface* by Smith and Wood (1981). The specification was designed to interconnect different synthesizers, facilitating compatibility among manufacturers and increasing sales. By the end of 1982, several other manufacturers joined the initiative, and the specification was renamed to **MIDI** (Moog 1986).

**Applications of MIDI.** **MIDI** is an ubiquitous format for digital musical instruments. It is mostly designed to capture a musical performance, for example, transmitting the pressed keys

6. https://music-encoding.org/conference/
7. https://music-encoding.org/guidelines/v4/elements/harm.html

(and the velocity at which they were pressed) of a keyboard controller into a **Digital Audio Workstation (DAW)**.

Because it is designed for musical performance, it can be generated in real-time and transmitted across hardware and software applications. For the same reason, it is also a very limited format, generally unable to capture music notation information, such as ties, note durations, pitch spelling, or phrasing. This limitation is usually important in automatic analysis models that rely on more information than pitch class and octave.

**MIDI in Roman Numeral Analysis.** Compared to other formats, such as ****kern**, **MusicXML**, or **MEI**, **MIDI** is a relatively inconvenient format for any **RNA** workflow. For example, as an input, it does not provide reliable pitch spelling information, which is crucial in recent **ARNA** models.

#### 3.1.1.4 MusicXML

**Origin of MusicXML.** **MusicXML** is an **XML**-based format for representing common Western music notation, introduced by Good (2001). It is designed mostly as an exchange format between music notation software. However, it is also useful for other applications, such as music analysis and retrieval. Originally, the **MusicXML** standard was heavily inspired by the MuseData and Humdrum (see Section 3.1.1.1) representations, adapting them into an **XML** context.

**Applications of MusicXML.** Because **MusicXML** is designed to represent music of the seventeenth century onwards, it has gained popularity as the standard exchange format among commercial applications, such as Dorico,[8] Finale,[9] MuseScore,[10] Sibelius,[11] and others.

8. https://www.steinberg.net/dorico/
9. https://www.finalemusic.com/
10. https://musescore.org/en
11. https://www.avid.com/sibelius

**MusicXML in Roman Numeral Analysis.** As of version 4.0 of the MusicXML format, there is a dedicated `<numeral>` element[12] to encode RNA annotations.

## 3.2 Deep Neural Networks

In recent years, deep neural networks have become an ubiquitous technology to approach perceptual problems. This has also been the case for music, where many music analysis problems are tackled using neural networks nowadays.

This section discusses supervised learning, the methodology used throughout the experiments in this dissertation, as well as convolutional, recurrent, and transformer network architectures, which are common in ARNA models.

### 3.2.1 Supervised Learning

Goodfellow, Bengio, and Courville (2016, 140) describe supervised learning algorithms as

> *algorithms that learn to associate some input with some output, given a training set of examples of inputs* $\boldsymbol{x}$ *and outputs* $\boldsymbol{y}$.

Commonly, the outputs are provided by a human annotator ("supervisor"), which in the case of RNA annotations, refers to the expert annotators providing the analyses.

### 3.2.2 Feed Forward Networks

An Artificial Neural Network (ANN) is a machine learning algorithm that models arbitrary functions by automatically learning *weights* (also known as *parameters*) connecting the different nodes of the neural network. Generally, a nonlinear activation function is applied to such weights, introducing a nonlinear behavior in the neural network that allows it to learn functions of higher complexity, which a linear model could not possibly learn. The learning

12. https://www.w3.org/2021/06/musicxml40/musicxml-reference/elements/numeral/

of the weights is not achieved by programming task-specific rules but instead, by identifying simple characteristics of the training examples and extending them into more complex, more abstract characteristics through the *backpropagation* algorithm. The process of decomposing an example into a combination of simpler features is known as *representation learning*, and it is one of the main ideas that differentiate an **ANN** from other classes of machine learning.

The study of **ANNs** started around the 1940s, and it has been known through different names throughout the years (Goodfellow, Bengio, and Courville 2016). The beginnings of **ANNs** can be traced back to the 1940s, when a bio-inspired *neuron* model was introduced (McCulloch and Pitts 1943). This neuron allowed to model very simple functions by manually setting the weights that connected the input into the neuron. This idea was later extended to propose the *Perceptron* (Rosenblatt 1958) and *Adaline* (Widrow and Hoff 1960) models, which were able to automatically learn the weights from the data. Although promising, these models lost popularity when it was demonstrated that they could not learn relatively simple functions, like the *XOR* function (Minsky and Papert 1972). This wave of research (1940–1960) is often referred as the *cybernetics* wave of neural networks research (Goodfellow, Bengio, and Courville 2016).

Following the wave of cybernetics, another wave extended from 1980 to 1990, colloquially known as *connectionism*. During the work of the "connectionists," the research community benefited from the development of the current form of the *backpropagation* algorithm (Rumelhart, Hinton, Williams, et al. 1988). The backpropagation algorithm became (and remains) an elemental process in the training of neural networks, which allows to propagate the error throughout the network by making use of the *chain rule*. Finding the derivatives of each parameter in the network, the values of such parameters can be updated in the "right direction" (against the gradient) to decrease the error in the next batch of training examples. This facilitates the automatic training of large and complicated neural networks, with a variety of layers, neurons, and nonlinear activation functions. Even though this and other improvements made neural networks a promising area of research, they were still very difficult to train in practice

(partly due to the difficulty of finding a good initialization of the weights) and were typically outperformed by domain-knowledge techniques. This resulted in many scientists losing interest in the technology until the next wave of research.

Finally, a third wave of research started around 2006, when new methods for training neural networks were introduced (Hinton, Osindero, and Teh 2006). These new methods not only facilitated the training of neural networks but the training of much larger neural networks. The interest in such larger architectures extended, and in a historical evaluation of the *ImageNet* dataset (Deng et al. 2009), in 2012, a neural network by Krizhevsky, Sutskever, and Hinton (2012) outperformed the most sophisticated methods of computer vision. This had a momentous impact in the way that neural networks were perceived by the research community, and motivated their application into different problems and fields of study. We know this last wave of research as *deep learning*, and it is currently an active and growing wave of research across many fields. Around this umbrella term of *deep learning*, many state-of-the-art machine learning techniques have been developed and continue to be improved. Among the landscape of existing technologies nowadays, Convolutional Neural Networks, Recurrent Neural Networks, and Transformer networks have been applied to **ARNA**.

### 3.2.3 Convolutional Neural Networks

**Convolutional Neural Networks (CNNs)** were introduced during the *connectionist* wave of research on neural networks, in LeCun (1989) and LeCun et al. (1989). An important innovation of **CNNs** is the idea of *shared parameters*. In a traditional feedforward network, for example, a **Multilayer Perceptron (MLP)**, every neuron of the network is typically connected to another neuron of the following layer using a unique parameter (i.e., a parameter used exclusively for connecting these two neurons). This gives the network more expressive power and the capability of modelling very complex functions. However, it also produces a combinatorial explosion of parameters as the neural network grows in number of layers and neurons. Training large networks with many unique parameters may be unfeasible (or even

impossible) due to the limitations in memory and computing power of modern technology. **CNNs**, by design, reuse parameters throughout the network. Thus, reducing the number of parameters compared to a fully-connected, feedforward network.

Sharing parameters is not only important for reducing the training time of the network, it is a bio-inspired design motivated by the mechanics of the visual system. It is customary, for example, to refer to the collection of neurons that make use of the same parameter as the *receptive field* of the parameter, a term taken from neurophysiology.

The shared parameters are modelled through a *kernel* vector. The kernel vector multiplies the inputs of the neural network layer in a way that resembles the mathematical operation of *convolution*, which motivated the use of the term *Convolutional Neural Networks*. After training, it is assumed that each of those kernels will learn a low-level, localized, feature, which is going to be searched across the entire input vector of the network and propagated into deeper (higher-level) kernels of the network.

Given the way that convolutional kernels work, **CNNs** have been a useful technique for dealing with fixed-length, grid-like structures (e.g., images), producing positive results in many tasks. For example, they were the technology behind every state-of-the-art *ImageNet* model from 2012 to 2020 (Krizhevsky, Sutskever, and Hinton 2012), identifying over 1,000 classes of objects in an image.

In **MIR**, **CNNs** have been used for genre recognition (Dieleman, Brakel, and Schrauwen 2011), chord recognition (Humphrey and Bello 2012), structural analysis (Ullrich, Schlüter, and Grill 2014; Grill and Schlüter 2015), music tagging (Choi, Fazekas, and Sandler 2016), instrument recognition (Lostanlen and Cella 2016), **OMR** (Calvo-Zaragoza, Valero-Mas, and Pertusa 2017; Pacha and Calvo-Zaragoza 2018), beat tracking (Gkiokas and Katsouros 2017), source separation (Miron, Janer, and Gómez 2017), syllable segmentation (Pons, Gong, and Serra 2017), key detection (Korzeniowski and Widmer 2018), and tempo estimation (Schreiber and Müller 2018, 2019).

Throughout the years, researchers have proposed different modifications to **CNNs**, which have improved the suitability of these models to certain problems and scenarios.

#### 3.2.3.1 1D, 2D, and 3D Convolutions

Originally, **CNNs** were proposed to deal with images, which are encoded in a two-dimensional grid of pixels. However, one-dimensional **CNNs** and three-dimensional architectures now exist, which can be applied to, for example, timeseries (1D tensors) and video data (3D tensors), respectively.

#### 3.2.3.2 Residual Connections

As researchers scaled the depth of **CNN** models, they noticed that it was increasingly difficult to train the larger models effectively. One strategy that helped to mitigate this problem was the use of *residual* connections. A residual connection consists of a connection between a layer of the network and a deeper one, "skipping" some layers in between (i.e., the two layers are not contiguous). This pattern of connections has allowed researchers to design much deeper networks more effectively, as the residual connections strengthen the signal of earlier representations in deeper layers of the network. One of the most well-known models of this kind is perhaps the *ResNet* by He et al. (2016).

### 3.2.4 Recurrent Neural Networks

**Recurrent Neural Networks (RNNs)** are a type of neural networks designed to deal with sequential data. Unlike most other neural network architectures, **RNNs** do not assume that inputs are independent from each other. Instead, they update their parameters considering not only the current input to the network but also the previous inputs processed by the network. Inputs are thus arranged as *sequences* of inputs.

**RNNs** were introduced after the backpropagation algorithm (Rumelhart, Hinton, Williams, et al. 1988) was extended into the **Backpropagation Through Time (BPTT)** algorithm,

around 1988 (Werbos 1988, 1990). Nevertheless, the difficulty of training such **RNN** architectures made them unfeasible in practical applications before the invention of the **Long Short-Term Memory (LSTM)** architecture (Hochreiter and Schmidhuber 1997). They were popularized during the *deep learning* wave of research and, since then, applied to many tasks involving sequential data.

Throughout the years, different strategies have been proposed to design **RNNs**, for example, connecting the output of one time step into the next time step (Jordan **RNN**), connecting the hidden state of one time step to the next (Elman **RNN**), and training the network in both directions (Schuster and Paliwal 1997). It is common to see several of these techniques combined in a single architecture, for example, the **Convolutional Bidirectional Long Short-Term Memory (CBLSTM)** by Vogl et al. (2017).

In **MIR**, **RNNs** and hybrid **Convolutional Recurrent Neural Networks (CRNNs)** have been applied in numerous tasks. For example, onset detection (Eyben et al. 2010), chord recognition (Boulanger-Lewandowski, Bengio, and Vincent 2013; Sigtia, Benetos, and Dixon 2016; Sears, Korzeniowski, and Widmer 2018), voice separation (P.-S. Huang et al. 2014), music transcription (Sigtia et al. 2014), tempo estimation (Böck, Krebs, and Widmer 2015), beat and downbeat tracking (Böck, Krebs, and Widmer 2016; Krebs et al. 2016), music generation (Liu and Randall 2016; Liang et al. 2017; Lim, Rhyu, and Lee 2017), music transcription (Rigaud and Radenen 2016; Sigtia, Benetos, and Dixon 2016; Southall, Stables, and Hockman 2016; Vogl, Dorfer, and Knees 2016; Southall, Stables, and Hockman 2017; Vogl et al. 2017; Basaran, Essid, and Peeters 2018), **OMR** (Calvo-Zaragoza, Vigliensoni, and Fujinaga 2017; Wel and Ullrich 2017; Calvo-Zaragoza and Rizo 2018), sequence modelling (Ycart and Benetos 2017), mood detection (Delbouys et al. 2018), and instrument recognition (Gururani, Summers, and Lerch 2018).

#### 3.2.4.1 LSTM

The **Long Short-Term Memory (LSTM)** architecture was introduced by Hochreiter and Schmidhuber (1997). Since its introduction, it has been increasingly used to approach multiple problems. **LSTM** architectures extend the basic structure of **RNNs** and it is relatively safe to assume that any problem that involves an **RNN** can substitute the **RNN** with an **LSTM**, matching (or often improving) the results obtained by the **RNN**.

The main contribution of **LSTMs** is to provide a solution to **vanishing gradients**. Given that vanishing gradients are associated with the loss of long-term dependencies, mitigating this issue would enable, in theory, the capability of the network to learn long-term dependencies between the inputs. In order to achieve these long-term dependencies, an **LSTM** substitutes the regular **RNN** unit with a more complex one. This new, more complex unit, receives the name of an *LSTM unit*.

The main difference between an **RNN** and **LSTM** units is that the **LSTM** adds a cell and three gates: the "forget" gate, the "input" gate, and the "output" gate. In conjunction, the cell and gates allow the network to control the flow of the information, selecting what information should the network store and what information should it forget. The gates of the **LSTM** unit act as "regulators" and each of them is responsible of a different part of the network. The forget gate controls what information of previous time steps in the network should be kept and what should be forgotten. The input gate regulates how much of the new inputs to the network should enter the cell. The output gate controls how much of previous time steps in the network should be used for computing the activation of the output of the network.

#### 3.2.4.2 GRU

The **Gated Recurrent Unit (GRU)** is a more recent recurrent architecture introduced by Cho et al. (2014). It is simpler than the **LSTM** and, similarly, can generally be used as a substitute of **RNN** or **LSTM** layers with similar (or better) results.

Beside the **LSTMs** and **GRUs**, further improvements have also been proposed to **RNNs**. For example, during the same year that the **LSTM** was proposed, another paper proposed a bidirectional **RNN** that could be used in *offline* systems (as it requires knowledge of future timesteps, it cannot be used in *realtime* applications). These advances have improved the capabilities of **RNNs**, as they are often not mutually exclusive and can be used in combination. Therefore, it is common to see nowadays architectures that combine these approaches, for example, **Bidirectional Long Short-Term Memory (BLSTM)** networks.[13] These type of models have been successful and provided state-of-the-art in multiple tasks. Given that **RNNs** and more specifically **LSTMs** and **GRUs** are useful architectures for modelling sequence-to-sequence processes, they can be useful for **MIR** applications.

### 3.2.5 Transformer Networks

The Transformer network is a sequential model introduced by Vaswani et al. (2017, 2). The Transformer differs from other sequential architectures (e.g., **RNN**) because it "relies entirely on an attention mechanism to draw global dependencies between input and output." The attention mechanism is a method designed to mitigate the limitations of previous sequence-to-sequence models. In a sequence-to-sequence model, the performance degrades as the length of the sequences increases, because the entire sequence is represented with a single vector in the encoder. Using attention, information from all timesteps in the sequence is available to the decoder. This allows the model to "attend" or give a higher weight to certain timesteps of the sequence. Allowing the model to attend certain parts of the sequence has been an important advancement in recent deep learning models. The Transformer architecture became a fundamental methodology in natural language processing after it reached state-of-the-art performance in comparison to **RNNs**. Nowadays, this architecture has been widely adopted for different problems, including **RNA**.[14]

13. This type of architecture has been applied to **ARNA** in the work by Chen and Su (2018).
14. See, for example, Chen and Su (2021).

## 3.3 Music Information Retrieval

This section presents a review of **MIR** models for tonal music analysis. Section Section 3.3.1 focuses on key estimation, Section 3.3.2 on chord recognition, Section 3.3.3 on pitch spelling, and Section 3.3.4 on **ARNA**.

### 3.3.1 Key Estimation

Identifying the musical key is a fundamental task in the analysis of tonal music. It is often a preliminary or concurrent step to other common musicological tasks like harmonic analysis and cadence detection. In particular, the knowledge of the musical key can help a music analyst to find boundaries in a musical piece, interpret the role of notes and chords, or suggest a musical form to which the analyzed piece conforms. Due to its importance, key estimation is a well-studied research topic in **MIR**, and multiple key-analysis algorithms have emerged during the last decades. Broadly, there are two types of key-estimation algorithms: those that find the *main key* of the piece (hereafter **Global-Key Estimation (GKE)** models) and those that find the changes of key *within* the piece (hereafter **Local-Key Estimation (LKE)** models). The global key is related with the key of the piece or work, for example, as written in the title of the score. A local key is related to the music-theoretical concepts that explain changes of keys, such as *modulations* and *tonicizations*.[15] One caveat is that the terms related to changes of keys are often loosely defined, which complicates the precise definition of a local key. This is further discussed in Section 3.3.1.2.

In this section, I describe the different **GKE** and **LKE** models that have been presented in the **MIR** literature over the years.

15. And maybe others, such as *modal mixture*, *applied chords*, and *secondary dominants*.

#### 3.3.1.1 Global-Key Estimation Models

**Symbolic Inputs.** The **GKE** algorithm by Longuet-Higgins (1971), arguably the first one ever designed, computed the key of a piece through an elimination process. The notes of the piece were read from left to right, one by one, discarding all the keys where the notes were not contained as diatonic steps of the key, until a single key was remaining. This often lead to no solutions, thus, the system contained several rules for deciding the key in difficult cases. The algorithm successfully explained the keys in Bach's *Well-Tempered Clavier*, however, it was relatively easy to find adversarial examples where it did not work (Temperley and Marvin 2008).

Another algorithm, known as the *Krumhansl-Schmuckler* algorithm, was introduced in Krumhansl (1990). Although introduced in 1990 as an automatic key finder, the main components of the algorithm, the **key profile**s, were introduced in 1982 (Krumhansl and Kessler 1982). This algorithm and its method based on **key profile**s influenced many other algorithms in the symbolic and audio domains.[16] It also motivated the research of new **key profile**s using alternative methods to the *probe-tone* technique used by Krumhansl and Kessler. The key was computed by correlating the pitch histogram of the musical score with the pitch-class distribution of the **key profile**. During a series of experiments, Sapp (2011) found that when mistaken, this algorithm tends to predict the dominant instead of the tonic.

In 1996, a model for single-voiced pieces of music was proposed by Vos and Geenen (1996). Similarly to the Longuet-Higgins (1971) model, it was tested in excerpts of the *Well-Tempered Clavier*, particularly, the themes of the fugues. Although the model seemed to improve the results of the Longuet-Higgins model, it did not become as influential as the Krumhansl-Schmuckler in future research.

Responding to some concerns about the original Krumhansl-Schmuckler algorithm, Temperley (1999) proposed two changes in a new **GKE** algorithm. The first modification consisted in replacing the *Pearson correlation* function with a dot product. The second modification was

16. See Nápoles López, Arthur, and Fujinaga (2019) for a discussion of several **key profile** methods.

a change in the probability distributions of the original Krumhansl-Kessler **key profiles**. The new distribution was fine-tuned by Temperley heuristically and through a trial-and-error process. This algorithm provided better results than the original algorithm and it was further extended into a probabilistic framework using Bayes' rule in Temperley (2002).

Through the *Spiral Array*, Chew (2002) facilitated the modelling of tonality as a spatial representation. In this representation, a sequence of notes could be localized in a point of the tonal space, known as the *Center of Effect*, and related to a key through a nearest-neighbours search. Although the geometric approach for localizing a point in space influenced further research on key-finding algorithms, the model did not perform better than the approaches based on distributions and **key profiles**.

Aarden (2003) proposed an alternative, data-driven, methodology for obtaining a distribution of scale degrees. This was used to generate a new set of key profiles that could substitute the ones by Krumhansl and Kessler (1982) in a key-finding algorithm. For this purpose, 1,000 songs from the Essen Folksong Collection and 250 music excerpts from the MuseData database were used for comparing the approaches. The distribution of scale degrees outperformed the **key profiles** from Krumhansl and Kessler (1982).

Similar efforts for scale-degree probability distributions were proposed by Rohrmeier (2007) and Bellmann (2006). Rohrmeier's model focused on the repertoire of Bach chorales, where a model used a sliding window of $n$-grams to retrieve key profiles. This sliding-window method recognized modulations in fragments of music where neither the dominants nor leading tones were involved. Bellman used the chord frequencies collected in the PhD dissertation by Budge (1943) to create a new scale degree probability distribution. This probability distribution was used to obtain the most likely key of a fragment of a score (typically, a measure long) by computing the **dot product** between the notes found in the excerpt and the scale degree distribution. The results of this algorithm were also favourable, showing that chord frequencies can be helpful in estimating the musical key.

In order to explore the capabilities of different key profiles at different window lengths, Sapp (2011) performed a series of experiments running every possible window in a piece with different **key profile**s. The results of such computations were visualized and rendered into what Sapp referred to as *keyscapes* (Sapp 2001). The algorithm used for computing the keys was the original Krumhansl-Schmuckler correlation algorithm, alternating the different **key profile**s. Additionally, Sapp introduced a new **key profile** named *simple weights*.

Albrecht and Shanahan (2013) introduced a new key-finding algorithm that improves the accuracy for pieces in the minor mode. This algorithm utilizes Euclidean distance to measure the similarity between the notes in the piece and a new key profile. The **key profile** was obtained by analyzing the first and last measures of a training dataset, which consisted of 982 pieces of music from the Baroque to the Romantic period. The model outperformed every other symbolic key-finding algorithm when used in this dataset.

In 2019, we introduced a new key-finding algorithm that works in the symbolic and audio domains (Nápoles López, Arthur, and Fujinaga 2019). This algorithm was able to operate as both a **GKE** and **LKE** model, and it is based on an **Hidden Markov Model (HMM)**. The algorithm provided state-of-the-art performance in the symbolic domain, however, it underperformed for pieces in minor modes, unlike the Albrecht and Shanahan (2013) algorithm, which performs well in the minor mode.

**Audio Inputs.** Arguably the first audio **GKE** algorithm was proposed by Leman (1992). It consisted of two stages. In the first stage, the algorithm extracted the strength of each pitch-class, including its harmonics, from the audio signal. In the second stage, the algorithm tried to match the pitch-class information of the first stage with a set of templates, analogous to a **key profile**, which were obtained through **Self-Organizing Maps (SOM)**.

Izmirli and Bilgen (1994) proposed a model that also consisted of two stages. The first stage converted monophonic audio signals into a sequence of note intervals and their occurrence times, similar to a *piano-roll* representation. The second stage used a series of finite state

automatas to match patterns of scales. In total, they used three pattern-matching automatas: major scale, natural minor scale, and harmonic minor scale.

Purwins, Blankertz, and Obermayer (2000) proposed a model that was based on the constant-Q transform, extracting pitch-class distributions from the audio signal based on a few basic music-theoretical assumptions (i.e., octave equivalence and the division of the octave in the chromatic scale). Using these distributions, the system tracked the key over time by comparing the distributions of the signals with the Krumhansl and Kessler (1982) **key profiles**. The model was not quantitatively evaluated but an example of the keys tracked in Chopin's Prelude Op. 28 No. 20 was provided, showing similar key segmentations as the ones provided by an expert annotator.

Gómez and Herrera (2004) proposed a comparison of what they named "cognition-inspired" models, against models derived from machine learning techniques. The cognition-inspired model was a modification of the Krumhansl-Schmuckler algorithm that worked with **Harmonic Pitch Class Profile (HPCP)** features as input, instead of note histograms. The machine learning models tested included binary trees, Bayesian estimation, **ANNs**, **Support Vector Machines (SVMs)**, boosting, and bagging. In an experiment with 661 audio files used for training and 217 used for testing, the best machine learning model had a slightly better performance than the cognitive-based model, however, the best overall performance resulted from the combination of both models.

Burgoyne and Saul (2005) provided a model for harmonic analysis in the audio domain, using a corpus of Mozart symphonies (15 movements) for training. The input for their model was a chromagram vector (denoted as pitch-class profile by the authors). They used an **HMM** to train Dirichlet distributions for major and minor keys on the chromagram vectors. Their system attempted to find chords and keys simultaneously. Unfortunately, the model was evaluated on a single example, the second movement of Mozart's KV550. Although it appears to do well in this piece, the reported results neither provide much insight about the generalization of the model nor its performance on different examples.

Chai and Vercoe (2005) proposed an **HMM** to detect keys in audio files. Their approach took as its input a chromagram vector of 24 bins, extracting the best match of a major-and-relative-minor pair of keys (this could be thought as an algorithm that extracted the key signature). A second step used heuristics to determine the mode. In order to evaluate their algorithm, they used 10 pieces of piano music from the Classical and Romantic period, which were manually annotated by the authors.

Chuan and Chew (2005b) proposed a real-time algorithm based on the extraction of pitches and pitch-strength through the **Fast Fourier Transform (FFT)**. The **GKE** model was based on the *Center of Effect Generator* algorithm by Chew (2002) and determined the key based on the pitch-strength information. In their evaluation, the model achieved a recognition rate of 96% within the first fifteen seconds of their test set, which consisted of 61 audio files with different performances of 21 Mozart Symphonies. Their system outperformed the Krumhansl-Schmuckler algorithm (Krumhansl 1990) and the modified version by Temperley (1999) when evaluated within the first seconds of the recordings. A further improvement was proposed in Chuan and Chew (2005a), which incorporated fuzzy analysis to improve the pitch-class distributions computed from the audio. This model was evaluated during the first *Audio Key-Finding* **Music Information Retrieval Evaluation eXchange (MIREX)** evaluation, in 2005. However, the model was outperformed by other approaches submitted to that evaluation campaign.[17]

Harte, Sandler, and Gasser (2006) proposed a new model that applied a similar principle to the spatial representation of Chew (2000). It had a 6-dimensional interior space contained by the surface of a hypertorus. The 6 dimensions of their model correspond to 2-axes for localizing a point in the circle of fifths, 2-axes for localizing a point in the circle of major thirds, and 2-axes for localizing a point in the circle of minor thirds. Using this spatial representation, a similar methodology to Chew's *Center of Effect* (Chew 2002) was used for computing the key.

17. https://www.music-ir.org/mirex/wiki/2005:Audio_Key_Finding_Results

The authors presented an evaluation of the model in the detection of chord changes within 16 songs of *The Beatles*.

Noland and Sandler (2006) proposed a model based on an **HMM**. The model comprised 24 hidden states, which represented each of the major and minor keys. As observations, the **HMM** considered a chord transition (a pair of consecutive chords). The emission probabilities of the model were initialized by assuming a strong correlation between the key and the key implied in a given chord transition. The transition probabilities of the model were obtained using a table of correlations between the transposed Krumhansl and Kessler (1982) **key profiles**. The model was evaluated in a dataset of 110 songs of the Beatles, for which the chord transitions were already annotated. The algorithm was able to predict the key of the songs in 91% of the cases. A further change on the observations of the **HMM** was necessary to work directly on audio inputs.

Peeters (2006) proposed a model that extracted information about the periodicity of pitches from the audio signals, mapped that information into the chroma domain, and decided the global key of the music piece from a succession of chroma-vectors over time. For the initial analysis, Peeters proposed a Harmonic Peak Substraction algorithm, which reduced the influence of the higher harmonics of each pitch. The preprocessed signal was mapped to the chroma domain and used as the observation for an **HMM** model that predicted the global key. The **HMM** classification consisted of one **HMM** for each of the 24 possible keys, which were trained using the Baum-Welch algorithm.[18] The model was tested in 302 audio files containing classical piano, chamber, and orchestral music, obtaining a 89.1% accuracy using the **MIREX** weighted evaluation score, shown in Table 3.1.

Catteau, Martens, and Leman (2007) devised a model that simultaneously predicts chords and keys in audio signals. Their model is an extension of a previous model by Bello and Pickens (2005), and it is based on music theory domain knowledge, requiring no training. The music theory components of the model were inspired by Lerdahl's Tonal Pitch Space (Lerdahl 2005).

18. See Rabiner (1989) for an introduction to the Baum-Welch algorithm in the context of **HMMs**.

Table 3.1: *The weighted evaluation score for key predictions proposed by* ***MIREX****.*

| Key relationship(ground truth, predicted) | Score |
|---|---|
| Same key | 1 |
| Dominant or SubDominant | 0.5 |
| Relative key | 0.3 |
| Parallel key | 0.2 |
| Other | 0 |

They tested their model with synthesized audio, recorded audio, and presented their model for evaluation in the Key and Chord detection tasks from **MIREX**.

Lee and Slaney (2007) proposed a model for chord and key estimation from audio sources. The model was trained by creating a separate **HMM** for each of the 24 keys. In order to obtain the training data, they annotated several **MIDI** files with harmonic analysis labels. These harmonic analyses were computed automatically using **Melisma** (Temperley 2004). The harmonic analysis corpus was obtained from *mididb.com*, and consisted of 1046 **MIDI** files of Rock music. The annotated files were synthesized with a sample-based synthesizer (*Timidity++*,[19] using the *FluidR3*[20] soundfont) and perfectly aligned to the annotations. Their chord detection model was able to distinguish only major and minor triads.

Campbell (2010) proposed an algorithm based on four stages: frequency analysis, pitch-class extraction, pitch-class aggregation, and key classification. A thorough experiment was performed with different solutions for each of the stages. The best performing model was evaluated in a dataset of classical music, another one of popular music, and an audio-synthesized dataset of **MIDI** files of classical music. The best performing model consisted of features extracted with the *jAudio* feature extractor (McEnnis et al. 2005) and a **K-Nearest Neighbours (KNN)** classifier.

Mauch and Dixon (2010b) proposed a model that predicted chords, bass notes, metric position of the chords, and the musical key. Although it is claimed that the model detects the key of the audio input, the model in fact computes the key signature, classifying a major key

19. http://timidity.sourceforge.net/
20. https://archive.org/details/fluidr3-gm-gs

and its relative minor with the same class. Due to this restriction, the model was not explicitly evaluated on musical key. Nevertheless, using the key classifications of the model as features, it showed to improve the results of their main task, chord recognition.

Korzeniowski and Widmer (2018) proposed a key-finding model that generalized across different genres of music. The model was based on a **CNN** that received a log-magnitude log-frequency spectrogram as its input. Initially, during training, the model received the full spectrogram, however, this computation turned out to be expensive. Thus, they proposed showing the network only *snippets* (20 seconds) of the spectrogram. This improvement decreased the training time but assumed that any arbitrary 20-second audio fragment would display the global key, disallowing the model to operate with music files featuring modulations. The model was trained on three datasets: the GiantSteps **Music Technology Group (MTG)** Key dataset by Faraldo et al. (2016), the McGill Billboard dataset by Burgoyne, Wild, and Fujinaga (2011), and an internal dataset with classical music. The model was submitted to the **MIREX** campaign of 2018 and it was successful at generalizing the key of different genres, outperforming existing audio **GKE** models.

We proposed a key-finding model in 2019 (Nápoles López, Arthur, and Fujinaga 2019). This model performed both **GKE** and **LKE**, and it was able to operate in the symbolic and audio domains. The model was originally designed for symbolic inputs, however, it was able to work in the audio domain by making use of the chroma features by Mauch and Dixon (2010a) and turning them into discrete values using a threshold. The model was submitted to the **MIREX** Audio Key Estimation in 2018 and 2019. In 2018, the model was submitted with a single **key profile**, which provided the best generalization in our experiments. In 2019, the model was improved by adding a meta-classifier which predicted the global key based on the predictions of several **key profile**. The results of the 2018[21] and 2019[22] models are available on the **MIREX** results website.

21. https://www.music-ir.org/mirex/wiki/2018:Audio_Key_Detection_Results
22. https://www.music-ir.org/mirex/wiki/2019:Audio_Key_Detection_Results

After the pioneering models, many different alternatives have been proposed for finding the musical key of symbolic and audio inputs. The models consider different methodologies, initially inspired by music perception and cognition and slowly transitioning into data-driven and machine learning approaches. As may be observed from the evaluation sections of the literature, it is difficult to know which models are better than the rest, mostly due to little agreement (except for the **MIREX** campaigns) in the methods used for comparing the capabilities of one model or another, or in the use of a similar test dataset. Furthermore, some of the models output different results, ranging from key signatures, local keys, global keys, and chords, which further complicates the comparison of key-finding models. Nevertheless, as a general trend, it can be seen that the research by Krumhansl and Kessler (1982) that introduced the **key profile**s, together with the use of chroma features, have been common approaches for designing newer **GKE** models throughout the years. These techniques are also relevant for **LKE** models, which are described next.

#### 3.3.1.2 Local-Key Estimation

Changes of key may belong to different categories. In music theory, terms like *modulation* and *tonicization* are helpful for interpreting the context of a change of key. In **MIR**, it is common to describe algorithms that model *changes of key* as **LKE** algorithms. The *local keys* being the predictions that these models generate. A local key is related to the concepts of *modulation* and *tonicization*. However, it is difficult to understand the distinction between these three terms.

In order to clarify this nomenclature, we investigated the relationship between the local keys, modulations, and tonicizations of the same musical fragment (Nápoles López et al. 2020). From this work, I take the definitions of local keys, modulations, and tonicizations, which might clarify what exactly an **LKE** algorithm predicts.

**Local Keys.** Are the predictions of the musical key provided by a **LKE** algorithm. These predictions are given at a finer level of granularity than the entire piece (e.g., notes, onsets,

fixed-duration timesteps, audio frames, etc.). In principle, no music-theoretical meaning is inferred from them. They may coincide with modulations or tonicizations.

**Modulation.** Is the change from one key to another. We refer to the initial key as the *departure* key, and the second key as the *destination* key.

**Tonicization.** Is a brief deviation to a different key, usually with the intention of emphasizing a certain scale degree or harmony. The tonicization often returns to the original key briefly after the deviation.

#### 3.3.1.3 Local-Key Estimation Models

Contrary to **GKE** approaches, **LKE** models have a relatively recent history.

Purwins, Blankertz, and Obermayer (2000) introduced a method for tracking changes of key in audio signals using cq-profiles, which were calculated with the constant-Q filter bank. Their goal was to track the tone center and its variation during the piece. Their dataset included annotations for both modulations and tonicizations but considered that the ground truth was the one indicated by the tonicizations.

Chew (2002) measured the distance from a sequence of pitches to a key using the *Spiral Array*. The succession of keys was then modeled as a sequence of *boundaries* dividing the score in different key areas.

Chai and Vercoe (2005, 469) designed a model based on a Hidden Markov Model (**HMM**) to detect changes of key. They described the term *modulation* as "the change of key at some point". Their model detected first the tonal center and then the mode of the key.

Catteau, Martens, and Leman (2007) introduced a model for scale and chord recognition, assuming a correspondence between a major scale and a major key, and between a harmonic minor scale and a minor key. Their model was based on the **key profile**s by Temperley (1999) and Lerdahl's *Tonal Pitch Space* (Lerdahl 2005).

Izmirli (2007, 1) proposed a model to find local keys in audio signals. As a preprocessing step, the model adapted the tuning of the signal before computing the spectral analysis and the chroma features. In a later stage, non-negative matrix factorization was used to segment contiguous chroma vectors, identifying potential local keys. The model identified segments that were candidates for unique local keys in relation to the neighbouring key centers. The model was evaluated in three datasets: 17 pop songs with at least one modulation, excerpts from the beginning of 152 classical music pieces from the Naxos website, and the examples from the S. M. Kostka and Payne (1984) harmony textbook. In the paper, the results for three different evaluation methods were provided for each of the datasets. Izmirli also attempted to disambiguate modulations and tonicizations in the following manner:

> *Secondary functions and tonicizations are heard as short deviations from the well-grounded key in which they appear—although the boundary between modulation and tonicization is not clear cut. A modulation unambiguously instigates a shift in the key center.*

This work is also, to the best of my knowledge, the first one where the term *local key* was mentioned in **MIR** research.

Papadopoulos and Peeters (2009) adopted a similar approach to Izmirli (2007) for audio **LKE**. Their model attempted to segment the score based on the points of modulation. They introduced key dependencies on the harmonic and metrical structures of **GKE** methods, in order to convert them into **LKE** ones. The method extended previous work (Papadopoulos and Peeters 2008), adding a stage of local key estimation using an **HMM**. The observations of the model were derived from different **key profile**s: Krumhansl and Kessler (1982), Temperley (1999), and a flat diatonic key profile (a uniform distribution for diatonic pitch-classes and *zero* elsewhere). The system was evaluated with five Mozart piano sonatas, where the local keys and chords were manually annotated. The maximum accuracy achieved was 80.22%

Rocher et al. (2010) introduced a model that provided duples of (chord, key) for each audio frame of an input excerpt. The model was based on a graph and the *best-path* estimation

method. For evaluating key distances, they used the **key profile**s by Temperley (1999). The authors alluded to the term modulation when discussing their key predictions.

Mearns, Benetos, and Dixon (2011) used an **HMM** to estimate modulations over audio transcriptions of Bach chorales. The **HMM** was trained with chord progressions. The emission probability distributions were obtained from two tables with the probabilities of chords existing in a given key. These tables were based on the work by Schoenberg and Krumhansl. Tonicizations were not described in these charts, therefore, the authors did not deal with tonicizations.

Pauwels and Martens (2014) presented a probabilistic framework for the simultaneous estimation of chords and keys in audio. They mentioned the importance of "integrating prior musical knowledge" into a **LKE** model, however, they did not allude to the terms modulation and tonicization. The same year, Weiß and Habryka (2014) proposed an audio scale estimator. They argued that this estimator could help to determine the local tonality based on Gárdonyi's scale analysis method. They did not use the term tonicization, however, they discussed "short-time local modulations", which resemble tonicizations.

Machine learning approaches, especially using neural networks, have recently gained popularity in **MIR** research, including **LKE** models. Chen and Su (2018, 2019) and Micchi, Gotham, and Giraud (2020) designed models that estimate local keys as part of their **ARNA** models. Tonicization information was implied by the **RNA** annotations.

We introduced a **GKE** and **LKE** model using an **HMM** (Nápoles López, Arthur, and Fujinaga 2019). The model was capable of working with symbolic and audio data. In that work, we did not allude to the terms modulation or tonicization when describing the nature of local keys, however, an experiment on Chopin's Op. 28 No. 20 revealed that different **key profile**s tended to be more or less sensitive to smaller fluctuations of key (e.g., tonicizations). This inspired future work on the relationship of local keys, modulations, and tonicizations (Nápoles López et al. 2020).

One of the most recent models for finding changes of key is the one by Feisthauer et al. (2020), which was designed to detect modulations in Mozart piano sonatas. It used three proximity measures established from pitch compatibility, tonality anchoring, and tonality proximity. The model computed the cost of being in a key on a given beat, and estimated the succession of keys using dynamic programming techniques.

### 3.3.2 Automatic Chord Recognition

**ACR** has been explored thoroughly in the field of **MIR**. **ACR** systems usually predict the root and quality of the chords throughout a piece of music via either an audio or a symbolic representation. **ACR** is arguably one of the most widely explored topics in **MIR**. In this dissertation, with an emphasis on **RNA**, the **ACR** methods that are directly related to **RNA** models are described in Section 3.3.4. For a more historical overview of general **ACR** models, please refer to the summaries presented by Pauwels et al. (2019) for audio and Ju (2021, 58–88) for symbolic.

### 3.3.3 Automatic Pitch Spelling

A pitch-spelling model is an algorithm that predicts the original spelling that a note had in a musical score when only the pitch-class, octave, and duration of the note are provided as inputs to the model.[23]

Compared to other **MIR** tasks, there are fewer pitch-spelling algorithms available in the literature. One caveat here is that, because pitch-spelling is an important feature for commercial software dealing with the conversion of **MIDI** files into music scores (e.g., music notation editors), other algorithms may exist among commercial applications. However, I focus on discussing the published algorithms.

One argument for the relevance of pitch spelling in **ARNA** is that recent models have benefitted from spelling information (Micchi, Gotham, and Giraud 2020). This is feasible for **Mu-**

23. Some researchers have also found helpful to know the *voice* (or *stream*, in psychoacoustics) to which the note belongs (Teodoru and Raphael 2007). However, this information is unavailable in most **MIDI** files available online. Therefore, algorithms of this kind should preferrably not rely on voice information.

**sicXML** and similar input representations, but not for **MIDI** inputs, which arguably account for the vast majority of the existing digital symbolic music files today. Pitch spelling in parallel (or as a preprocessing step) to **ARNA** will thus be a viable approach in future models. This is also true for implementing **ARNA** models in the audio domain, which will likely deal with chromagram representations (without spelling).

#### 3.3.3.1 Pitch-Spelling Models

Longuet-Higgins (1976) presented what is probably the first pitch-spelling algorithm, which was constrained to monophonic melodies. Longuet-Higgins considered that any note could be assigned a number, $p$, based on 3-variables that related the note with respect to *middle C*: the distance in perfect fifths, the distance in major thirds, and the octave. Using that methodology, Longuet-Higgins extended the notation to add a "sharpness" feature, $q$, which was defined based on the distance in fifths and thirds to *middle C*, regardless of the octave. This "sharpness" value, in conjunction with the **MIDI** note number, was used to indicate the spelling of the note.

After a gap of around 25 years, a series of new algorithms were developed by different researchers independently, almost at the same time. Cambouropoulos (2003) presented an approach that relied on intervals. The algorithm used a shifting overlapping window (suggested to be of 9 to 12 pitches long). The window moved along the sequence of pitches, from left to right, until the sequence concluded. For each window, the pitch-spelling process optimized two aspects: 1) that notes make the minimum use of accidentals (something that Cambouropoulos referred to as *notational parsimony*), and 2) the avoidance of 8 classes of augmented and diminished intervals (something that Cambouropoulos referred to as *interval optimization*). The algorithm was evaluated on Mozart piano sonatas and Chopin Waltzes.

The *Spiral Array* introduced by Chew (2000) in her dissertation was used for multiple tasks that involved tonal analysis. Namely, chord labeling, key-finding, and pitch-spelling. The Spiral Array is a spatial representation of tonality that allows a sequence of pitches to be positioned in a—theoretical—tonal space, which facilitates the computation of tonal features in different

ways. Regarding pitch-spelling, three different algorithms were proposed by Chew and Chen (2003) that were built on top of each other. In the paper, their evaluation of the algorithms was restricted to a few movements of Beethoven's piano sonatas.

Throughout the late 1990s and early 2000s, Temperley developed a series of algorithms for modelling different aspects of musical structure (i.e., meter, phrasing, counterpoint, harmony, key, and pitch-spelling). These models were implemented by Daniel Sleator in **Melisma** and explained in detail in the book by Temperley (2004). Regarding pitch-spelling, the approach by Temperley consists of a rule-based system based on his concepts of **Tonal Pitch Class (TPC)** and the **line of fifths** (Temperley 2000). Additionally, the algorithm depended on the metrical analysis provided by **Melisma** before extracting the spelling of the notes.

Meredith (2003) introduced an algorithm that received **MIDI** note numbers and onset times as ordered pairs and determined the spelling of the notes through two stages. The first stage consisted of a rule-based system with eight sequential rules. After this stage, a spelling was already determined. The second stage corrected the outputs of the first stage, taking neighbouring and passing notes into account, which could have been erroneously spelled in the first stage. In a subsequent study, Meredith (2005) compared its pitch-spelling algorithm against other pitch spelling algorithms, reporting that his algorithm was more consistent across composers and styles.

Stoddard, Raphael, and Utgoff (2004) proposed a data-driven algorithm for pitch-spelling. It required ground-truth spelling information in order to be trained. Additionally, the algorithm ran on top of the probabilistic framework for harmonic analysis by Raphael and Stoddard (2003), conditioned by the accuracy of that harmonic analysis model. The algorithm was evaluated over a total of 22,593 notes, scoring 96.87% accuracy. During their study, they found that the detection and resolution of voice-leading considerations (i.e., the relationship between a note and its immediate neighbours) was the most important feature to consider in a pitch-spelling algorithm.

Teodoru and Raphael (2007) tackled the problem by assuming that the spelling of a note depended on voice-leading and local harmony. They observed principles for spelling notes in music theory textbooks (Aldwell and Schachter 1978; Rimsky-Korsakov, Achron, and Hopkins 2005) and built a probabilistic model that captured these principles through *Markov chains*. In their evaluation, they showed that their model outperformed other models, including the one by Meredith (2006). Nevertheless, an important limitation of their model was that it relied on *voice* information being provided as part of the input, which is not the case for most **MIDI** data available online and, therefore, limits the practical application of the algorithm. In their study, they found that **LKE** is fundamental for pitch spelling. Additionally, they considered that the relationship between local keys and spelling is bidirectional. That is, spelling can influence the choice of key as much as the key can influence the choice of spelling.

### 3.3.4 Automatic Roman Numeral Analysis

**RNA** refers to the syntax used to annotate chords, which evolved from the sporadic use of Roman numerals in the late eighteenth century to the complex analysis system used today (see Chapter 2). In computational contexts, **ARNA** has been thought, starting with Temperley (1997), as a "chord-finding plus key-finding" problem. In recent years, Chen and Su (2018) subdivided the problem further into six sub-tasks: chord quality, chord root, local key, inversion, primary degree, and secondary degree. As an **MIR** problem, **ARNA** can be expressed as the task of correctly predicting enough features in order to reconstruct the correct Roman numeral labels. This problem is challenging for various reasons (see Section 1.2). In the next section, I present a survey of existing approaches to solve this problem.

#### 3.3.4.1 Roman Numeral Analysis Models

**Pioneering Works.** The pioneering work in **ARNA** is the one by Winograd (1968). This model was evaluated over music by Bach. It required a preliminary, hand-made, representation of the score as a sequence of four-part perfect chords. This representation was processed

by a model implemented in the **LISt Processor (LISP)** programming language. The hand-made representation required *nonchord*s to be removed before processing the input with the model, a crucial limitation, because this process was complex and could not be automated. In addition to this limitation, Temperley (1997) provided insight about other limitations in Winograd's model, for example, its incapacity to deal with arpeggios and contrapuntal textures.

After Winograd, an *expert system* model was introduced by Maxwell, first in his PhD dissertation (Maxwell 1984), and subsequently in a book chapter (Maxwell 1992, 334–353). This model is one of the best examples of rule-based systems for **MIRs**. The model consisted of fifty five rules that reduced vertical sonorities into chord sequences, and decided the location of changes of key. Initially, the rules of the system were short and concise:

> *Perfect and imperfect consonant intervals constitute a consonant interval. Every other is a dissonant interval.*

However, as new rules were introduced, they became increasingly more complex and cryptic:

> *If the goal chord falls on a strong beat and it is a major triad or major-minor seventh, and the root movement from the pre-cadence is an ascending or descending perfect fifth or major second or a descending minor second, and when the root motion is a descending fifth, the pre-cadence is not a potential dominant, and when the root motion is an ascending fifth the pre-cadence is triadic, then the pseudo-cadence is a half cadence, and its strength increases by 10.*

The system was also criticized by Temperley and others for its use of arbitrary or "hard-coded" values to determine the strength of a cadence. An example shown in the last sentence of the previous rule.

Regardless of the complex rules, Maxwell's model provided convincing music-theoretical outputs when analyzing three movements of the Bach Six French Suites: the sarabande from

Suite No. 1, the minuet from Suite No. 2 and the gavotte from Suite No. 5. The pieces featured distinct types of complexities: four-part harmony with several **nonchord tone**s, 2-voice contrapuntual textures, and a varying contrapuntual texture, respectively. Unfortunately, because these are only three musical examples by the same composer, it is unlikely that the system would generalize to other compositions. Temperley (1997) listed the limitations of Winograd and Maxwell's approaches, summarizing them in the following:

- Sequences of notes that are not displayed simultaneously (vertically), as arpeggiations of chords.
- Missing pitches in the spelling of a full chord that can be deduced from the context.
- Ornamental notes. Maxwell proposes specific rules to deal with these notes, but according to Temperley, neither Maxwell's or Winograd's are good enough to correctly detect ornamental notes.

**The Melisma Music Analyzer.** The first end-to-end **ARNA** system can be attributed to the contributions of Temperley, Sleator, and Sapp. First, Temperley (1997) proposed an algorithm to find harmonic roots, which was complemented with preference rules for meter analysis (Temperley and Sleator 1999) and a key estimation model (Temperley 1999). The collective algorithms were published by Temperley (2004) and implemented in collaboration with Sleator as a suite of programs called **Melisma**.[24] The system operated with intermediate representations for both inputs and outputs. Sapp filled in the remaining gap with additional stages that allowed **Melisma** to digest ****kern** inputs and generate ****harm RNA** annotations, aligning them with the original score. The full end-to-end workflow was facilitated by a program called *tsroot* (Sapp 2009) and used to generate fully automatic **RNA** labels of the music scores in the *KernScores* database (Sapp 2005).

After the first version of **Melisma**, which is based on preference rules, Temperley favored probabilistic models in subsequent work. Notably, Temperley (2009) proposed a Bayesian ap-

24. https://www.link.cs.cmu.edu/music-analysis/

proach to simultaneously model harmonic analysis, meter induction, and melodic seggregation. This motivated the *Melisma Music Analyzer (version 2)*,[25] however, this program was not extended to provide **RNA** annotations.

**Probabilistic and Grammar-Based Models.** Notable subsequent studies include an **HMM**-based approach for functional harmony in Raphael and Stoddard (2004), a method based on dynamic programming in Illescas, Rizo, and Quereda (2007), an application of **pcset**s to search for chord "syntacticality" in Rohrmeier and Cross (2008), grammar-based approaches in Magalhães and Haas (2011), Quick (2016), Harasim, Rohrmeier, and O'Donnell (2018), and Harasim, O'Donnell, and Rohrmeier (2019), and a generative syntax to parse chord progressions featuring modulations in Rohrmeier (2011).

**Deep Learning Models.** More recently, deep neural networks have become the preferred tool for approaching this problem. Chen and Su (2018) were the first to introduce **Multitask Learning (MTL)** (Ruder 2017) to the problem as a suitable way for the neural network to share representations between related tonal tasks. Chen and Su's model consists of a bidirectional **LSTM** (Hochreiter and Schmidhuber 1997) followed by task-specific dense layers, which implement the **MTL** aspect. In this work, the authors also introduced the **Beethoven Piano Sonatas (BPS)** dataset for evaluating such models. The **MTL** layout outperformed single-task configurations and it has continued to prove the best-performing approach in subsequent deep learning studies. In subsequent work, the same authors have adopted Transformer-based networks to deal with functional harmony and **ACR** (Chen and Su 2019, 2021). The work with these networks has explored the capability of the attention mechanisms to improve the performance of **ACR**, paying special consideration to chord segmentation and its evaluation.

Micchi, Gotham, and Giraud (2020) proposed a *DenseNet*-like (G. Huang et al. 2017) **CNN**, followed by a recurrent component. The recurrent component consists of a bidirectional **GRU**

25. http://davidtemperley.com/melisma-v2/

(Cho et al. 2014) connected to task-specific dense layers, similar to those of Chen and Su (2018). In their experiments, the *DenseNet*-like convolutions outperformed dilated convolutions and a **GRU** by itself (i.e., with pooling instead of the convolutional blocks). Micchi et al. also demonstrated the positive effect of using *pitch spelling* in the inputs and outputs. This provided at least two advantages: a more informative output (e.g., not only the correct key, but the correct spelling between two enharmonic keys), and an increased (theoretical) number of transpositions available for data augmentation. In Micchi et al. (2021), the same architecture was extended to include a **Neural Autoregressive Density Estimator (NADE)** layer, which models a dependency between the **MTL** layers, improving the overall performance of the network.

McLeod and Rohrmeier (2021) proposed a new network architecture for **RNA**. In their work, they depart from the multitask learning approach used by previous researchers, substituting it with a modular workflow where the different tonal attributes are computed in sequential stages of the model.

Around the same time that Micchi et al. (2021) and McLeod and Rohrmeier (2021) have proposed their models, we have also introduced an earlier version of the neural network presented in this dissertation (Nápoles López, Gotham, and Fujinaga 2021).

The method in this dissertation follows the line of research based on **MTL**. This methodology is described below.

#### 3.3.4.2 Multitask Learning Approaches

Chen and Su (2018) proposed decomposing an **RNA** label into a set of multitask classification problems. Furthermore, these classification problems are solved concurrently using a **MTL** configuration. In the initial work by Chen and Su (2018), the **MTL** configuration is based on the following tonal features: local key, primary degree, secondary degree, inversion, and chord quality. This approach has been adopted in other research (Chen and Su 2019; Micchi, Gotham, and Giraud 2020; Micchi et al. 2021).

**Local Key.** The local key describes the reference key used to disambiguate the Roman numeral scale degrees. It is arguably one of the most important features because if it is mislabeled, then all RNA labels will be mislabeled.

In the initial multitask learning model by Chen and Su (2018), this task had 24 output classes. Each of the classes represented the pitch class of the tonic, which were encoded twice to account for major mode keys and minor mode keys.

In a subsequent model by Micchi, Gotham, and Giraud (2020), the number of classes was extended to 30, as the spelling of the tonic pitch was taken into account (i.e., $C\sharp \neq D\flat$ major).

**Primary Degree.** The primary degree of the chord represents the "numerator" part of a Roman numeral. For example, in the annotation $\mathbf{V^7/v}$, the primary degree is $\mathbf{V^7}$. In the multitask learning model by Chen and Su (2018), this task had 21 classes.

**Secondary Degree.** The secondary degree represents the "denominator" part of a Roman numeral label. For example, in the annotation $\mathbf{V^7/v}$, the secondary degree is $\mathbf{V}$. In the multitask learning model by Chen and Su (2018), this task had 21 classes. Secondary degrees only appear in the context of a tonicization. Thus, the majority of the RNA annotations will have an empty secondary degree, and a special class to denote this empty secondary degree needs to be included. The scarcity of secondary degrees also results in a highly imbalanced distribution of examples, which is problematic for training.

**Inversion.** The inversion of the chord indicates the note acting as the bass of the chord. In the RNA system, the arrangement of the notes above the bass is irrelevant for the annotation. For example, the first two chord realizations shown in Figure 3.1 have an equivalent Roman numeral label, despite the different arrangement of the upper voices. However, modifying the bass modifies the inversion, regardless of the arrangement of the upper voices remaining unchanged.

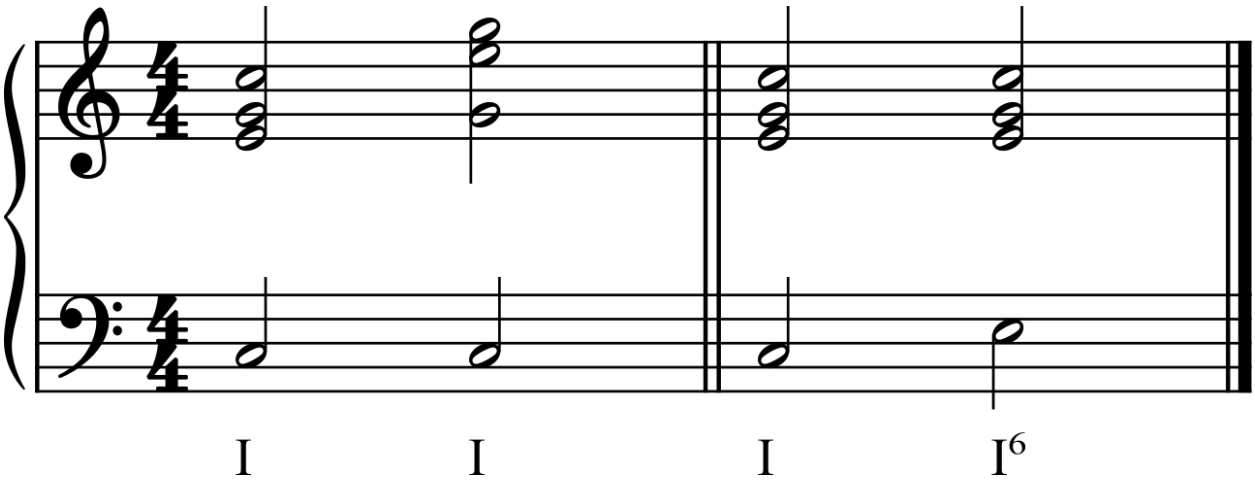


Figure 3.1: *Example of chord inversions in the **RNA** syntax. The inversion changes when the bass changes, regardless of the arrangement of the upper voices.*

The inversion task generally requires 4 output classes: root position (no inversion), 1st inversion, 2nd inversion, and 3rd inversion (only for seventh chords).

**Quality.** The chord quality task depends greatly on the chord vocabulary. The vocabulary introduced in Chen and Su (2018) considered 10 classes. The one by Micchi, Gotham, and Giraud (2020) considered the same number of classes. This classification also suffers from a large class imbalance, because the vast majority of chords found in tonal music are major triads.[26]

**Root.** The chord root task, as the name implies, predicts the pitch of the chord's root. In the multiclass classification problem introduced by Chen and Su (2018), there are 12 possible chord roots (i.e., as many as pitch classes). In the work by Micchi, Gotham, and Giraud (2020), this was extended to 35 classes (see Section 5.1.3.1), as spelling was taken into account. The system was trained to predict any root within two flats and two sharps.

26. See, for example, the distributions of chords in Figure 4.15 or Table 6.13.

# Chapter 4

# Data Acquisition and Preparation

This chapter introduces the datasets used for this research, as well as the the data curation and data-augmentation workflows used to process the data. Section 4.1 describes the general preprocessing applied to all the datasets utilized, before aggregating them. Section 4.2 introduces the publicly available datasets that were aggregated. Section 4.3 summarizes the size and **Roman Numeral Analysis (RNA)** distribution of the aggregated dataset. Section 4.4 describes the process to partition the dataset into training, validation, and test splits. Section 4.5 introduces the flavors of data augmentation performed to the original data, notably a new method based on synthesizing musical scores from the **RNA** annotations.

## 4.1 General Preparation of the Data

The **Automatic Roman Numeral Analysis (ARNA)** system developed for this dissertation, **AugmentedNet**, was trained and evaluated with publicly available digital **RNA** annotations.

**ARNA** datasets often differ in their size, format, and musical genre. One of the main characteristics that distinguishes them from, for example, chord labeling datasets, is that **RNA** annotations require a clear indication of the key (see Section 3.3.1). Thus, knowing the key is a necessity for any dataset to be considered an **RNA** dataset. Preferrably, a full Roman numeral annotation in string form, such as the ones described in the digital standards (see Section 3.1.1)

is provided as well. However, this is not always the case, or the **RNA** strings in one digital standard may be incompatible with the ones of another digital standard. For this reason, the approach taken in this dissertation is to standardize all the annotations based on two pieces of information: the key provided, and the **pcset** representation of the chord annotation. This is achieved through the methodss described in Appendix A.

### 4.1.1 Standardizing the Annotations

In order to facilitate the aggregation of all datasets, **RomanText** was chosen as the "container" digital format for all **RNA** annotations. However, the underlying vocabulary of Roman numerals is the one described in Section A.2, $\mathcal{N}$, which consists of 31 classes of Roman numeral numerators. This helps to perform the supervised learning workflow in a more controlled manner during training and evaluation.

Regarding **RomanText**, there are a few motivations for choosing it as the container format for all annotation files:

1. It is a stand-alone format, which does not require access to the musical score; this is useful for datasets where the scores were not provided, in which case the annotations can be aligned with a third-party score.

2. An existing effort has already been done in the **When in Rome (WiR)** dataset (Gotham, Tymoczko, and Cuthbert 2019; Gotham and Jonas 2022) to convert other datasets into **RomanText**. Thus, taking advantage of that effort is desirable.

3. The **RomanText** has been closely integrated to the *music21* Python library (Cuthbert and Ariza 2010), which is one of the most advanced software tools for processing musical scores, and facilitates the integration of annotations with **MusicXML** files, which is the symbolic music format of most digital music scores.

Beyond the conversion to **RomanText**, one of the challenges of aggregating the different datasets is that errors are introduced at different stages of the process. Some of the problems

observed were that: 1) the annotations and scores were misaligned, 2) several chords were mislabeled (due to errors in the conversion or errors in the annotations themselves), and 3) the inversions of the chords were incorrectly annotated.

### 4.1.2 Detecting Errors in the Annotations

The process of inspecting each dataset to ensure that they can be aggregated is very time consuming, and generally requires an extensive correction of errors. Performing this work manually is unrealistic, as each of these datasets involves, presumably, hundreds of hours of work by experts in tonal harmony, which can doubtly be reviewed by a single person.

In order to lessen this effort, I developed a semi-automatic workflow to identify potential issues with the quality of an annotation file in a dataset. This semi-automatic workflow is based on detecting three common error patterns. These patterns are based on common corrections that I needed to perform in the datasets that I initially aggregated manually.

#### 4.1.2.1 Common Error Patterns

**Alignment between Scores and Annotations.** When the annotations and scores are not perfectly aligned, this results in a very negative effect, where a model learns the wrong chord associations for a musical context. I found this one of the most pressing problems to be addressed when aggregating the available datasets. However, it is also very difficult to detect.

An initial approach is to verify that the duration and numbering of a measure matches between the annotation and score files. If a measure number changes inconsistently between the two, this might indicate a misalignment.

**Pitch Correspondence between Chord Annotations and Scores.** Although the presence of **nonchord tone**s and absence of chord tones can make it difficult to automatically detect whether a chord has been mislabeled, there is usually some shared pitch content between the annotation and the score.

Whenever the correspondence of the pitch content and the chord annotation is unusually low, this might be an indication that the chord has been mislabeled. Designing a routine to compare this pitch correspondence helps detecting such instances (e.g., a C chord has been mislabeled as a C♯ chord, which is unambiguosly incorrect). The most common version of this problem is an annotator indicating the wrong key at some point of the **RNA** annotations, and what follows is that all subsequent chords are relative to the wrong key. This results in an unusually low pitch correspodence between the annotations and the score.

**Correspondence between Lowest-Sounding Note and Bass of the Chord.** One of the most common errors done by harmonic analysis annotators (even if they are experts) is to mislabel the inversion of the chord. This happens likely because the annotators pay less attention to this property of the chord than, for example, the root. In certain genres, such as chorales or string quartets, a higher voice may cross the bass momentarily, confusing the annotator into thinking that the lowest sounding note of the chord is in the bass voice, when it is in fact not the case.

These inconsistencies can be verified in a similar way to the pitch correspondence. An unusually low match between the annotated bass and the lowest note in the score may indicate that the annotator mislabeled the inversion of the chord. Nonetheless, looking for this pattern will often result in flagging pedal tones as mislabeled inversions, which is not the case.

Software implementations for detecting the three types of errors discussed can be found in the accompanying source code of the dissertation (see Section 7.3).

### 4.1.3 Summary of the Preparation

In summary, each of the publicly available datasets was standardized to a reduced vocabulary of Roman numerals. This reduced vocabulary was encoded into a **RomanText** container format, that was independent of the score. In order to spot errors and misalignments, three common error patterns were searched across the pairs of **RomanText** and **MusicXML** scores.

The files with the highest number of anomalies were reviewed and corrected. This process was repeated a few times until a satisfactory level of quality was achieved. The next section describes each of the publicly available datasets that were aggregated in this way.

## 4.2 Publicly Available Datasets

There are currently, to the best of my knowledge, eight **RNA** datasets: **Annotated Beethoven Corpus (ABC)**, **Beethoven Piano Sonatas (BPS)**, **Haydn "Sun" String Quartets, Op. 20 (HaydnSun)**, **Hooktheory Lead Sheet Dataset (HLSD)**, **Key Modulations and Tonicizations (KMT)**, **Mozart Piano Sonatas (MPS)**, **Theme and Variation Encodings with Roman Numerals (TAVERN)**, and **WiR**. All but **HLSD** focusing on music from the **common-practice period** and annotated by expert human annotators.[1]

In this section, I describe the datasets focusing on music from the **common-practice period**, as these can be aggregated into a unified dataset to train a machine learning model.

### 4.2.1 Annotated Beethoven Corpus (ABC)

The **ABC** dataset was introduced by Neuwirth et al. (2018). It is a dataset consisting of Roman numeral annotations for all string quartets by Beethoven, except for Op. 133 ("Große Fuge"), where the authors claim that contrapuntal principles prevail over harmony. The corpus of scores was annotated by two of the co-authors (Neuwirth and Moss), each one annotating half of the dataset and cross-reviewing the other annotator's work. Both annotators have a background in Musicology and are experts in tonal harmony.

The dataset comprises sixteen string quartets. These account for 70 music files (i.e., string quartet movements), 15,806 measures, 240,462 notes, and 27,962 chord labels, according to the statistical summary presented by the authors.

1. **HLSD** is a crowdsourced dataset.

#### 4.2.1.1 Format of ABC

All of the annotations in the **ABC** dataset are encoded in the **Digital and Cognitive Musicology Lab (DCML)** standard for **RNA** (see Section 2.3.1.3).

**Original Annotations (Version 1).** In the originally published version of the dataset, each annotation consists of nine parts, which are parsed using regular expressions: 1) key, 2) pedal note, 3) Roman numeral (numerator), 4) chord quality, 5) numeric inversion, 6) chord alterations (e.g., added notes), 7) Roman numeral (denominator), 8) pedal ending, and 9) phrase ending.[2]

**Revised Annotations (Version 2).** A revision of the **DCML** standard notation (see Section 2.3.1.3) lead to a new version of this dataset in 2021.[3] The new version of the dataset, in addition to the new harmonic analysis standard, introduces corrections in the annotations and additional files.

#### 4.2.1.2 Summary of the ABC Dataset

According to the count performed in this dissertation, the **ABC** dataset has a total duration in quarter notes of 48,034, across 15,746 measures. After preprocessing and preparation, the **ABC** dataset contributed 29,427 **RNA** chord annotations, which have an average harmonic rhythm of 1.63 **Quarter notes (♩s)** notes.

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.1. As expected, all of the $\mathbf{Cad}_{4}^{6}$ annotations correspond to a "second inversion" position. While not all of the datasets provide this type of annotation (**cadential six-four** chords) this dataset provides nearly 110 examples of them. The **ABC** dataset is also one of the few datasets (the others being **TAVERN** and **WiR**) that provide examples of $\mathbf{III}^{+7}$ chords, which is one of the least common types of Roman numeral numerators in the vocabulary.

2. https://github.com/DCMLab/ABC
3. https://github.com/DCMLab/ABC/tree/v2

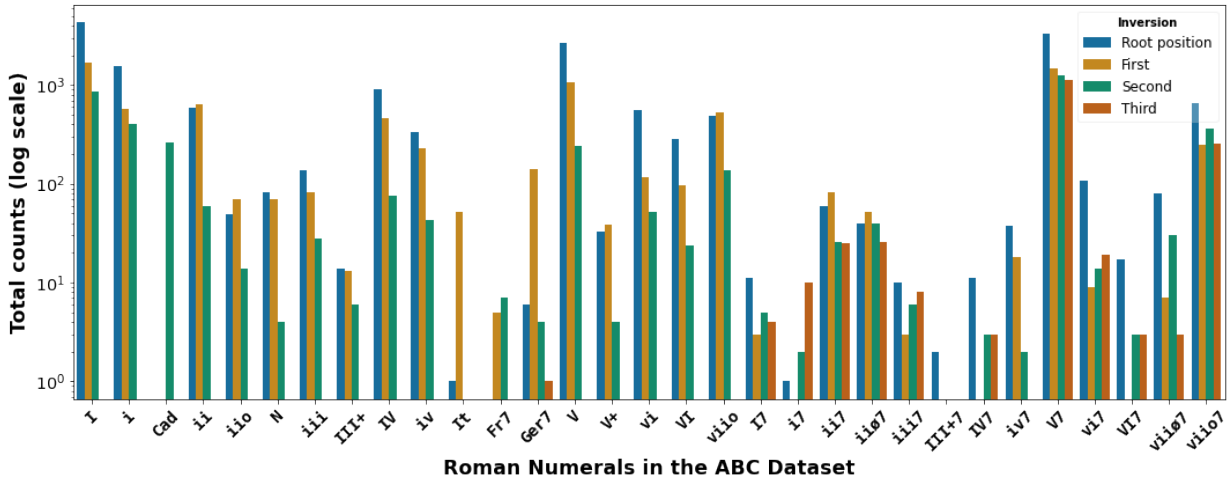


Figure 4.1: *All the RNA labels taken from the ABC dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

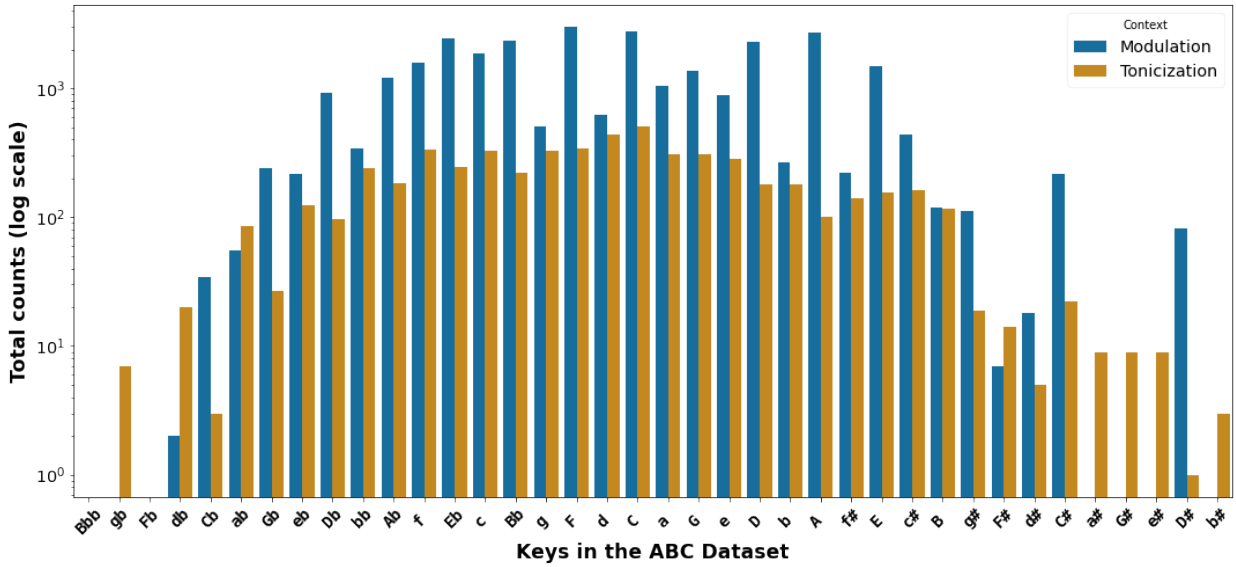


Figure 4.2: *All the keys spanned by the RNA annotations of the ABC dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations .*

Figure 4.2 shows the distribution of the keys in the **ABC** dataset. The dataset spans a total of 36 keys out of the 38 in the vocabulary $\mathcal{K}$ (see Section A.4), making it one of the datasets with more representation of the keys in the vocabulary. It is only outnumbered by the **KMT** and **WiR**, which span 37 keys each. Most of the keys in **ABC** lie within the center of the **line of fifths** (i.e., few accidentals), as expected. However, there is an unusually large occurrence of the **D♯** key. After closer inspection, it seems that this resulted from a mislabeled (long)

modulation in Op. 131 No. 7 (around measure 119). Thus, the high count of **D♯** is unjustified and an example of the second error pattern discussed in Section 4.1.2.1. Errors like this one are occassional in all of the datasets. The search for the common error paterns helps to identify them, but it remains to be one of most time-consuming issues to address in the development of an **ARNA** system. The visualization of the data, as demonstrated by this example, is also helpful to find and correct annotation issues.

### 4.2.2 Beethoven Piano Sonatas (BPS)

The **BPS** dataset was introduced by Chen and Su (2018). It consists of annotations for all the first movements of piano sonatas by Beethoven. The annotations were provided by an expert musicologist, according to the authors. The process for encoding the annotations was divided in four steps: 1) identify the local key, 2) decide the segmentation of the chords (i.e., where one chord ends and the next one begins), 3) labeling the chord, taking into account the absence of chord tones and/or presence of **nonchord tone**s, and 4) specifying the chord inversion. Out of these steps, the authors mention that step 3 was particularly complicated, because of the factors that need to be taken into account.

The **BPS** dataset contains 32 files (i.e., piano sonata movements), 86,950 notes, 7,394 Roman numeral labels, and 531 key modulations.

#### 4.2.2.1 Format of BPS

The annotations of the **BPS** dataset are provided in a tabular **Comma-Separated Values (CSV)** file format. These **CSV** files contain an offset, in **Sixteenth notes (𝅘𝅥𝅯s)** notes from the beginning of the score, of the location of each chord. The annotations are accompanied by another **CSV** file containing the **Musical Instrument Digital Interface (MIDI)** notes of the score (in a similar format for the offsets), but not a full symbolic score (e.g., in **MusicXML** format). Thus, for this research, it was necessary to find an external collection of musical scores in **MusicXML** format to accompany the annotations of this dataset.

#### 4.2.2.2 Acquiring Matching Symbolic Scores

Two sets of symbolic music encodings were considered as a match for the **BPS** annotations, one encoded by the user *ClassicMan* in the MuseScore community website[4] and another one encoded by Sapp in the ****kern** format.[5]

Two encodings of the same score can often feature discrepancies when encoded by different persons, in different formats, or converted from one format to another (Nápoles López, Vigliensoni, and Fujinaga 2018, 2019). I found that to be the case between these two sets of scores when they were compared with each other.[6]

In the end, the set by *ClassicMan* was preferred because the file format did not require any conversion, and the measure information was more consistent with the annotation files.

#### 4.2.2.3 Summary of the BPS Dataset

According to the count performed in this dissertation, the **BPS** dataset has a total duration in quarter notes of 23,540, across 7,080 measures. After preprocessing and preparation, the **BPS** dataset contributed 10,584 **RNA** chord annotations, which have an average harmonic rhythm of 2.22 quarter notes (♩).

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.3. The **BPS** dataset is one of the two datasets (the other one being **HaydnSun**) that do not provide examples of $\mathbf{Cad}^{\mathbf{6}}_{\mathbf{4}}$ chords. It is unlikely that this means there is no presence of them, but instead that they have been annotated as either $\mathbf{I}^{\mathbf{6}}_{\mathbf{4}}$ or $\mathbf{V}^{\mathbf{6}}_{\mathbf{4}}$. This is one of the reasons why the $\mathbf{Cad}^{\mathbf{6}}_{\mathbf{4}}$ symbol is useful, because then these chords can be disambiguated.

Figure 4.4 shows the distribution of the keys in the **BPS** dataset. The dataset spans a total of 34 keys out of the 38 in the vocabulary $\mathcal{K}$ (see Section A.4). The distribution of keys is slightly skewed towards the "flatter" side of the **line of fifths**, up to the last key in the vocabulary, **B**𝄫. The occurrence of the flatter keys is mostly in the form of tonicizations.

4. https://musescore.com/user/19710/sets/54311
5. https://github.com/craigsapp/beethoven-piano-sonatas
6. https://napulen.github.io/beethoven_piano_sonatas_comparison/

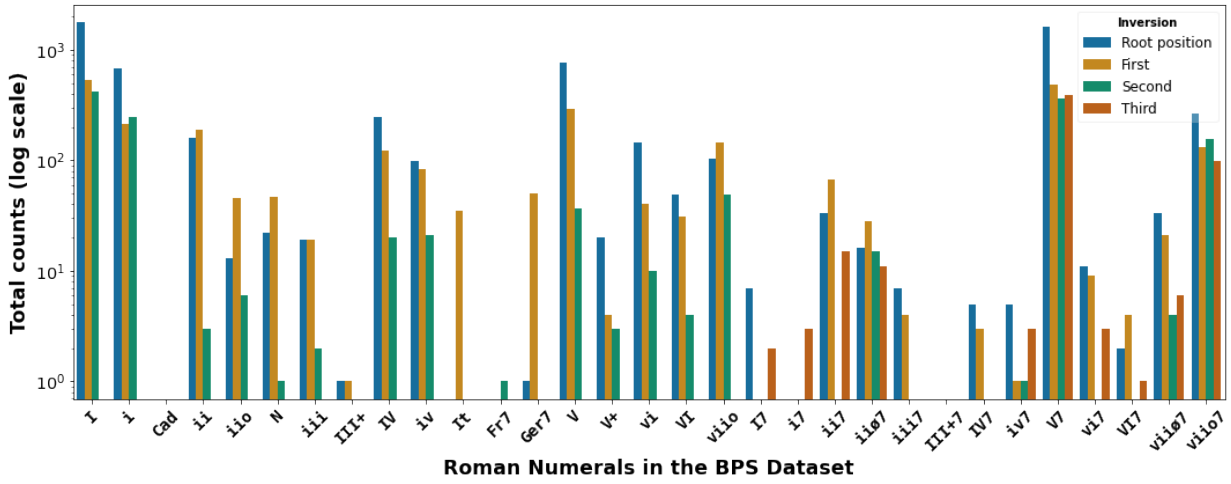


Figure 4.3: *All the RNA labels taken from the BPS dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

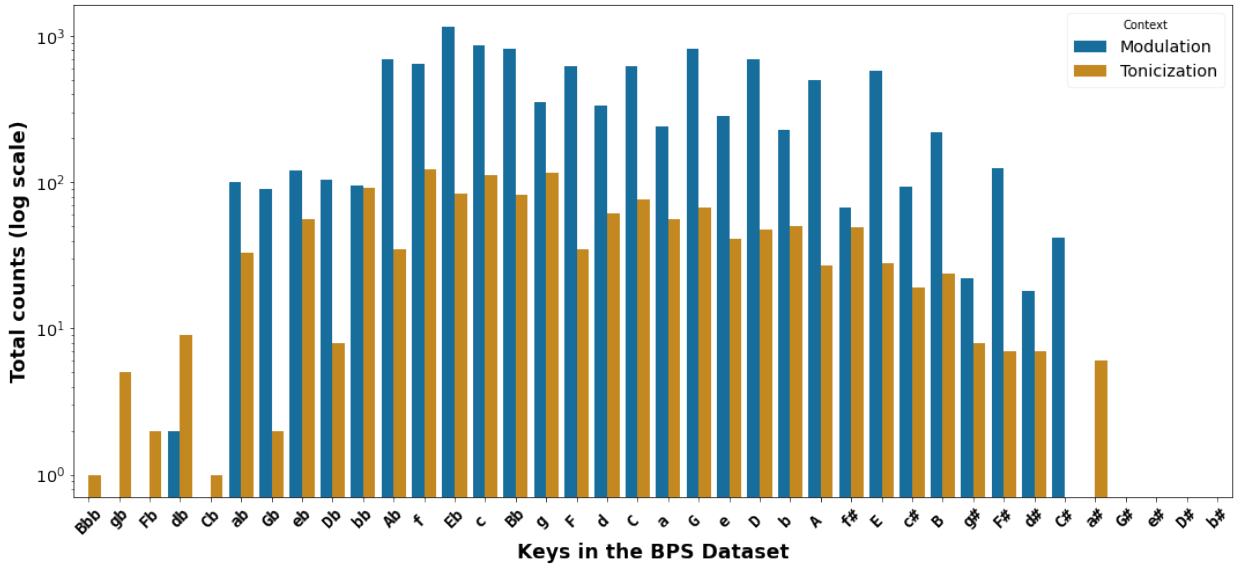


Figure 4.4: *All the keys spanned by the RNA annotations of the BPS dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations .*

### 4.2.3 Haydn "Sun" String Quartets (HaydnSun)

The **HaydnSun** dataset was introduced by Nápoles López (2017). It consists of annotations for every movement in the string quartets Op. 20 by Haydn (commonly known as the "Sun" quartets). The annotations are written on top of the symbolic music scores provided at the KernScores library (Sapp 2005). The dataset was annotated by Caro Repetto and Nápoles López for

the first movements of each quartet (6 files), and by Nápoles López for all remaining movements (18 files). The dataset consists of 24 files, 2,921 measures of music, and 5,357 Roman numeral labels.

#### 4.2.3.1 Format of HaydnSun

The annotations of the **HaydnSun** dataset are provided in the ****harm** syntax (see Section 2.3.1.1). This is the same digital format used by the **TAVERN** dataset and the first end-to-end **ARNA** system. In addition to the annotations, a "description" field was included in these files, which provides descriptions for difficult/ambiguous annotations.

#### 4.2.3.2 Summary of the HaydnSun Dataset

According to the count performed in this dissertation, the **HaydnSun** dataset has a total duration in quarter notes of 9,095, across 2,921 measures. After preprocessing and preparation, the **HaydnSun** dataset contributed 5,357 **RNA** chord annotations, which have an average harmonic rhythm of 1.7 quarter notes (♩).

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.5. Something surprising among the **RNA** distribution of this dataset is that all of the **It** and **Ger**$^{7}$ annotations have been encoded as in "root position," which is unusual for these types of chords. It is more likely that this indicates a mistranslation of the annotations for this particular dataset. Due to the cascading effects that a re-translation of the dataset could have in other areas of this dissertation (e.g., experimental evaluation), a possible re-translation of the dataset is left for future work. As long as the annotations are correctly labeling **It** or **Ger**$^{7}$ chords, they will remain useful for classifying the appropriate **pcset** and key contexts, despite the inversions being incorrect. Notice also the lack of $\mathbf{Cad}^{6}_{4}$ chords in this dataset. This is due to the lack of a $\mathbf{Cad}^{6}_{4}$ symbol in the ****harm** syntax used to encode the dataset. The $\mathbf{Cad}^{6}_{4}$ chords were instead encoded as $\mathbf{I}^{6}_{4}$. They are thus undistiguishable from noncadential $\mathbf{I}^{6}_{4}$ chords (see Section 2.3 for further discussion).

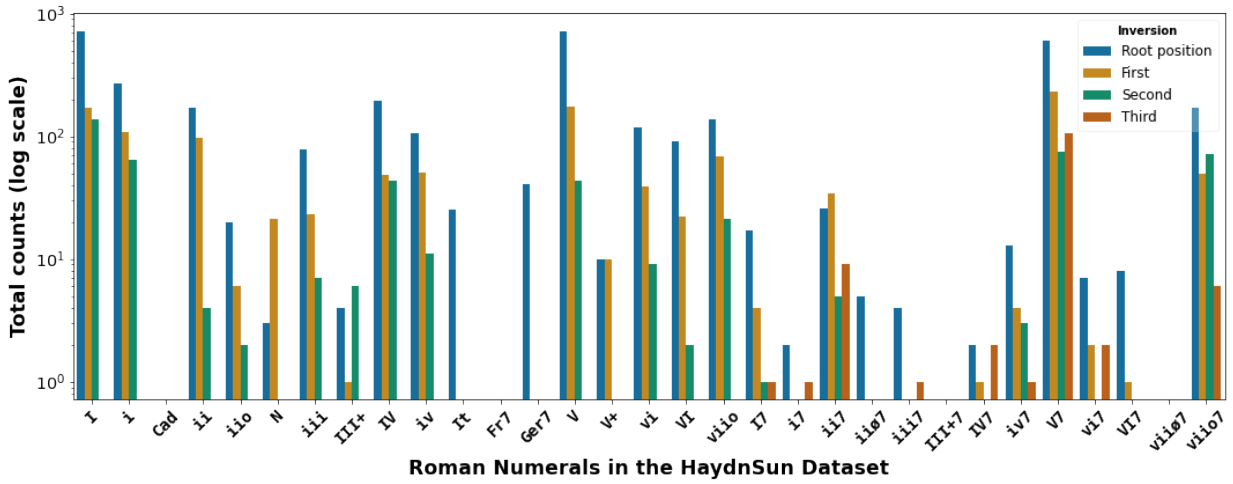


Figure 4.5: *All the **RNA** labels taken from the **HaydnSun** dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

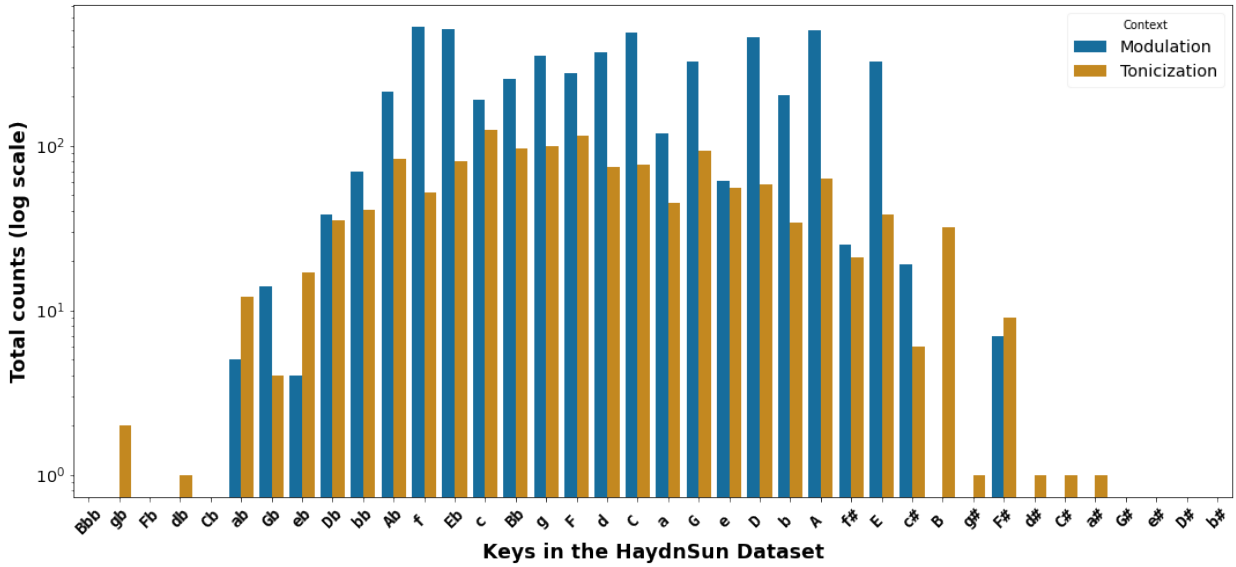


Figure 4.6: *All the keys spanned by the **RNA** annotations of the **HaydnSun** dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations .*

Figure 4.6 shows the distribution of the keys in the **HaydnSun** dataset. The dataset spans a total of 31 keys out of the 38 in the vocabulary $\mathcal{K}$ (see Section A.4). The occurrence of keys around the center of the **line of fifths** is higher, as expected. The keys on the extremes occur mostly in the context of tonicizations and for one or two chord annotations, which is consistent with a brief tonicization throughout a piece of music.

### 4.2.4 Key Modulations and Tonicizations (KMT)

The **KMT** dataset was introduced in Nápoles López et al. (2020). All the labels in the dataset were obtained from the modulation excerpts of five music theory textbooks: Aldwell, Schachter, and Cadwallader (2019), S. Kostka and Payne (2008), Reger (1904), Rimski-Korsakov (1886), and Tchaikovsky (1872).

The dataset contains, in total, 201 excerpts of music with annotated modulations and tonicizations. The annotations are encoded in the form of **RNA** annotations of all the chords in the dataset.

When the theorists of the original sources provided **RNA** annotations, those were preserved in the digital transcriptions. Otherwise, Nápoles López and Feisthauer provided them. All the annotations related to modulations were taken exclusively from the textbooks. Tonicizations rely on the **RNA** annotations of the chords and these were not always provided in the textbooks, therefore, they were often supplied. For some onsets, multiple key annotations were provided by the theorists. For these excerpts, the authors of the dataset encoded the keys in chronological order.

#### 4.2.4.1 Format of KMT

The music notation of each file was encoded in the **KMT** symbolic music representation. The **RNA** annotations were digitally encoded as an additional Humdrum **spine** in the *harmalysis* format (see Section 2.3.1.4).

#### 4.2.4.2 Summary of the KMT Dataset

According to the count performed in this dissertation, the **KMT** dataset has a total duration in quarter notes of 2,107, across 548 measures. After preprocessing and preparation, the **KMT** dataset contributed 1,415 **RNA** chord annotations, which have an average harmonic rhythm of 1.49 quarter notes (♩).

The distribution of the RNA annotations and their inversions are shown in Figure 4.7. This dataset has the smallest RNA vocabulary among all datasets considered. The majority of the annotations being **I**, **i**, **V**, and $\mathbf{V}^7$ chords. However, there are nearly 50 examples of $\mathbf{Cad}^{6}_{4}$ annotations and an unusually large number, nearly 70, of Neapolitan chords, **N**. Due to the nature of this dataset, which consists of examples of modulations in music theory textbooks, it is possible that the focus of the examples has been on modulations across different keys, rather than exotic chord progressions. The distribution of keys discussed below supports this idea.

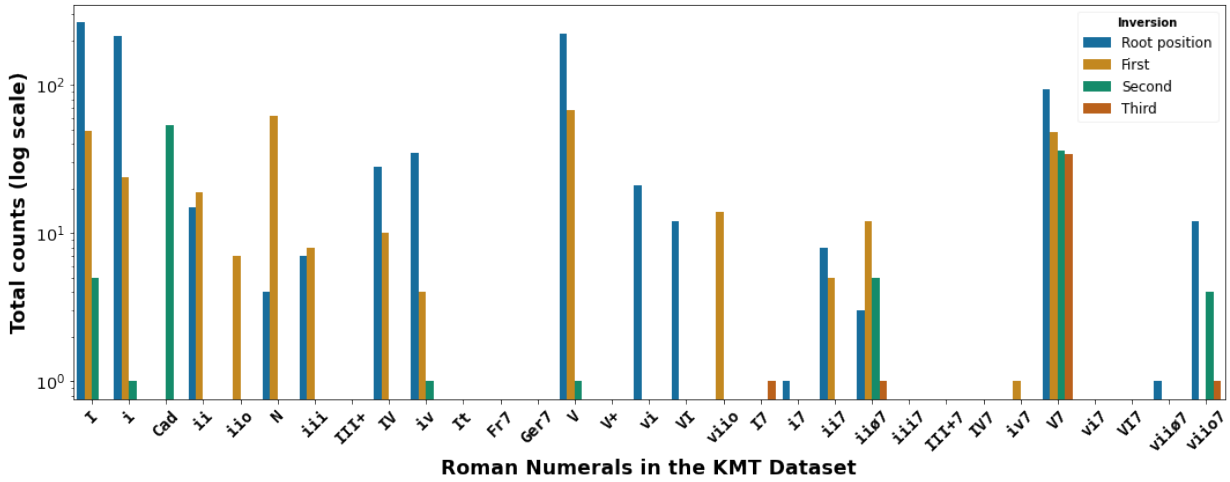


Figure 4.7: *All the RNA labels taken from the KMT dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

Figure 4.8 shows the distribution of the keys in the KMT dataset. The dataset spans a total of 37 keys out of the 38 in the vocabulary, missing only the occurrence of the key of **B**𝄫. Among all datasets, this is perhaps the most interesting in terms of the key distribution. It was discussed above that the vocabulary of Roman numerals in this dataset was visibly smaller than all other datasets. The opposite happens with the distribution of keys, especially when considering the presence of a key in the context of a modulation. This dataset is more uniform and the use of the vocabulary more extensive. This is probably because the dataset is a compilation of textbook examples of modulations. Particularly, one of the sources of this dataset, Reger (1904), contains an extensive set of modulations in different, sometimes unusual, keys. This is reflected in the key vocabulary observed here.

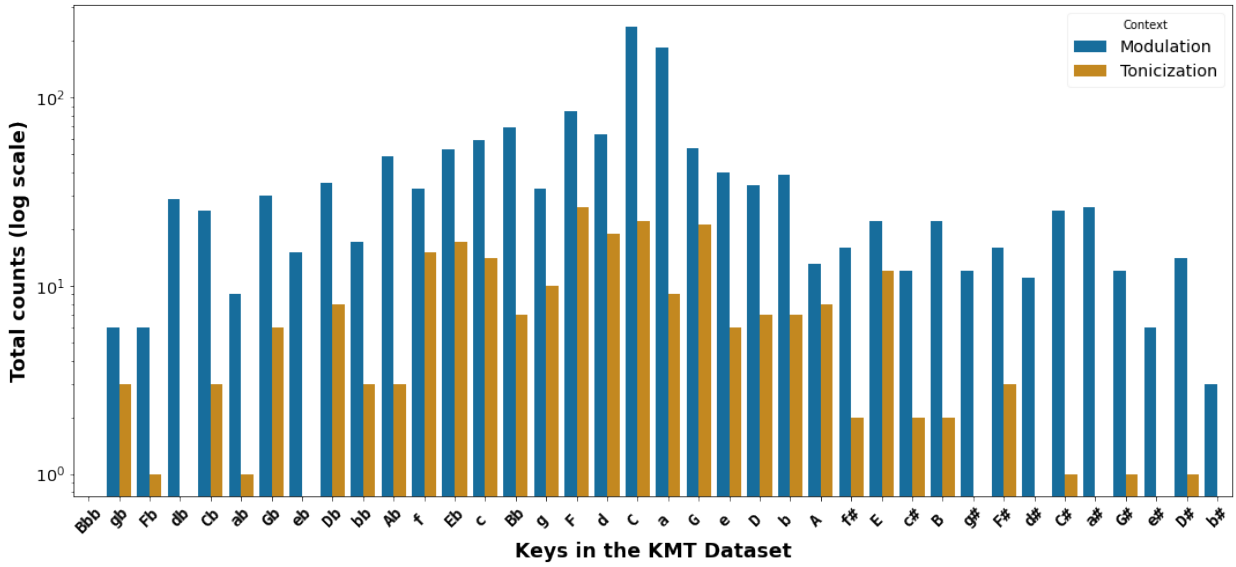


Figure 4.8: *All the keys spanned by the **RNA** annotations of the **KMT** dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations* .

### 4.2.5 Mozart Piano Sonatas (MPS)

The **MPS** dataset was introduced by Hentschel, Neuwirth, and Rohrmeier (2021). The dataset consists of musical scores in **MusicXML** format, as well as annotations of the harmony and cadence information. The format of the annotations is similar to the one in the revised (i.e., version 2) **ABC** dataset.

**MPS Preprocessing.** The scores were preprocessed before aggregated to the dataset.

The first step was to detect collisions in the position of two annotations. Collisions happen because the "Chord" objects in MuseScore do not encode a position when they are exported to **MusicXML**. The location of the annotations is inferred from the surrounding note. However, in instances where an annotation is added to a longer note (e.g., half way of a whole note), the position is impossible to retrieve from the **MusicXML** file. One way to solve this problem is to create an additional part in MuseScore with a sufficient note/rest onsets to accomodate the annotation's location. I processed all the scores that required this modification (all but 8 scores in the dataset). This process removed the majority of the collisions, however, a few of them

persisted. Additional collisions happen when a measure is divided by a repetition bar (e.g., in anacrusic music). In these instances, the offset of the annotation resets for the portion of the measure beyond the repetition bar and it collides with the annotations before the repetition bar. These second round of collisions were removed by removing repetition bars, effectively avoiding to reset the offset of the annotations within a single measure.

#### 4.2.5.1 Summary of the MPS Dataset

According to the count performed in this dissertation, the **MPS** dataset has a total duration in quarter notes of 22,305, across 7,465 measures. After preprocessing and preparation, the **MPS** dataset contributed 15,865 **RNA** chord annotations, which have an average harmonic rhythm of 1.41 quarter notes (♩).

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.9. In this dataset, all occurrences of **N** chords are in first inversion, which is not the case for any other dataset. Although this is also the most common occurrence of **Neapolitan** chords, so it is consistent with the musical practice of the period. The **MPS** dataset is also one of two datasets lacking any examples of $\mathbf{III}^{+}$ chords (the other one being **KMT**). Unlike **KMT**, however, **MPS** does present examples of $\mathbf{V}^{+}$ chords. This is an interesting case, because a $\mathbf{III}^{+}$ chord is an enharmonic of a $\mathbf{V}^{+}$ chord in the same key, as both are made of the same **pcset** but with different note spellings. For this reason, when designing the vocabulary, I decided to make $\mathbf{III}^{+}$ an exclusive chord of the minor mode, and $\mathbf{V}^{+}$ an exclusive chord of the major mode, so both labels can coexist.[7] What this indicates here, is that all occurrences of augmented triads found in **MPS** occur in a major-key context, which results in only $\mathbf{V}^{+}$ labels. Perhaps this suggests that Mozart only used augmented triads in major keys, although that musicological claim would require further inspection beyond the scope of this dissertation.

Figure 4.10 shows the distribution of the keys in the **MPS** dataset. The dataset spans a total of 26 keys out of the 38 in the vocabulary, making it the dataset with the smallest key

7. See Section A.2 for further discussion on $\mathbf{III}^{+}$ and $\mathbf{V}^{+}$ chords.

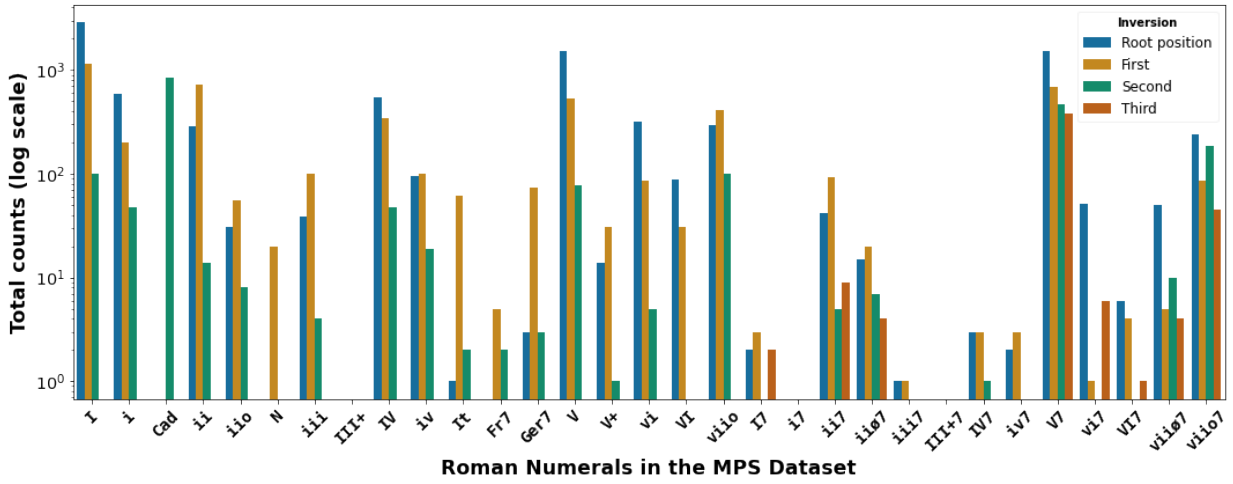


Figure 4.9: *All the RNA labels taken from the MPS dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

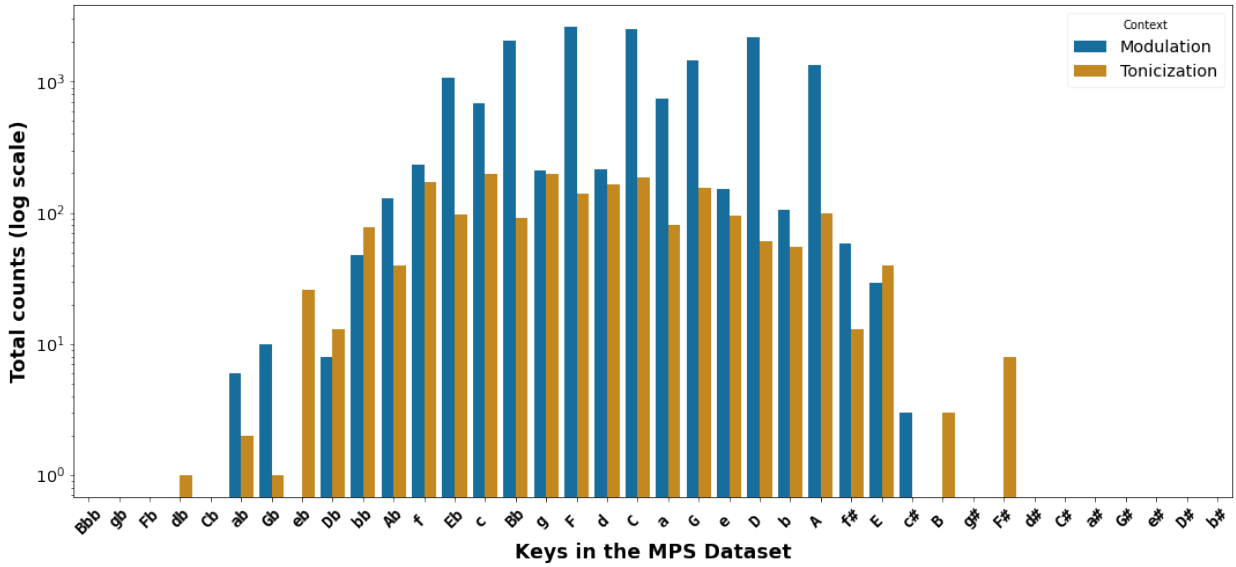


Figure 4.10: *All the keys spanned by the RNA annotations of the MPS dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations .*

vocabulary, among the datasets presented here. As expected, most of the keys found in this dataset lie within the center of the **line of fifths**. The most extreme keys in the dataset are **d**♭ and **F**♯, both in the context of tonicizations.

### 4.2.6 Theme and Variation Encodings with Roman Numerals (TAVERN)

The **TAVERN** dataset was introduced by Devaney et al. (2015). This dataset consists of **RNA** annotations for sets of piano theme and variations by Beethoven and Mozart. The dataset contains a total of 27 sets of theme and variations, 10 by Mozart and 17 by Beethoven. Unlike most datasets of this kind, the **TAVERN** dataset provides two sets of annotations for each piece. Three experts annotated a set of 18 theme and variations each. Presumably, a set of 9 pieces annotated by annotators 1 and 2, a set of 9 pieces by annotators 2 and 3, and a set of 9 pieces by annotators 1 and 3. The full distribution of the pieces among annotators is not provided, but the dataset provides both sets of annotations per piece, as well as an integrated analysis resulting from combining both annotations (and resolving any differences between annotators). All three annotators were PhD-level music theorists.

The dataset was divided in phrases rather than pieces. Thus, each file in the dataset corresponds to a phrase. The dataset consists of a total of 1,060 phrases.

#### 4.2.6.1 Format of TAVERN

The annotations are provided in the ****harm** format, which accompanies the ****kern** encodings of the musical scores. In addition to the Roman numeral labels, the annotators also included the **Tonic, Subdominant, and Dominant (TSD)** function of each chord in a separate column of the Humdrum files.[8]

#### 4.2.6.2 Summary of the TAVERN Dataset

According to the count performed in this dissertation, the **TAVERN** dataset has a total duration in quarter notes of 40,899, across 15,534 measures. After preprocessing and preparation,

8. Referred to as *spines* in the Humdrum format.

the TAVERN dataset contributed 24,544 RNA chord annotations, which have an average harmonic rhythm of 1.67 quarter notes (♩).

The distribution of the RNA annotations and their inversions are shown in Figure 4.11. The TAVERN dataset is one of the few datasets (the others being ABC and WiR) that provide examples of $\mathbf{III}^{+7}$ chords, which are the least common type of chord in the vocabulary.

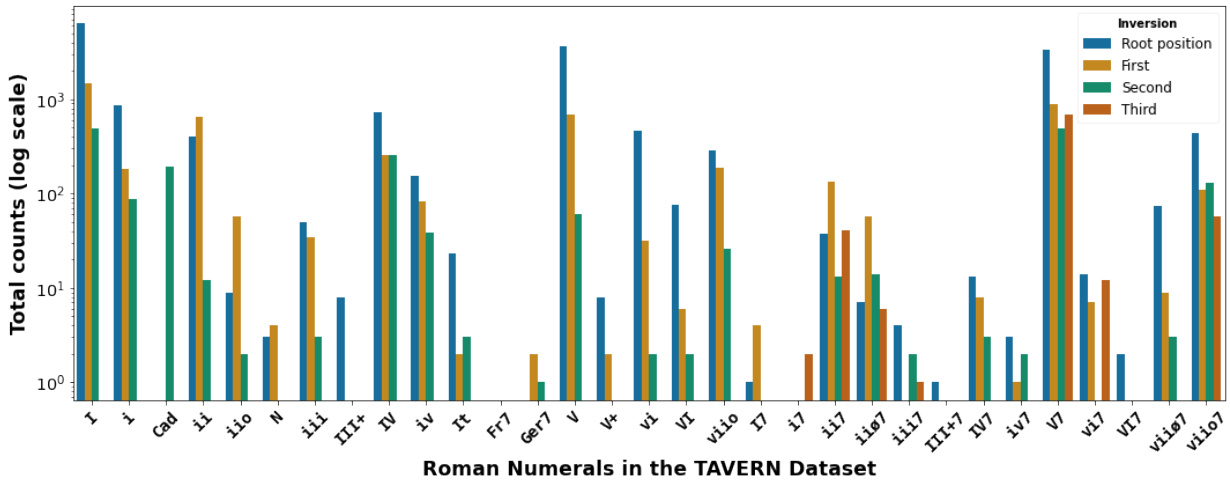


Figure 4.11: *All the RNA labels taken from the TAVERN dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

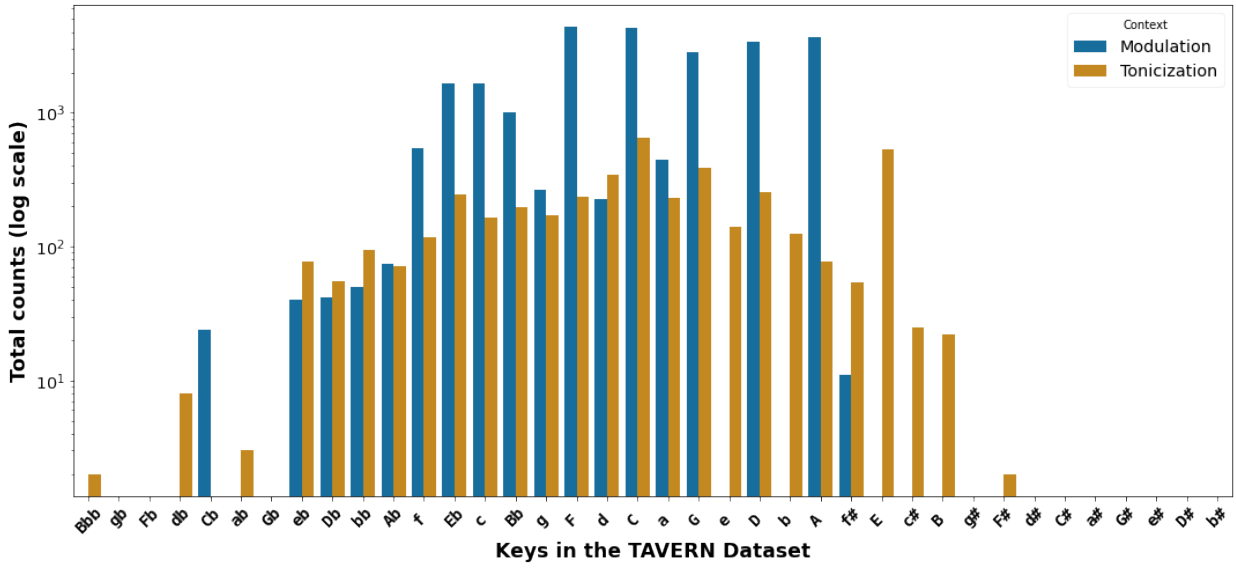


Figure 4.12: *All the keys spanned by the RNA annotations of the TAVERN dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations .*

Figure 4.12 shows the distribution of the keys in the **TAVERN** dataset. The dataset spans a total of 27 keys out of the 38 in the vocabulary $\mathcal{K}$ (see Section A.4). After **MPS**, it has the second-smallest key vocabulary among the datasets presented here. The vocabulary is also slightly skewed towards the "flatter" side of the **line of fifths**. The "sharpest" key that appears in a modulation is **f♯** (3 sharps), whereas the flattest key is **C♭** (7 flats).

### 4.2.7 When in Rome (WiR)

The **WiR** dataset is a growing repository of **RNA** annotations[9] introduced by Gotham, Tymoczko, and Cuthbert (2019) and Gotham and Jonas (2022). This dataset consists of original annotation work as well as conversions of existing datasets. The purpose of the conversions is to present external digital formats in the **RomanText** format.

#### 4.2.7.1 Format of WiR

All annotations in the repository are provided in the **RomanText** format. In cases where the annotations were converted from an external format (e.g., **DCML**'s standard), the conversions are generally reviewed/corrected according to the conventions of the **RomanText** format.

#### 4.2.7.2 Converted Corpora

The datasets that have been converted in the **WiR** dataset include, chronologically, **ABC**, **TAVERN**, and **HaydnSun**. The **ABC** was converted and manually corrected by Gotham, which was presented in the first mention of **WiR** (Gotham, Tymoczko, and Cuthbert 2019). Similarly, **TAVERN** was converted by Gotham and appended to the repository. **HaydnSun** was converted by Nápoles López, with the purpose of increasing the reach of the annotations among the users of the **RomanText** format.

9. https://github.com/MarkGotham/When-in-Rome

#### 4.2.7.3 Original Corpora

There is a set of analyzed corpora that is exclusive of the **WiR** dataset. The analyzed corpora were annotated by an extensive group of annotators and compiled into **RomanText** in Gotham, Tymoczko, and Cuthbert (2019).

**OpenScore Lieder.** A collection of annotated 19th-century French and German songs from the OpenScore corpus in Gotham and Jonas (2022).

**Well-Tempered Clavier.** The preludes of the first book from Bach's **[The] Well-Tempered Clavier (WTC)**.

**Bach Chorales.** A set of 20 annotated Bach chorales.

**Monteverdi Madrigals.** The set of Monteverdi madrigals in Books 3 to 5 (48 pieces).

#### 4.2.7.4 Summary of the WiR Dataset

According to the count performed in this dissertation, the **WiR** dataset has a total duration in quarter notes of 29,951, across 9,099 measures. After preprocessing and preparation, the **WiR** dataset contributed 17,734 **RNA** chord annotations, which have an average harmonic rhythm of 1.69 quarter notes (♩).

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.13. This dataset is one of the two datasets (the other one being **ABC**) that provides at least one example of every Roman numeral class in the vocabulary. In addition to that, the **WiR** dataset is the only dataset that provides examples in third inversion of every seventh chord in the vocabulary, and the only dataset providing examples of any inversions at all in a $\mathbf{III}^{+7}$ chord.

Figure 4.14 shows the distribution of the keys in the **WiR** dataset. The dataset spans a total of 37 keys out of the 38 in the vocabulary $\mathcal{K}$ (see Section A.4). This is as many keys as **KMT** and more than any other dataset. The occurrence of keys around the center of the **line of**

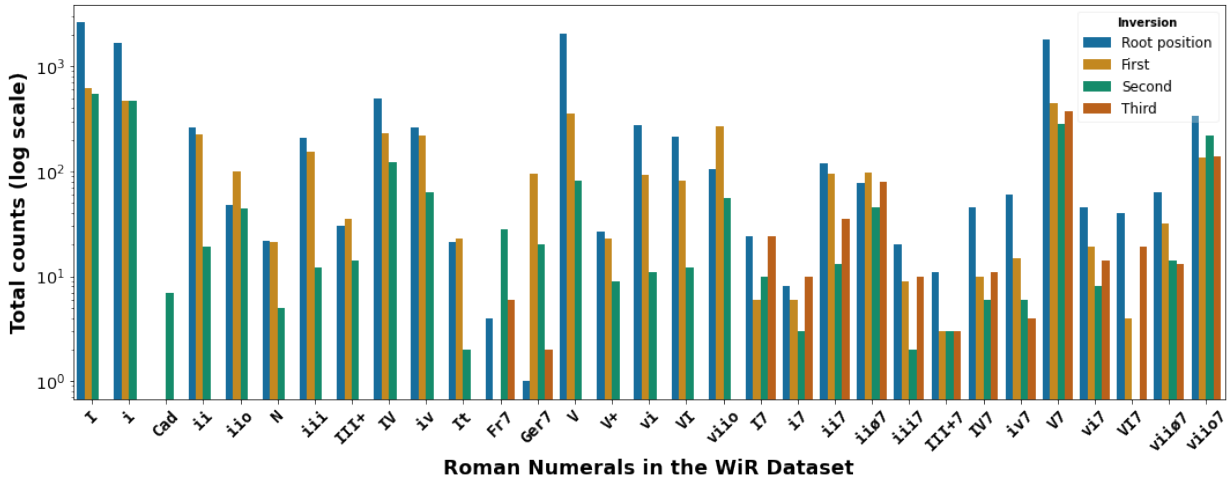


Figure 4.13: *All the RNA labels taken from the WiR dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

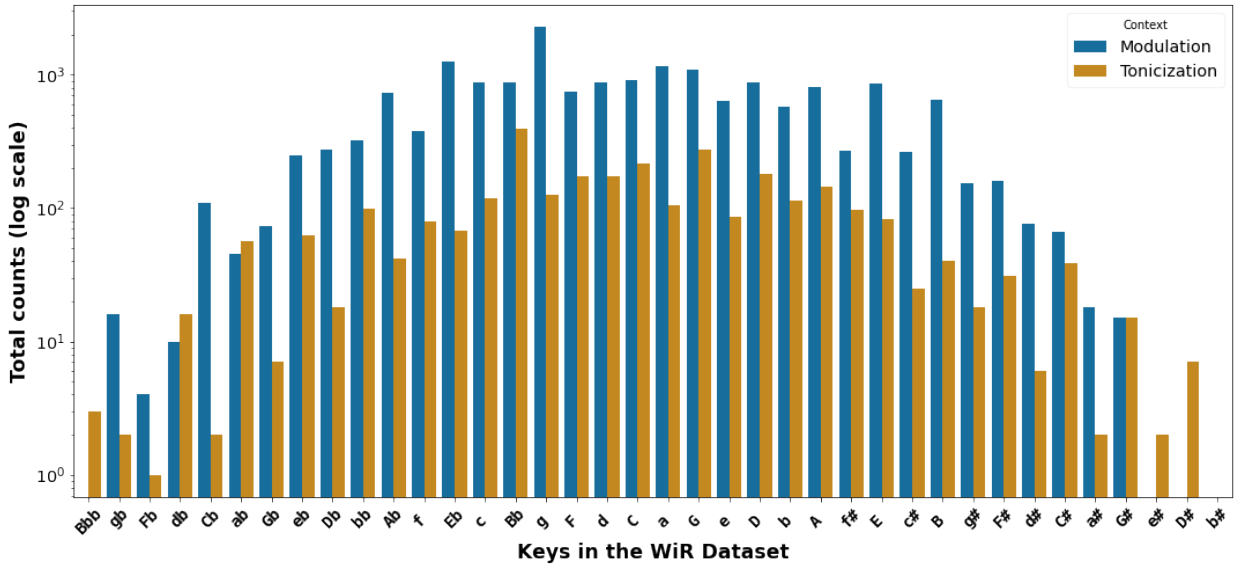


Figure 4.14: *All the keys spanned by the RNA annotations of the WiR dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations.*

**fifths** is higher and it slowly smooths toward the extremes. The only key of the vocabulary that is missing from this dataset is **b♯**. The distribution of modulations appears to be less uniform than in **KMT**. It is possible that this is because the examples of **WiR** are real musical examples, instead of textbook examples of modulations.

## 4.3 The Aggregated Dataset

The "aggregated" dataset is the resulting collection of datapoints after applying the general preparation to each of the publicly available datasets described above. When these datasets are aggregated, the resulting dataset has a total duration in quarter notes of 175,930, across 58,393 measures. In terms of annotations, the aggregated dataset has 104,926 **RNA** chord annotations, which have an average harmonic rhythm of 1.68 quarter notes (♩).

The distribution of the **RNA** annotations and their inversions are shown in Figure 4.15.

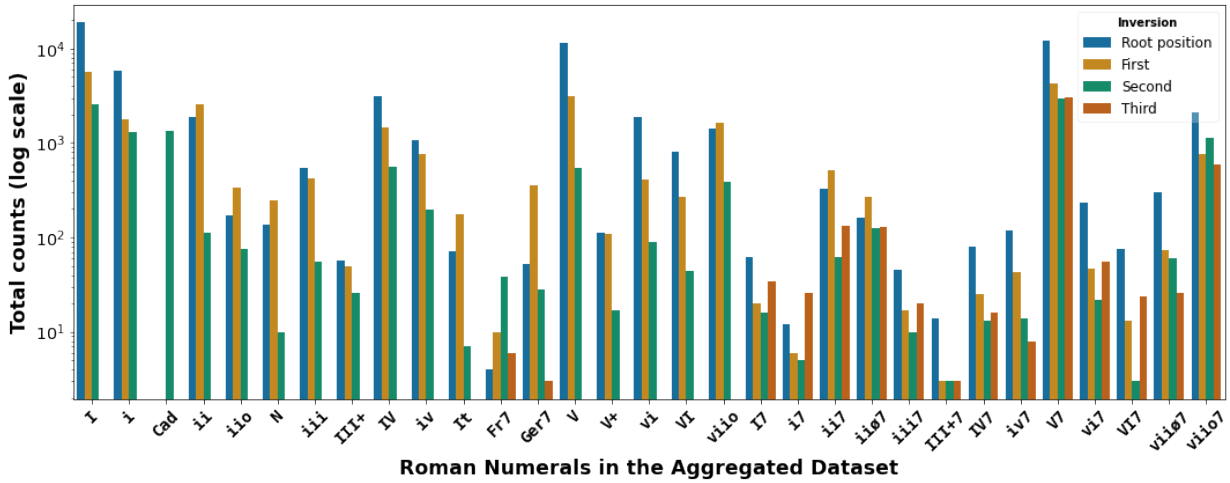


Figure 4.15: *All the **RNA** labels in the aggregated dataset. Each bar indicates the counts of the Roman numeral class in different inversions.*

Figure 4.16 shows the distribution of the keys in the aggregated dataset.

After the data aggregation process, the next step is to split the data into training, validation, and test sets that can be used to run supervised learning experiments.

## 4.4 Generating Training, Validation, and Test Splits

In supervised learning machine learning methods, it is customary to evaluate the generalization of a model on an unseen portion of data. This is necessary as models tend to **overfit** on the data they are trained with.

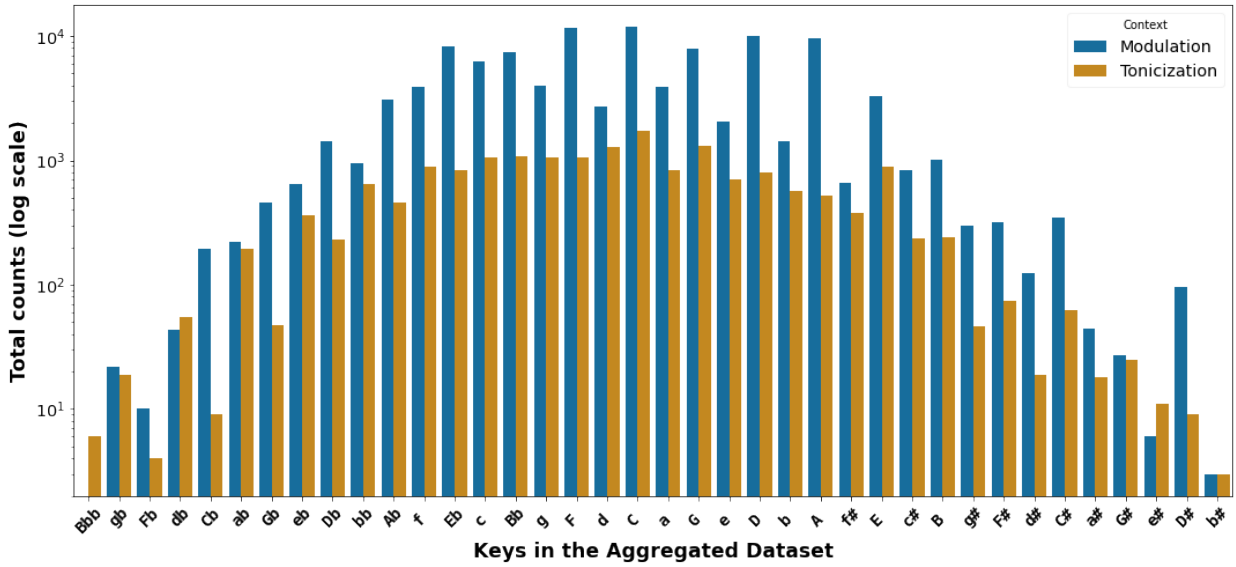


Figure 4.16: *All the keys spanned in the aggregated dataset. For each key, the counts indicate which ones correspond to modulations (local key regions) and tonicizations.*

Furthermore, there are several ways to designate that portion of unseen data. In this research, I opted for training, validation, and test sets. The training set is the data used for training the model. The validation set is to verify the generalization of the model while tuning hyper-parameters, designing the neural network architecture, or making other decisions affecting the model. The test set is an unseen portion of data that remains unused throughout the entire process of designing model, and it is used once (and only once) to evaluate the generalization of the final model.

In some datasets, such as **BPS**, these splits were already provided by other researchers. In those instances, the splits were respected for direct comparison with their models. When not specified, the splits were generated randomly, providing around 70% of the data for training, 15% for validation, and 15% for testing.

Once all datasets have been aggregated, standardized, and split into three sets, the dataset is ready for training the model and running the experiments.

The last step, described in Section 4.5, is to perform data augmentation on the training set.

## 4.5 Data Augmentation

The annotations in the aggregated dataset account for nearly 105,000 chords. However, the chord classes are imbalanced and there may not be examples of a chord class across all the keys in the vocabulary. One way to compensate for this is by transposing the training examples to different keys, which is a popular data-augmentation method in **ARNA**, **Automatic Chord Recognition (ACR)**, and **Global-Key Estimation (GKE)** research. This method is applied to the aggregated dataset to increase the number of examples and balance the distribution of chords across all keys.

In addition to transposition, an additional method for data augmentation is proposed, which consists of generating artificial examples from the chord annotations presented in the dataset. This method is inspired by pedagogical exercises often found in harmony textbooks.

### 4.5.1 Synthesis of Artificial Training Examples

In tonal music theory textbooks, exercises like the one shown in Figure 4.17 are common. Notice that the exercise indicates the key, time signature, duration of the chords (i.e., harmonic rhythm), and Roman numeral labels. Moreover, no indication of the arrangement of the chords is given, which is left to the student to decide.

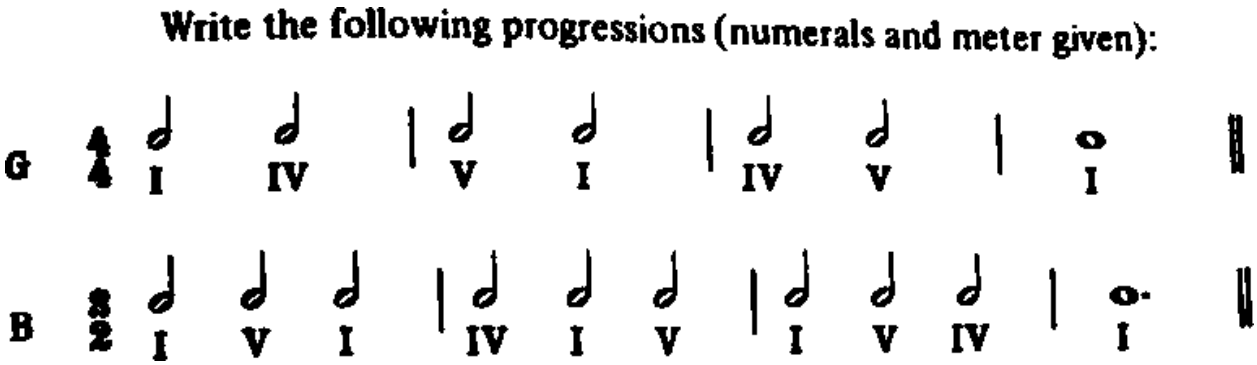


Figure 4.17: *Harmony exercise in Hindemith (1943, 7), where a student realizes the chords indicated by the Roman numerals and note durations.*

The information provided in the exercise is similar to the one contained in a digitized **RNA** file in the **RomanText** (or similar) format. Thus, an "exercise" can be synthesized from the annotations and aggregated to the dataset as a new training example for the same set of **RNA** annotations. This approach was implemented as a data-augmentation workflow to increase the number of examples in the training set.

#### 4.5.1.1 Block Chord Sequences

In harmony exercises like the one shown in Figure 4.17, the goal is often to **realize** the chords in a four-part harmonization that respects the voice-leading rules explained in the textbook, as shown in Figure 4.18.

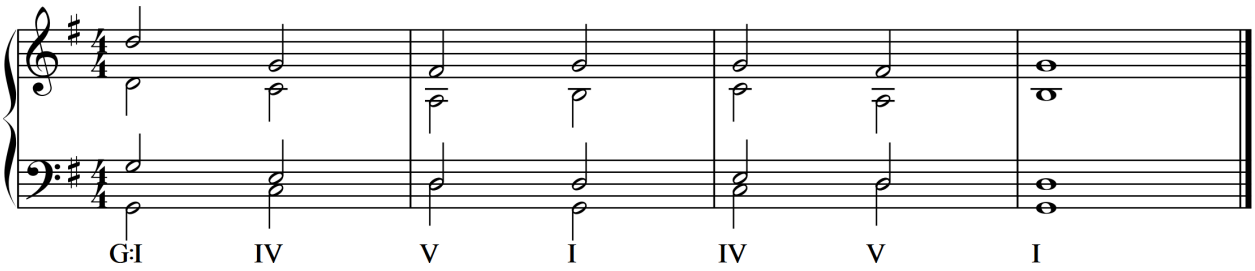


Figure 4.18: *A possible solution to the first harmony exercise proposed Figure 4.17.*

Although it is possible to algorithmically generate such harmonizations with awareness of voice-leading rules, this is computationally expensive using a rule-based voice-leading algorithm.[10] Moreover, in the neural network architecture of this dissertation, **AugmentedNet** (see Chapter 5), the voice-leading information is not taken into account, because the octave of the notes is not encoded. For that reason, an alternative approach is to synthesize training examples as simple block chords, such as the ones shown in Figure 4.19. These block chords are useful because they indicate the notes of the chord and which of those notes is at the bottom (i.e., the lowest-sounding one).

We introduced a data-augmentation approach of this kind in Nápoles López and Fujinaga (2020b), where it showed an improved accuracy when predicting the chords in a dataset of

10. https://github.com/napulen/romanyh

Bach chorales when compared to using only the "real" training set. Nevertheless, in preliminary experiments with the datasets presented in Section 4.2, this approach was not sufficient to work with the more complicated textures found, for example, in piano sonatas and string quartets.

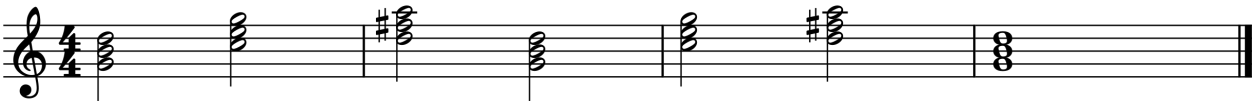

Figure 4.19: *Another version of Figure 4.18 with block chords, without any consideration for voice-leading rules.*

A solution to this problem, originally proposed by co-author Gotham in Nápoles López, Gotham, and Fujinaga (2021) was to texturize the block chords to approximate the textures found in more complicated music. I approached this solution by implementing texturization patterns on top of the block chords.

#### 4.5.1.2 Texturization Patterns

The block chords have all a homorhythmic texture. Thus, they lack the textural complexity of polyphonic or homophonic pieces of music, which are often found in the aggregated dataset.

For this reason, an automatic texturization approach is proposed to process block-chord annotations, turning them into more realistic data-augmentation examples.

I explored three texturization patterns, which were designed to deal with specific problems found in the real examples: labeling arpeggios and chords implied in a melodic line, labeling chords where the bass is played in isolation from the upper notes of the chord, and labeling chords where the bass is temporally displaced from the location of the chord. The patterns designed to deal with these problems, respectively, are called the *Alberti bass* pattern, the *bass split* pattern, and the *syncopation* pattern.

**Alberti Bass.** This pattern consists of a 4-note melodic pattern with a pitch contour of *low-high-middle-high*. It is used to turn block chords into melodic lines. The goal is to occasionally

play chords using a monophonic texture. Figure 4.20 shows an example of this texturization pattern.

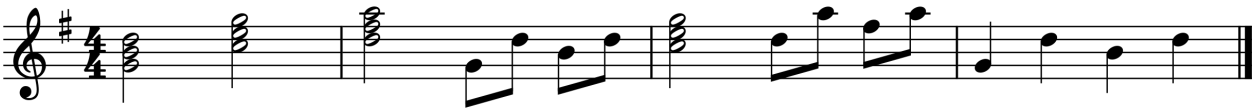

Figure 4.20: *A random texturization of some of the block chords in Figure 4.19 with an* Alberti bass *pattern.*

**Bass Split.** This is a pattern where the original chord duration is divided by half, playing the bass in the first half, and the remaining notes in the second. The goal is to occasionally separate the bass from all other notes. Figure 4.21 shows an example of this texturization pattern.

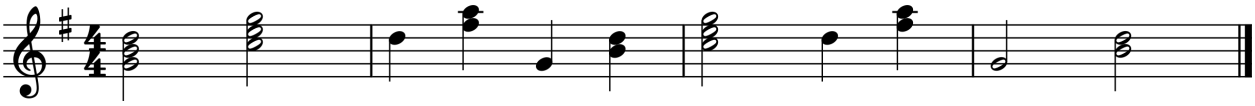

Figure 4.21: *A random texturization of some of the block chords in Figure 4.19 with a* Bass split *pattern.*

**Syncopation.** This is a pattern where the highest note is played first, followed by the rest of the notes, played in syncopation. The goal is to occasionally shift the onset of the bass from the onset of the chord. It is intended to provide the trainable system with examples where the location of the bass is different from the location of the chord onset. Figure 4.22 shows an example of this texturization pattern.

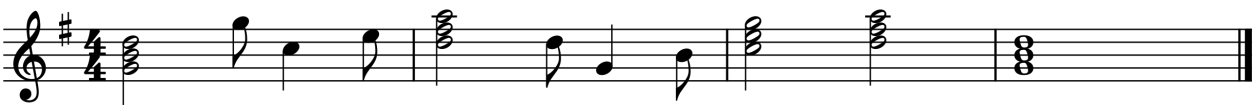

Figure 4.22: *A random texturization of some of the block chords in Figure 4.19 with a* Syncopation *pattern.*

In the experiments performed in this dissertation, these patterns were applied randomly and recursively to the annotations. A hypothetical training example synthesized from the Hindemith annotations is shown in Figure 4.23. This example combines the three patterns.[11]

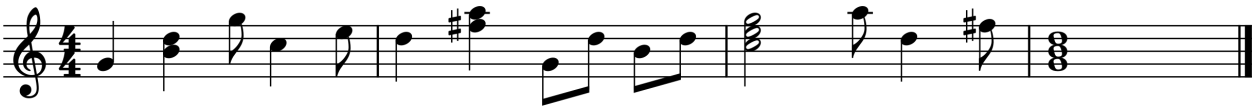

Figure 4.23: *A synthesized training example from the annotations in the first exercise of Figure 4.17. The example has been texturized with the three texturization patterns applied randomly.*

The results of this data-augmentation technique, applied with and without the use of key transpositions is discussed in Section 6.2.

As a final note on the data augmentation technique, notice the lack of any **nonchord tone**s in the synthesis. This was a chosen limitation to keep the complexity of the texturization algorithm low.

11. One of the possibilities is also to leave the block chord unmodified, as depicted in measures 3 and 4 of Figure 4.23.

# Chapter 5

# Model Design

This chapter introduces the design and implementation of the end-to-end system for **Automatic Roman Numeral Analysis (ARNA)** proposed in this dissertation. The main component of this system is a **Convolutional Recurrent Neural Network (CRNN)** dubbed **AugmentedNet**, which is designed to provide predictions of different tonal attributes. These predictions are used to produce the final **ARNA** annotations. Section 5.1 describes the input of the system. Section 5.2 describes the convolutional layers of the neural network model. Section 5.3 describes the dense and recurrent layers of the neural network model. Section 5.4 describes the multitask learning configurations in the output layer of the neural network model. Section 5.5 describes the methods to generate the Roman numeral labels from the predictions of the network.

## 5.1 Input

The goal of this system is to process any symbolic music files provided as input, annotating them with Roman numeral labels. In practice, there are a few limitations and constrains regarding the input files, which are described in Section 5.1.1. After a score has been imported into the system, it is divided in sequences of a fixed number of timesteps, with each of those timesteps sampled at fixed-duration note events. This step is described in Section 5.1.2. The

sampled scores of fixed number of timesteps are encoded into a set of vector representations, described in Section 5.1.3. These vector-form representations become the input to the neural network model.

### 5.1.1 Importing a Digital Music Score

As mentioned in Section 3.1.1, there are several symbolic music formats, which prioritize different aspects of the musical representation (e.g., engraving, performance, or pitch information). When an **ARNA** system is designed, some considerations about symbolic music files are important, such as the support for a given format and/or the quality of conversions. Additionally, some formats (e.g., **Musical Instrument Digital Interface (MIDI)**) lack the minimum information required to encode the representations needed by the neural network.

#### 5.1.1.1 Supported Formats

Unfortunately, aggregating datasets of various symbolic music formats (and annotated by different people) often leads to discrepancies and misrepresentation in digital systems (Nápoles López, Viglensoni, and Fujinaga 2018, 2019). In the **ARNA** model proposed here, the *music21* Python library (Cuthbert and Ariza 2010) was used to process incoming symbolic music files. The support for some formats (e.g., **Music Encoding Initiative (MEI)**) is currently inadequate for guaranteeing that the score representation will be properly imported. Furthermore, Micchi, Gotham, and Giraud (2020) found that the spelling of the notes was meaningful in their proposed **ARNA** machine learning model. We followed a similar approach in Nápoles López, Gotham, and Fujinaga (2021). Thus, in the proposed system here, pitch spelling is also an expected property of the inputs. For that reason, **MIDI** files are not acceptable inputs. Thus, the allowed inputs are restricted to **MusicXML** and ****kern** symbolic music formats.

### 5.1.2 Sampling of the Score

Neural network models often work with a lossy representation of the original input. For example, by lowering the original resolution of an image or the sampling rate of an audio file.

This is also the case in symbolic music scores. A symbolic music file often includes characteristics of the music notation, such as articulations, dynamics, tempo, beaming, and stem directions. However, this information is frequently removed from the encoded representation sent to a neural network model.[1] In the proposed system, the encoding of the score is limited to fragments of music of fixed length and sampled at regularly-spaced note durations.

#### 5.1.2.1 Note Duration of each Timestep

Following the practice of Micchi, Gotham, and Giraud (2020), the **Thirty-second notes (𝅘𝅥𝅯s)** note was taken as the reference value used to sample the input score. One drawback of this sampling method and reference value is that most tuplets and notes shorter than a 𝅘𝅥𝅯s cannot be appropriately encoded. These note durations are thus approximated. However, looking at a distribution of note durations across the dataset, tuplets and duration values shorter than a 𝅘𝅥𝅯s note are rare. The majority of the note durations being **Sixteenth notes (𝅘𝅥𝅰s)** notes, as shown in Figure 5.1.

#### 5.1.2.2 Number of Timesteps

Neural network models, particularly **Recurrent Neural Networks (RNNs)** or Transformer-based models are often used to process sequential information (e.g., timeseries). Symbolic music files are naturally sequential data. Although it is possible to deal with sequences of arbitrary length, it is common to set a fixed number of timesteps in a given sequence. In the architecture proposed, this is the case, where the fixed length input is set to 640 𝅘𝅥𝅯s note timesteps. That is, the longest musical encoding that the system can process is 80 𝅗𝅥s notes long (equivalent of 20

1. Notice that this does not necessarily mean that such information is detrimental to the model. Most likely, the omission of the information is related to lowering the number of trainable parameters in the network, as well as decreasing the data-curation effort.

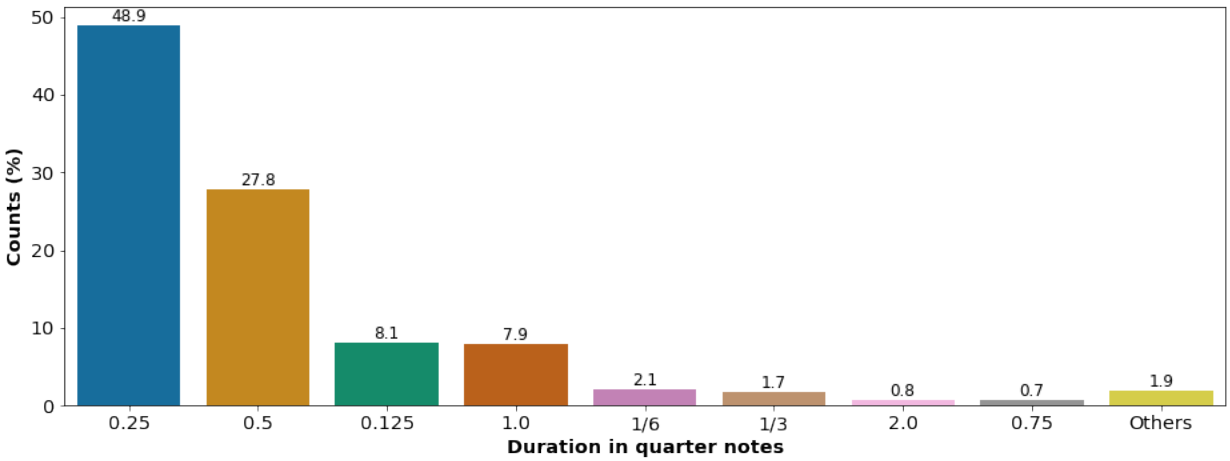


Figure 5.1: *Distribution of duration values across the training portion of the aggregated dataset. The durations are indicated in multiples of a* ***Quarter notes (𝅘𝅥s)*** *note. Thus, the sampling reference 𝅘𝅥𝅯s note is indicated as a duration of 0.125 𝅘𝅥s notes.*

measures in a **Common times (Cs)** time signature). For scores that are longer, the input is divided in nonoverlapping sequences of 640 timesteps. For scores that are shorter, the input is padded with a special symbol after the entire musical content is encoded, in order to meet the exact length of 640 timesteps.

### 5.1.3 Encoding the Score

The steps in the previous sections describe the methods to reduce a symbolic music file into a fixed-length sequence of pitch values. As mentioned in Section 5.1.2, the score in the symbolic music file is stripped from some of its original information when it is encoded for the neural network model. Furthermore, the score is sampled at regular note intervals of 𝅘𝅥𝅯s notes, encoding the musical information at each of those timesteps. The encoding dispatched to the neural network takes the form of three *input representations*, which are described in Section 5.1.4. However, these representations require a method to encode pitch spellings and onsets as numerical vectors. Thus, the methods to encode pitch spellings and onsets are introduced here.

#### 5.1.3.1 Encoding Notes with Spelling

Most tonal music analysis neural network models collapse all enharmonic spellings of the same note, as shown in Equation 5.1.

$$\text{F}\sharp = \text{G}\flat \quad \text{enharmonic equivalence} \tag{5.1}$$

$$\text{F}\sharp \neq \text{G}\flat \quad \text{enharmonic nonequivalence} \tag{5.2}$$

In **AugmentedNet**, however, the spelling of a pitch is encoded and taken into account, as shown in Equation 5.2. Two methods are described to encode pitch spellings in this way.

**35 One-Hot Encoding.** Perhaps the easiest method to encode a pitch spelling is to assume that only a number of accidentals are allowed next to a generic note letter. The use of **A "sharp" accidental (♯)** and **A "flat" accidentals (♭s)** accidentals is very common. The use of **A "double-sharp" accidentals (×s)** and **A "double-flat" accidentals (𝄫s)** accidentals is less common. Any accidental sharper than **×s** or flatter than **𝄫s** is rare. Thus, a reasonable vocabulary for spelled pitch classes[2] is one with 35 classes, from C𝄫 to B×, as shown in Equation 5.3. A "one-hot" encoding of this form means that, at each timestep, only one of the 35 possible classes of pitch spelling will be set as active (i.e., "hot").

$$\begin{aligned}\mathcal{S}_{35} = \{&\text{C}\doubleflat, \text{D}\doubleflat, \text{E}\doubleflat, \text{F}\doubleflat, \text{G}\doubleflat, \text{A}\doubleflat, \text{B}\doubleflat,\\ &\text{C}\flat, \text{D}\flat, \text{E}\flat, \text{F}\flat, \text{G}\flat, \text{A}\flat, \text{B}\flat,\\ &\text{C}, \text{D}, \text{E}, \text{F}, \text{G}, \text{A}, \text{B},\\ &\text{C}\sharp, \text{D}\sharp, \text{E}\sharp, \text{F}\sharp, \text{G}\sharp, \text{A}\sharp, \text{B}\sharp,\\ &\text{C}\times, \text{D}\times, \text{E}\times, \text{F}\times, \text{G}\times, \text{A}\times, \text{B}\times\}\end{aligned} \tag{5.3}$$

2. Note that the octave is ignored.

This method was used in Micchi, Gotham, and Giraud (2020) and Micchi et al. (2021) to encode the inputs of two **ARNA** models. Using this approach, a 35-dimensional pitch-spelling vector is encoded every timestep, indicating the active pitch (or pitches, if several) among the 35 available classes. One of the limitations of this method is that it cannot encode any notes sharper than a 𝄪s or flatter than a 𝄫s. Additionally, the resulting vector is almost three times larger than an encoding based on pitch classes, which only requires 12 classes,[3] as shown in Equation 5.4.

$$\mathcal{C} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\} \tag{5.4}$$

**19 Two-Hot Encoding.** Another encoding method, first proposed in Nápoles López, Gotham, and Fujinaga (2021), provides an alternative way to encode pitch spellings. Instead of encoding ♯ and ♭s notes explicitly, it extends the pitch class vector representation, by concurrently encoding a generic note letter class, as shown in Equation 5.5.

$$\begin{aligned} \mathcal{L} &= \{\mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{G}, \mathrm{A}, \mathrm{B}\} \\ \mathcal{S}_{19} &= \mathcal{C} \times \mathcal{L} \end{aligned} \tag{5.5}$$

This encoding method results in duples of the form $(c, l)$, where $c \in \mathcal{C}$ and $l \in \mathcal{L}$. For example, C♯ is represented by the duple $(1, \mathrm{C})$, whereas D♭ is represented by the same pitch class but a different generic note letter, $(1, \mathrm{D})$. D♮ has the same generic note letter but a different pitch class, $(2, \mathrm{D})$, and so on. Using this representation, a spelled note, beyond two flats and two sharps can be encoded with a 19-feature vector. Notice, however, that this vector is a two-hot encoding representation. This means that at any given timestep, each pitch spelling will be encoded as two active classes: one for the pitch class and one for the generic note letter. The output representations in the **Multitask Learning (MTL)** configuration (see Section 5.4)

3. When enharmonic equivalence is assumed in a **Twelve-Tone Equal Temperament (12-TET)** system, a set of 12 pitch classes spans all the note classes of the Western chromatic scale.

use the first encoding method. The input representations described in Section 5.1.4 employ the second encoding method.

#### 5.1.3.2 Encoding Measure, Note, and Chord Onsets

A way to encode onsets is often needed in inputs and outputs of an **ARNA** model. For example, to indicate where a measure, note, or chord begins. In order to encode this kind of information, the following representation is proposed, with 7 features.

$$\mathcal{O} = \{\checkmark, \text{𝅘𝅥𝅯}, \text{𝅘𝅥𝅮}, \text{♪}, \text{♩}, \text{𝅗𝅥}, \text{𝅝}\} \tag{5.6}$$

$$|\mathcal{O}| = 7 \tag{5.7}$$

In this encoding method, ✓ describes an *onset*, and each of the subsequent classes represents the time elapsed since the onset, measured in note durations. In the input representation **Onsets14**, it is used to encode measure and note onsets (see Section 5.1.4.3). In the output representation **HarmonicRhythm7**, it is used to predict the onset of **Roman Numeral Analysis (RNA)** labels (see Section 5.4.9).

The two methods discussed for encoding pitch spellings and onsets, based on the vocabularies $\mathcal{S}_{19}$ and $\mathcal{O}$, respectively, are used in the three *input representations* dispatched to the neural network.

### 5.1.4 Input Representations

Using the fixed-length sequences and the vocabularies of pitch spelling (see Section 5.1.3.1) and onsets (see Section 5.1.3.2), the score can be encoded into a set of three *input representations*, which are dispatched to the trainable layers of the neural network. These input representations are the lowest-sounding pitch of a timestep (**LowestNote19**), all-sounding pitches of

each timestep (**Notes19**), and the measure-and-note onsets (**Onsets14**). Each representation is described below.

#### 5.1.4.1 Lowest Sounding Note

The first input representation, **LowestNote19**, encodes the lowest-sounding note at each timestep. The encoding of pitch spelling is done using the method based on duples of pitch class and generic note letters, introduced in Section 5.1.3.1. When this encoding strategy is applied, it results in a two-hot encoded vector like the one shown in Figure 5.2.

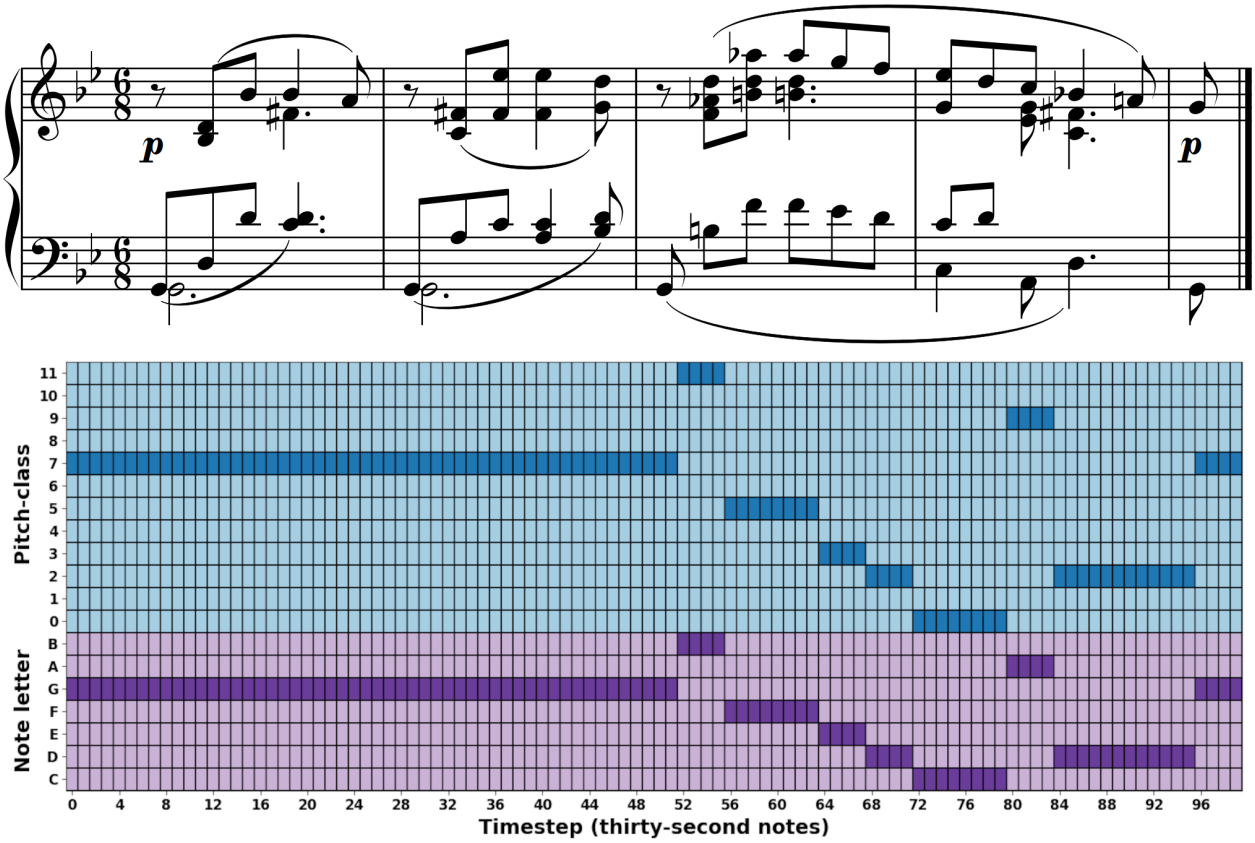


Figure 5.2: *An encoding of the **LowestNote19** representation in Clara Schumann's Op. 13 No. 2, mm. 1–4.*

#### 5.1.4.2 All Sounding Notes

The second input representation, **Notes19**, encodes all the sounding notes at each timestep. In Nápoles López, Gotham, and Fujinaga (2021), we colloquially referred to this representation as the "spelled chroma" input representation. This is due to its similarity with the commonly used "chromagram" representation in the audio domain. The main difference being, beyond operating in the symbolic domain, that chromagram features often ignore note spellings, whereas here they are taken into account. The encoding of **Notes19** is also done using the pitch-spelling duples discussed in Section 5.1.3.1. However, in this case, the representation results in a "multi" two-hot encoding, as each note will result in a two-hot encoding, but several spelled notes may be encoded per timestep. An example of this encoding is shown in Figure 5.3.

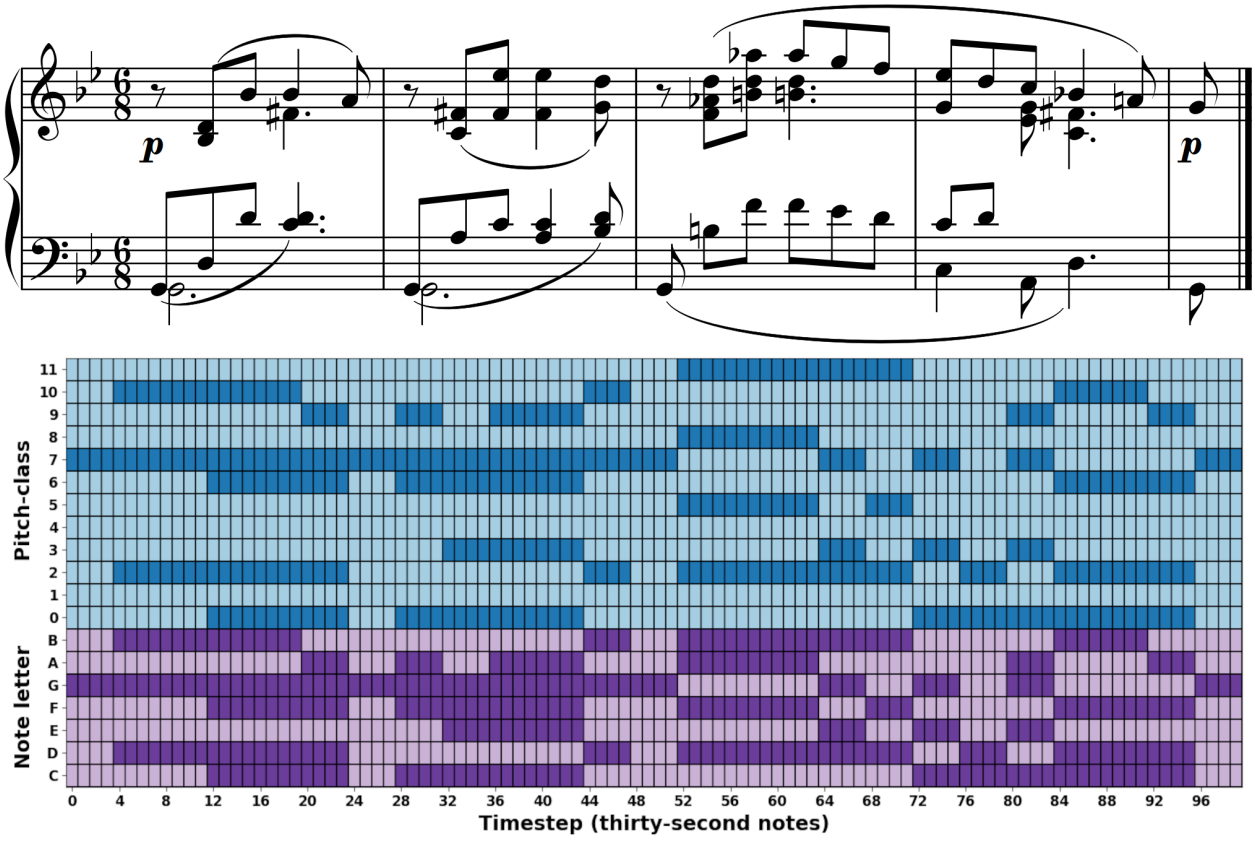


Figure 5.3: *An encoding of the **Notes19** representation in Clara Schumann's Op. 13 No. 2, mm. 1–4.*

#### 5.1.4.3 Measure and Note Onsets

The encoding of pitch information in the **LowestNote19** and **Notes19** input representations does not take into account any information about note onsets, measures, or metrical structure. In order to compensate for that, a third input representation, **Onsets14**, is introduced to process measure and note onsets. The process for encoding the onsets utilizes two 7-dimensional vectors, one for the onset of measures, and one for the onset of notes (see Section 5.1.3.2). Figure 5.4 shows an example of the representation.

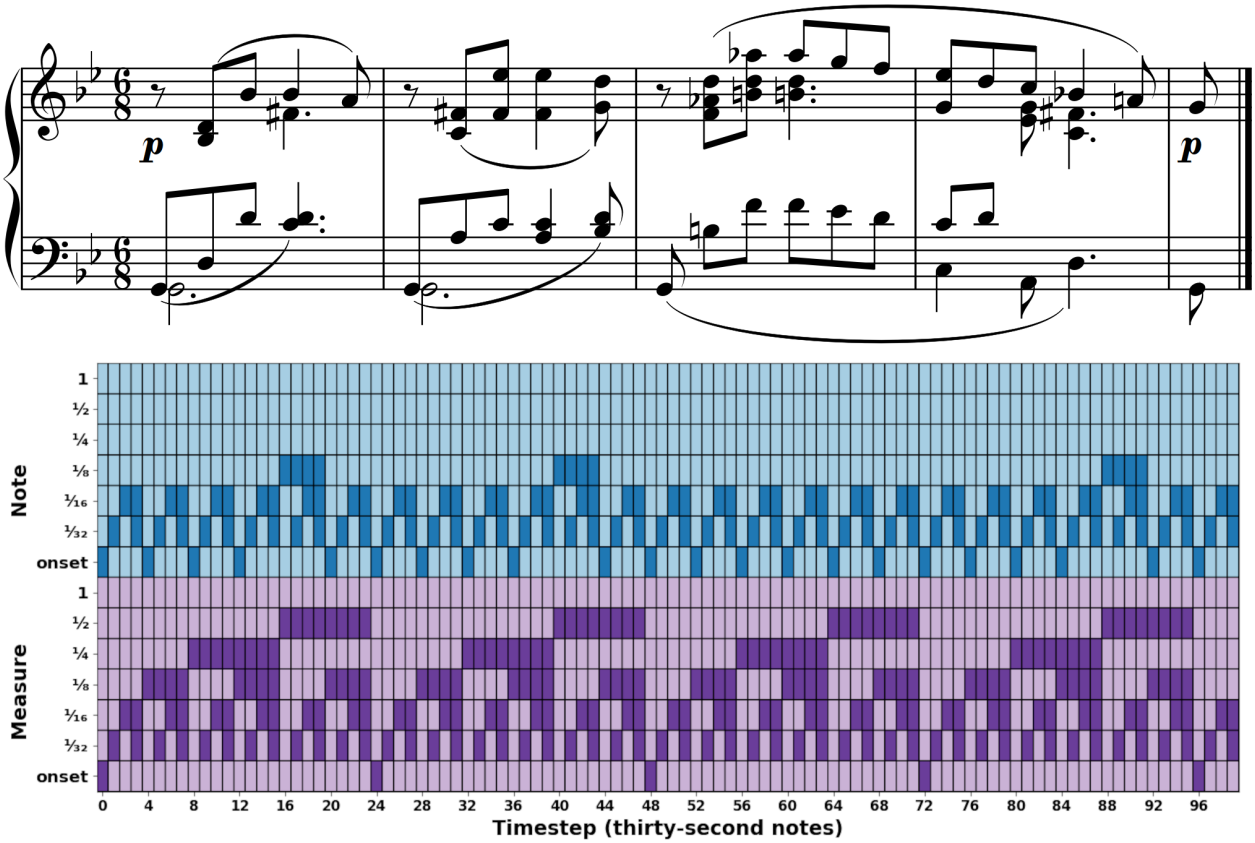


Figure 5.4: *An encoding of the **Onsets14** representation in Clara Schumann's Op. 13 No. 2, mm. 1–4.*

After the three input representations are encoded for all the 640 timesteps, the resulting encoded sequences are dispatched to the trainable layers of the neural network.

## 5.2 The Convolutional Blocks

The first trainable layers of the **CRNN** are the convolutional ones. The network begins by processing each input representation in its own convolutional block and concatenating the resulting outputs, as shown in Figure 5.5. Each of the convolutional blocks comprises a sequence of 1D convolutional layers.

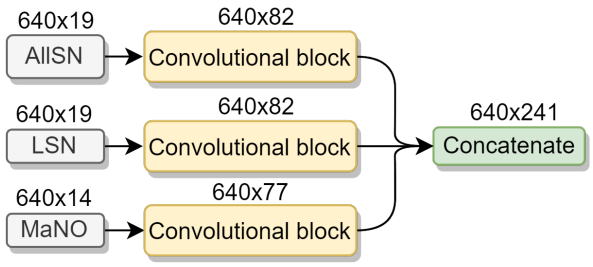


Figure 5.5: *The "convolutional" part of the network, consisting of three convolutional blocks that process each of the input representations independently. The outputs of the blocks are later concatenated.*

### 5.2.1 1D Convolutional Layers Inside a Block

Each of the previous input representations, **LowestNote19**, **Notes19**, and **Onsets14** are processed by a block of convolutional layers. These convolutional blocks have a similar configuration, as shown in Figure 5.6.

All the layers in the convolutional block are 1D convolutional layers. In this configuration of the layers, the size of the kernel grows with each layer, whereas the number of filters is reduced.

#### 5.2.1.1 Kernel Size

In a 1D convolutional layer, the kernel size, $k$, generally represents the number of timesteps in the signal that each filter will multiply at a time. That is, a kernel $k = 3$ will multiply the signal at the timesteps $t$, $t + 1$, and $t + 2$.

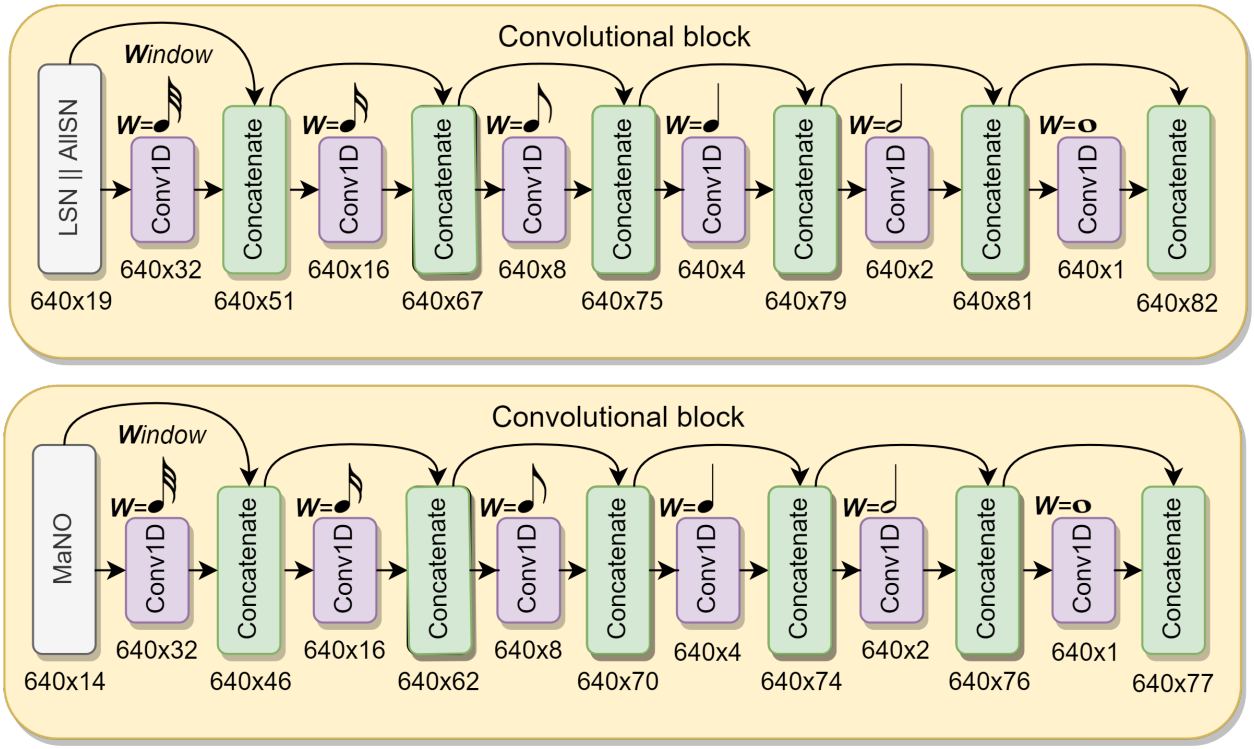


Figure 5.6: *The architecture of each convolutional block. The blocks of the* ***LowestNote19*** *and* ***Notes19****, because they share the same dimensionality, have identical convolutional blocks. The size of the block for the* ***Onsets14*** *representation is different, and indicated on the bottom of the figure.*

In the proposed architecture, $k$ is doubled for each 1D convolutional layer, as shown in Equation 5.8:

$$k_n = 2^n \tag{5.8}$$

where $n$ is the n$^{th}$ 1D convolutional layer. The resulting filters of each layer are concatenated with the ones from the previous layer. Thus, the musical context of the signal is gradually increased.

#### 5.2.1.2 Number of Filters

In a **Convolutional Neural Network (CNN)** layer, the number of filters represents the number of "patterns" that the network will learn from the input. In the visual domain, these pat-

terns could result in, for example, edge-detection filters. In the musical domain, the possible patterns are related with combinations of pitch classes, pitch names, and note/measure onsets.

In this architecture, the number of filters is halved for each consecutive 1D convolutional layer, as shown in Equation 5.9:

$$f_n = 2^{N-n} \tag{5.9}$$

where $N$ is the number of 1D convolutional layers and $n$ is the n$^{\text{th}}$ layer. The network is allowed to store more filters (higher $f$) for short-term patterns (lower $k$) and fewer filters (lower $f$) for long-term patterns (higher $k$). Preliminary experiments showed that prioritizing short-term patterns in the **CNN** layers facilitated the network to learn certain features, such as the ones related with chord segmentation. Moreover, longer-term patterns benefitted key-finding tasks, but these are mostly delegated to the **RNN** layers of the network.

#### 5.2.1.3 Number of Convolutional Layers

The **CNN** blocks of the neural network are implemented with a configurable number of layers, $N$, which is provided as a hyperparameter. Initially, this number is set to $N = 6$. With that configuration, each **CNN** block is able to process inputs with a context of $[t, t + 32]$ for each timestep $t$. That is, at each 𝅘𝅥𝅯s note timestep, the **CNN** will capture information up to a **Whole notes (𝅝s)** note in the future, prioritzing the patterns learned at the level of the individual 𝅘𝅥𝅯s note timestep.

### 5.2.2 Merging the Blocks

The size of each convolutional block depends on the number of features of each input tensor. The resulting convolutional block of the **LowestNote19** and **Notes19** inputs contains 82 features per timestep each, whereas the one in the **Onsets14** input contains 77 features per timestep. Before proceeding to the next layers of the neural network, the tensors of all these

convolutional blocks are concatenated, resulting in a tensor of 640 timesteps with 241 (i.e., 82 + 82 + 77) features each, as shown in Figure 5.5.

## 5.3 Dense and Recurrent Layers

After the convolutional blocks are concatenated, the resulting tensor is processed by a set of dense layers, and eventually the recurrent layers. Figure 5.7 shows the complete neural network architecture.

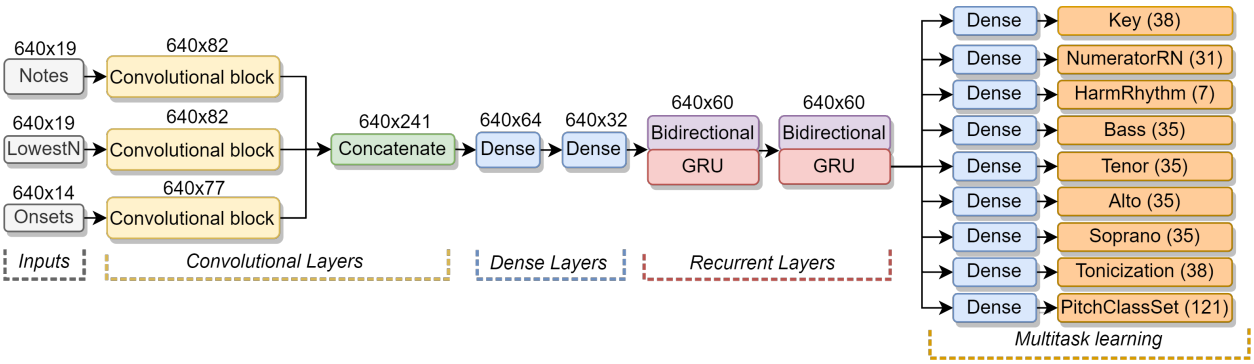


Figure 5.7: *End-to-End neural network architecture.*

### 5.3.1 Dense Layers

After merging the convolutional blocks, the representation has as many features per timestep as the sum of number of features of each convolutional block.

Two dense layers are introduced to reduce the dimensionality of each timestep before the **Gated Recurrent Unit (GRU)** layers.

The first dense layer has 64 neurons. The second dense layer has 32 neurons.

### 5.3.2 Recurrent Layers

Following the two dense layers, the representation of the music has a shape of 640x32. That is, 640 $\mathfrak{s}$ note timesteps, and 32 features in each timestep. This is the input shape leading to

the first of two **GRU** layers. Both **GRU** layers are bidirectional, introducing 30 features from beginning to end of the sequence and 30 features from the end-to-begining of the sequence.

## 5.4 Multitask Learning Outputs

After the recurrent layers, the last component of the **CRNN** is a set of nine classification tasks, which are arranged in a **MTL** configuration. The **MTL** configuration is perhaps one of the main contributions of this dissertation, as it defines the tonal attributes of chords and keys that will be predicted by the network. In previous work (Nápoles López, Gotham, and Fujinaga 2021), we showed that having additional tasks is sometimes beneficial to the performance of the network, especially in combination with the synthesized training examples described in Section 4.5.1.

The proposed **MTL** configuration consists of nine multiclass classification problems: **Bass35**, **Tenor35**, **Alto35**, **Soprano35**, **PitchClassSet121**, **Numerator31**, **LocalKey38**, **Tonicization38**, and **HarmonicRhythm7**. The tasks are codenamed with their number of output classes appended.[4] Each of the output tasks is also tightly coupled with the musical vocabularies and encoding methods introduced throughout the dissertation. More specifically, the following vocabularies and encoding methods:

$\mathcal{S}_{35}$ The encoding method for spelled pitches with 35 classes, described in Section 5.1.3.1.

$\mathcal{O}$ The encoding method for onsets with 7 classes, described in Section 5.1.3.2.

$\mathcal{P}$ The vocabulary of 121 **pcset**s, described in Section A.6.

$\mathcal{N}$ The vocabulary of 31 Roman numeral numerators, described in Section A.2.

$\mathcal{K}$ The vocabulary of 38 keys, described in Section A.4

4. I opted for this naming convention to more easily track the evolution of the multiclass classification problems. Some of them, such as the **PitchClassSet121** or **Numerator31** went through various revisions, as the vocabularies were adjusted during data exploration and experiments. The names, definitions, and number of classes presented in this dissertation represent the latest definition of these problems.

Each of these tasks is subsequently used to generate the final **RNA** annotations in string form, such as **vii$^{o7}$/ii**. The process to generate the **RNA** annotations from the predictions of the classification tasks will be introduced in Section 5.5.

The first four output representations, **Bass35**, **Tenor35**, **Alto35**, and **Soprano35**, are all related to the realization of the annotated chord in a **closed-position form**. Collectively, I refer to these four classifiers as the **SATB35** tasks. Note that these four tasks encode as the target class a spelled note. However, unlike the **LowestNote19** and **Notes19** input representations, the spelled notes in the **SATB35** classifiers have been encoded using the $\mathcal{S}_{35}$ vocabulary, instead of $\mathcal{S}_{19}$. This was done in order to use the same loss function (sparse categorical cross entropy) across all the **MTL** tasks, which would not be possible for the two-hot encoding required in $\mathcal{S}_{19}$. Figure 5.8 shows the chord realization of the **RNA** annotations in the ground truth. The resulting chords define the target class of each of the four **SATB35** classifiers.

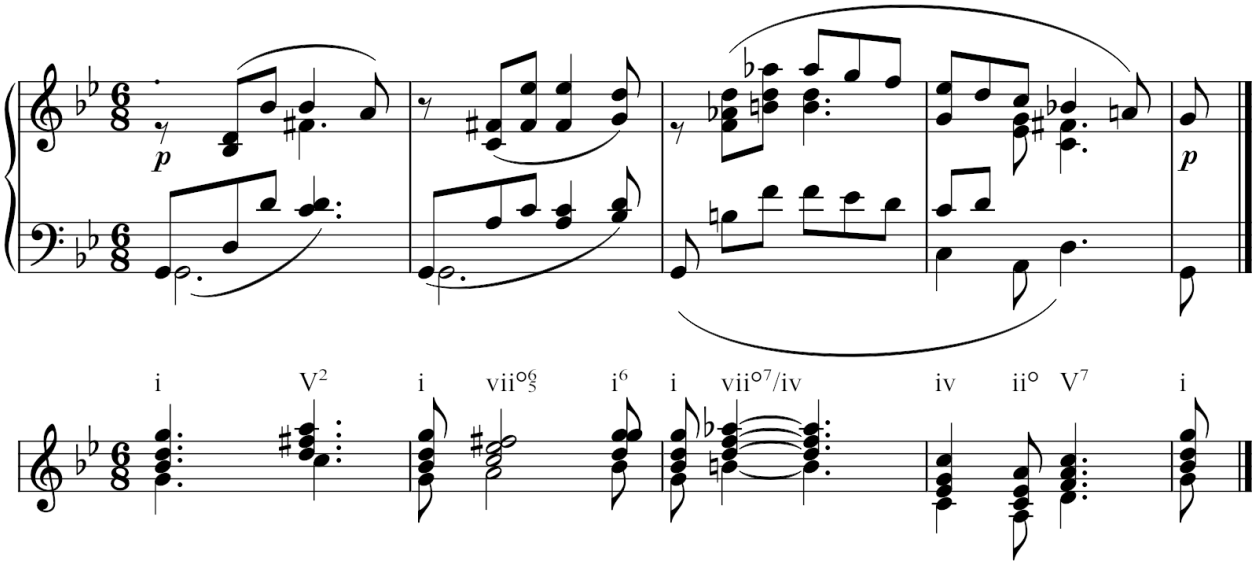


Figure 5.8: *Example of the **SATB35** classifiers, where each classifier learns to predict one of the four notes in the **closed-position form** realization of the chord.*

### 5.4.1 Bass35

The **Bass35** class represents the chord tone acting as the bass. In the chord realizations shown in Figure 5.8, it encodes the pitch spelling of the lowest note of each chord. When the **CRNN**

is processing new music inputs, the **Bass35** classifier will try to predict the lowest note of that chord realization. The goal is that all of the four **SATB35** classifiers predict their corresponding note correctly. An additional role of the **Bass35** classifier is that it determines the inversion predicted by the **ARNA** model. If the **Bass35** prediction is incorrect, then the inversion will necessarily be incorrect. Figure 5.9 shows the encoding of the bass according to the realization shown in Figure 5.8.

Notice also that the bass does not necessarily correspond with the **LowestNote19** representation encoded in the input. Figure 5.2 is a good example of a situation where they do not coincide. In the original score, the two-voice pattern of the bass in measures 1 and 2 could easily persuade the listener to think (or hear) the lower G note as the bass throughout measure 3, however, it is only the lowest-sounding note during the first eighth note. Thus, it can simultaneously be the *bass* of the chord throughout measure 3, but not the lowest-sounding note.

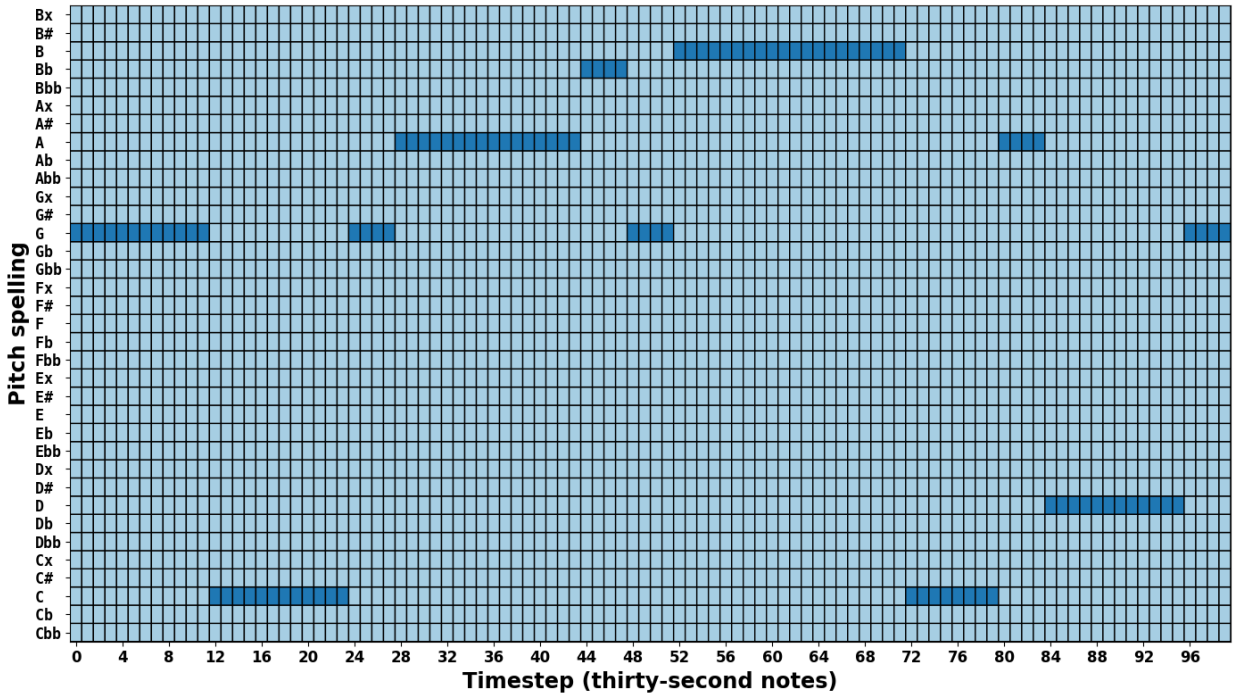


Figure 5.9: *Example of the **Bass35** encoding for the chords in Figure 5.8.*

### 5.4.2 Tenor35

The **Tenor35** task is similar to the **Bass35** task, except that the target label at a given timestep is the "tenor" note of the **closed-position form** chord realization. That is, the second note from bottom to top. Figure 5.10 shows the encoding of the tenor according to the realization shown in Figure 5.8.

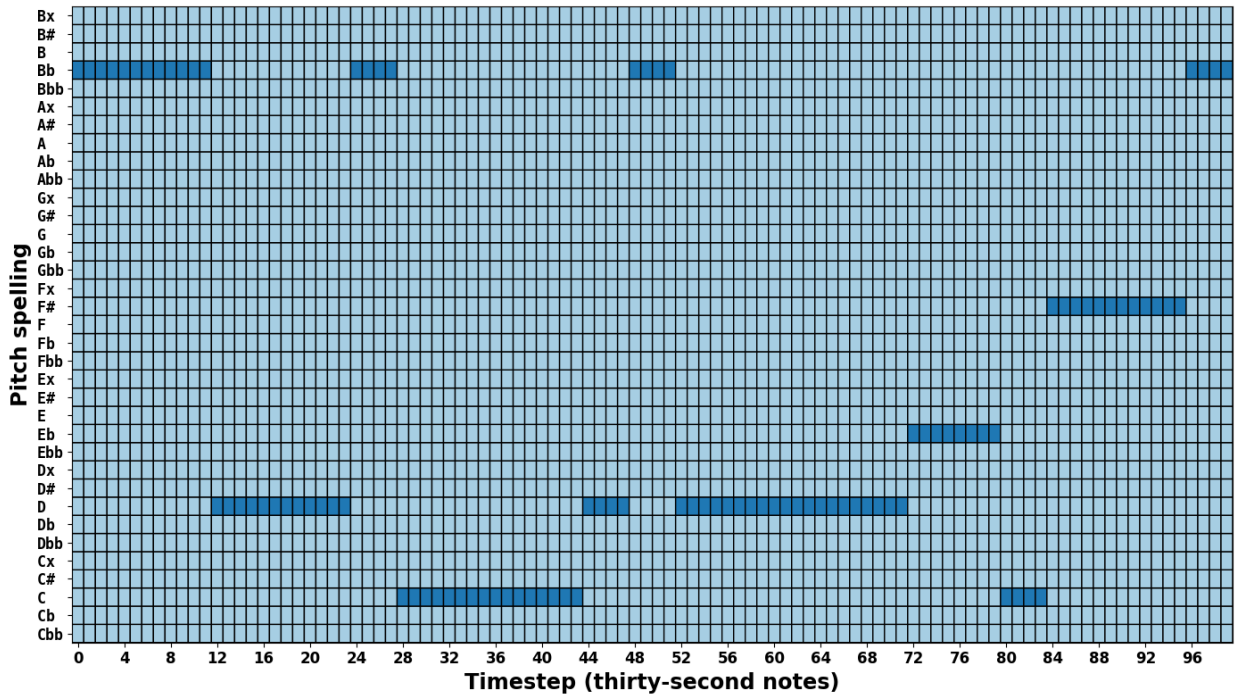


Figure 5.10: *Example of the **Tenor35** encoding for the chords in Figure 5.8.*

### 5.4.3 Alto35

The **Alto35** task is similar to the **Bass35** task, except that the target label at a given timestep is the "alto" note of the **closed-position form** chord realization. That is, the third note from bottom to top. Figure 5.11 shows the encoding of the alto according to the realization shown in Figure 5.8.

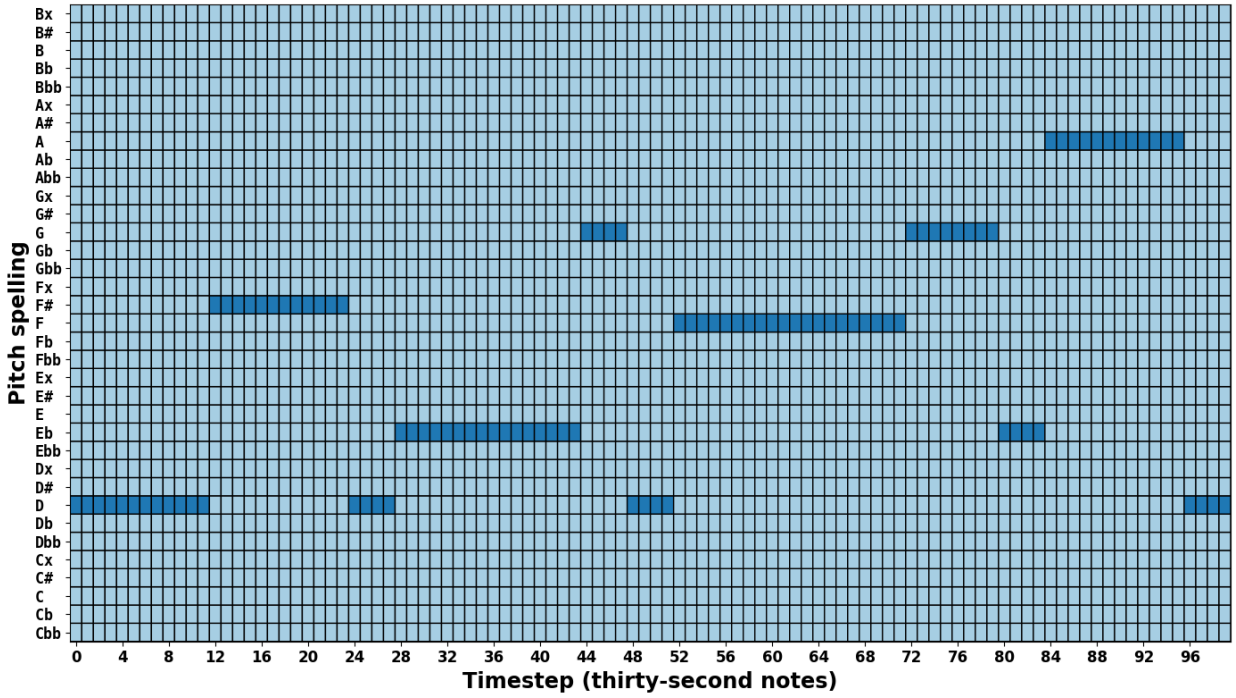


Figure 5.11: *Example of the **Alto35** encoding for the chords in Figure 5.8.*

### 5.4.4 Soprano35

The **Soprano35** task is similar to the **Bass35** task, except that the target label at a given timestep is the "soprano" note of the **closed-position form** chord realization. That is, the highest note of each of the **closed-position form** chords. Figure 5.12 shows the encoding of the soprano according to the realization shown in Figure 5.8.

### 5.4.5 LocalKey38

The **LocalKey38** is a classification task that predicts the $\kappa$ component of an **RNA** annotation. It uses the $\mathcal{K}$ vocabulary to encode one of the 38 key classes available. In the **RNA** annotations of the ground truth, a key $\kappa$ must be indicated at least once in the piece (generally, at the beginning of the piece). If there are no modulations, this will be the key throughout for all timesteps. The **LocalKey38** tends to remain unchanged for longer periods of time than the **Tonicization38** task. However, this depends on the specific dataset and the musical con-

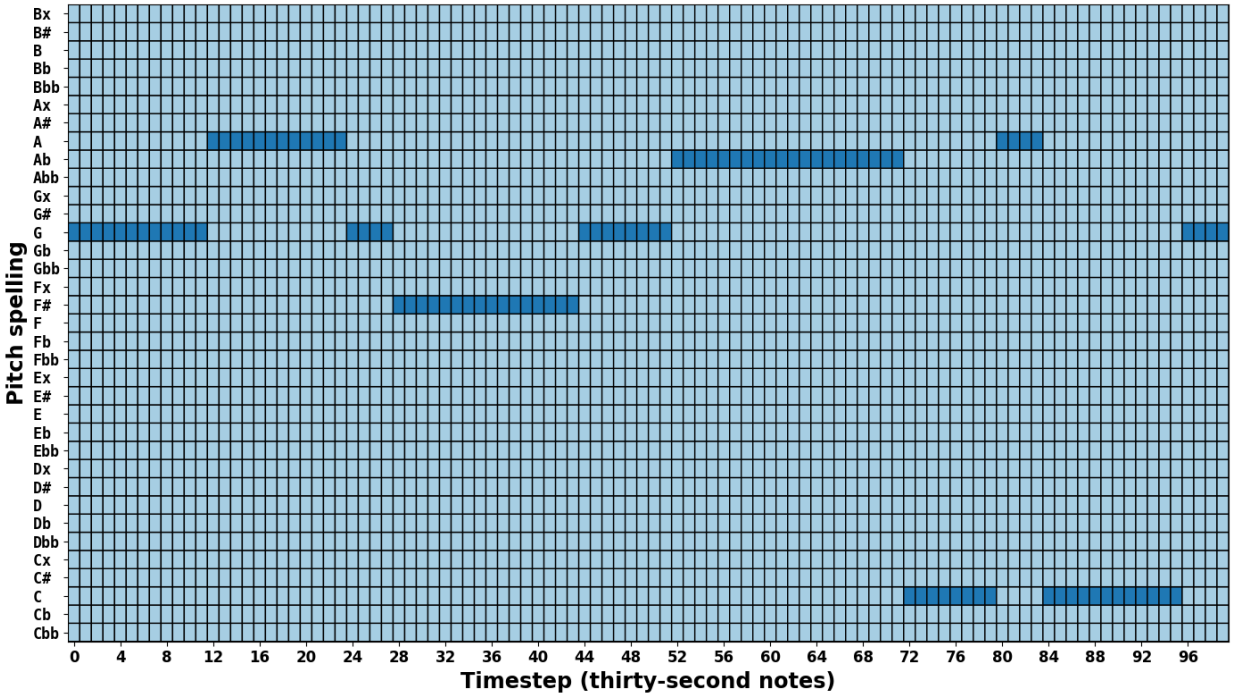


Figure 5.12: *Example of the **Soprano35** encoding for the chords in Figure 5.8.*

ventions of the annotator. For example, some annotators prefer to annotate long sequences of tonicizations. The prediction of the **LocalKey38** task can be used directly as the $\kappa$ component of the **RNA** annotation generated by the **ARNA** model. This will be discussed in Section 5.5.1.

### 5.4.6 TonicizationKey38

The **Tonicization38** is a classification task that predicts the $\tau$ component of an **RNA** annotation. In other approaches (Chen and Su 2021; Micchi et al. 2021), this is addressed as a "secondary degree" task. Nonetheless, here the tonicization is modeled as a key, instead of a relative scale degree. Using the notation of the Appendix A, what the **Tonicization38** task predicts is the key implied by the tonicization, $\tau_\kappa$. The motivation for this departure is the high class imbalance that exists in predicting secondary degrees. Tonicizations are scarce, thus, most of the times there will be no tonicization in a given timestep, which tends to be detrimental for the performance of the model, because the model may learn to predict that tonicizations never happen. By predicting $\tau_\kappa$ instead, the **Tonicization38** learns key fluctua-

tions that are generally shorter than the ones in **LocalKey38** task. When generating the **RNA** label from the model predictions, $\tau_\kappa$ can be easily converted to $\tau$ by encoding the key $\tau_\kappa$ as a scale degree that is relative to $\kappa$. Finally, because **Tonicization38** is a classification task for keys, it uses the $\mathcal{K}$ vocabulary, with 38 classes available for the tonicized keys.

### 5.4.7 PitchClassSet121

The **PitchClassSet121** is a classification task that predicts the **pcset** $\rho$ of an **RNA** annotation. The **pcset** can also be derived from the **SATB35** tasks. However, this approach may lead to a set of predictions that do not form a valid **pcset** $\rho \in \mathcal{P}$. The advantage of the **PitchClassSet121** is that it is always bounded to a valid set of **pcset**s, which is useful as a "fallback" method to generate a chord annotation that does not rely on other tasks.

### 5.4.8 Numerator31

The **Numerator31** is a classification task that predicts the numerator $\eta$ of an **RNA** annotation. In a similar way than the **LocalKey38**, the **Numerator31** can be used directly to generate the $\eta$ component of an **RNA** annotation from the predictions of the **CRNN**. This will be discussed further in Section 5.5.1.

### 5.4.9 HarmonicRhythm7

The last classification task is the **HarmonicRhythm7**. One of the main problems of the **MTL** configuration, especially with the larger number of tasks proposed here, is that it is difficult to infer the segmentation of the chord labels, because the different classifiers may segment the score differently (e.g., if the **PitchClassSet121** predicts a change of chord in timestep $t$, but the **Numerator31** does not).

There have been multiple solutions proposed to deal with this problem. Chen and Su (2021) explored a different neural network architecture and metrics to improve the segmentation.

Micchi et al. (2021) proposed a **Neural Autoregressive Density Estimator (NADE)** layer to model the interdependency of the different classification tasks. McLeod and Rohrmeier (2021) and Wu et al. (2021) tackled the problem of chord segmentation (or, harmonic rhythm) by implementing dedicated models.

Here, another multiclass classifcation problem within the **MTL** layout is proposed instead. This results in an easier implementation, compared to developing a dedicated harmonic rhythm model, and the training can be done jointly with the other tasks in an end-to-end fashion.

The encoding of the **HarmonicRhythm7** is based on the method for encoding onsets described in Section 5.1.3.2, with a vocabulary of 7 classes. The way that chord onsets are encoded is by assigning one of those classes as a chord onset and all others as the "time elapsed since the last chord", in a similar way to how it is done in the **Onsets14** input representation (see Section 5.1.4.3). When the **HarmonicRhythm7** predicts a chord change at timestep $t$, the predictions of all other classifiers at timestep $t$ are used to generate the **RNA** output label.

## 5.5 From Output Predictions to Roman Numeral Labels

Although the predictions provided by the **CRNN** are crucial to determine the **RNA** labels generated by the **ARNA** model, there is another important step left, which is to turn those predictions into **RNA** labels.

As mentioned in Section A.1, a Roman numeral label $\gamma \in \mathcal{R}$ could be defined as a string with four components: a key $\kappa \in \mathcal{K}$, a numerator $\eta \in \mathcal{N}$, a tonicization (denominator) $\tau \in \mathcal{T}$, and an inversion $\iota \in \mathcal{I}$, as shown in Equation 5.10. Thus, if those four pieces of information are known, an **RNA** label that conforms to this structure can be generated.

$$\gamma = \kappa : \eta^{\iota} \,/\, \tau \tag{5.10}$$

Two methods are proposed to retrieve these four pieces of information from the predictions of a machine learning model. The first one, the *direct* method, is tightly coupled with the configuration of the **CRNN** proposed here. Particularly, it benefits from the **MTL** configuration and the chosen output tasks. The second method is more generic, as it only requires a **pcset** $\rho$ and key $\kappa$. In the second method, the numerator $\eta$ and tonicization $\tau$ are estimated from $\rho$ and $\kappa$. Thus, I refer to the second method as the *indirect* method.

### 5.5.1 Direct Method

The direct method consists of using the multiclass classification tasks of the **AugmentedNet** to generate the **RNA** label. The first step is to use the predictions of the **HarmonicRhythm7** task to decide the location of the chord onsets. Once the location of timesteps that represent **RNA** annotations are known, the next step is to use the predictions of the relevant multitask classifiers.

The prediction of the **LocalKey38** task becomes the key $\kappa$.

The prediction of the **Numerator31** task becomes the Roman numeral numerator $\eta$.

The prediction of the **Tonicization38** becomes the key implied by the tonicization $\tau_\kappa$; if $\tau_\kappa \neq \kappa$, then it needs to be converted into a scale degree $\tau$ that is relative to key $\kappa$.

The inversion $\iota$ requires a slightly more involved process. First, using the other components of the **RNA** annotation, the chord implied by the numerator $\eta$ needs to be converted into a sequence of notes, arranged in order from the root upwards. Then, retrieving the prediction $\beta \in \mathcal{S}_{35}$ of the **Bass35** classifier, the location of $\beta$ is searched within the sequence of notes of the chord. If the note is found, then the index of the bass $\beta$ in the sequence of notes is the inversion $\iota$. With these features, an **RNA** label can be generated, as shown in Equation 5.11, where $\hat{\gamma}$ indicates that the generated **RNA** label is an automatic prediction from the **ARNA** model. This process is repeated for each of the chord onsets predicted by the **HarmonicRhythm7** classifier.

$$\hat{\gamma} = \kappa : \eta^{\iota} \ / \ \tau \tag{5.11}$$

### 5.5.2 Indirect Method

In the indirect method, instead of retrieving predictions for $\eta$ and $\tau$ directly, these are estimated from a **pcset** $\rho \in \mathcal{P}$ and key $\kappa \in \mathcal{K}$. The estimation is done using a new algorithm called Numerator and Tonicization Estimation Method (**NaTEM**). This algorithm provides a Roman numeral numerator given the **pcset** and key. In instances where no Roman numeral numerator is suitable for the **pcset** and key provided, the algorithm searches for an appropriate tonicization that fits the given tonal context. For example, given the **pcset** $\rho = \{2, 6, 9\}$ and key $\kappa = \mathbf{C}$, the algorithm would return a "dominant of the dominant" annotation of the form **V**/**V**. The technical details of the **NaTEM** algorithm are described in Section A.7. The indirect method, which is facilitated by the **NaTEM** algorithm, is useful in several situations:

1. When running an evaluation with several **ARNA** models that have a distinct set of classification tasks, as in Section 6.4.

2. When several classification tasks need to be combined to retrieve the chord (e.g., the $\rho$ implied by combining the **SATB35** classifiers).

3. When there is no information available other than a chord label and a key, which occurs in some methods.

In this case, the **NaTEM** algorithm is used to retrieve the numerator $\eta$ and tonicization $\tau$. The generation of the **RNA** label is done as shown in Equation 5.12.

$$\begin{aligned} \hat{\eta}/\hat{\tau} &= \textbf{NaTEM}(\rho, \kappa) \\ \hat{\gamma} &= \kappa : \hat{\eta}/\tau \end{aligned} \tag{5.12}$$

Note that the indirect method does not retrieve the inversion $\iota$. If the inversion of the chord is of interest, there needs to be a classifier for it. The **NaTEM** algorithm can only be used to obtain the Roman numeral numerator and tonicization.

# Chapter 6

# Experimental Evaluation

This chapter introduces the experiments that evaluate the end-to-end **Automatic Roman Numeral Analysis (ARNA)** system proposed here. Section 6.1 introduces a series of ablation studies to assess the contributions of the different components of the neural network. Section 6.2 describes the effects of the data-augmentation techniques. Section 6.4 describes the experimental set up to compare the resulting model against previous approaches. Section 6.5 presents the results of the evaluation. Section 6.6 discusses the overall observations of the experiments.

## 6.1 Ablation Studies

It is difficult to understand what kind of representations a deep learning model learns during training. Ablation studies are an useful way to inspect the contribution of its different components. Ablation studies refer to a concept from neuroscience related to the removal of components of an organism (Meyes et al. 2019). When applied to deep learning models, ablation studies refer to experiments where components of the model are removed or modified. The purpose of the modifications is to observe the effects in the performance of the model, as well as to understand the contributions of the different parts of the model. This section introduces a series of experiments to evaluate the performance of the model after modifying or removing

some of the components of the neural network. The experiments are divided by sections of the **Convolutional Recurrent Neural Network (CRNN)**: input representations, convolutional layers, dense layers, and recurrent layers.

The ablation studies were run over the full aggregated dataset (see Section 4.3) using 5-fold cross-validation. There was no use of data augmentation, neither in the form of transposition nor synthetic files, whose effects is evaluated separately in Section 6.2.

### 6.1.1 Baseline Model

All the ablation studies are compared against a baseline model. The baseline model consists of the **CRNN** described in Chapter 5, which comprises three input representations (**Lowest-Note19**, **Notes19**, and **Onsets14**) described in Section 5.1.3, a block of $n = 6$ convolutional layers described in Section 5.2, two dense layers, and two **Recurrent Neural Network (RNN)** layers with **Gated Recurrent Unit (GRU)** units (see Section 5.3). The baseline model has 9 classification outputs in the **Multitask Learning (MTL)** layout described in Section 5.4, however, these remain constant in the ablation studies. The baseline model is shown in Figure 6.1.

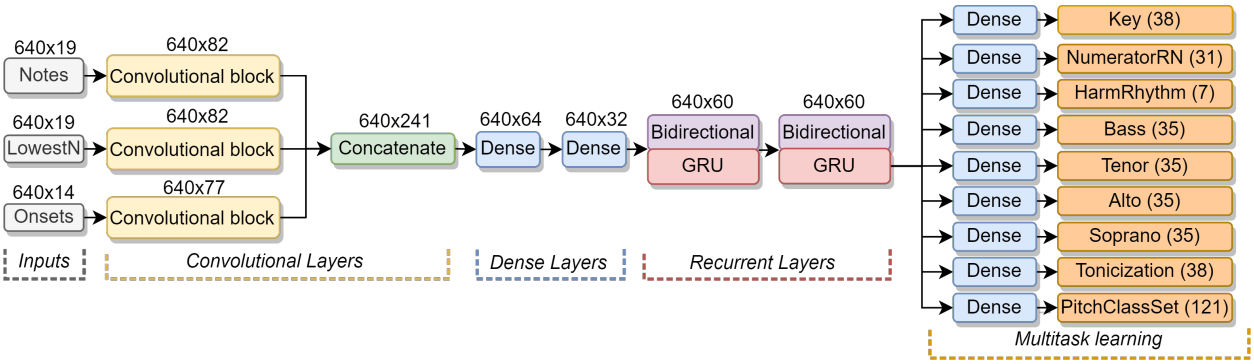


Figure 6.1: *The model that is considered the baseline during the ablation studies.*

### 6.1.2 Changing the Input Representations

The input information provided to the neural network is arguably one of the most important decisions that needs to be made. In the proposed system, the input consists of three representations: **LowestNote19**, **Notes19**, and **Onsets14**. Furthermore, the pitch information considers the spelling of the notes, but discards the octave,[1] and it is divided in two parts: the bottom note of the timestep (**LowestNote19**) and the remaining notes of the timestep (**Notes19**). The experiment in Section 6.1.2.1 considers the effects of modifying the pitch spelling encoding method. The subsequent experiments consider the effects of removing pitch and onset information entirely.

#### 6.1.2.1 Encoding Pitch Spelling with an Alternative Method

In Section 5.1.3.1, a new method to encode pitch spelling was proposed using a 19-dimensional vector. This method was used in the baseline model of the ablation studies to encode the pitch spelling of the **LowestNote19** and **Notes19** input representations. The pitch-spelling method is a replacement of a 35-dimensional vector encoding used by a previous model (Micchi et al. 2021). The new encoding method, among other things, reduces the number of trainable parameters. The current ablation study proposes to use the 35-dimensional representation in Micchi et al. (2021) instead of the method chosen for the baseline. The hypothesis of the experiment is that there will be no noticeable gains in performance by using the 35-dimensional representation, and that the 19-dimensional one is an adequate replacement. The modifications proposed are shown in Figure 6.2.

1. This particular aspect, ignoring the octave of a pitch, was not assessed in the ablation studies below. However, relevant experiments were presented by Micchi, Gotham, and Giraud (2020). In preliminary experiments, the results obtained for the proposed model were consistent with the previous study. That is, a representation without pitch height leads to a higher accuracy than a representation with pitch height.

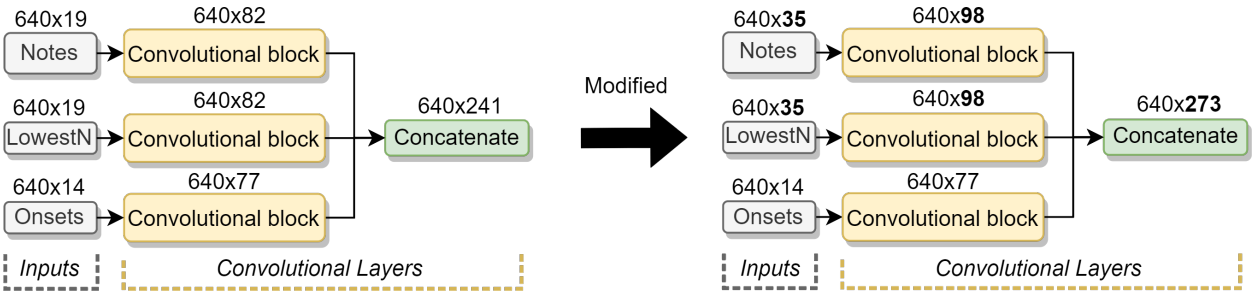


Figure 6.2: *Modification proposed in the first ablation study, where the method in Micchi, Gotham, and Giraud (2020) is used for encoding pitch spellings, instead of the one in the baseline. See Section 5.1.3.1 for further details on the difference between the two methods.*

#### 6.1.2.2 No Lowest-Sounding Note Information

Intuitively, there is a clear correlation between the lowest sounding note in the musical staff and the "bass" note of the chord. If a human analyst was required to indicate the inversion of a chord knowing which notes are sounding in the staff, but not knowing which of them is sounding the lowest, it would make the task very difficult, if not impossible. The lowest-sounding note is provided to the network in the **LowestNote19** input representation. This ablation study proposes removing that representation, which should be detrimental to the **Bass35** classification task (i.e., predicting the bass of the chord). The **Bass35** is the task used to retrieve the inversion of the chord. The modifications proposed are shown in Figure 6.3.

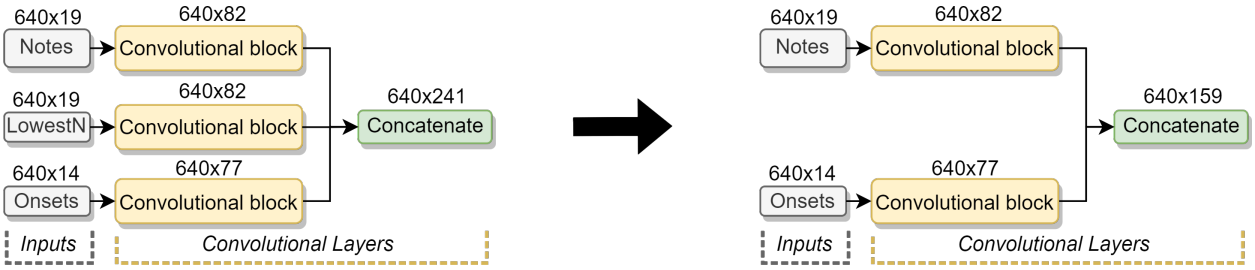


Figure 6.3: *Modification proposed in the second ablation study, where the **LowestNote19** input representation is removed. This is expected to affect the prediction of the inversion.*

#### 6.1.2.3 No Upper Notes Information

An analog of the previous experiment would be to guess the chord knowing only the lowest-sounding note at a given time, but none of the other notes in the staff. The **Notes19** input

representation provides all the note information at a given time, without it, only the lowest-sounding one is known (which is provided by the **LowestNote19** representation). An ablation study is also proposed to demonstrate the effects of removing the **Notes19** input representation. The hypothesis is that it should be detrimental to tasks related to chords: **PitchClassSet121**, **Numerator31**, as well as the three upper **SATB35** tasks: **Soprano35**, **Alto35**, and **Tenor35**. Figure 6.4 summarizes the changes to the input representation.

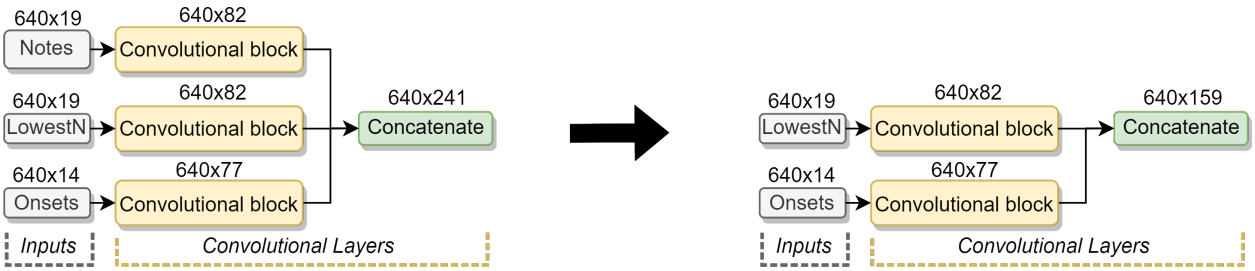


Figure 6.4: *Modification proposed in the third ablation study, where the **Notes19** representation is removed. This is expected to affect most tasks in the **MTL** configuration.*

#### 6.1.2.4 No Onset Information

Chords often occur at the beginning of measures. The **Onsets14** input representation provides the network with this kind of structural information. Namely, the timesteps where a new measure starts, and the timesteps where a new note onset starts. This complements the pitch-related information provided by the other inputs, **LowestNote19** and **Notes19**. An ablation study is proposed here, where the **Onsets14** input representation is removed. The hypothesis is that this will be detrimental to the performance of the **CRNN**, particularly, the chord segmentation. The position of the chords is determined from the predictions of the **HarmonicRhythm7** task. Thus, losing performance in this task results in worse chord segmentation, and it is hypothesized that the **Onsets14** input is correlated with the performance of **HarmonicRhythm7**. The modifications proposed in the ablation study are shown in Figure 6.5.

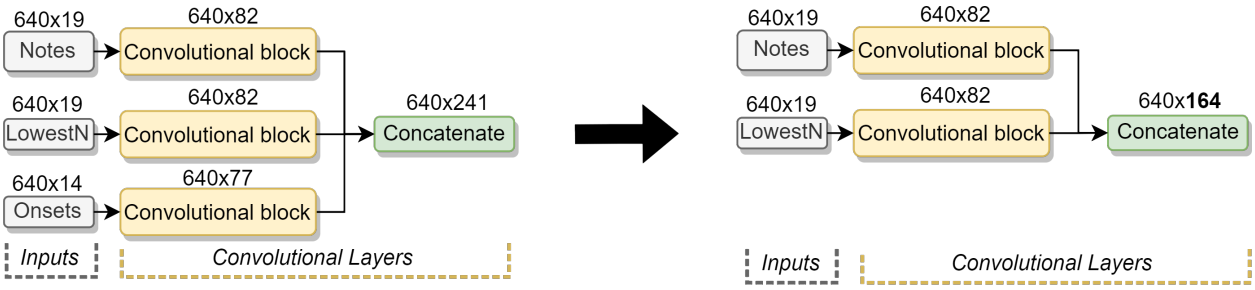


Figure 6.5: *Modifications proposed in the fourth ablation study, where the* ***Onsets14*** *input representation is removed. This is expected to affect the chord segmentation.*

### 6.1.3 Changing the Convolutional Layers

In the baseline model, the convolutional blocks process the three inputs independently, which are later concatenated. The convolutional blocks for all three inputs are similar: they have the same number of layers ($n = 6$, see Section 5.2.1.3) and use the same strategy to adjust the kernel size (see Section 5.2.1.1) and number of filters (see Section 5.2.1.2) of each layer. The structure of these blocks are shown in Figure 6.6.

The experiments in this section explore modifications to the convolutional blocks or the convolutional layers within. Section 6.1.3.1 explores the performance of the neural network with a single convolutional block, stacking the three inputs into a single vector before dispatching them to the convolutional block. Section 6.1.3.2 explores the performance of the neural network with a constant number of filters in each convolutional layer, testing the hypothesis that capturing short-term patterns is preferred. The last experiment in Section 6.1.3.3 explores the effects of removing all convolutional layers (and blocks) entirely.

#### 6.1.3.1 Single Convolutional Block

The number of convolutional blocks is defined by the number of input representations sent to the neural network. In the baseline model, there are three convolutional blocks, one for each of the **LowestNote19**, **Notes19**, and **Onsets14** input representations. An experiment is proposed to explore the effects of using a unique convolutional block. That is, learning convolutional filters from all the inputs at once, instead of independently. This is achieved by

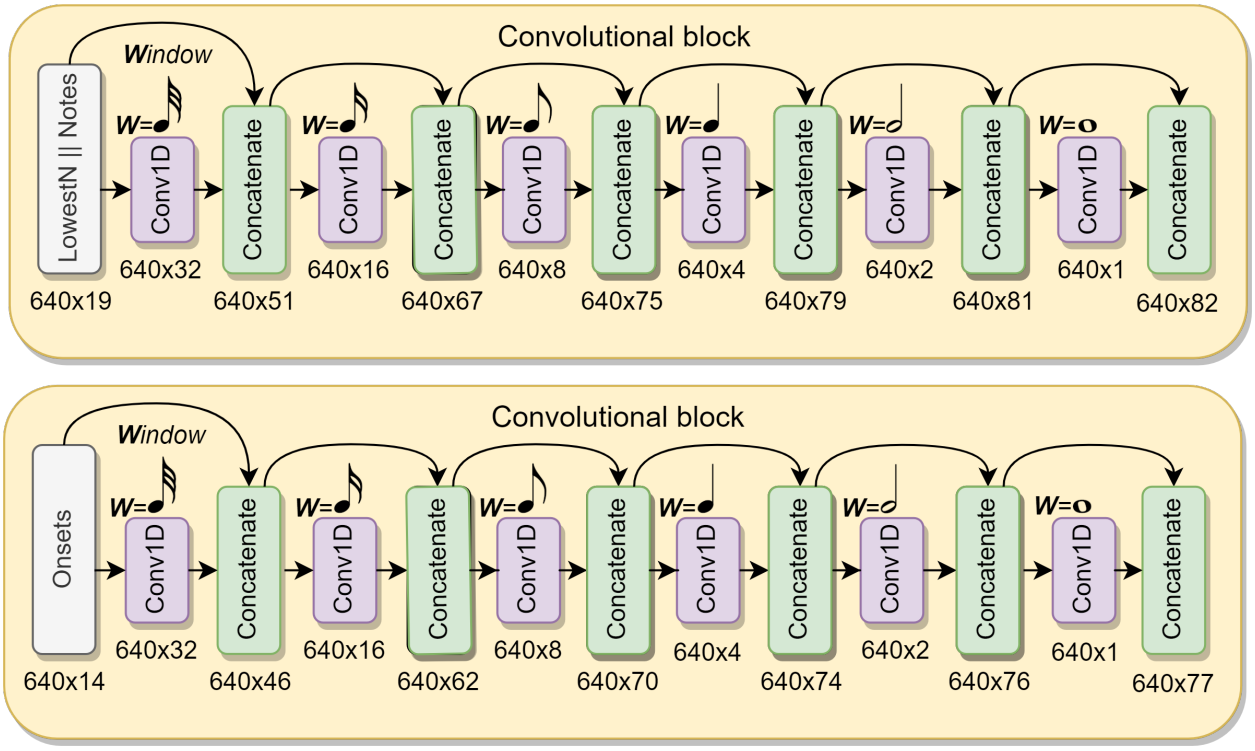


Figure 6.6: *The configuration of the convolutional blocks in the baseline **CRNN** used for the ablation studies. Note that the blocks processing **LowestNote19** and **Notes19** are identical in structure. Thus, two blocks are shown.*

concatenating (i.e., stacking) all the inputs into a single vector per timestep, and processing the resulting sequence with one convolutional block, as shown in Figure 6.7.

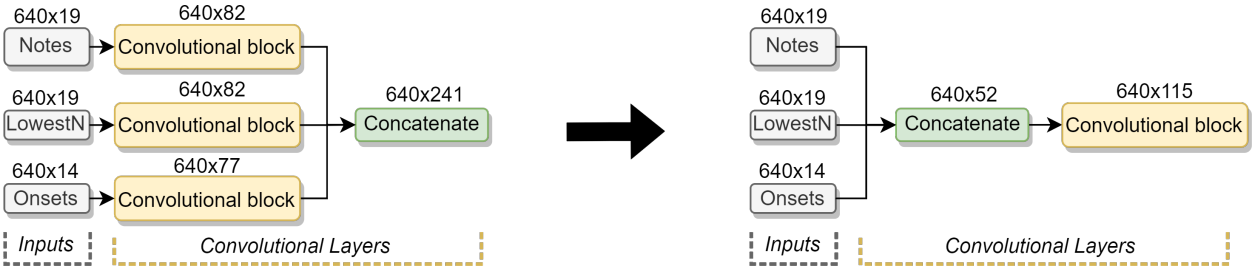


Figure 6.7: *Modification proposed in the ablation study, where all the input representations are concatenated and processed by a single convolutional block.*

#### 6.1.3.2 Constant Number of Filters

In the baseline model, the number of filters decreases in each layer, while the kernel size increases, as shown in Figure 6.8. An ablation experiment is proposed here, where the number

of filters across all convolutional layers remains constant, keeping the number of trainable parameters as close as possible to the baseline. A constant of $f = 5$ was chosen for the number of filters in the ablation experiment, as this results in a similar number of trainable parameters as in the variable $f$ approach of the baseline.

Using this configuration, the network will learn the same number of patterns at the level of **Thirty-second notes (𝅘𝅥𝅯s)** notes as it does for **Whole notes (𝅝s)** notes. In the baseline configuration, I hypothesize that more patterns in the short-term kernel sizes (e.g., 𝅘𝅥𝅯s notes) are beneficial to the performance of the model. The modifications proposed are shown in Figure 6.8.

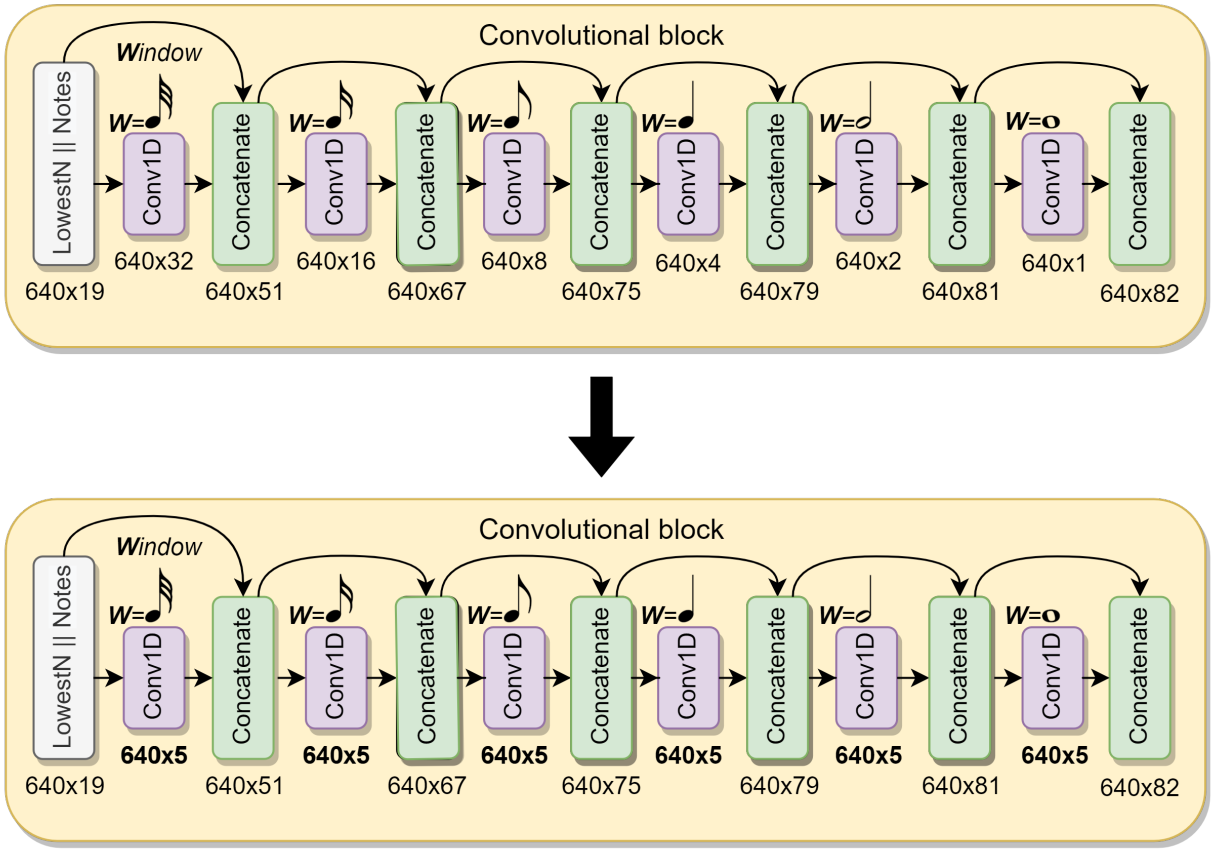


Figure 6.8: *Experiment with a constant number of filters in each convolutional layer. The top figure is from the baseline model, with a variable number of filters. The bottom figure is the modified version. The affected sizes are shown in bold typeface.*

#### 6.1.3.3 No Convolutional Layers

The number of layers in each convolutional block determine how much context is provided to the **Convolutional Neural Network (CNN)**. It also affects the number of trainable parameters on the **CNN** part of the network. This experiment explores removing all the convolutional layers. The modifications proposed are shown in Figure 6.9.

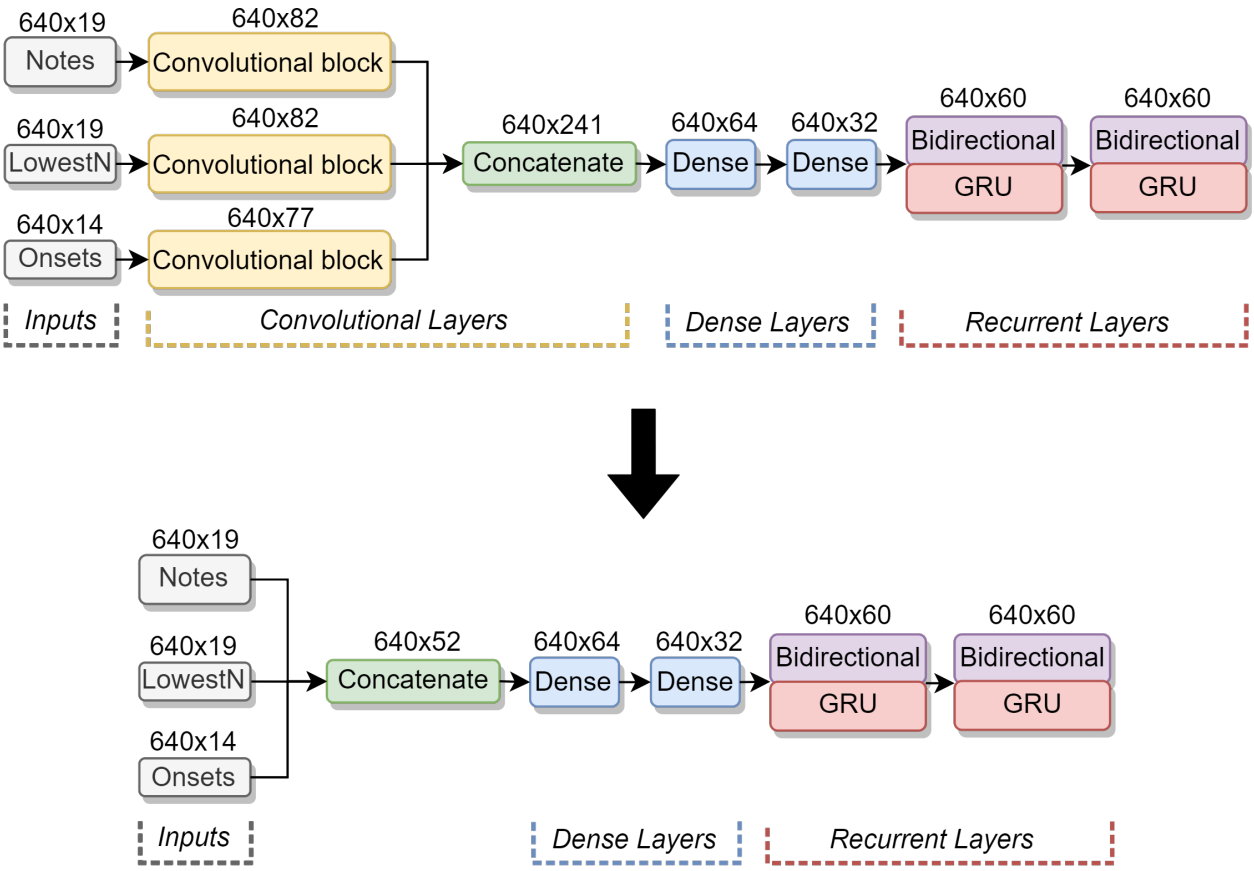


Figure 6.9: *Modifications proposed in the seventh ablation study, where all the convolutional layers have been removed. This is expected to affect the chord segmentation and other "short-term" musical tasks.*

### 6.1.4 Single Dense Linear Layer

In the baseline model, two dense layers are positioned between the convolutional and recurrent layers to reduce the dimensionality of the representation after the convolutional layers. The two dense layers have a **Rectified Linear Unit (ReLU)** nonlinear activation function.

An experiment is proposed to explore the effects of reducing the dimensionality without the addition of nonlinearities. This is achieved by replacing the two **ReLU**-activated dense layers with a single dense linear layer without activation function. The latter exclusively projects the output of the convolutional layers into a low-dimensionality representation before the recurrent layers. Figure 6.10 shows the changes proposed in the experiment.

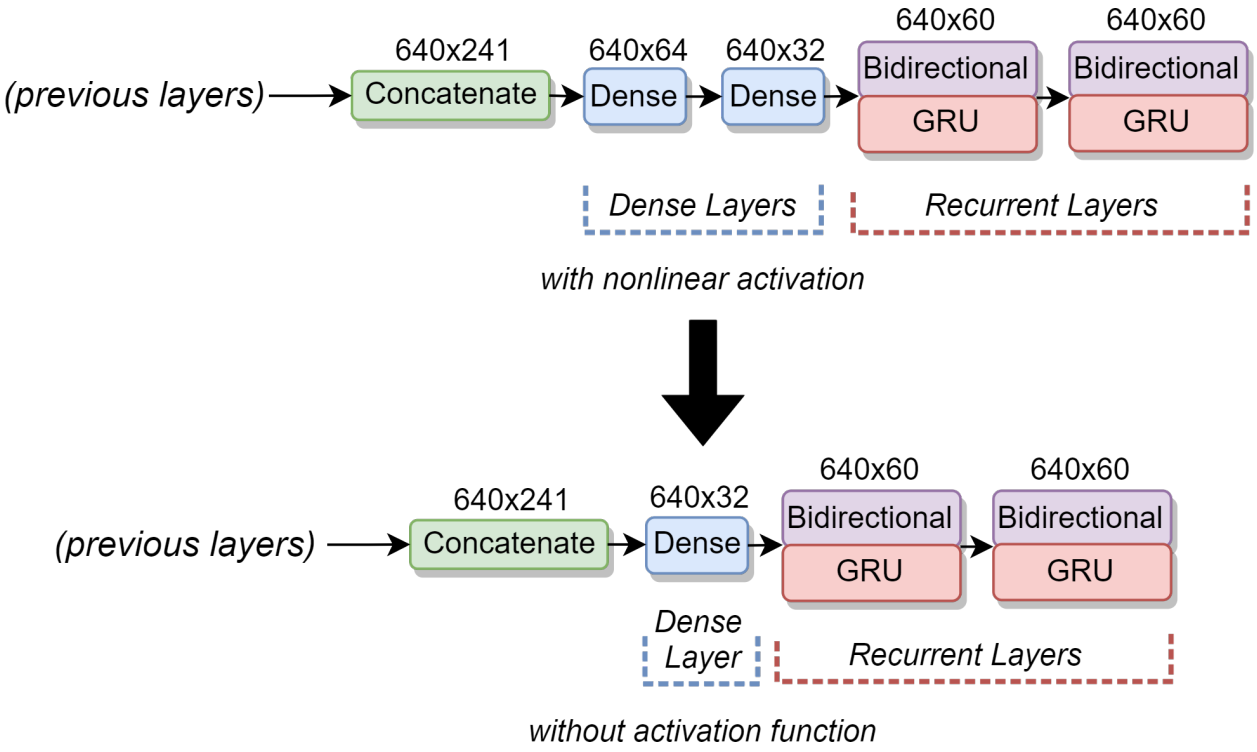


Figure 6.10: *An experiment replacing two nonlinear dense layers with a single linear dense layer.*

### 6.1.5 Changing the Recurrent Layers

The recurrent layers of the network are arguably the most important trainable component of the **CRNN** proposed in this dissertation. In the baseline model, two bidirectional recurrent layers (**GRUs**) are used. Section 6.1.5.1 explores the removal of both recurrent layers and Section 6.1.5.2 explores the effects of using a unidirectional configuration instead of bidirectional.

#### 6.1.5.1 Removing the Recurrent Layers

The recurrent layers in the baseline model process the full sequence of timesteps (i.e., 640 notes). Except for the convolutional layers 2–6 (shown in Figure 6.6), all other layers in the network learn parameters at the level of an individual timestep. Furthermore, the convolutional layer with the longest time window (i.e., kernel size) learns patterns across 32 timesteps (i.e., a note). Any musical patterns that occur beyond a note benefit from the recurrent layers. An ablation experiment is proposed where the two recurrent layers in the baseline model are removed. This should confirm the effects of losing longer-term dependencies in the network. For example, affecting the key estimation tasks: **LocalKey38** and **Tonicization38**. The modifications proposed are shown in Figure 6.11.

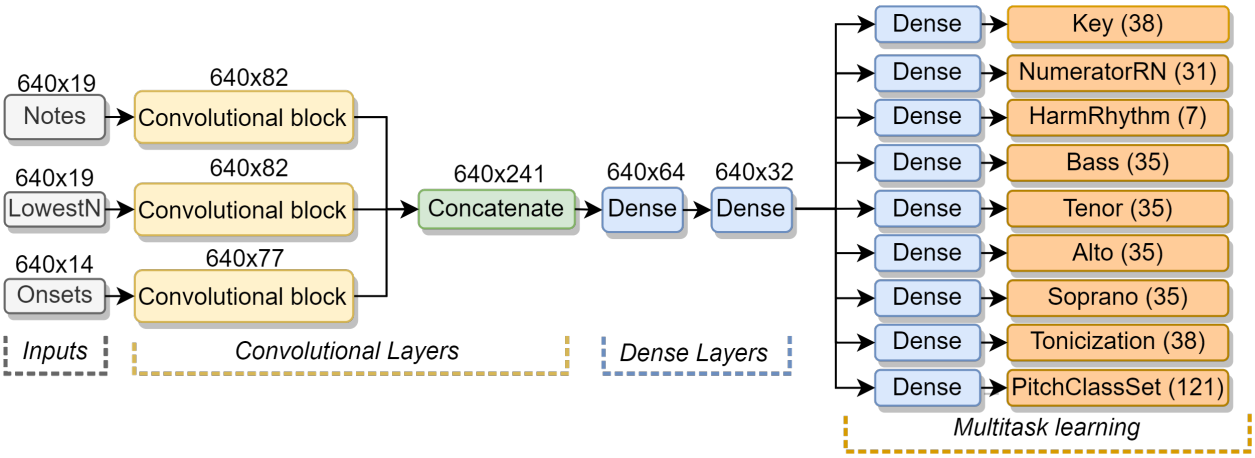


Figure 6.11: *An ablation experiment where the recurrent layers have been removed entirely, compared to Figure 6.1.*

#### 6.1.5.2 Unidirectional Recurrent Layers

In the baseline model, a bidirectional recurrent layer is used. Using this configuration doubles the number of parameters that each recurrent layer needs to learn. An experiment is proposed to verify the effects of processing the musical only from beginning to end, and not from end to beginning. The modifications proposed are shown in Figure 6.12.

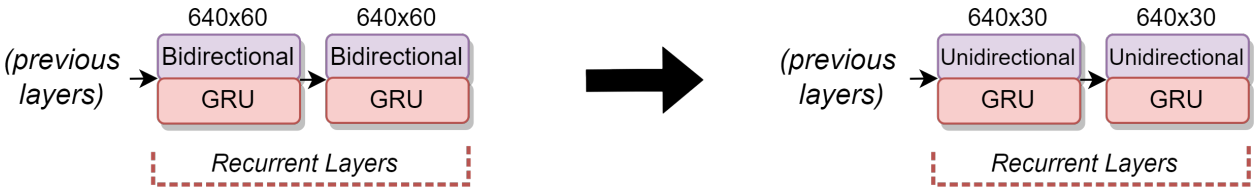


Figure 6.12: *An experiment replacing the bidirectional recurrent layers with a unidirectional (left-to-right) configuration.*

### 6.1.6 Summary of the Ablation Studies

Table 6.1: *Performance obtained in the ablation studies, compared to the baseline configuration of the network. The "Baseline" row shows the average accuracy obtained in each task across the 5-fold cross validation. Each row of the ablation studies shows the difference in accuracy between the value obtained in the experiment and the baseline. The biggest drop in performance for each task column is highlighted in bold font.*

| Model | A35 | B35 | HR7 | K38 | PCS121 | N31 | S35 | T35 | KT38 |
|---|---|---|---|---|---|---|---|---|---|
| Baseline ($\mu_\sigma$) | $68.1_{2.2}$ | $74.5_{2.2}$ | $78.9_{0.9}$ | $78.8_{3.4}$ | $73.5_{1.9}$ | $61.3_{2.8}$ | $72.3_{1.7}$ | $71.3_{2.5}$ | $80_{1.8}$ |
| Pitch spelling | 0.0 | -0.2 | +0.2 | -0.1 | +0.1 | +0.1 | +0.1 | -0.1 | +0.1 |
| No **LowestNote19** | **-13.1** | **-16.5** | 0.0 | +0.3 | +0.3 | +0.4 | -5.0 | **-15.4** | +0.3 |
| No **Notes19** | -11.1 | -1.7 | -4.7 | -6.4 | **-22.2** | **-18.2** | **-16.7** | -10.3 | -10.0 |
| No **Onsets14** | -1.4 | -2.0 | **-21.2** | +0.6 | -1.1 | -0.1 | -0.8 | -1.7 | +0.7 |
| Single block | -0.3 | -0.4 | +0.1 | -0.4 | +0.1 | -0.2 | 0.0 | -0.2 | 0.0 |
| Constant filters | -0.3 | -0.2 | 0.0 | -0.5 | -0.4 | -1.0 | -0.6 | -0.2 | -0.3 |
| No convolutional | -0.4 | -0.3 | -0.3 | +0.1 | 0.0 | -0.4 | -0.1 | -0.1 | +0.8 |
| Linear dense | +0.2 | +0.5 | +0.4 | -0.5 | +0.8 | +0.3 | +0.7 | +0.3 | +0.1 |
| No recurrent | -5.5 | -5.0 | -7.0 | **-18.5** | -6.0 | -10.9 | -5.0 | -6.2 | **-12.4** |
| Unidirectional | -1.9 | -1.1 | -0.6 | -6.7 | -2.7 | -5.2 | -2.4 | -2.0 | -5.2 |

Table 6.1 shows a summary of the performance of the different configurations in the ablation experiments. Some of these results confirm the expected results. For example, removing the **Onsets14** input representation affects the **HarmonicRhythm7** more than any other change in the model configuration. This might indicate that measure onsets are important for a better chord segmentation. Removing the **Notes19** input representation has noticeable effects on the prediction of chords (e.g., **PitchClassSet121**, **Numerator31**). Unexpectedly, it also affects the **Soprano35** task more than any other **SATB35** task. On the contrary, removing the **LowestNote19** input representation affects the performance of the **Tenor35** and **Alto35** tasks, in addition to the **Bass35**, which was the expected outcome.

Overall, as predicted, the biggest drop in performance happens when the recurrent layers of the network are removed. The recurrent layers have a notable effect on the "key" tasks: **LocalKey38** and **Tonicization38**. This supports the hypothesis that a longer musical context is needed to estimate musical keys, in comparison to estimating chords. It is interesting that among the chord tasks, however, **Numerator31** is the one affected the most by the removal of the recurrent layers. This might be because Roman numeral numerators are relative to the key, and thus sensitive to the key context.

In the row of the baseline model, the standard deviation ($\sigma$) across the 5-fold cross validation is provided for reference. All the differences in performance highlighted in Table 6.1 are at least 2 standard deviations below the performance of the baseline. Furthermore, most modifications in the ablation studies have a negative effect compared to the baseline model, which is consistent with the observations during preliminary experiments designing the network. One exception is the use of a linear dense layer, which appears to be slightly above the baseline in most tasks. In order to explore this further, a subsequent experiment was performed between the baseline and linear dense variations, adding data augmentation. In that experiment, the advantage of the linear dense variation faded away, with an average drop of $-2.7\%$ across all tasks compared to the baseline. This brings an important point about the ablation studies presented here, which is that they do not provide information about how the modifications scale with more (or less) data. Thus, they provide hints about the contributions of the different components of the network using a fixed amount of data. The design of a neural network architecture is importantly an empirical process, which requires continuous experimentation in different scenarios. For this reason, the baseline model presented in this section was used in subsequent experiments, as it is the version of the network that went through more experiments and datasets of different sizes. Nevertheless, I still consider the ablation studies interesting, as they permit to confront musical intuitions with the components of the neural network.

## 6.2 Effects of Data Augmentation

The effects of two data-augmentation techniques were explored: *transposition* and *synthesis*. Transposition refers to the transposition of training examples to a different key. Synthesis refers to a new data-augmentation technique where artificial training examples are generated from the **Roman Numeral Analysis (RNA)** annotations. Both data-augmentation techniques are described in Section 4.5. These techniques were applied on the same baseline model described in the ablation studies (see Section 6.1.1). In the four experiments conducted, each of the seven individual datasets (see Section 4.2) was used to train the baseline model, as shown in Table 6.2. The first time, with the original training data. The second time, with the original training data augmented by *transposition*. The third time, with the original training data augmented by *synthesis*. The fourth time, with the original training data augmented by both *transposition* and *synthesis*. As these two methods are not mutually exclusive, the last experiment verifies whether they benefit from each other.

Table 6.2: *Experiments performed on the baseline regarding data augmentation.*

| Experiment | Data Augmentation Strategy |
|---|---|
| 1 | No data augmentation (baseline) |
| 2 | Synthesis |
| 3 | Transposition |
| 4 | Transposition and Synthesis |

### 6.2.1 No Data Augmentation (Baseline)

In the first experiment, the baseline model (see Section 6.1.1) was trained with the original training data of each publicly available dataset: **Annotated Beethoven Corpus (ABC)**, **Beethoven Piano Sonatas (BPS)**, **Haydn "Sun" String Quartets, Op. 20 (HaydnSun)**, **Key Modulations and Tonicizations (KMT)**, **Mozart Piano Sonatas (MPS)**, **Theme and Variation Encodings with Roman Numerals (TAVERN)**, and **When in Rome (WiR)**. The full training set of each dataset was used in the experiment, with the reserved test portion

used to run the evaluation. Table 6.3 shows the number of training examples, test examples, and the training time of the experiment.

Table 6.3: *Parameters of the baseline experiment with no data augmentation.*

| Dataset | Training set | Test set | Training time |
|---|---|---|---|
| **ABC** | 60 files | 10 files | 18 min |
| **BPS** | 25 files | 7 files | 8.9 min |
| **HaydnSun** | 20 files | 4 files | 5.6 min |
| **KMT** | 159 files | 30 files | 6.8 min |
| **MPS** | 46 files | 8 files | 9.4 min |
| **TAVERN** | 46 files | 8 files | 15.7 min |
| **WiR** | 146 files | 27 files | 12.8 min |

### 6.2.2 Synthesis

In the second experiment, the data-augmentation technique proposed in Nápoles López, Gotham, and Fujinaga (2021) was applied to the seven training sets. When the synthesis was used in an experiment, it was used to double the size of the training set: one real example, and one synthesized from its **RNA** annotations. Thus, the augmented training set was twice the size of the original one in the previous experiment. Table 6.4 shows the parameters and training time of the experiment.

Table 6.4: *Parameters of the experiment with data-augmentation by* synthesis.

| Dataset | Augmented training set | Test examples | Training time |
|---|---|---|---|
| **ABC** | 120 files | 10 files | 31.9 min |
| **BPS** | 50 files | 7 files | 15.9 min |
| **HaydnSun** | 40 files | 4 files | 9.0 min |
| **KMT** | 318 files | 30 files | 11.3 min |
| **MPS** | 92 files | 8 files | 17.9 min |
| **TAVERN** | 92 files | 8 files | 29.3 min |
| **WiR** | 292 files | 27 files | 26.1 min |

### 6.2.3 Transposition

In a large number of **Music Information Retrieval (MIR)** works, transposition has been used as a data augmentation technique to overcome the scarcity of data. When pitch spelling is taken into account, it requires the additional consideration that transpositions into two enharmonic keys are distinct from each other. This was first investigated in Micchi, Gotham, and Giraud (2020), where the transposition takes into account the spelling of the notes and the vocabulary of the key modulations within the piece. The approach here is to make sure that all transpositions done over a piece result in key modulations and tonicizations that fit within the vocabulary of keys $\mathcal{K}$ (see Section A.4). Table 6.5 shows the parameters and training time of this experiment. Note that in two datasets with the same number of files, **TAVERN** and **MPS**, the augmentation resulted in a different number of transpositions. This is related to the possible number of transpositions where the modulations lie within the vocabulary $\mathcal{K}$.

Table 6.5: *Parameters of the experiment with data-augmentation by* transposition.

| Dataset | Augmented training set | Test examples | Training time |
|---|---|---|---|
| **ABC** | 625 files | 10 files | 132.0 min |
| **BPS** | 276 files | 7 files | 55.4 min |
| **HaydnSun** | 255 files | 4 files | 33.3 min |
| **KMT** | 2134 files | 30 files | 54.3 min |
| **MPS** | 612 files | 8 files | 72.0 min |
| **TAVERN** | 581 files | 8 files | 144.0 min |
| **WiR** | 2035 files | 27 files | 132.0 min |

### 6.2.4 Synthesis and Transposition

In the fourth experiment, both techniques of *synthesis* and *transposition* were applied to the original training data. This is because the techniques are not mutually exclusive, and they might achieve a better performance in combination.

Table 6.6: *Parameters of the experiment with data-augmentation by* synthesis *and* transposition.

| Dataset | Augmented training set | Test examples | Training time |
|---|---|---|---|
| **ABC** | 1250 files | 10 files | 294.0 min |
| **BPS** | 552 files | 7 files | 126.0 min |
| **HaydnSun** | 510 files | 4 files | 72.0 min |
| **KMT** | 4268 files | 30 files | 114.0 min |
| **MPS** | 1224 files | 8 files | 180.0 min |
| **TAVERN** | 1162 files | 8 files | 312.0 min |
| **WiR** | 4070 files | 27 files | 294.0 min |

### 6.2.5 Summary of the Effects of Data Augmentation

The results in Figures 6.13 and 6.14 show the summary of the four experiments described above. All the experiments were run for a fixed number of 200 epochs. The lines shown correspond to the average validation loss (Figure 6.13) and validation accuracy (Figure 6.14) achieved at epoch $n$ over all seven datasets. The validation accuracy is slightly better in the experiment where both data-augmentation techniques were included. Table 6.7 supports this reading of the accuracy plot by showing average accuracy achieved in each dataset by the end of the experiment.

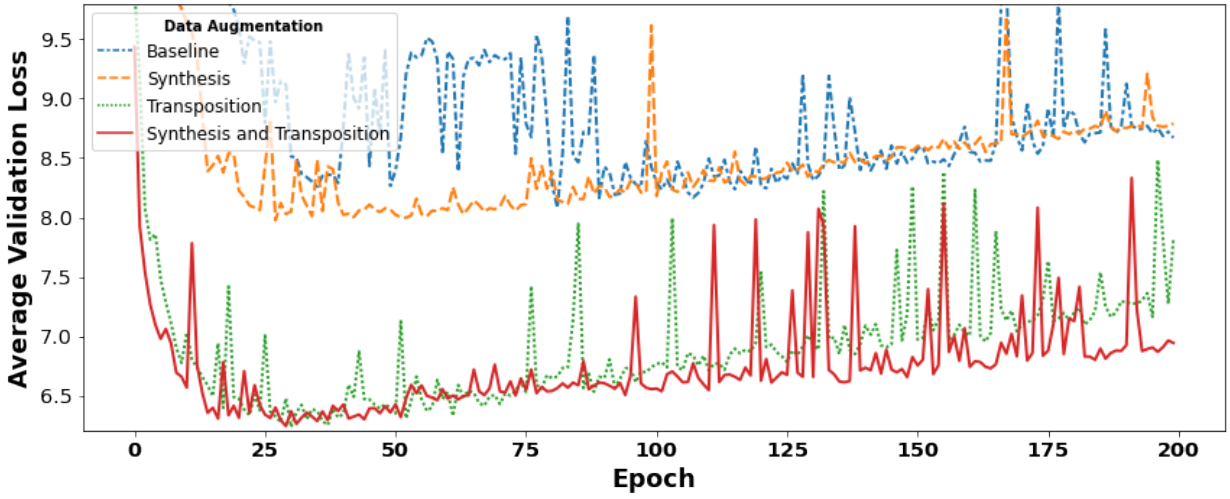


Figure 6.13: *Average validation loss achieved in the four experiments with data augmentation. The average accuracy value is across the seven datasets at the given epoch.*

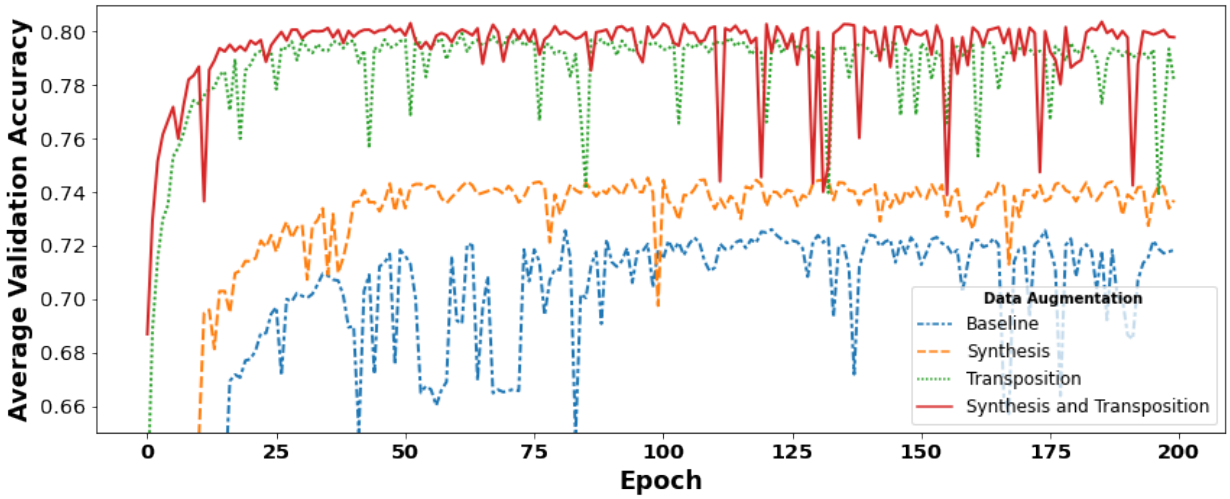


Figure 6.14: *Average validation accuracy achieved in the four experiments with data augmentation. The average accuracy value is across the seven datasets at the given epoch.*

Table 6.7: *Average performance and standard deviation of the four experiments with data augmentation over the seven publicly available datasets.*

| | No augmentation | Synthesis | Transposition | Synthesis and Transposition |
|---|---|---|---|---|
| **ABC** | $65.2_{7.04}$ | $67.2_{7.00}$ | $70.7_{6.09}$ | $\mathbf{72.2}_{5.71}$ |
| **BPS** | $62.5_{9.25}$ | $64.6_{6.62}$ | $74.7_{5.89}$ | $\mathbf{75.7}_{5.70}$ |
| **HaydnSun** | $54.4_{11.76}$ | $59.4_{9.64}$ | $68.2_{8.61}$ | $\mathbf{68.7}_{6.87}$ |
| **KMT** | $78.8_{6.02}$ | $81.6_{6.65}$ | $85.3_{5.84}$ | $\mathbf{88.7}_{3.60}$ |
| **MPS** | $73.2_{7.52}$ | $75.2_{6.86}$ | $\mathbf{81.8}_{5.90}$ | $81.5_{5.89}$ |
| **TAVERN** | $69.1_{7.55}$ | $71.3_{7.66}$ | $76.8_{6.07}$ | $\mathbf{77.0}_{6.44}$ |
| **WiR** | $69.9_{8.79}$ | $71.7_{7.88}$ | $78.1_{5.40}$ | $\mathbf{78.6}_{4.74}$ |

Both data augmentation techniques have a positive effect in the performance of the model compared to only using the original data. Transposition continues to be the most effective data-augmentation technique on its own, however, the experiments showed that both techniques do not interfere with each other. In fact, the most effective technique was to use both, transposition and synthetic examples, although the improvement was not extensive and the training time increased. However, considering the difficulty of obtaining expert-curated data for **ARNA**, using both techniques seems to provide a consistent improvement at the expense of more computation. Thus, in future experiments, the proposed model is trained with both data augmentation techniques simultaneously.

## 6.3 Training on the Aggregated Dataset

Section 6.1 introduced a baseline **CRNN** model and a series of experiments to explore its underlying components. This baseline model was subsequently used in Section 6.2 to explore the effects of data-augmentation techniques when training the model on seven publicly available datasets. In the experiments of Section 6.2, the model was trained on each dataset individually. However, the best performance of a deep learning model is often achieved when more data is used to train it.

This section reports the results of training the **AugmentedNet** model with the aggregated dataset, consisting of all the training data across the seven publicly available datasets. Furthermore, the model was evaluated on the test portion of each individual dataset. The results are reported in Table 6.8. In this experiment, both data-augmentation techniques, *transposition* and *synthesis*, were used in combination.

Table 6.8: *Validation accuracy of the model trained with the aggregated dataset. The validation accuracy is reported in the test set of each dataset and for each of the 9 classification tasks of the model.*

| | ABC | BPS | HaydnSun | KMT | MPS | TAVERN | WiR |
|---|---|---|---|---|---|---|---|
| **Alto35** | 68.8 | 71.7 | 73.3 | 91.5 | 74.7 | 72.9 | 77.2 |
| **Bass35** | 74.4 | 74.5 | 80.6 | 91.8 | 81.8 | 77.2 | 82.3 |
| **HarmonicRhythm7** | 80.4 | 84.5 | 84.3 | 92.9 | 84.4 | 79.6 | 83.8 |
| **LocalKey38** | 81.8 | 81.2 | 73.6 | 59.8 | 90.9 | 87.5 | 72.9 |
| **PitchClassSet121** | 75.5 | 81.8 | 77.5 | 94.1 | 83.9 | 78.7 | 80.6 |
| **Numerator31** | 66.8 | 70.3 | 63.2 | 77.0 | 73.0 | 70.2 | 64.6 |
| **Soprano35** | 73.1 | 77.6 | 75.4 | 95.2 | 80.1 | 76.8 | 78.8 |
| **Tenor35** | 71.0 | 73.9 | 77.2 | 92.2 | 80.8 | 73.9 | 78.9 |
| **Tonicization38** | 80.7 | 85.7 | 78.8 | 78.0 | 90.9 | 86.2 | 78.2 |

## 6.4 Evaluation against Previous Models

Several **ARNA** models have been proposed in the past. This section introduces the experimental set up to compare the proposed model against four previous ones: **Melisma**, Chen and Su (2021), Micchi et al. (2021), and McLeod and Rohrmeier (2021). Whenever a pretrained model

(or program) was provided, it has been used in the experiments. If no pretrained model was available, one was trained from scratch using the dataset and hyperparameters of the original publication. Each of the models required a unique workflow in order to be integrated into the common evaluation. These methods are described below.

### 6.4.1 Baseline Models

The first model, **Melisma**, is a rule-based system that does not require training. The three remaining models are recent deep learning approaches (Chen and Su 2021; Micchi et al. 2021; McLeod and Rohrmeier 2021). The models often differ in their input and output representations. For example, the symbolic music formats they accept as inputs, and the format of their **RNA** annotations.

#### 6.4.1.1 Melisma (2003)

A comparison against the **Melisma** model is presented for historical reference. **Melisma** was arguably the first end-to-end **ARNA** system that could annotate any arbitrary musical score. After the development of the *tsroot* program (Sapp 2009), the **Melisma** system was capable of processing music scores in the ****kern** representation. For example, it was used to annotate scores in the *KernScores* library (Sapp 2005). This model, to the best of my knowledge, has never been evaluated against other methods. Thus, an evaluation is presented here. Because the **Melisma** model only supports ****kern** inputs, all of the **MusicXML** examples in the test set of this evaluation were converted to ****kern**.

During the first attempt to make such conversion, many mistranslations were found among the files of the test set, which lead to **Melisma** being unable to process them. In order to mitigate this problem, the **MusicXML** files were preprocessed. The preprocessing consisted of removing all the *voice* information of the score, by collapsing all the staves and voices in the score into a single staff with chord blocks and tied notes. This process seemed useful for all other models as well, thus, the preprocessing step of the **MusicXML** files was applied to

the entire test set in all experiments. The preprocessing was facilitated by the `chordify()` function of the *music21* Python library (Cuthbert and Ariza 2010).

#### 6.4.1.2 Chen and Su (2021)

Over the years, Chen and Su have proposed three **ARNA** models (Chen and Su 2018, 2019, 2021). Each of these models improving over the previous one. Thus, the latest model (Chen and Su 2021) was chosen for comparison. The original publication does not provide a pretrained model for this neural network. Thus, one was trained from scratch. The data and hyperparameters used for training the model was the same as in the original publication. This model also lacked a code implementation for running it in inference mode (i.e., to predict unseen examples). I provided an implementation, which is made publicly available, together with the resulting pretrained model.[2]

#### 6.4.1.3 Micchi et al. (2021)

The model introduced in Micchi et al. (2021) is an extension of a previous model, Micchi, Gotham, and Giraud (2020). The more recent model, with **Neural Autoregressive Density Estimator (NADE)** layers was chosen for comparison.

The published model was trained on several of the collections[3] introduced in Chapter 4. A pretrained model was unfortunately not provided by the authors due to copyright reasons. However, the source code and dataset necessary to train and run the model was provided. Thus, it was easy to train one from scratch. The outputs of the model were written in the **RomanText** format, so no translation was necessary.

In the original publication (Micchi et al. 2021), this model was trained by randomly splitting the dataset between training and validation portions. In the comparison done here, the model was trained from scratch with nearly all of of their dataset used for training, except for 49 pieces that overlapped with the test set used for this evaluation.

2. https://github.com/napulen/ChenSu21
3. All of them except **KMT** and **MPS**.

#### 6.4.1.4 McLeod and Rohrmeier (2021)

McLeod and Rohrmeier (2021) proposed an **ARNA** model that differs in methodology to the ones presented in Chen and Su (2021), Micchi et al. (2021), and the approach presented here. For example, it substituted the **MTL** configuration with a modular approach based on dedicated models per task. The published model was trained twice with different datasets: a public dataset similar to the one in Micchi et al. (2021), and a larger private dataset of their own. The authors provided source code and pretrained models for both versions of the model. The output annotations of the model were provided in a tabular **Tab-Separated Values (TSV)** format, which comprises chord labels and key annotations. Another output format based on the **Digital and Cognitive Musicology Lab (DCML)** standard for **RNA** annotations (see Section 2.3.1.3) was made available by the authors, but the tabular **TSV** format was used in these experiments. The pretrained model used in this comparison was the one with the best results according to the publication, using the private dataset.

### 6.4.2 Experimental Set up

The test set used for the comparison was the same one used in the experiment of Section 6.3. A total of 94 files were used for testing, which were sampled from the seven publicly available datasets (see Section 4.2). All of the input files in the test set were provided in the **MusicXML** format, and their ground-truth annotations in **RomanText**. The **Melisma** model does not support **MusicXML** inputs, in this case, all the **MusicXML** files were translated into ****kern**. Similarly, three models, **Melisma**, Chen and Su (2021) and McLeod and Rohrmeier (2021), provided their output annotations in other formats. For these models, the original outputs were translated into **RomanText**. All the comparisons were performed between the ground-truth **RomanText** in the test set, and each of the **RomanText** files generated by the models (either directly or through a translation). All the models were evaluated with the same **RNA** vocabulary, which required standardization.

#### 6.4.2.1 Standardizing the Chord Vocabulary

The vocabulary of each model is different, as shown in Table 6.9. In addition to the different qualities of chords supported by each model, the **RNA** syntax is also often distinct (see Section 2.3).

Table 6.9: *The chord vocabularies of the compared models. C&S21 refers to Chen and Su (2021), Mi21 to Micchi et al. (2021), and M&R21 to McLeod and Rohrmeier (2021).*

| Chord Quality | **Melisma** | C&S21 | Mi21 | M&R21 | **AugmentedNet** |
|---|---|---|---|---|---|
| Major triad | ✓ | ✓ | ✓ | ✓ | ✓ |
| Minor triad | ✓ | ✓ | ✓ | ✓ | ✓ |
| Augmented triad | | ✓ | ✓ | ✓ | ✓ |
| Diminished triad | ✓ | ✓ | ✓ | ✓ | ✓ |
| Dominant seventh | ✓ | ✓ | ✓ | ✓ | ✓ |
| Minor seventh | ✓ | ✓ | ✓ | ✓ | ✓ |
| Minor major seventh | | | | ✓ | |
| Major seventh | | ✓ | ✓ | ✓ | ✓ |
| Augmented major seventh | | | | ✓ | ✓ |
| Augmented minor seventh | | | | ✓ | |
| Fully diminished seventh | ✓ | ✓ | ✓ | ✓ | ✓ |
| Half diminished seventh | ✓ | ✓ | ✓ | ✓ | ✓ |
| Augmented sixth | | ✓ | ✓ | | ✓ |
| Augmented sixth (German) | | | | | ✓ |
| Augmented sixth (French) | | | | | ✓ |
| Augmented sixth (Italian) | | | | | ✓ |

Some of the models provide two layers of key analysis (**AugmentedNet**, Micchi et al. 2021; Chen and Su 2021), whereas others provide only one key prediction (**Melisma**, McLeod and Rohrmeier 2021). This difficults a fair comparison between the models. Furthermore, the models might be in fact able to recognize chords that are not explicitly considered in their vocabulary. For example, neither the **Melisma** nor the McLeod and Rohrmeier (2021) models explicitly recognize **augmented sixth** chords. However, a label of $\mathbf{V^7}$ may be interpreted as an **augmented sixth** chord in a certain key context, even if the predictions of the model do not indicate it as such.[4] This is true for other types of chords too, such as **Neapolitan** chords. All of these issues combined were a motivation to propose the **NaTEM** algorithm (see Section A.7),

4. See the examples in Section A.7 where a $\mathbf{V^7}$ chord in one key is the same **pcset** as a $\mathbf{Ger^7}$ chord in a different key.

which makes it possible to use a common representation for all the annotations of the five models, regardless of their original syntax. The inputs to the **NaTEM** algorithm are a **pcset** $\rho \in \mathcal{P}$ (see Section A.6) and a key $\kappa \in \mathcal{K}$ (see Section A.4). Given those inputs, the algorithm provides one of the 31 numerators $\eta \in \mathcal{N}$ (see Section A.2), and a tonicization $\tau \in \mathcal{T}$, if necessary. This standardization is also applied to the ground truth annotations, providing the same **RNA** vocabulary for all models and ground truth for direct comparison. This process of running the annotations and the ground truth through the **NaTEM** algorithm is explained below.

#### 6.4.2.2 Evaluation Procedure

In order to perform the evaluation, all the annotations in the **RomanText** files, including the ground truth, were decomposed into their **pcset** $\rho$, key $\kappa$, and inversion $\iota$ components. The components $\rho$ and $\kappa$ were processed by **NaTEM** to retrieve a numerator $\eta$ and tonicization $\tau$. This process is illustrated for a ground-truth example of the test set in Table 6.10. The process performed on the predictions of each model was identical, using their **RomanText** files.

Table 6.10: *Translation process of the ground-truth annotations in one of the test files of the **KMT** dataset. The original annotations (leftmost column) are parsed from a **RomanText** file, and decomposed into a **pcset** $\rho$, key $\kappa$, and inversion $\iota$ components (middle columns). Then, a numerator $\eta$ and tonicization $\tau$ were retrieved from **NaTEM** (rightmost columns).*

| Original annotation | **pcset** ($\rho$) | Key ($\kappa$) | Inversion ($\iota$) | **NaTEM** ($\eta$) | **NaTEM** ($\tau$) |
|---|---|---|---|---|---|
| **C:I** | {0, 4, 7} | **C** | 0 | **I** | **I** |
| **V$^{2}$** | {2, 5, 7, 11} | **C** | 3 | **V$^{7}$** | **I** |
| **I$^{6}$** | {0, 4, 7} | **C** | 1 | **I** | **I** |
| **vii$^{o6}$** | {2, 5, 11} | **C** | 1 | **vii$^{o}$** | **I** |
| **I** | {0, 4, 7} | **C** | 0 | **I** | **I** |
| **Cad$^{6}_{4}$** | {0, 4, 7} | **C** | 2 | **I** | **I** |
| **V** | {2, 7, 11} | **C** | 0 | **V** | **I** |
| **I** | {0, 4, 7} | **C** | 0 | **I** | **I** |
| **G:viio$^{6}$** | {0, 6, 9} | **G** | 1 | **vii$^{o}$** | **I** |
| **vii$^{o6}$/ii** | {2, 8, 11} | **G** | 1 | **vii$^{o}$** | **ii** |
| **ii$^{6}$** | {0, 4, 9} | **G** | 1 | **ii** | **I** |
| **V$^{7}$** | {0, 2, 6, 9} | **G** | 0 | **V$^{7}$** | **I** |
| **I** | {2, 7, 11} | **G** | 0 | **I** | **I** |

Three evaluation experiments were performed. The first one, measuring the time it took for the models to generate their chord predictions. Two subsequent experiments evaluated the performance of the models on their **RNA** predictions:

1. The models were evaluated in the accuracy of their $\rho$, $\kappa$ and $\iota$ components, compared to the $\rho$, $\kappa$ and $\iota$ of the ground-truth.

2. The models were also evaluated using the $\eta$ retrieved from **NaTEM**, which was compared against the $\eta$ obtained for the ground-truth.

## 6.5 Results

This section introduces the results of comparing the five **ARNA** models. Three evaluations were performed. The first evaluation compared the inference time of the models in the test set. That is, the time it took for the models to output their predictions. The second evaluation experiment measured the average accuracy obtained for the "individual tasks" of **pcset** $\rho$, key $\kappa$, and inversion $\iota$, compared to the same labels in the ground-truth. The third evaluation experiment measured the accuracy of the numerators $\eta$ obtained by the models against the ground-truth. These numerators refer to the ones retrieved by the **NaTEM** algorithm.

### 6.5.1 Time Performance on Inference

The time that each model took to compute its predictions on the test set of 94 **MusicXML** files was measured.[5] In order to run the measurement, each of the files was processed as an independent process in the operating system, running the model with the input and output arguments for the file in turn. For the models based on deep learning approaches, no **Graphics Processing Unit (GPU)** was used to compute the predictions, nor they were processed in batch. Table 6.11 shows the performance of the models.

5. In the case of **Melisma** the time it took to compute its predictions over the set of 94 ****kern** files.

Table 6.11: *Time elapsed for each model to provide the output predictions on the 94 files of the test set.*

| Model | Total time | Mean | SD |
|---|---|---|---|
| **Melisma** | 2.7 min | 1.7s | 0.8s |
| Micchi et al. (2021) | 18.6 min | 11.9s | 6.0s |
| **AugmentedNet** | 22.6 min | 14.4s | 12.5s |
| Chen and Su (2021) | 26.2 min | 16.8s | 9.0s |
| McLeod and Rohrmeier (2021) | 183.0 min | 116.8s | 154.6s |

By far, the fastest model is **Melisma**. This model is a compiled program written in the *C programming language*. Thus, as a compiled program, it is intrinsically faster than all other approaches, which are based on the Python programming language. Out of the deep learning models, the one in Micchi et al. (2021) seems to be the fastest, followed by **AugmentedNet** and Chen and Su (2021). The model in McLeod and Rohrmeier (2021) is the slowest one to run on raw **MusicXML** inputs, averaging nearly 2 minutes to annotate each file. It is possible that one of the drawbacks of the modular approach is that the computations are not run in an end-to-end fashion as in **MTL** approaches. However, considering the time it takes for a human annotator to annotate a long musical score (e.g., piano sonata), any of the models would be faster than a human annotator.

### 6.5.2 Accuracy on Individual Tasks

Table 6.12: *Comparison of the accuracy achieved by the five models on the individual tasks $\rho$, $\kappa$, and $\iota$.*

| model | **pcset** ($\rho$) | Key ($\kappa$) | Inversion ($\iota$) |
|---|---|---|---|
| **Melisma** | 54.6 | 58.2 | 70.3 |
| McLeod and Rohrmeier (2021) | 57.7 | 51.5 | 69.3 |
| Chen and Su (2021) | 59.8 | 52.6 | 70.8 |
| Micchi et al. (2021) | 74.7 | 72.9 | **82.0** |
| **AugmentedNet** | **79.4** | **79.4** | 81.9 |

The accuracy of the models on the individual tasks of **pcset** $\rho$, key $\kappa$, and inversion $\iota$ is shown in Table 6.12. In this evaluation, **Melisma** model seemed to be the worst-performing model. Surprisingly, the performance of two deep learning approaches, McLeod and Rohrmeier

(2021) and Chen and Su (2021), was not too far ahead of **Melisma**. Given the nature of **RomanText** files, which have discrete chord annotations located at measure and beat positions, it is possible that the lower performance shown in these models is heavily penalizing the chord segmentation of the models. If the predictions are correct but misaligned with the annotations of the ground-truth, it is possible that the performance of the models appears to be worse than it would be perceived in a different evaluation workflow. The Micchi et al. (2021) and **AugmentedNet** models had a similar performance, with **AugmentedNet** slightly ahead on the $\rho$ and $\kappa$ tasks.

### 6.5.3 Roman Numeral Accuracy

In the third experiement, the accuracy of the models on the numerator $\eta$ was evaluated. One of the purposes of the **NaTEM** algorithm was to standardize the conditions that lead to a tonicization. As long as the key context and **pcset** are the same between the ground-truth and the prediction, the Roman numeral numerator $\eta$ and tonicization $\tau$ will be the same. This might also be useful to help the models recognize chords that do not exist in the vocabulary of their annotations, such as **Neapolitan** and **augmented sixth** chords. For each model, a confusion matrix is shown with the target chord classes in each row, and the predicted chord classes in each column.

#### 6.5.3.1 Confusion Matrix for Melisma (2003)

Figure 6.15 shows the confusion matrix for the numerators ($\eta$) obtained by the predictions of the **Melisma** model, compared against the ones obtained for the ground-truth. Compared to other models, **Melisma** has a very limited chord vocabulary. The model does not explicitly indicate what is the theoretical vocabulary it supports, however, an inspection of the predictions it generated revealed that it does not support as many chords as the recent models do (see Table 6.9). This is confirmed in the confusion matrix with the thick vertical lines across the **I**, **V**, and $\mathbf{V^7}$ columns, indicating that the model mostly predicts tonic and dominant harmonies

in most instances. This is often misleading when seen from the point of view of accuracy on chord recognition, as chords are heavily skewed towards tonic and dominant harmonies. A model that mostly predicts tonic chords may receive a good global accuracy score, whereas the confusion matrix is better at demonstrating the real limitations of the model. Nevertheless, the model seemed to provide some of the difficult annotations correct, such as the **iv**$^{7}$.

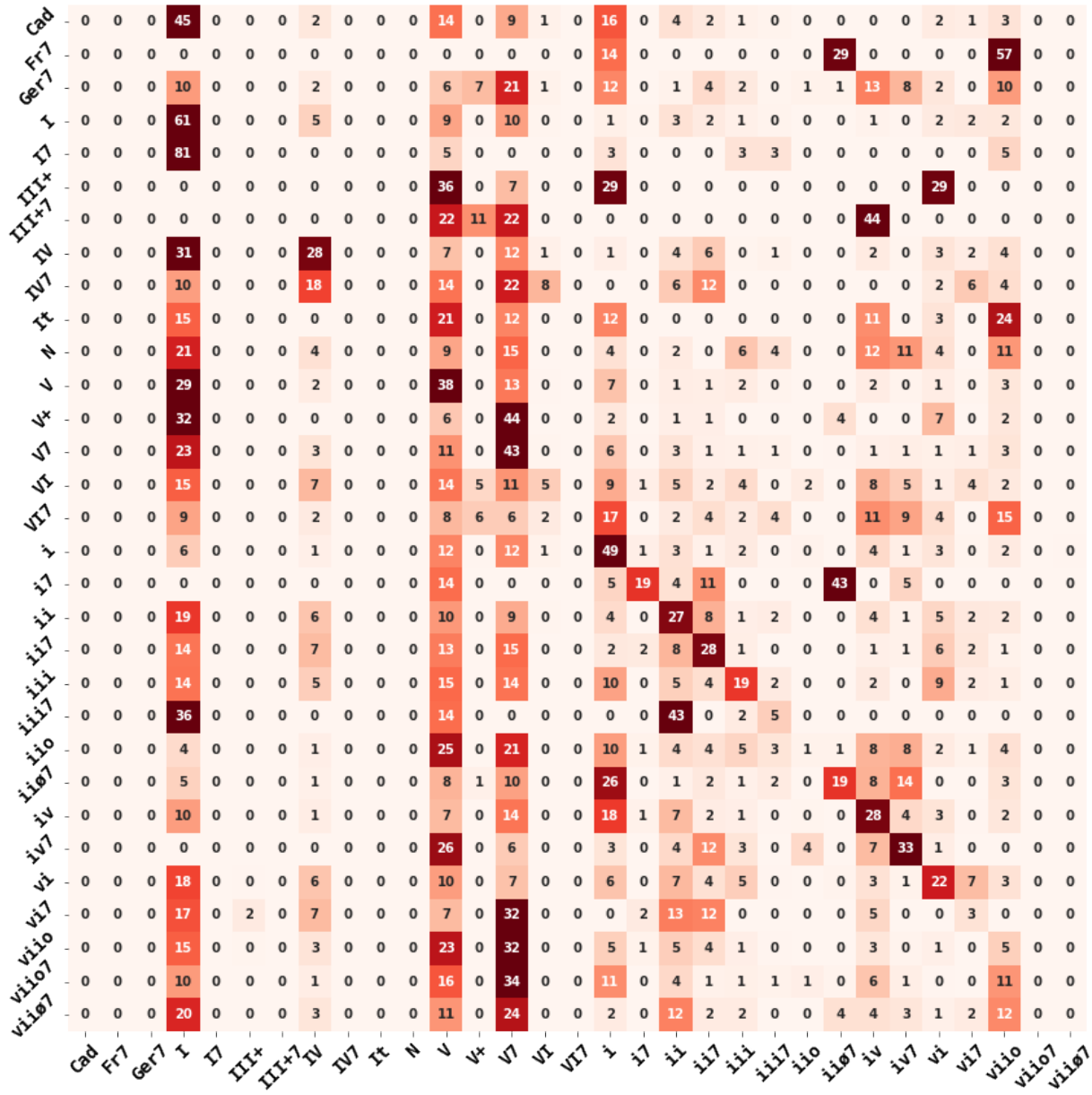


Figure 6.15: *Confusion matrix of the Roman numeral numerators (η) for the* ***Melisma*** *model. Rows represent the target class and columns the predicted class. Values reported in percentage, rounded to the closest integer. The color intensity is mapped to the percentage value.*

#### 6.5.3.2 Confusion Matrix for Chen and Su (2021)

Figure 6.16 shows the confusion matrix for the Chen and Su (2021) model. The model seemed to have a noticeably larger vocabulary than **Melisma**. Although the model did not explicitly recognize **Neapolitan** chords, the confusion matrix showed that 43% of the time it was able to provide the correct tonal context to decode a triad acting as a **Neapolitan**. This confirms the utility of the standardization provided by **NaTEM**. The Chen and Su (2021) model also seemed to have the highest recognition of some chords, for example, $\mathbf{iv^7}$, where the accuracy is higher than in any other model. Nevertheless, the model seemed to have a strong tendency to mislabel many chords with $\mathbf{V^7}$.

#### 6.5.3.3 Confusion Matrix for Micchi et al. (2021)

Figure 6.17 shows the confusion matrix for the Micchi et al. (2021) model. Among all the models, the Micchi et al. (2021) had the best recognition of **Neapolitan** chords, predicting 64.4% of them correctly. The vertical lines on the $\mathbf{V}$ and $\mathbf{V^7}$ are less prominent than in the Chen and Su (2021) and McLeod and Rohrmeier (2021) models, indicating perhaps that when the model mislabels a chord, it tends to be a distinct chord, rather than the often overused $\mathbf{V^7}$ chords, which most models emphasize in their predictions.

#### 6.5.3.4 Confusion Matrix for McLeod and Rohrmeier (2021)

Figure 6.18 shows the confusion matrix for the McLeod and Rohrmeier (2021) model. Below **AugmentedNet**, the McLeod and Rohrmeier (2021) seemed to have the best performance in this evaluation among all other models. Although the performance looks better than the previous models, it seems that the model did not benefit too much from the standardization provided by **NaTEM**. For example, it appears to have missed all the **augmented sixth** chords, except for 1% of the instances where it seemed to recognize a $\mathbf{Ger^7}$. Nevertheless, the performance on **Neapolitan** chords was good. This model and **Melisma** seemed to be the only

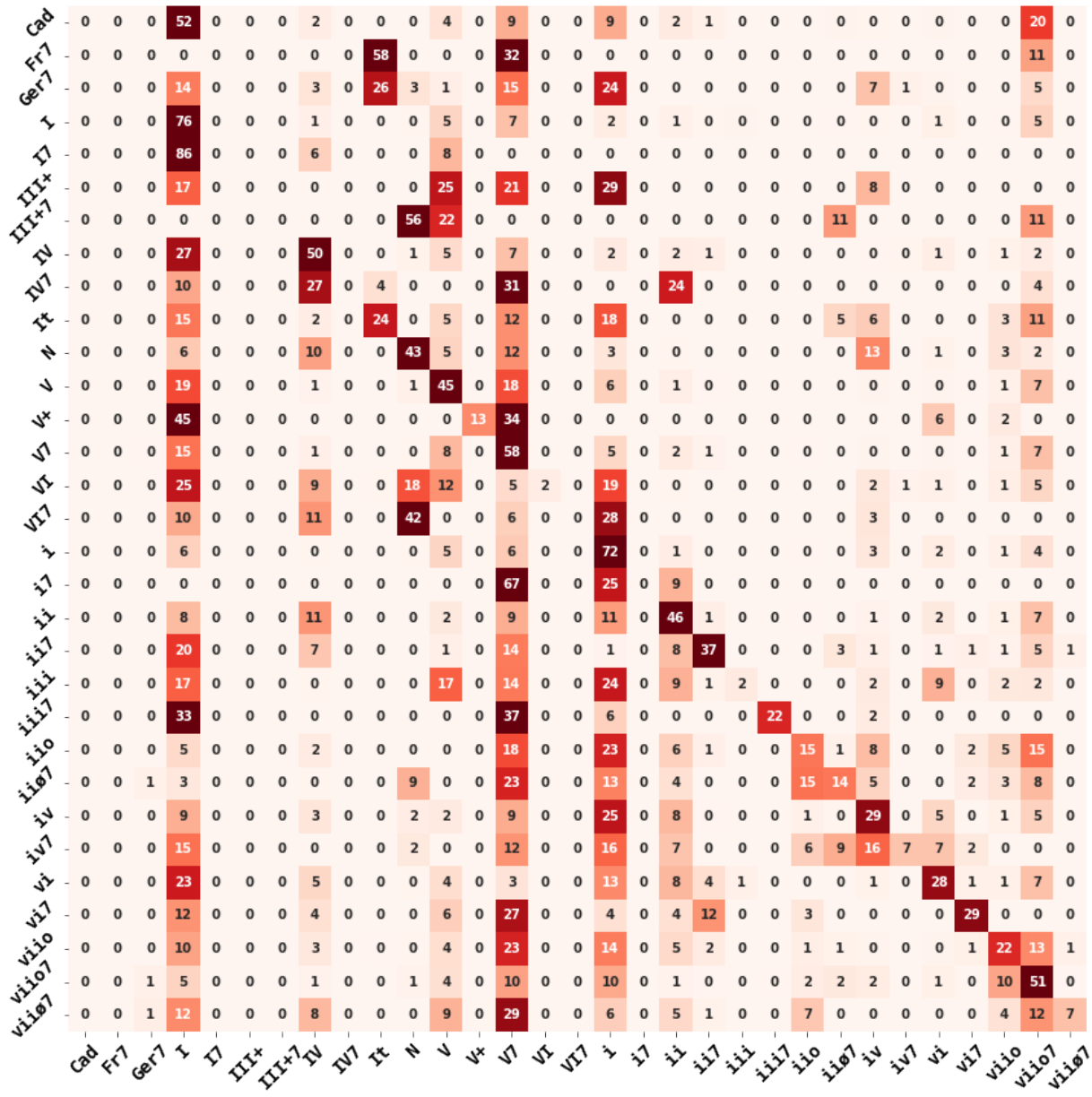


Figure 6.16: *Confusion matrix of the Roman numeral numerators (η) for the model in Chen and Su (2021). Rows represent the target class and columns the predicted class. Values reported in percentage, rounded to the closest integer. The color intensity is mapped to the percentage value.*

ones recognizing $\mathbf{i}^7$ chords, a strange situation considering that other forms of minor seventh chords were recognized by the other models.

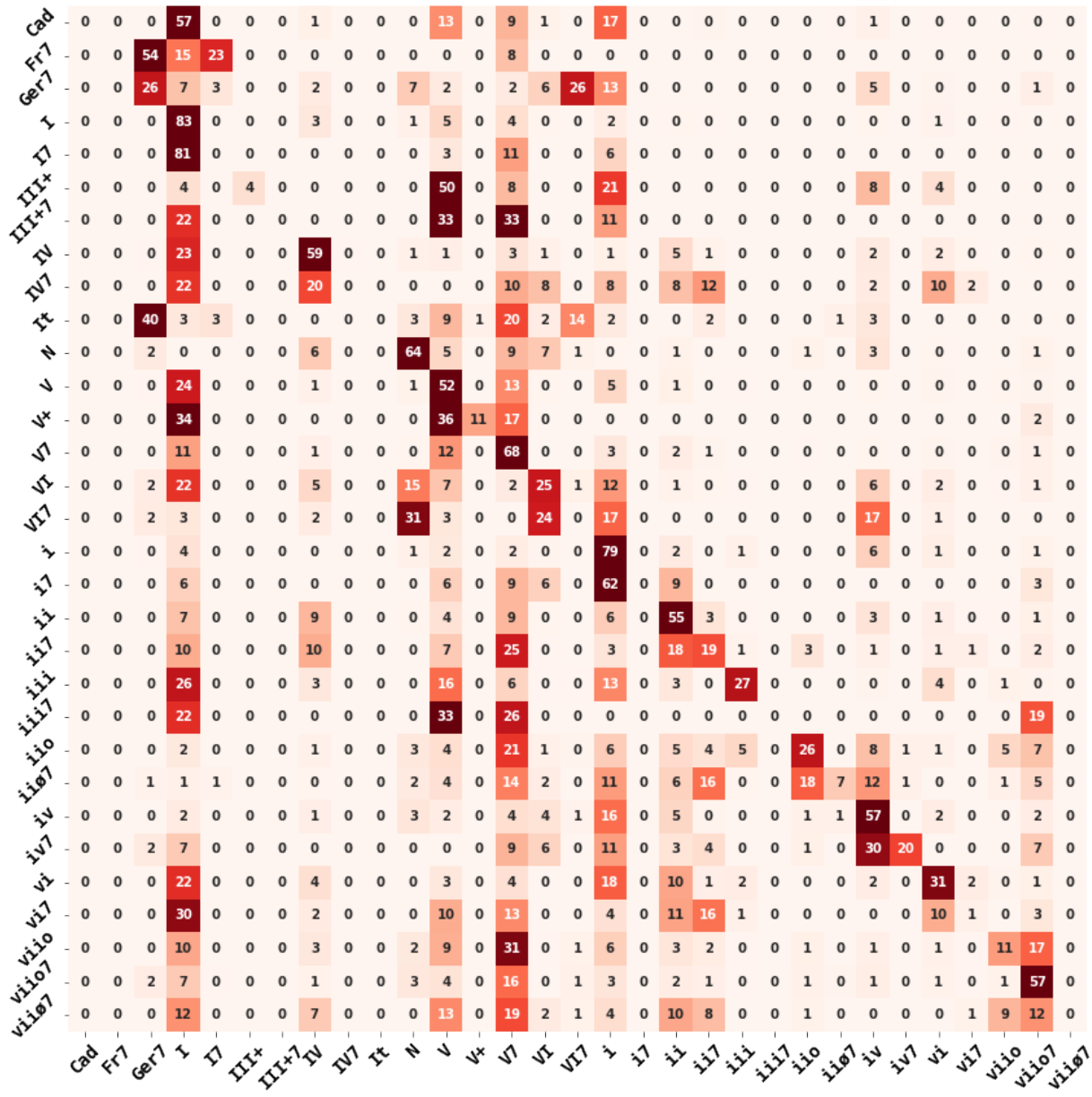


Figure 6.17: *Confusion matrix of the Roman numeral numerators (η) for model in Micchi et al. (2021). Rows represent the target class and columns the predicted class. Values reported in percentage, rounded to the closest integer. The color intensity is mapped to the percentage value.*

#### 6.5.3.5 Confusion Matrix for AugmentedNet

Figure 6.19 shows the confusion matrix for the **AugmentedNet** model. In this evaluation, the performance of **AugmentedNet** seemed to be above all other models. This is an interesting finding, because the results on the individual tasks shown in Table 6.12 seemed to present

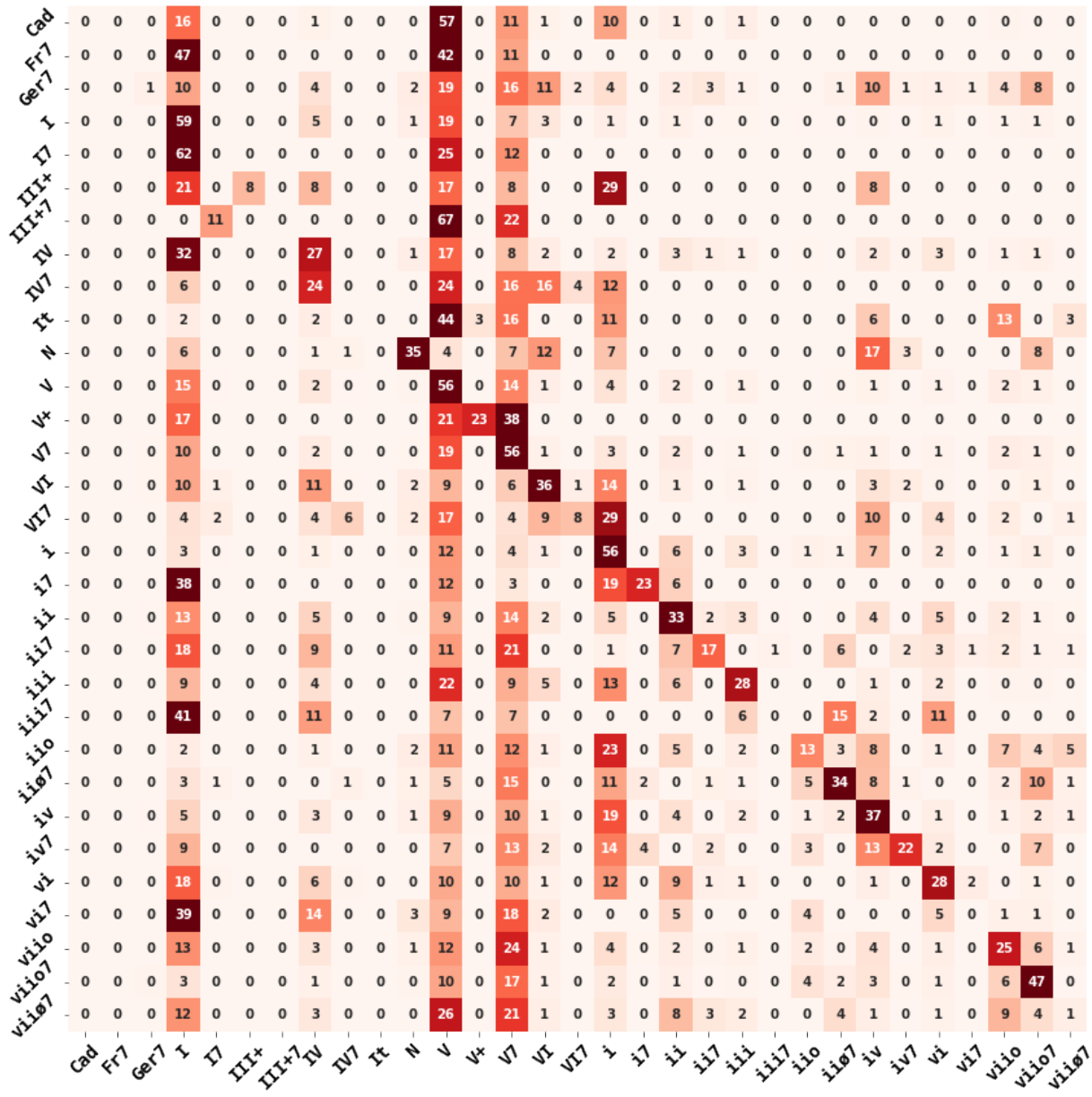


Figure 6.18: *Confusion matrix of the Roman numeral numerators (η) for model in McLeod and Rohrmeier (2021). Rows represent the target class and columns the predicted class. Values reported in percentage, rounded to the closest integer. The color intensity is mapped to the percentage value.*

a very similar performance between **AugmentedNet** and Micchi et al. (2021). However, in the confusion matrix of numerator predictions, it seems that McLeod and Rohrmeier (2021) provided a better performance, in general, than Micchi et al. (2021), and also that the gap between **AugmentedNet** and the other models seems larger. This may be an indication of the

misleading nature of chords. Being heavily skewed towards certain "common" classes (e.g., **I**, **V**, $\mathbf{V}^7$) in the musical practice, often the task of evaluating global accuracy does not show the full picture of which chords are overused by a model.

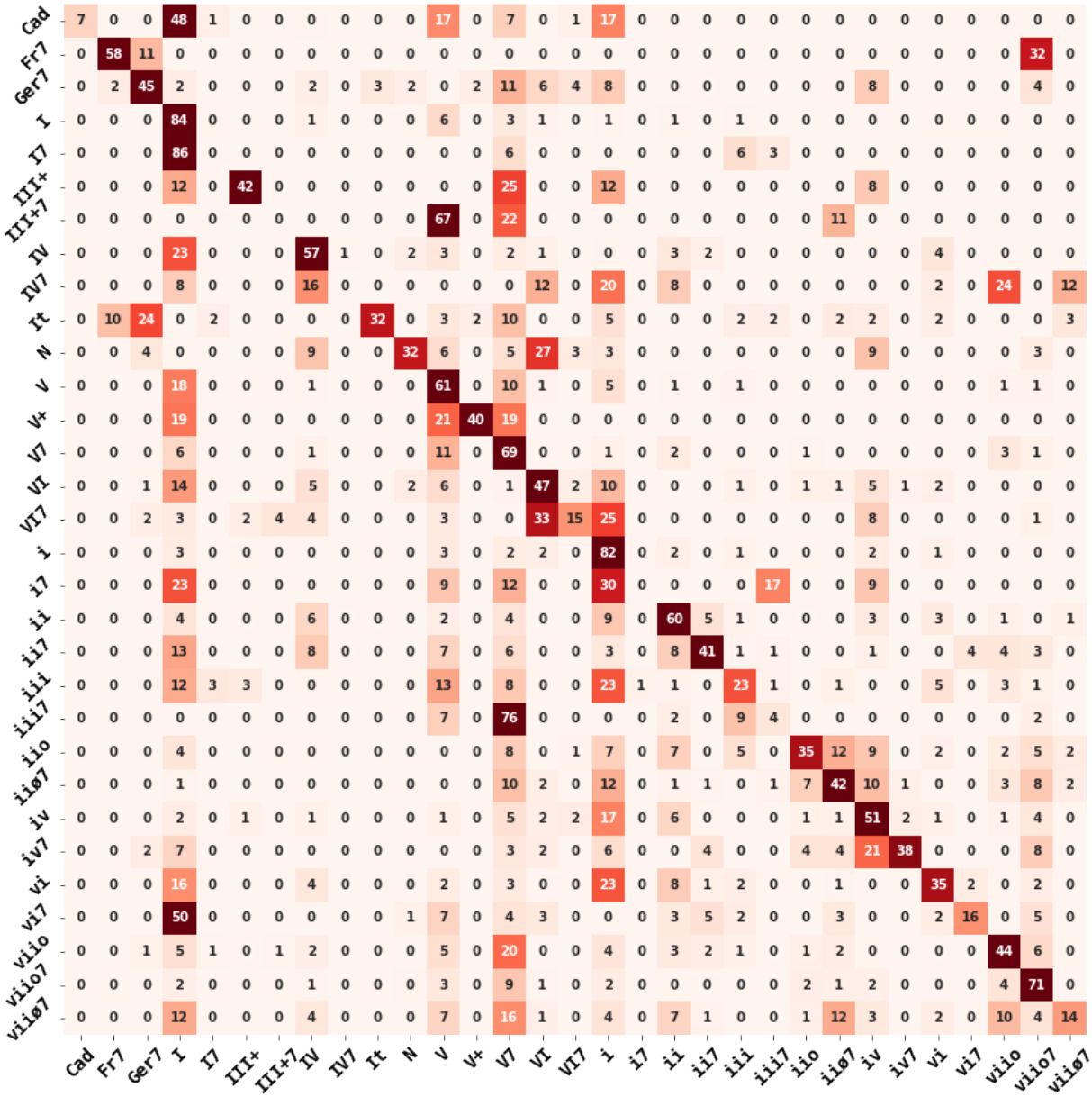


Figure 6.19: *Confusion matrix of the Roman numeral numerators (η) for AugmentedNet. Rows represent the target class and columns the predicted class. Values reported in percentage, rounded to the closest integer. The color intensity is mapped to the percentage value.*

#### 6.5.3.6 Summary of Performance on Difficult Chords

Perhaps, one of the most important aspects of **RNA** is that it provides an analytical framework for special chords, which are usually not considered in chord labels. For example, **Neapolitan** or **augmented sixth** chords.

These chords are extremely difficult to recognize because, in some instances, they occur in less than 1% of the annotations. Table 6.13 complements the results of the confusion matrices by comparing the accuracy of each model on each chord class, when the chord classes are sorted by least occurrence in the test set.

The results of this table are interesting, because it seems that different models succeed and fail in different sets of chords. For example, one of the most limited models, **Melisma**, was nevertheless capable of recognizing 19% of the $\mathbf{i}^7$ chords, whereas all the deep learning models except for McLeod and Rohrmeier (2021) struggled to recognize such chords. Out of all the models, **AugmentedNet** was the only one able to recognize $\mathbf{Fr}^7$ chords, which occur in only 0.03% of the annotations of the test set. Perhaps the reason why the model was fairly successful in this chord class is because $\mathbf{Fr}^7$ chords have a very specific intervallic structure. They are not enharmonics to any other chord class, except for another $\mathbf{Fr}^7$ chord in a different key. Thus, the **PitchClassSet121** classification task in **AugmentedNet** was particularly helpful in recognizing the unique **pcset** configuration of these chords.

## 6.6 Discussion

An **ARNA** model is more than a neural network architecture. It should be seen as an end-to-end model that consists both of machine learning components and musical decisions. When I started this project, I assumed that the goal was to create a "machine learning model" that would solve the problem. However, from the beginning, parsing different datasets in different annotation formats soon made me realize that domain knowledge was necessary to clean the datasets, standardize their musical assumptions, define the chord vocabulary, and guide the

Table 6.13: *Accuracy performance of the models in all 31 numerator classes, sorted by least occurrence in the test set.*

| Numerator | Occurrence (%) | Melisma | C&S21 | Mi21 | M&R21 | AugmentedNet |
|---|---|---|---|---|---|---|
| **III$^{+7}$** | 0.02 | 0 | 0 | 0 | 0 | 0 |
| **Fr$^{7}$** | 0.03 | 0 | 0 | 0 | 0 | **57.9** |
| **III$^{+}$** | 0.04 | 0 | 0 | 4.1 | 8.3 | **41.7** |
| **i$^{7}$** | 0.06 | 19.0 | 0 | 0 | **23.2** | 0 |
| **I$^{7}$** | 0.07 | 0 | 0 | 0 | 0 | 0 |
| **V$^{+}$** | 0.09 | 0 | 12.8 | 10.6 | 23.4 | **40.4** |
| **IV$^{7}$** | 0.09 | 0 | 0 | 0 | 0 | 0 |
| **iii$^{7}$** | 0.10 | 5.4 | **22.2** | 0 | 0 | 3.7 |
| **iv$^{7}$** | 0.16 | 33.3 | 7 | 20.0 | 22.2 | **37.8** |
| **vi$^{7}$** | 0.19 | 3.2 | **29.3** | 1.0 | 0 | 15.6 |
| **VI$^{7}$** | 0.19 | 0 | 0 | 0 | 7.7 | **15.3** |
| **It** | 0.22 | 0 | 24.3 | 0 | 0 | **32.4** |
| **vii$^{ø7}$** | 0.25 | 0 | 7.3 | 0 | 1.5 | **14.3** |
| **Ger$^{7}$** | 0.43 | 0 | 0 | 25.7 | 0.9 | **44.6** |
| **ii$^{o}$** | 0.47 | 0.6 | 14.6 | 26.0 | 13.5 | **35.0** |
| **N** | 0.51 | 0 | 43.4 | **64.2** | 35.0 | 32.0 |
| **ii$^{ø7}$** | 0.69 | 19.1 | 14.4 | 6.6 | 34.2 | **41.5** |
| **iii** | 0.78 | 18.7 | 2.0 | 27.0 | **27.8** | 23.2 |
| **ii$^{7}$** | 0.91 | 27.8 | 37.0 | 19.1 | 17.0 | **41.0** |
| **Cad$^{6}_{4}$** | 1.13 | 0 | 0 | 0 | 0 | **6.9** |
| **VI** | 1.40 | 5.5 | 1.6 | 25.0 | 36.0 | **47.4** |
| **vi** | 1.82 | 21.6 | 28.5 | 31.2 | 28.2 | **35.2** |
| **iv** | 1.88 | 28.1 | 29 | **56.6** | 37.2 | 51.4 |
| **vii$^{o}$** | 2.37 | 5.3 | 21.9 | 11.4 | 25.2 | **43.8** |
| **ii** | 3.70 | 27.4 | 45.5 | 54.9 | 33.0 | **60.3** |
| **IV** | 3.78 | 28.0 | 50.5 | **58.9** | 26.7 | 57.0 |
| **vii$^{o7}$** | 5.46 | 0 | 50.7 | 56.8 | 47.0 | **71.0** |
| **i** | 10.93 | 49.4 | 71.6 | 78.8 | 56.0 | **82.3** |
| **V** | 13.89 | 37.6 | 44.9 | 51.9 | 56.3 | **60.9** |
| **V$^{7}$** | 23.17 | 43.1 | 57.9 | 68.0 | 55.9 | **68.6** |
| **I** | 25.18 | 60.5 | 75.7 | 83.1 | 59.5 | **84.3** |

design of the network. This is true for the output of the model as well. For example, determining the number of classification outputs and how these are defined, the number of key layers in the musical analysis, or the spelling of certain Roman numerals (e.g., that infamous $\mathbf{I}^{6}_{4}$ vs $\mathbf{V}^{6}_{4}$). All of these aspects shape the design of an **ARNA** model.

Out of the ideas explored in the comparison process, the **NaTEM** algorithm seems particularly promising to me in future experiments. The design of tonal music analysis models is open to the interpretation of the researchers implementing them. This makes it very difficult to compare them. Perhaps the distribution of Table 6.13 summarizes my current thoughts on the design of the **ARNA** models. Even if a model is not competitive in most metrics, it may be useful in certain situations, or adopt certain design patterns that result beneficial for certain types of music or certain types of chords. Thus, it is worth the effort to search for tools that allow these models to "communicate" with each other, and obtain feedback on the things they do well and do poorly. Taking the performance on the $\mathbf{Fr^7}$ as an example, noticing the peculiar intervallic configuration of the chord and designing a representation that exploits it might be the difference between recognizing all $\mathbf{Fr^7}$ or none.

In order to complement the quantitative results presented in this section, Figure 6.20 shows a musical excerpt analyzed by the various **ARNA** systems. For reference, the example also shows the human annotations ("ground truth") in the dataset. The key of the excerpt is **g**, which all of the models except **Melisma** predict correctly. A brief discussion of the results by each model is presented below.

**Analysis of the Melisma model.** In general, the **Melisma** model seems to present more flaws than all other **ARNA** models. The most important one perhaps being mislabelling the key. After that, it misses the initial annotation of the tonic triad (m.1, beat 1). The model also has a tendency to provide long streams of chord annotations (e.g., m. 3), which greatly differs from the harmonic rhythm suggested by the human annotator or the other **ARNA** models.

**Analysis of the Chen and Su model.** After inspecting the annotations of **Melisma**, the Chen and Su (2021) model seems to provide better predictions. Importantly, all but one of the annotations (the $\mathbf{It^6}$ chord on m. 8, beat 2) in Chen and Su (2021) are either a tonic or dominant chord. This speaks about the bias that exists for the Roman numeral classes that appear more often in the dataset, as shown in Table 6.13.

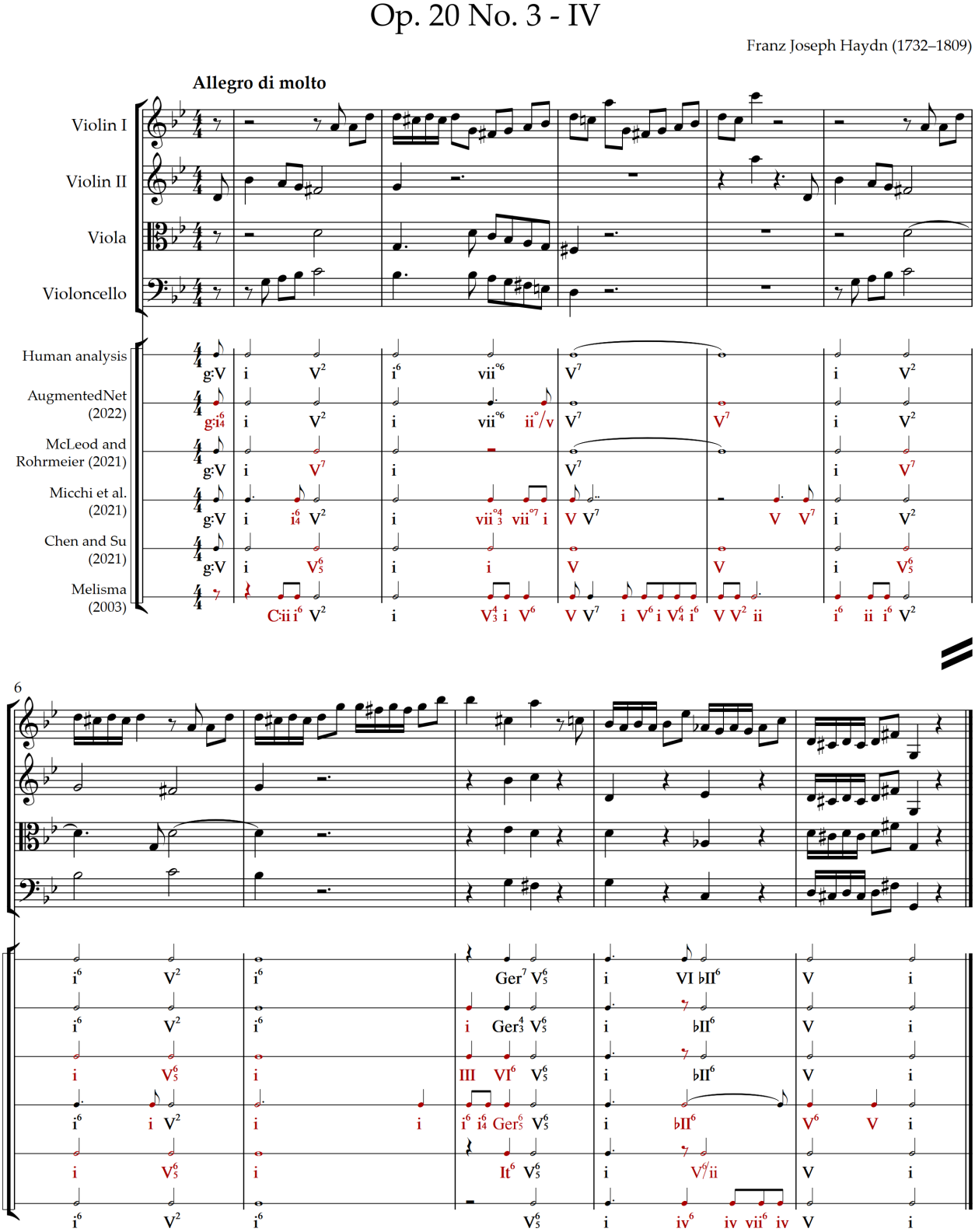


Figure 6.20: *Comparison of the annotations provided by a human analyst and various **ARNA** models. The musical excerpt is from Haydn's Op. 20 No. 3 - IV, a piece in the **HaydnSun** test set. The annotations from the **ARNA** models that differ from the human analyst's are marked in red.*

**Analysis of the Micchi et al. model.** The Micchi et al. (2021) model is less biased towards tonic and dominant chords. For instance, it recognizes instances of diminished chords (e.g., m. 2, beat 3). The model seems to struggle with the chord segmentation, as chords are often in odd locations. For example, the precipitated **Neapolitan** (m. 9, beat 3), or the two tonic triads in the previous measure (m. 8, beat 1). This is perhaps because the musical excerpt features an anacrusis, which makes it more complicated for the model to predict the appropriate harmonic rhythm.

**Analysis of the McLeod and Rohrmeier model.** The McLeod and Rohrmeier (2021) model does not seem to suffer from segmentation problems. In general, it also seems fairly similar to the human annotator. Two drawbacks of this model is that it often predicts the wrong inversion (e.g., m. 1, beat 3; m. 5, beat 3; and m. 6, beat 3). In addition to this, it is unable to recognize the **Ger$^{7}$** chord in the musical excerpt (m. 8, beat 2), proposing an annotation of **VI$^{6}$** instead. The **VI$^{6}$** annotation is a reasonable one when ignoring the C♯ in the first violin. However, this intervallic relationship between E♭4 and C♯5 is precisely what characterizes the **augmented sixth** chord here. This is perhaps a more subtle criticism, considering that **Ger$^{7}$** chords occur in less than 1% of the annotations, however, one could argue that it is in these subtleties that an **ARNA** model could appear to be better than others, in the eyes of a human analyst.

**Analysis of the AugmentedNet.** The proposed model in this dissertation, **AugmentedNet**, provides annotations that are generally similar to the human analysis. Among the points of discrepancy, the model predicts a diminished triad of a tonicized minor dominant (m. 2, beat 4.5), which coincides with the chord and inversion appearing in the music score, but that was not annotated by the human analyst. Another discrepancy is a second instance of the **V$^{7}$** chord in measure 4, which is considered by the analyst (and the McLeod and Rohrmeier (2021) model) as a continuation of the chord in measure 3. Lastly, the model (as well as all

other models) does not predict the **VI** chord instance expressed by the annotator before the **Neapolitan** (m.9, beat 2.5). Yet, most other annotations coincide with the human analysis.

Notice also two errors in the human analysis. The first one, in the $\mathbf{i^6}$ label (m. 2, beat 1), where the analyst indicated a first inversion. Nonetheless, because the viola crosses the violoncello at that point, the chord is in fact in root position. All **ARNA** models unequivocally get this annotation correct. The second error in the human analysis lies in the inversion of the $\mathbf{Ger^7}$ chord, indicated by the annotator as in "root position." The **AugmentedNet** model gets the annotation in the proper inversion, according to the conventions used in the **RomanText** format. This particular set of chords, **augmented sixth**, are often a source of disagreement regarding the definition of "root position." In the **RomanText** format, used to compare these annotations, the root of a $\mathbf{Ger^7}$ chord is located an augmented fourth above the tonic (i.e., C♯, in this case). The Micchi et al. (2021) correctly predicts the $\mathbf{Ger^7}$ chord, but the inversion suggests E♭ as the bass, which is not the case.

The discussion above regarding the $\mathbf{Ger^7}$ chord highlights another issue. Because digital **RNA** standards often differ in their digitization approach for certain chords, it is still challenging to distinguish a model's misperformance from a mistranslation of the data. In this dissertation, a meticulous effort was put in the quantitative evaluation and comparison of the models. However, more involvement from the community is needed here, in the translation between digital **RNA** standards. This would result not only in better **ARNA** models, but also in better evaluations for these models.

# Chapter 7

# Conclusions

This chapter presents the closing thoughts of this dissertation. Section 7.1 summarizes the main points made throughtout the dissertation. Section 7.2 discusses the main contributions. Section 7.3 describes the code repository accompanying this work as well as the location of pre-trained models, data, and experiments to reproduce or extend this work. Section 7.4 discusses future work.

## 7.1 Summary of the Dissertation

This dissertation presented an end-to-end system for **Automatic Roman Numeral Analysis (ARNA)**. Chapter 1 introduced the problem and the motivation to solve it. **Roman Numeral Analysis (RNA)** is a popular analytical system for Western tonal music, which allows analysts to explain complicated concepts (e.g., chromatic changes of harmony) using a relatively compact syntax. From an **Music Information Retrieval (MIR)** perspective, it is also an interesting problem because it encapsulates the tasks of recognizing chords and keys simultaneously. Thus, it provides a framework to design more complex tonal music analysis models and unified datasets to train and evaluate such models.

Chapter 2 discussed the **RNA** notation from the music theory perspective. In particular, a timeline of the evolution of the syntax was presented. This timeline scrutinized the use of sym-

bols and notations over historical textbooks, and how that probably led to the notation used today. This study linked tightly into the topic of digitization of **RNA** annotations, where the conventions and specificities of the notation are crucial to formalize what each of the symbols means. Several digital formats and their conventions were discussed at the end of this chapter.

Chapter 3 introduced a literature review on various relevant technical topics, such as music representation, deep neural networks, music information retrieval, and computational music theory. The end of chapter introduced the relevant literature on **ARNA** and most recent approaches tackling the problem.

Chapter 4 discussed the datasets used for training and testing the proposed system. The datasets were provided by different researchers and came in various formats. Thus, this chapter also discussed the challenges in aggregating the datasets, and the process of preparing the data. Near the end of the chapter, a new data augmentation technique was presented, which consisted of the synthesis of new training examples.

Chapter 5 discussed the components of the **ARNA** model. For example, it described the inputs and outputs, convolutional layers, and recurrent layers of the neural network. Additionally, this chapter also introduced the methods to turn the predictions of the neural network into **RNA** annotations.

Chapter 6 introduced the experiments to evaluate the proposed system. First, a number of ablation studies were proposed to demonstrate the role of different components of the neural network in the tasks of the **Multitask Learning (MTL)** layout. The model was later compared and evaluated against existing approaches for **ARNA**. The comparison considered common representations of the predicted features across algorithms, as well as the performance on difficult chords of the **RNA** vocabulary. Based on those experiments, I concluded that the proposed model generally provides a better performance than previous approaches on rare chords of the vocabulary, which makes it more useful for real applications. The last section of this chapter discussed a real musical example annotated by existing **ARNA** systems, and the implications of those analyses.

## 7.2 Summary of the Contributions

Throughout the dissertation, various contributions towards **ARNA** have been made in different dimensions of the problem. Chapter 2 presented a historical analysis of the **RNA** syntax. Although several accounts of the history of harmony exist,[1] most of these do not focus on the syntax of **RNA** annotations. The review presented in this dissertation focused solely in the use of symbols, conventions, and specificities of the **RNA** notation that were relevant for its digitization and standardization.

Chapter 4 documented the aggregation process of the, to the best of my knowledge, largest publicly available dataset of **ARNA** labels. The process to overcome the alignment between annotations and scores was discussed through the use of quantitive metrics and other resources that facilitate the aggregation of the data. Furthermore, in the upcoming Section 7.3.3 of this chapter, the preprocessed data is made publicly available for other researchers who may be interested in using it. The distributed dataset includes the explicit training, validation, and test splits used for the experiments conducted in this dissertation. Lastly, in the context of data augmentation, a new technique[2] based on the synthesis of training examples was presented in Section 4.5.1. Although less impactful than the more commonly used transposition, it presents the advantage that both can be used in combination, increasing the performance obtained by the network on the same dataset with both techniques in combination.

Chapter 5 described a novel neural network architecture for **ARNA**. The architecture is based on a previous version presented in Nápoles López, Gotham, and Fujinaga (2021), however, the structure of the **MTL** configuration presented here shows several improvements. Unlike other models, a set of tasks that has a more balanced distribution of target classes was proposed for different components. For example, rather than predicting chord qualities, which were heavily skewed towards major and dominant seventh examples in the dataset, chords are

1. See, for example, Wason (1985), Grave and Grave (1988), Hyer (2002), Laitz and Barlette (2010), or Sansa Llovich (2013).
2. First introduced in Nápoles López, Gotham, and Fujinaga (2021).

modeled by predicting four note classification tasks.[3] Each of these tasks represent a note in a **closed-position form** representation of the annotated chord. This is an equivalent task to predicting the notes that conform the chord, instead of the chord itself, which leads to a more balanced distribution of the target classes in a supervised learning scenario. Other contributions of the model include the use of a **PitchClassSet121** task, which summarizes the chord vocabulary of the system, a **HarmonicRhythm7** task based on 7 classes of outputs, and a vocabulary of 31 Roman numeral numerators, **Numerator31**. The input representations sent to the neural network also present contributions. For example, the **Onsets14** encoding of note and measure onsets improved the segmentation of the chords in the ablation studies shown in Section 6.1.6. The alternative representation of pitch spelling described in Section 5.1.3.1 was also shown to be an adequate subtitute for the more common representation $\mathcal{S}_{35}$, reducing the number of training parameters in the **Convolutional Recurrent Neural Network (CRNN)**.

Chapter 6 introduced contributions regarding the evaluation of an **ARNA** system. For example, the performance on rare chords was discussed among recent approaches in Section 6.5.3. This form of evaluation revealed that some of the chords have a tiny representation in the dataset, however, represent the chords and tonal situations that most often benefit from the **RNA** notation. In this respect, one of the contributions is to propose a system that can improve on such difficult cases, and to indicate to other researchers that methods for evaluating these rare occurrences of chords are needed in the future.

Lastly, the source code of the model, experiment logs, and general documentation of the software is offered as a contribution to future researchers in the field. These contributions are discussed in the next sections of this chapter.

3. Referred throughout the dissertation as the **SATB35** tasks.

## 7.3 Source Code and Reproducibility

The end-to-end system presented in this dissertation has been implemented as a software project in the Python programming language (Rossum et al. 2007). The code is publicly available on GitHub and it has been thoroughly version controlled and tested. The project's repository informally started in November 2020 and went through continuous revisions until the version presented in this dissertation.

Section 7.3.1 documents the releases of the project since 2020. These changes are often related to bug fixes, unit tests, and general software implementation. However, they also refer to the implementation of intellectual contributions highlighted in this dissertation. For example, `v1.6.0` introduced the **Onsets14** representation discussed in Section 5.1.4.3.

### 7.3.1 Releases

A comprehensive list of the releases of the system proposed in this dissertation is presented below. The tagged releases are ordered chronologically, with a brief description of the changes they introduce. Some of the tagged releases were left out for brevity.

**First Commit - February 12, 2021** This is the date of the initial commit of the project, with the commit identifier `f721860`.

**v0.0.1 - April 12, 2021** Initial tagged version of the software. Unstable. Closed source. Introduced the basic workflow for pairing annotation and scores into a tabular format.

**v0.1.0 - May 14, 2021** This is the commit version accompanying the paper submission to the **International Society for Music Information Retrieval (ISMIR)** 2021 Conference. The code is not considered to be in a state of "production," but it produces the final results in the submitted paper. The code continues to be closed source for double-blind review purposes.

**v1.0.0 - August 5, 2021** The published version of the **AugmentedNet** network in Nápoles López, Gotham, and Fujinaga (2021). This is the version of the code used for all the revised experiments in the camera-ready paper. Accompanying this release are the experiment logs, preprocessed data, and data splits of the paper. One (and maybe the only) substantial difference between the architecture submitted in the initial **ISMIR** submission and `v1.0.0`, is the use of *rmsprop* as the optimizer, instead of *Adam* (Kingma and Ba 2014).

**v1.1.0 - February 3, 2022** This version introduced the **HarmonicRhythm7** output task as a substitute of a previous binary classifier, which was similar to the one used in Chen and Su (2021). This addresses the class imbalance in the previous version of the task, which leads into a better chord segmentation. The four **SATB35** tasks were also added to the architecture in this version.

**v1.2.0 - February 5, 2022** Turned the **SATB35** tasks into the default ones computed by the network. These tasks are also used to compute the chords in the final **RNA** outputs of the model.

**v1.2.2 - February 5, 2022** The output **RNA** annotations produced by the system are reconstructed from the **SATB35** tasks by default. The `README` file was updated with instructions for inference and training. A *Jupyter* notebook (Kluyver et al. 2016) to run an inference demo was included. Lastly, an updated version of the dataset with the new output representations was distributed.

**v1.2.3 - February 14, 2022** Introduced the "key distance" algorithm (see Section A.7.2). This algorithm is used to find the closest key to a given key, which is useful to force tonicizations when a chord does not exist in the vocabulary.

**v1.2.4 - February 14, 2022** Extended the vocabulary of PitchClassSets to 121 classes. The resulting task **PitchClassSet121** has a more robust vocabulary than the previous version based

on 94 classes. This means that `v1.2.4` is backwards-incompatible with models trained on previous versions.

**v1.3.0 - February 14, 2022** Replacing the use of the `romanNumeralFromChord()` function of the *music21* (Cuthbert and Ariza 2010) library with Roman numerals from the custom chord vocabulary. Using the chord vocabulary facilitates the standardization of the chord names. The vocabulary is implemented to standardize both the training annotations that enter the system, and the output annotations generated in inference mode.

**v1.3.1 - February 14, 2022** Fixing unit tests to work with the pretrained model of `v1.3.0`. Fine-tuning the chord vocabulary and improving the implementation of the algorithm to force tonicizations (see Section A.7.2).

**v1.3.3 - February 14, 2022** The transposition algorithm for data augmentation is now based on tonicizations rather than modulations. Forcing all **ii**$^{\mathbf{o7}}$ annotations to be interpreted as **ii**$^{\mathbf{\emptyset 7}}$. Collapsing other annotations, such as $\mathbf{i}^{\mathbf{5}}_{\mathbf{4}}$ into the closest chord in the vocabulary using cosine similarity.

**v1.4.4 - February 21, 2022** Extending the vocabulary of keys from 35 to 38 clases. The new vocabulary spans the keys [**B**$\flat\flat$/, **D**$\sharp$] $\cup$ [**g**$\flat$, **b**$\sharp$]. Additionally, the translation of Roman numeral labels before training was improved.

**v1.5.0 - March 13, 2022** Padding the fixed-length inputs with a special token rather than a value of zero. This makes it possible to distinguish between "rest" and "padding." Added an additional option for texturizing the synthetic examples at each transposition (as opposed to one texturization per training file).

**v1.5.1 - March 26, 2022** This version removed the *dropout* layers (Dahl, Sainath, and Hinton 2013) that were originally part of the architecture. It was discovered that these layers were detrimental for the **HarmonicRhythm7** task.

**v1.6.0 - June 13, 2022** This version introduced the new input representation **Onsets14**. This input representation encodes the measure-and note-onset information. That is, it indicates when a new measure begins and the duration elapsed since the last measure. Additionally, it indicates when a new onset begins, and the duration elapsed since the last note onset. It dramatically improves the performance of the **HarmonicRhythm7** task (around 30% of accuracy improvement).

**v1.7.0 - June 17, 2022** The main contribution of this version is that it introduces the **Mozart Piano Sonatas (MPS)** dataset into the aggregated dataset. The code that organizes all the individual datasets was restructured as well.

**v1.7.1 - June 20, 2022** Patching an issue where percussion tracks would introduce nonsensical pitch information. Patching the **RomanText** output files, which were sometimes invalid. The `README` is pointed to the latest preprocessed dataset automatically. Removing drum parts from `music21.stream.Stream` objects before processing a **MusicXML** file. Improve the quality of the **RomanText** outputs generated by the system.

**v1.8.0 - June 25, 2022** Texturization patterns can handle durations of **Dotted half notes (𝅗𝅥.s)** and **Dotted quarter notes (♩.s)** notes. Renaming the **Onsets14** task in the code. The **Onsets14** was also divided into `MeasureOnset7` and `NoteOnset7`, for easier experimentation with note onsets and measure onsets. The **Haydn "Sun" String Quartets, Op. 20 (HaydnSun)** dataset was renamed internally, and so were the **Tonicization38** and **LocalKey38** tasks to correspond with the new vocabulary of 38 keys. Improving the evaluation of the model by removing any padding from the input test sequences, which were considered before.

**v1.9.0 - August 4, 2022** This version introduced the **Key Modulations and Tonicizations (KMT)** dataset into the aggregated dataset. This is the last dataset to be integrated into the set of seven publicly available datasets described in Chapter 4.

**v1.9.1 - August 7, 2022** This is a minor release. It fixed a bug where the annotations would not generate properly on **MusicXML** files with multiple voices. It also fixed another issue where $\mathbf{Cad}_{4}^{6}$ chords would not be labeled properly although the predictions were correct. Lastly, it removed the caching of **MusicXML** files from the *music21* Python library, as this lead to unpredictable behaviour during the final evaluation of the model.

### 7.3.2 Pretrained Model

A pretrained model of the **AugmentedNet** is distributed to the public. This model is the one trained for the experiment on the aggregated dataset, discussed in Section 6.3. More specifically, the baseline model described in Section 6.1.1, trained with the aggregated dataset discussed in Section 4.3, and applying the *transposition* and *synthesis* data-augmentation techniques as discussed in Section 6.2.4. The model can be used to annotate unseen **MusicXML** scores, or to compare the predictions of a newer model against **AugmentedNet**. The training session lasted for 16 hours, and it consisted of 200 epochs, where the metric of average validation accuracy across the 9 multitask classification tasks was used to choose the best checkpoint out of the 200 epochs of training. The model was trained on August 1, 2022 in the cluster allocations of the Digital Research Alliance of Canada (formerly known as Compute Canada).

The pretrained model was written using the *Tensorflow* and *Keras* libraries (Abadi et al. 2016; Chollet 2021). It is distributed in the **Hierarchical Data Format (v5) (HDF5)** format with the rest of the source code.[4]

4. https://github.com/napulen/AugmentedNet

### 7.3.3 Preprocessed Data

The aggregated dataset was processed in two sequential steps: a "human-friendly" **Tab-Separated Values (TSV)** file and a "machine-friendly" tensor of sequences. The **TSV** file was generated by processing **MusicXML** and **RomanText** source files with the *music21* (Cuthbert and Ariza 2010) and *pandas* (McKinney et al. 2011) packages into a "tabular" representation. During this process, the common error patterns were searched and measured in the **TSV** file, which made it easier to spot files that were misaligned or presented bad quality annotations. The tensor of sequences, written as a binary *numpy* (Oliphant 2006) multidimensional array was generated by encoding each **TSV** file into a numeric representation. The encoding of these tensors was based on the definition of the input (**LowestNote19**, **Notes19**, and **Onsets14**) and output (**Alto35**, **Bass35**, **HarmonicRhythm7**, **LocalKey38**, **PitchClassSet121**, **Numerator31**, **Soprano35**, **Tenor35**, and **Tonicization38**) representations.

The generation of the initial **TSV** file was generally slower[5] and did not change often. The generation of the tensor sequences was generally faster[6] and changed frequently for experimentation. Thus, a preprocessed collection of **TSV** files was distributed with each of the releases of the model. This can be used to reproduce the results of the experiments presented here or to facilitate the exploration of new tonal tasks or input/output representations.

### 7.3.4 Experiment Logs

Throughout the development of **AugmentedNet**, the experiments were documented with the *MLflow* Python library, introduced by Zaharia et al. (2018). Among other things, the software facilitates the recording of *git*[7] commits (i.e., version of the source code), name identifiers, and results achieved during an experiment. The logs for each of the experiments reported in the dissertation are shared in their original *MLflow* format (i.e., as an `mlruns` folder). One draw-

5. Nearly 50 minutes to compute on an Intel i7 10750 processor.
6. Nearly 3 minutes to compute on an Intel i7 10750 processor, considering the aggregated dataset (see Section 4.3) without data augmentation.
7. https://git-scm.com/

back is that the software requires some experience setting up a local development environment to visualize the results. Thus, as an additional convenience, the logs per training epoch of each experiment have also been uploaded to the *Tensorboard.dev* platform,[8] which is freely available. All the resources can be found in the source code repository of the model.

### 7.3.5 API Documentation

An **Application Programming Interface (API)** documentation of the system is provided.[9] Each of the Python modules of the system is documented with *docstrings*.[10] Although the documentation does not explain in great detail the implementation of the system, it provides useful hints about the structure of the code and the purpose of each Python module.

## 7.4 Future Work

There are several avenues where this work can be explored further. For example, researching better texturization techniques, improving the chord vocabulary, standardizing the **RNA** notation, supporting audio representations, and developing applications with the existing technology. Some thoughts on each of these topics are shared below.

### 7.4.1 Texturization

The synthesis technique explored for data augmentation relied on artificial texturization of the chords. The proposed method made use of three simple texturization patterns, described in Section 4.5.1.2. However, there might be better ways to perform this texturization, such as data-driven texturizaton patterns or introducing texturizations with synthetic **nonchord tone**s.

8. https://tensorboard.dev/
9. https://napulen.github.io/AugmentedNet
10. https://peps.python.org/pep-0257/

#### 7.4.1.1 Data-Driven Texturization

Recent work on generative deep learning models has shown latent representations that capture texture, such as the one by Wang et al. (2020). It would be interesting to apply these models to the texturization strategies used for **RNA** models. Another option would be to obtain common texturization patterns from existing music, rather than designing them with heuristics, as it was done here.

#### 7.4.1.2 Synthesizing Nonchord Tones

An approach that was not explored in the texturization techniques proposed here was the use of **nonchord tone**s. In all the texturization examples, the notes were exclusively chord tones. This is not, however, the case for most music of the **common-practice period**, because it often introduces passing notes, neighbouring notes, and other forms of **nonchord tone**s. The use of synthetic **nonchord tone**s was avoided because it naturally introduces additional problems, but these could be overcome and result beneficial to the quality of the data-augmentation examples.

### 7.4.2 Extending the Chord Vocabulary

The chord vocabulary proposed in this dissertation consists of 121 **pcset**s (see Section A.6) and 31 classes of Roman numeral "numerators" (see Section A.2). This vocabulary is extensive, but still neglects several chords that are used in practice. For example, it did not consider *common-tone diminished seventh chords* (**CT°7**), *minor-major seventh chords*, nor suspended chords of any kind. Figure 7.1 shows an example of such chords, which are left for future work.

### 7.4.3 Standardization

Annotating musical scores with **RNA** annotations continues to be, fortunately and unfortunately, an idiosyncratic process. The digitization standards presented in Section 2.3.1 will con-

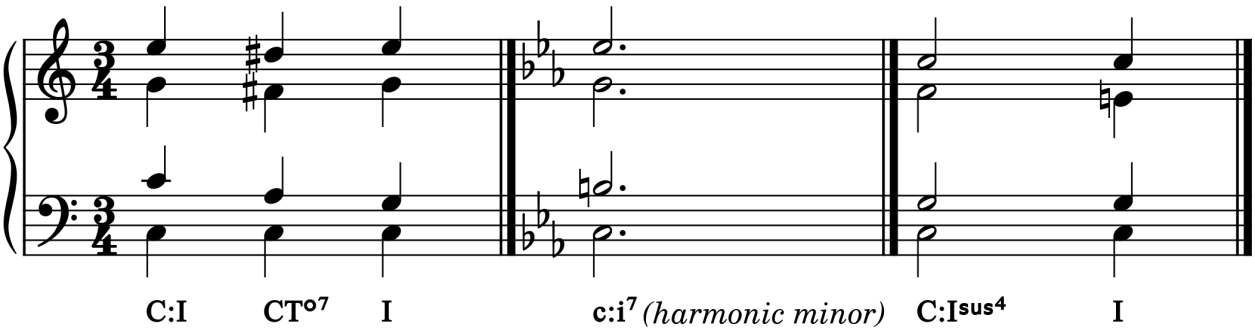


Figure 7.1: *Three types of chords that could extend the existing vocabulary: common-tone diminished seventh chords, major-minor seventh chords, and suspended chords, respectively.*

tinue to exist and update. New standards are expected to be introduced, and researchers will continue to prioritize different goals in their analytical work. For this reason, I consider that standardization will continue to be a challenge in this research. Nevertheless, as the tools mature, hopefully the compatibility and portability of different points of view will require less manual work. Some of the existing points of friction that I consider deserve more attention are the distinction between modulations and tonicizations, and the consistency in chord annotations (e.g., the conventions for inversions in certain chords).

#### 7.4.3.1 Consistency in Modulations and Tonicizations

In Nápoles López et al. (2020), we discussed the problem of a lack of definition of modulations and tonicizations, which has implications for **MIR** research. In that work, different harmony textbooks featured different degrees of consistency in the use of modulations and tonicizations. Considering that these represented easy "textbook" examples of modulations, it is only logical to expect a greater inconsistency in more complicated corpora, such as the dataset used in this dissertation. This is a difficult problem, but at the same time, a fascinating phenomenon of tonal music, the fluctuation of musical keys, how they are perceived by the human ear, and the way they are annotated in an analytical framework like **RNA**. This line of research probably requires more involvement of the **Music Perception and Cognition (MPC)** and music theory communities. I am less hopeful that the **MIR** community will, on its own, propose general solutions to formalize key changes.

#### 7.4.3.2 Consistency in Chords and Inversions

The problem of having consistent labels of chords and inversions is less intellectually profound than tackling modulations and tonicizations. Some researchers have different conventions for the definitions of chords that result in incompatibilities when aggregating datasets. For example, the "root" of a French **augmented sixth**, or the label for a so-called **cadential six-four**. Perhaps a conversation among analysts and dataset curators could fix these problems one day. They certainly take time away while performing **ARNA** research.

### 7.4.4 Audio Support

The proposed system and others such as the one by Micchi et al. (2021) and McLeod and Rohrmeier (2021) operate exclusively in the symbolic domain. One of the advantages of the representation proposed here for pitch information is that it is composed of the combination of pitch classes and generic note letters:

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\} \times \{C, D, E, F, G, A, B\} \tag{7.1}$$

The pitch-class part of this representation can be substituted by a chromagram vector in the audio domain, enabling different sorts of experiments. For example, training the network with only pitch classes in the audio, symbolic, or both domains. In the past, we have been successful implementing a hybrid model for symbolic and audio representations in Nápoles López, Vigliensoni, and Fujinaga (2019). A few preliminary experiments indicate that this would be possible in **RNA** as well, but more formal experiments are needed.

### 7.4.5 Applications

Although much research work is still necessary around **ARNA**, the existing technology might benefit from a concurrent development of applications. Some examples of possible applications are presented here.

#### 7.4.5.1 Batch Corpus Analysis

The pretrained end-to-end model described in Section 7.3.2 could be used to annotate batches of scores automatically for different purposes. For example, annotating all of the Bach Chorales[11] can be done in approximately 6 minutes using **AugmentedNet**. In the past, human-in-the-loop methods like the one by Ju et al. (2019) have been proposed, where an expert annotator corrects a set of auto-generated annotations to reduce the time required to annotate a dataset. Using an existing model to batch annotate scores could be a way of producing more training data for future models.

#### 7.4.5.2 Harmonic Reduction

The annotations generated by existing **ARNA** models are not generally sonified. However, this can be solved in various ways. One such way is to create a voice-leading algorithm that receives **RNA** inputs and generates a four-part harmonization of the annotations. This could help to, for example, produce the automatic harmonic reduction of a musical score analyzed by an **ARNA** model.

#### 7.4.5.3 Melody Harmonization

During the last year of my doctoral program, I participated in a research internship as part of the Mitacs organization.[12] In this project, I developed a generative algorithm to propose harmonizations to a given melody. Although the algorithm developed was of a generative nature, the core technology behind it was the proposed model in this dissertation. Figure 7.2 shows an example of this application. Other research directions at the intersection of analysis and generation could be explored.

11. https://github.com/craigsapp/bach-370-chorales
12. https://www.mitacs.ca/en/projects/deep-learning-automatic-melody-harmonization

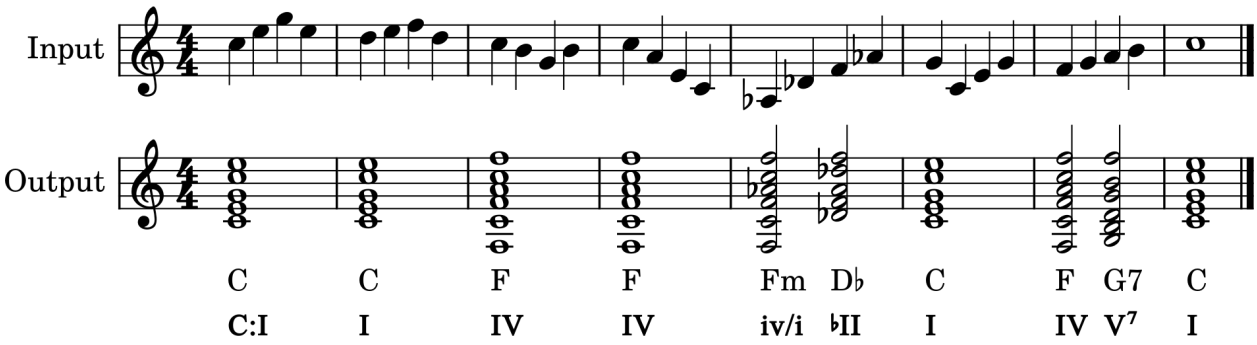


Figure 7.2: *A generative system that suggests chord accompaniments for a given melody. The input prompt to the system is the melody in the upper staff, which is harmonized with the labels in the lower staff. The notes in the lower staff are automatically realized from the labels.*

## 7.5 Closing Remarks

In conclusion, this dissertation described a two-year research effort on **ARNA**, with the aid of a many-year effort to understand music theory, music encoding, and computer science principles. The main contribution of the dissertation is the documented account of experimental results, machine learning and musical decisions, and software development considerations. If this dissertation was written six months from now, the **ARNA** model would look different to the one presented here. That is, there is much to do, many experiments to try, and many ideas worth implementing. Hopefully, the work presented here will be of help to future researchers tackling this and similar problems by illustrating some of the findings and lessons learned. In a not-too-long future, I foresee these systems being used by end-user musicians and students. In general, I think of **ARNA** systems as a type of "music theory" calculators. I do not think they are to replace the role of thorough—human—musical analysis, but they will automate some analytical tasks, like labelling clear unambiguous chords in clear unambiguous tonal contexts. It is common to see statisticians relying on tools that compute averages, standard deviations, and graphs automatically for them. I think it will be common too to see musicians using tools that predict chord labels, changes of key, and **RNA** labels for them.

# Appendix A

# A Method for Systematic Roman Numeral Analysis

This annex chapter presents a few technical ideas about **Roman Numeral Analysis (RNA)** that are relevant in various parts of the dissertation (e.g., Chapters 5 and 6). Thus, it is presented as a self-contained chapter referenced throughout other chapters.

Section A.1 describes the basic structure of an **RNA** label and its underlying components. The subsequent sections describe the vocabularies of Roman numeral numerators, denominators, keys, inversions, and other musical symbols that are relevant for systematic **RNA**.

## A.1 The Structure of a Roman Numeral Analysis Label

If we consider an **RNA** label to be a "fraction-like" expression, such as the one shown in Equation A.1, then the symbol $\kappa$ preceding the colon represents a key, the numerator $\eta$ represents a chord, the inversion $\iota$ indicates the bass of the chord, and the denominator $\tau$ represents a tonicized scale degree.

$$\kappa : \eta^{\iota} / \tau \tag{A.1}$$

Thus, in the annotation shown in Equation A.2, the preceding symbol $\kappa = \mathbf{C}$ indicates the key of **C**; the denominator $\tau = \mathbf{V}$ indicates a tonicization of $\kappa$'s dominant, **G**; and $\eta = \mathbf{vii^{ø7}}$ indicates a chord relative to the tonicized scale degree $\tau$, an F♯ half-diminished seventh chord. The inversion is generally a stack of Arabic numerals, which in this case is absent (see Section A.5 for further discussion on inversions).

$$\mathbf{C{:}vii^{ø7}/V} \tag{A.2}$$

The mode (major or minor) implied by the keys $\kappa$ and $\tau$ is indicated with a case-sensitive notation. For example, in the annotation shown in Equation A.3, the denominator $\tau = \mathbf{ii}$ indicates a key of **d**. In this case, the numerator $\eta = \mathbf{vii^{o7}}$ indicates a chord that is relative to the key of the denominator $\tau$, namely, a C♯ diminished seventh chord.

$$\mathbf{C{:}vii^{o7}/ii} \tag{A.3}$$

Tonicizations are optional and do not occur often. If there is no symbol $\tau$ found in the annotation, it can be assumed that $\tau = \mathbf{I}$ in major keys and $\tau = \mathbf{i}$ in minor keys. For example, in the annotation shown in Equation A.4, $\tau = \mathbf{I} \Rightarrow \mathbf{G}$.

$$\mathbf{G{:}V^7} \tag{A.4}$$

The "modulation" key[1] $\kappa$ is mandatory for the tonicization $\tau$ and numerator $\eta$ to have any meaning, but it can be omitted in subsequent annotations if it remains unchanged. For example, in the subsequent **RNA** annotations shown in Equation A.5, which are read from left to right, the symbols can be interpreted as in Table A.1.

$$\mathbf{a{:}iv} \qquad \mathbf{V^7/V} \qquad \mathbf{V} \tag{A.5}$$

1. I like to refer to key $\kappa$ as the "modulation" key. However, it is also common to refer to it as the "local key" in the **Music Information Retrieval (MIR)** literature (Nápoles López et al. 2020).

Table A.1: *Interpretation of the RNA annotations in Equation A.5.*

| Annotation | $\kappa$ (implied) | $\tau$ (implied) | $\eta$ |
|---|---|---|---|
| **a**:**iv** | **a** | (**i** $\Rightarrow$ **a**) | **iv** $\Rightarrow$ D minor |
| $\mathbf{V}^7/\mathbf{V}$ | (**a**) | **V** $\Rightarrow$ **E** | $\mathbf{V}^7$ $\Rightarrow$ B dominant seventh |
| **V** | (**a**) | (**i** $\Rightarrow$ **a**) | **V** $\Rightarrow$ E major |

In most instances, an RNA label will be made of only a numerator $\eta$, with keys $\kappa$ appearing either at the beginning of the piece or when modulations occur, and denominators $\tau$ appearing when there is a tonicized key, such that $\tau \neq \mathbf{I}$ and $\tau \neq \mathbf{i}$.

These general rules allow the interpretation of virtually any RNA label. However, one of the confusing aspects of the notation is perhaps the vocabulary of $\eta$, $\kappa$, and $\tau$ symbols that one can encounter in the notation, and the meaning of those when interpreted as chords.

The next sections describe the vocabularies of musical symbols $\mathcal{N}$, $\mathcal{K}$, and $\mathcal{T}$, among others, that satisfy $\forall\ \eta \in \mathcal{N}$, $\forall\ \kappa \in \mathcal{K}$, and $\forall\ \tau \in \mathcal{T}$ for all the annotations presented throughout the chapters. This complementary information facilitates the interpretation of RNA labels in the context of this dissertation.

## A.2 The Vocabulary of Roman Numeral Numerators

After a key $\kappa$ has been established in at least one annotation, a Roman numeral numerator $\eta$ is the only compulsory symbol in any subsequent RNA label. The other—optional—symbols being a new key $\kappa$, a tonicization $\tau$, and an inversion $\iota$.

A Roman numeral numerator always indicates a chord relative to a key. When a tonicized key $\tau$ is provided, the numerator $\eta$ is relative to this key. However, when $\tau$ is omitted, the numerator $\eta$ is relative to the key $\kappa$, which should have been indicated in at least the first RNA label of a piece. One additional simplification is to assume that if a tonicization $\tau$ is not indicated for a particular RNA label, then $\tau = \mathbf{I}$ or $\tau = \mathbf{i}$, depending on the mode of $\kappa$. Following this assumption, then the chord indicated by $\eta$ is always relative to the key implied

by $\tau$. The vocabulary of chords allowed for $\eta$ depends on the mode of the key implied by $\tau$. For brevity, I will refer to the key implied by $\tau$ as $\tau_\kappa$.

Some of the numerators are common to both major and minor modes. These are shown in Equation A.6.

$$\begin{aligned} \mathcal{N}_{M/m} &= \{\mathbf{Cad}_{\mathbf{4}}^{\mathbf{6}},\ \mathbf{V},\ \mathbf{vii}^{\mathbf{o}}, \mathbf{V}^{\mathbf{7}}, \mathbf{N},\ \mathbf{It},\ \mathbf{Fr}^{\mathbf{7}},\ \mathbf{Ger}^{\mathbf{7}}\} \\ |\mathcal{N}_{M/m}| &= 8 \end{aligned} \tag{A.6}$$

The set of numerators for major and minor keys consist of the union between the numerators that are common to both modes $\mathcal{N}_{M/m}$ and the numerators that are exclusive of a given mode. The set of numerators for major keys, $\mathcal{N}_M$, is shown in Equation A.7, and the set for minor keys, $\mathcal{N}_m$ is shown in Equation A.8. If $\tau_\kappa$ is a major key, there are 20 possible values for $\eta$; if $\tau_\kappa$ is a minor key, there are 19 possible values for $\eta$.

$$\begin{aligned} \mathcal{N}_M &= \mathcal{N}_C \cup \{\mathbf{I},\ \mathbf{ii},\ \mathbf{iii},\ \mathbf{IV},\ \mathbf{vi}, \mathbf{I}^{\mathbf{7}},\ \mathbf{ii}^{\mathbf{7}},\ \mathbf{iii}^{\mathbf{7}},\ \mathbf{IV}^{\mathbf{7}},\ \mathbf{vi}^{\mathbf{7}},\ \mathbf{vii}^{\mathbf{ø7}}, \mathbf{V}^{+}\} \\ |\mathcal{N}_M| &= 20 \end{aligned} \tag{A.7}$$

$$\begin{aligned} \mathcal{N}_m &= \mathcal{N}_C \cup \{\mathbf{i},\ \mathbf{ii}^{\mathbf{o}},\ \mathbf{III}^{+},\ \mathbf{iv},\ \mathbf{VI}, \mathbf{i}^{\mathbf{7}},\ \mathbf{ii}^{\mathbf{ø7}},\ \mathbf{III}^{+\mathbf{7}},\ \mathbf{iv}^{\mathbf{7}},\ \mathbf{VI}^{\mathbf{7}},\ \mathbf{vii}^{\mathbf{o7}}\} \\ |\mathcal{N}_m| &= 19 \end{aligned} \tag{A.8}$$

The complete vocabulary of Roman numeral numerators $\mathcal{N}$, shown in Equation A.9, comprises all numerators for major and minor modes, including the set of numerators shared by both modes. Thus, $\mathcal{N}$ indicates the set of Roman numeral numerators where $\forall\ \eta \in \mathcal{N}$. Notice that the cardinality of the vocabulary is of 31 Roman numeral numerators, because the duplicate numerators ($\mathcal{N}_{M/m}$) are only counted once.

$$\begin{aligned} \mathcal{N} &= \mathcal{N}_M \cup \mathcal{N}_m \\ |\mathcal{N}| &= 31 \end{aligned} \tag{A.9}$$

Regarding the meaning of the 31 numerators in the vocabulary, these correspond mainly to diatonic harmonies that can be constructed from a major or harmonic minor scale. A few chords are exceptions to this rule, which are indicated as "special chords." Table A.2 summarizes the 31 numerators and their chord context. An explanation of all the chords is provided below.

Table A.2: *Vocabulary $\mathcal{N}$ of valid Roman numeral numerators.*

| | Diatonic triads | Diatonic seventh chords | Special chords |
|---|---|---|---|
| Major mode | **I**, **ii**, **iii**, **IV**, **vi** | $\mathbf{I^7}$, $\mathbf{ii^7}$, $\mathbf{iii^7}$, $\mathbf{IV^7}$, $\mathbf{vi^7}$, $\mathbf{vii^{ø7}}$ | $\mathbf{V^+}$ |
| Minor mode | **i**, $\mathbf{ii^o}$, $\mathbf{III^+}$, **iv**, **VI** | $\mathbf{i^7}$, $\mathbf{ii^{ø7}}$, $\mathbf{III^{+7}}$, $\mathbf{iv^7}$, $\mathbf{VI^7}$, $\mathbf{vii^{o7}}$ | |
| Both modes | $\mathbf{Cad^6_4}$, **V**, $\mathbf{vii^o}$ | $\mathbf{V^7}$ | **N**, **It**, $\mathbf{Fr^7}$, $\mathbf{Ger^7}$ |

## A.2.1 Chords of the Major Mode

Except for **V**, $\mathbf{vii^o}$, and a few other chords, the RNA numerators constructed from the major mode are exclusive of a major key. That is, they do not exist in a minor key. Those chords are explained in this section.

### A.2.1.1 Major-Mode Triads

There are 5 Roman numerals generated from the diatonic triads of the major mode: **I**, **ii**, **iii**, **IV**, and **vi**.

### A.2.1.2 Major-Mode Seventh Chords

There are 6 Roman numerals generated from the diatonic seventh chords of the major mode: $\mathbf{I^7}$, $\mathbf{ii^7}$, $\mathbf{iii^7}$, $\mathbf{IV^7}$, $\mathbf{vi^7}$, and $\mathbf{vii^{ø7}}$.

#### A.2.1.3 Major-Mode Augmented Dominant

A chord occasionally found in the dataset is an augmented dominant chord. This chord is also common in the minor mode, however, it is an enharmonic of the $\mathbf{III}^{+}$ triad of that mode. In order for the algorithm described in Section A.7 to work, it is required that a given **pcset** $\rho$ has only one Roman numeral numerator $\eta$ association in a given key, a condition that would not be met for $\mathbf{III}^{+}$ and $\mathbf{V}^{+}$ coexisting in a minor key. One way to address this is to consider $\mathbf{V}^{+}$ a chord exclusive of the major mode and $\mathbf{III}^{+}$ a chord exclusive of the minor mode. When the method proposed in Section A.7 is trying to resolve a **pcset** $\rho$ that conforms to an augmented triad, it will be resolved as either $\mathbf{III}^{+}$ or $\mathbf{V}^{+}$ depending on the mode of the key.

### A.2.2 Chords of the Minor Mode

In a similar way to the diatonic chords of the major mode, the ones for the minor mode are described below.

#### A.2.2.1 Minor-Mode Triads

There are 5 Roman numerals generated from the diatonic triads of the minor mode: $\mathbf{i}$, $\mathbf{ii}^{o}$, $\mathbf{III}^{+}$, $\mathbf{iv}$, and $\mathbf{VI}$.

#### A.2.2.2 Minor-Mode Seventh Chords

There are 6 Roman numerals generated from the diatonic seventh chords of the minor mode: $\mathbf{i}^{7}$, $\mathbf{ii}^{\emptyset 7}$, $\mathbf{III}^{+7}$, $\mathbf{iv}^{7}$, $\mathbf{vi}^{7}$, and $\mathbf{vii}^{o7}$.

### A.2.3 Chords Shared in both Modes

Most of the chords in either mode are exclusive of that mode. That is, the same scale degree in the opposite mode has a different chord quality. Three diatonic chords are shared between both modes, however: $\mathbf{V}$, $\mathbf{V}^{7}$, and $\mathbf{vii}^{o}$.

The $\mathbf{Cad}_4^6$ is a special case among the diatonic triads. Unlike all other Roman numerals, which have one and only one associated chord quality, a $\mathbf{Cad}_4^6$ may be a major or minor triad, depending on the key's mode. In addition to the $\mathbf{Cad}_4^6$ chord, there are other special chords shared between the two modes.

#### A.2.3.1 Special Chords

One of the main uses of the RNA notation is to indicate chords that occur in special tonal contexts. The chords considered here are examples of those special contexts.

**Neapolitan Chord.** A Neapolitan triad is the major triad that forms from the "flatted" second degree in either mode, $\flat\mathbf{II}$. It is often used to substitute a "predominant" chord, for example, the **IV** of a major key. It is also common to find it in first inversion, with the third as the bass.

**Augmented Sixth Chords.** There are at least three kinds of augmented sixth chords: **It**, $\mathbf{Fr}^7$, and $\mathbf{Ger}^7$. These chords share commonalities. For example, they all feature an interval of an augmented sixth between the sixth degree (lowered sixth degree, if in major mode) and the raised fourth degree. The main difference between the three types of chords lies in the additional pitches that complete their configurations. An **It** chord is a triad, adding the root of the key as part of the chord. For example, in the key of $C$, the **It** chord is made of the notes $\{F\sharp, A\flat, C\}$. Using the same key as an example, the $\mathbf{Fr}^7$ chord includes the second degree, $\{F\sharp, A\flat, C, D\}$. The $\mathbf{Ger}^7$ chord, instead of the second degree, includes the third degree (lowered third degree, if in major mode). Thus, $\{F\sharp, A\flat, C, E\flat\}$.

The pcset configuration of **It** chords is unique. That is, there is no other Roman numeral numerator $\eta$ with the same pcset $\rho$ of an **It** chord, except for the same **It** in an enharmonic key. This is not true for $\mathbf{Fr}^7$ and $\mathbf{Ger}^7$. A $\mathbf{Fr}^7$ chord has the same pcset configuration as another $\mathbf{Fr}^7$ in a different (non-enharmonic) key. A $\mathbf{Ger}^7$ chord has the same pcset configuration as a $\mathbf{V}^7$ chord in a different key. These properties need to be considered when designing the chord

vocabulary, as the subtle distinction of one chord (**pcset**) in different keys, results in a very distinct **RNA** label.

## A.3 The Vocabulary of Roman Numeral Denominators

A denominator $\tau$ in an **RNA** label indicates a tonicized scale degree. The tonicized scale degree is always relative to the current key $\kappa$, which should be known at any given moment of the piece.

The tonicization $\tau$ is indicated with case-sensitive Roman numerals, where the Roman numeral indicates the scale degree, and the case of the Roman numeral indicates the mode. When the scale degree is not a diatonic scale degree of the key $\kappa$, an accidental can be used to alter it accordingly. Thus, the vocabulary $\mathcal{T}$ that satisfies $\forall\ \tau \in \mathcal{T}$ comprises all case-sensitive scale degrees relative to key $\kappa$. This vocabulary is shown in Equation A.10.

$$\mathcal{T} = \{\dots, \flat\flat, \flat, \natural, \sharp, \times, \dots\} \times \{\mathbf{i}, \mathbf{I}, \mathbf{ii}, \mathbf{II}, \mathbf{iii}, \mathbf{III}, \mathbf{iv}, \mathbf{IV}, \mathbf{v}, \mathbf{V}, \mathbf{vi}, \mathbf{VI}, \mathbf{vii}, \mathbf{VII}\} \tag{A.10}$$

In the design of the **AugmentedNet** model proposed in this dissertation, this vocabulary is never used, because it results inconvenient for multiclass classification. Instead, tonicizations are encoded as keys $\tau_\kappa$, which can always be reversed to their scale degree representation when writing the output **RNA** labels. See Section 5.4.6 for further details on the implementation of tonicizations.

## A.4 The Vocabulary of Musical Keys

If all enharmonic major-and-minor keys were collapsed into the same key class, then there would be a set of 24 keys. However, if two enharmonic keys are considered different keys, then the set of keys becomes an infinite set. In the first case, the keys can be arranged in a circular structure, such as the circle of fifths, because eventually the classes will repeat. In the

second case, the keys can be arranged in a **line of fifths** (Temperley 2000), such as the one shown in Table A.3.

Table A.3: *A **line of fifths** for major (top) and minor (bottom) keys.*

| ... | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | B𝄫 | F♭ | C♭ | G♭ | D♭ | A♭ | E♭ | B♭ | F | C | G | D | A | E | B | F♯ | C♯ | G♯ | D♯ | ... |

| ... | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | g♭ | d♭ | a♭ | e♭ | b♭ | f | c | g | d | a | e | b | f♯ | c♯ | g♯ | d♯ | a♯ | e♯ | b♯ | ... |

Designing a computational system that distinguishes between enharmonic keys is easier if the vocabulary of keys is bounded. Furthermore, many of the keys in the **line of fifths** exist in theory, but would rarely (or never) occur in practice. For example, the key of $C\times$ major.

In this dissertation, I bounded the vocabulary of major-and-minor keys to be finite. The set of available keys was determined in a data-driven way. By inspecting the examples provided in the aggregated dataset of **RNA** annotations (see Section 4.3), it was determined that most annotations lie within the range of keys between [**B𝄫**, **D♯**] for the major keys, and [**g♭**, **b♯**] for the minor keys. Thus, the vocabulary of keys considered in this dissertation spans all the keys in that range, as shown in Equation A.11. This results in 38 keys.

$$
\begin{aligned}
\mathcal{K}_M &= \{B\flat\flat, F\flat, C\flat, G\flat, D\flat, A\flat, E\flat, B\flat, F, C, G, D, A, E, B, F\sharp, C\sharp, G\sharp, D\sharp\} \\
\mathcal{K}_m &= \{g\flat, d\flat, a\flat, e\flat, b\flat, f, c, g, d, a, e, b, f\sharp, c\sharp, g\sharp, d\sharp, a\sharp, e\sharp, b\sharp\} \\
\mathcal{K} &= \mathcal{K}_M \cup \mathcal{K}_m \\
|\mathcal{K}| &= 38
\end{aligned}
\tag{A.11}
$$

## A.5 The Vocabulary of Arabic Numeral Inversions

A Roman numeral numerator $\eta \in \mathcal{N}$ is said to be in "root position" if the root of its chord is also acting as the bass. If one of the other chord tones is taking the role of the bass, the

Roman numeral numerator is said to be in either a first, second, or third inversion.[2] Thus, the vocabulary of inversions $\mathcal{I}$ for a numerator $\eta$ comprises the root position and three possible inversions, as shown in Equation A.12.

$$\mathcal{I} = \{0, 1, 2, 3\} \tag{A.12}$$

However, the annotation of the inversion in an **RNA** label is not written with the inversion number, but with a stack of Arabic numerals.[3] The precise notation depends on whether the numerator $\eta$ is a triad or a seventh chord, and it is summarized in Table A.4.

Table A.4: *Notation for the vocabulary $\mathcal{I}$ of inversions depending on the characteristics of $\eta$.*

| Inversion | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| When $\eta$ is a triad | $\eta$ | $\eta^{6}$ | $\eta^{6}_{4}$ | (absent in triads) |
| When $\eta$ is a seventh chord | $\eta^{7}$ | $\eta^{6}_{5}$ | $\eta^{4}_{3}$ | $\eta^{2}$ |

## A.6 The Vocabulary of Pitch-Class Sets

Given the set $\mathcal{C}$ of pitch classes in the Western chromatic scale $\mathcal{C} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, a pitch class set (**pcset**) is, in the context of this dissertation, a subset $\rho \subset \mathcal{C}$ comprising the pitch classes of a given triad or seventh chord. Several **RNA** annotations may share the same **pcset**, as shown in Equation A.13.

$$\begin{aligned} \text{pcset}(\mathbf{C{:}vii^{\emptyset 7}}) &= \{2, 5, 9, 11\} \\ \text{pcset}(\mathbf{a{:}ii^{\emptyset 7}}) &= \{2, 5, 9, 11\} \end{aligned} \tag{A.13}$$

2. Third inversions only occur if $\eta$ indicates a seventh chord, as a third inversion requires a chord with at least four notes.

3. See Section 2.2.3 for further discussion on why this is the case.

One of the advantages of formulating a vocabulary of chords $\mathcal{P}$ based on **pcset**s,[4] instead of a vocabulary of chords $\mathcal{R}$ based on **RNA** labels is that, because of the collisions of many chords in terms of their **pcset**, $|\mathcal{P}| \ll |\mathcal{R}|$. The main disadvantage of this is that, because several **RNA** labels will result in the same **pcset**, it is difficult to retrieve the original **RNA** back from the **pcset**. Nevertheless, using the musical key and the algorithm described in Section A.7, it is possible to retrieve the original numerator of the **RNA** annotation. Thus, I assume that a smaller vocabulary of chords $\mathcal{P}$, which is based on **pcset**s, will be useful to generate **RNA** labels if the **pcset** $\rho$ and key $\kappa$ can be predicted by a machine learning model.

The vocabulary of **pcset**s is constructed by combining all keys $\kappa \in \mathcal{K}$ with the Roman numeral numerators $\eta \in \mathcal{N}$, and extracting the **pcset** of the resulting chord. However, because there are Roman numeral numerators that are exclusive of the major or minor mode, the approach shown in Equation A.14 is preferred, where this process is divided among major and minor keys. The union of both sets for major and minor keys results in a set of 121 **pcset**s.

$$\begin{aligned}
\mathcal{P}_M &= \{\rho \mid \rho \text{ is the } \textbf{pcset} \text{ of } \kappa{:}\eta, \text{ and } \kappa \in \mathcal{K}_M \text{ and } \eta \in \mathcal{N}_M\} \\
\mathcal{P}_m &= \{\rho \mid \rho \text{ is the } \textbf{pcset} \text{ of } \kappa{:}\eta, \text{ and } \kappa \in \mathcal{K}_m \text{ and } \eta \in \mathcal{N}_m\} \\
\mathcal{P} &= \mathcal{P}_M \cup \mathcal{P}_m \\
|\mathcal{P}| &= 121
\end{aligned} \tag{A.14}$$

## A.7 The Numerator and Tonicization Estimation Method

This section describes the **NaTEM** algorithm, which is useful to retrieve a Roman numeral numerator $\eta$ and tonicization $\tau$ from a **pcset** $\rho$ and key $\kappa$, as shown in Equation A.15. This is useful when a machine learning system only generates chord labels and keys, which is often the case.

4. See Rohrmeier and Cross (2008) for another example of **pcset**s applied to a vocabulary of chords.

$$\textbf{NaTEM}(\rho, \kappa) = \eta/\tau \tag{A.15}$$

The retrieval process can be summarized in two steps. The first step is to acquire, if possible, the numerator $\eta$ from the given $\rho$ and $\kappa$, assuming a tonicization $\tau = \mathbf{I}$ or $\tau = \mathbf{i}$ (depending on $\kappa$'s mode). If this is not possible, the second, optional step, is to force a tonicization to retrieve a tonicization $\tau$ such that $\tau_\kappa \in \mathcal{K}_\rho$. Both steps are described in more detail below.

### A.7.1 Retrieving the Numerator

A **pcset** $\rho$ has an interpretation as one of the Roman numeral numerators $\eta \in \mathcal{N}$ in a subset of keys $\mathcal{K}_\rho \subset \mathcal{K}$ that is described in Equation A.16.

$$\mathcal{K}_\rho = \{\kappa \mid \kappa \in \mathcal{K}, \text{ and } \rho \text{ has an interpretation in key } \kappa\} \tag{A.16}$$

For example, given the **pcset** $\rho = \{0, 4, 7, 9\}$, there is an interpretation for that **pcset** in several keys $\kappa \in \mathcal{K}$, as shown in Equation A.17. This set of keys represents the subset $\mathcal{K}_\rho$ for this $\rho$, as shown in Equation A.18.

$$\text{The } \textbf{pcset } \rho = \{0, 4, 7, 9\} \text{ in key } \kappa = \begin{cases} \mathbf{C} & \text{is the pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{vi}^7 \\ \mathbf{e} & \text{is the pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{iv}^7 \\ \mathbf{F} & \text{is the pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{iii}^7 \\ \mathbf{G} & \text{is the pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{ii}^7 \\ \mathbf{a} & \text{is the pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{i}^7 \end{cases} \tag{A.17}$$

$$\text{if } \rho = \{0, 4, 7, 9\}\text{, then } \mathcal{K}_\rho = \{\mathbf{C}, \mathbf{e}, \mathbf{F}, \mathbf{G}, \mathbf{a}\} \tag{A.18}$$

Given a **pcset** $\rho \in \mathcal{P}$ and a key $\kappa \in \mathcal{K}_\rho$, the function $\Gamma(\rho, \kappa) = \eta$ returns the corresponding numerator $\eta$ such that pcset($\kappa$:$\eta$) = $\rho$, as shown in Equation A.19. This is the simple case of the **NaTEM** algorithm, as shown in Equation A.20.

$$\begin{aligned} \rho = \{0, 4, 7, 9\}, \;\; \kappa = \mathbf{C} \quad \Gamma(\rho, \kappa) = \mathbf{vi}^7 \\ \rho = \{0, 4, 7, 9\}, \;\; \kappa = \mathbf{e} \quad \Gamma(\rho, \kappa) = \mathbf{iv}^7 \end{aligned} \tag{A.19}$$

$$\textbf{NaTEM}(\rho, \kappa) = \Gamma(\rho, \kappa) \tag{A.20}$$

A limitation of the function $\Gamma$ is that $\forall \; \kappa \notin \mathcal{K}_\rho \; \Gamma(\rho, \kappa) = \varnothing$. This is a plausible problem when this method is used to retrieve a numerator $\eta$ from a predicted pair of $\rho$ and $\kappa$, generated by a machine learning system. The reason is that a machine learning system could predict a $\hat{\kappa}$ such that $\hat{\kappa} \notin \mathcal{K}_\rho$.[5] In that case, the Roman numeral numerator cannot be retrieved directly. However, it can be retrieved if a tonicized scale degree $\tau$ is imposed into the annotation (i.e., the key changes for that particular chord), such that $\tau_\kappa \in \mathcal{K}_\rho$.

In the more complicated case for a given $\rho$ and $\kappa$, where $\kappa \notin \mathcal{K}_\rho$, the numerator $\eta$ will be retrieved with an estimated $\tau$, such that $\tau_\kappa \in \mathcal{K}_\rho$. Thus, $\eta = \Gamma(\rho, \tau_\kappa)$ will lead to a $\eta \in \mathcal{N}$ and $\eta \neq \varnothing$. This will return an **RNA** label of the form shown in Equation A.21.

$$\textbf{NaTEM}(\rho, \kappa) = \Gamma(\rho, \tau_\kappa) \; / \; \tau \tag{A.21}$$

In the **RNA** of Equation A.21, the key $\kappa$ and $\rho$ are given inputs, the tonicization $\tau$ is a forced tonicization computed by this method, and the numerator $\eta$ is a chord relative to the tonicized key $\tau_\kappa$. The next section describes the process of obtaining $\tau$.

5. It is customary in machine learning literature to use the "hat" symbol to refer to the predictions of a machine learning system.

### A.7.2 Forcing a Tonicization

Consider the case of $\Gamma(\rho, \kappa)$ when $\rho = \{0, 2, 6, 9\}$ and $\kappa = \mathbf{C}$. In that case, $\rho$ has an associated numerator $\eta \in \mathcal{N}$ with the subset of keys shown in Equation A.22. Thus, that subset of keys is $\mathcal{K}_\rho$, as shown in Equation A.23.

$$\rho = \{0, 2, 6, 9\} \text{ in key } \kappa = \begin{cases} \mathbf{f}\sharp & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{Ger}^7 \\ \mathbf{F}\sharp & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{Ger}^7 \\ \mathbf{g}\flat & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{Ger}^7 \\ \mathbf{g} & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{V}^7 \\ \mathbf{G}\flat & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{Ger}^7 \\ \mathbf{G} & \text{is pcset}(\kappa{:}\eta) \text{ if } \eta = \mathbf{V}^7 \end{cases} \tag{A.22}$$

$$\mathcal{K}_\rho = \{\mathbf{f}\sharp, \mathbf{F}\sharp, \mathbf{g}\flat, \mathbf{g}, \mathbf{G}\flat, \mathbf{G}\} \tag{A.23}$$

However, we can see that for the given key $\kappa = \mathbf{C}$, $\kappa \notin \mathcal{K}_\rho$. In this case, a tonicization $\tau$ will be computed, such that $\tau_\kappa \in \mathcal{K}_\rho$. The computed tonicization $\tau$ must imply a "closely related" key to $\kappa$, in order for the tonicization to be easily interpretable in an **RNA** label.

One way to compute a "closely related" key is to use a metric of key distance between $\kappa$ and all scale degrees $\tau$ that satisfy $\tau_\kappa \in \mathcal{K}_\rho$. The tonicization $\tau_{\text{min}}$ with the minimal distance to $\kappa$ is a good candidate for a tonicization.

The key distance metric used is the Euclidean distance $d$ between $\kappa$ and all $\tau_\kappa \in \mathcal{K}_\rho$ in the Weber tonal chart of keys (Weber 1818). The chart is arranged in the layout shown in Table A.5.

Thus, the estimation of tonicization $\tau_{\text{min}}$ is shown in Equation A.24. For the example above with $\kappa = \mathbf{C}$, this results in the key-distance estimations shown in Table A.6.

$$\tau_{\text{min}} = \text{argmin}_{\forall \tau_\kappa \in \mathcal{K}_\rho} d(\kappa, \tau_\kappa) \tag{A.24}$$

Table A.5: *Weber's tonal chart of neighbouring keys. Column-wise, the keys follow a* ***line of fifths****. Row-wise, each key is surrounded by its relative and parallel major (or minor) keys.*

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E♭ | c | C | a | A | f♯ | F♯ | d♯ | D♯ | b♯ | B♯ | gx | Gx |
| A♭ | f | F | d | D | b | B | g♯ | G♯ | e♯ | E♯ | cx | Cx |
| D♭ | b♭ | B♭ | g | G | e | E | c♯ | C♯ | a♯ | A♯ | fx | Fx |
| G♭ | e♭ | E♭ | c | C | a | A | f♯ | F♯ | d♯ | D♯ | b♯ | B♯ |
| C♭ | a♭ | A♭ | f | F | d | D | b | B | g♯ | G♯ | e♯ | E♯ |
| F♭ | d♭ | D♭ | b♭ | B♭ | g | G | e | E | c♯ | C♯ | a♯ | A♯ |
| B♭♭ | g♭ | G♭ | e♭ | E♭ | c | C | a | A | f♯ | F♯ | d♯ | D♯ |
| E♭♭ | c♭ | C♭ | a♭ | A♭ | f | F | d | D | b | B | g♯ | G♯ |
| A♭♭ | f♭ | F♭ | d♭ | D♭ | b♭ | B♭ | g | G | e | E | c♯ | C♯ |
| D♭♭ | b♭♭ | B♭♭ | g♭ | G♭ | e♭ | E♭ | c | C | a | A | f♯ | F♯ |
| G♭♭ | e♭♭ | E♭♭ | c♭ | C♭ | a♭ | A♭ | f | F | d | D | b | B |
| C♭♭ | a♭♭ | A♭♭ | f♭ | F♭ | d♭ | D♭ | b♭ | B♭ | g | G | e | E |
| F♭♭ | d♭♭ | D♭♭ | b♭♭ | B♭♭ | g♭ | G♭ | e♭ | E♭ | c | C | a | A |

Table A.6: *Distance between the given key $\kappa$ and tonicizations $\tau_\kappa \in \mathcal{K}_\rho$. The $\tau_{\min}$ tonicization with the smallest distance, $d(\kappa, \tau) = 1.0$, is highlighted.*

| $\tau_\kappa$ | $d(\kappa, \tau_\kappa)$ |
|---|---|
| **f♯** | 3.0 |
| **F♯** | 3.6 |
| **g♭** | 4.2 |
| **g** | 1.4 |
| **G♭** | 3.6 |
| **G** | **1.0** |

In the example provided, the tonicization $\tau_{\min}$ chosen would correspond to $\tau = \mathbf{V}$ and $\tau_\kappa = \mathbf{G}$. When this tonicization is notated in relation to the given key $\kappa$, the resulting **RNA** label becomes $\mathbf{C{:}V^7/V}$. An experienced analyst that observes the given **pcset** $\rho = \{0, 4, 7, 9\}$ and key $\kappa = \mathbf{C}$ would perhaps provide the same answer in a short time. However, the relationships between chords and keys are not always trivial, and this method provides a programmatic way to obtain such **RNA** annotations. The quality of the **RNA** labels provided by this method is also heavily affected by the quality of the $\rho$ and $\kappa$ predictions provided, which is the duty of the machine learning model.

# Bibliography


Aarden, Bret J. 2003. “Dynamic Melodic Expectancy.” PhD diss., The Ohio State University.

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. “Tensorflow: A System for Large-Scale Machine Learning.” In *12th USENIX Symposium on Operating Systems Design and Implementation,* 265–283.

Abraham, Lars Ulrich. 1965. *Harmonielehre.* 1st ed.

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

Alchin, Carolyn A. 1921. *Applied Harmony.* 1st ed.

Aldwell, Edward, and Carl Schachter. 1978. *Harmony and Voice Leading.* 1st ed.

Aldwell, Edward, Carl Schachter, and Allen Cadwallader. 2019. *Harmony and Voice Leading.* Cengage Learning. ISBN: 978-1-337-56057-3. https://books.google.fr/books?id=T69EDwAAQBAJ.

Anger, Joseph Humfrey, and Henry Clough-Leighter. 1919. *A Treatise on Harmony.* 1st ed.

Bairstow, Edward Cuthbert. 1945. *Counterpoint and Harmony.* 1st ed.

Barnes, Archie Fairbairn. 1937. *Practice in Modern Harmony.* 1st ed.

Basaran, Dogac, Slim Essid, and Geoffroy Peeters. 2018. “Main Melody Estimation with Source-Filter NMF and CRNN.” In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 82–89. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492349. https://doi.org/10.5281/zenodo.1492349.

Bazin, François. 1857. *Cours d’harmonie théorique et pratique.* 1st ed.

Bellmann, Héctor. 2006. “About the Determination of Key of a Musical Excerpt.” In *Computer Music Modeling and Retrieval,* edited by Richard Kronland-Martinet, Thierry Voinier, and Sølvi Ystad, 76–91. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-34028-7.

Bello, Juan Pablo, and Jeremy Pickens. 2005. “A Robust Mid-Level Representation for Harmonic Content in Music Signals.” In *Proceedings of the 6th International Conference on Music Information Retrieval,* 304–311. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1417431. https://doi.org/10.5281/zenodo.1417431.

Bengio, Y., P. Simard, and P. Frasconi. 1994. “Learning Long-Term Dependencies with Gradient Descent Is Difficult.” *IEEE Transactions on Neural Networks* 5, no. 2 (March): 157–166. ISSN: 1045-9227, 1941-0093. https://doi.org/10.1109/72.279181.

Böck, Sebastian, Florian Krebs, and Gerhard Widmer. 2015. “Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters.” In *Proceedings of the 16th International Society for Music Information Retrieval Conference,* 625–631. Málaga, Spain: ISMIR, October. https://doi.org/10.5281/zenodo.1416026. https://doi.org/10.5281/zenodo.1416026.

———. 2016. “Joint Beat and Downbeat Tracking with Recurrent Neural Networks.” In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 255–261. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1415836. https://doi.org/10.5281/zenodo.1415836.

Boise, Otis Bardwell. 1898. *Harmony Made Practical.* 1st ed.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent. 2013. “Audio Chord Recognition with Recurrent Neural Networks.” In *Proceedings of the 14th International Society for Music Information Retrieval Conference,* 335–340. Curitiba, Brazil: ISMIR, November. https://doi.org/10.5281/zenodo.1418319. https://doi.org/10.5281/zenodo.1418319.

Bowman, Edward Morris. 1881. *Harmony: Historic Points and Modern Methods of Instruction.* 1st ed.

Bridge, Frederick. 1900. *A Course of Harmony.* 1st ed.

Broekhoven, John Andrew. 1889. *A System of Harmony for Teacher and Pupil.* 1st ed.

Buck, Percy Carter. 1920. *Unfigured Harmony.* 2nd ed.

Budge, Helen. 1943. “A Study of Chord Frequencies Based on the Music of Representative Composers of the Eighteenth and Nineteenth Centuries,” OCLC: 3300662. PhD diss., Teachers College, Columbia University.

Burgoyne, John Ashley, and Lawrence K. Saul. 2005. "Learning Harmonic Relationships in Digital Audio with Dirichlet-Based Hidden Markov Models." In *Proceedings of the 6th International Conference on Music Information Retrieval,* 438–443. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1414870. https://doi.org/10.5281/zenodo.1414870.

Burgoyne, John Ashley, Jonathan Wild, and Ichiro Fujinaga. 2011. "An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis." In *Proceedings of the 12th International Society for Music Information Retrieval Conference,* 11:633–638. Miami, FL.

Bussler, Ludwig. 1878. *Praktische Musikalische Compositionslehre in Aufgaben.* 1st ed.

Buwa, Johann. 1893. *Schule Der Accord-Verbindungen: Eine Harmonielehre Fur Schulen Und Zum Selbstunterrichts.* 2nd ed.

Calvo-Zaragoza, Jorge, Jan Hajič Jr., and Alexander Pacha. 2020. "Understanding Optical Music Recognition." *ACM Comput. Surv.* 53, no. 4 (July). ISSN: 0360-0300. https://doi.org/10.1145/3397499. https://doi-org.proxy3.library.mcgill.ca/10.1145/3397499.

Calvo-Zaragoza, Jorge, and David Rizo. 2018. "Camera-Primus: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 248–255. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492395. https://doi.org/10.5281/zenodo.1492395.

Calvo-Zaragoza, Jorge, Jose J. Valero-Mas, and Antonio Pertusa. 2017. "End-to-End Optical Music Recognition Using Neural Networks." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 472–477. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1418333. https://doi.org/10.5281/zenodo.1418333.

Calvo-Zaragoza, Jorge, Gabriel Vigliensoni, and Ichiro Fujinaga. 2017. "One-Step Detection of Background, Staff Lines, and Symbols in Medieval Music Manuscripts with Convolutional Neural Networks." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 724–730. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1417493. https://doi.org/10.5281/zenodo.1417493.

Cambouropoulos, Emilios. 2003. "Pitch Spelling: A Computational Model." *Music Perception: An Interdisciplinary Journal* 20 (4): 411–429.

Campbell, Spencer. 2010. "Automatic Key Detection of Musical Excerpts from Audio." Masters Thesis, McGill University.

Campbell-Watson, Frank. 1930. *Modern Elementary Harmony.* 1st ed.

Capellen, Georg. 1908. *Fortschrittliche Harmonie-und Melodielehre.* 1st ed.

Carter, Elliott. 2002. *Harmony Book.* 1st ed.

Catteau, Benoit, Jean-Pierre Martens, and Marc Leman. 2007. "A Probabilistic Framework for Audio-Based Tonal Key and Chord Recognition." In *Advances in Data Analysis,* edited by Reinhold Decker and Hans -J. Lenz, 637–644. Studies in Classification, Data Analysis, and Knowledge Organization. Springer Berlin Heidelberg. ISBN: 978-3-540-70981-7.

Chadwick, George Whitefield. 1897. *Harmony: A Course of Study.* 1st ed.

Chai, Wei, and Barry Vercoe. 2005. "Detection of Key Change in Classical Piano Music." In *Proceedings of the 6th International Conference on Music Information Retrieval,* 468–473. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1415538. https://doi.org/10.5281/zenodo.1415538.

Chen, Tsung-Ping, and Li Su. 2021. "Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models." *Transactions of the International Society for Music Information Retrieval* 4 (1).

———. 2018. "Functional Harmony Recognition of Symbolic Music Data with Multi-Task Recurrent Neural Networks." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 90–97. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492351. https://doi.org/10.5281/zenodo.1492351.

———. 2019. "Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition." In *Proceedings of the 20th International Society for Music Information Retrieval Conference,* 259–267. Delft, The Netherlands: ISMIR, November. https://doi.org/10.5281/zenodo.3527794. https://doi.org/10.5281/zenodo.3527794.

Chew, Elaine. 2002. "The Spiral Array: An Algorithm for Determining Key Boundaries." In *International Conference on Music and Artificial Intelligence,* 18–31. Springer.

———. 2000. "Towards a Mathematical Model of Tonality." PhD diss., Massachusetts Institute of Technology. Accessed August 5, 2019. https://dspace.mit.edu/handle/1721.1/9139.

Chew, Elaine, and Yun-Ching Chen. 2003. "Determining Context-Defining Windows: Pitch Spelling Using the Spiral Array." In *Proceedings of the 4th International Conference on Music Information Retrieval.* Baltimore, MD: ISMIR, October. https://doi.org/10.5281/zenodo.1418037. https://doi.org/10.5281/zenodo.1418037.

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation." *arXiv preprint arXiv:1406.1078.*

Choi, Keunwoo, György Fazekas, and Mark B. Sandler. 2016. "Automatic Tagging Using Deep Convolutional Neural Networks." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 805–811. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1416254. https://doi.org/10.5281/zenodo.1416254.

Chollet, François. 2021. *Deep Learning with Python.* 2nd ed. OCLC: 1289290141. ISBN: 978-1-61729-686-4.

Chuan, Ching-Hua, and Elaine Chew. 2005a. "Fuzzy Analysis in Pitch-Class Determination for Polyphonic Audio Key Finding." In *Proceedings of the 6th International Conference on Music Information Retrieval,* 296–303. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1417297. https://doi.org/10.5281/zenodo.1417297.

———. 2005b. "Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm." In *2005 IEEE International Conference on Multimedia and Expo,* 21–24. 2005 IEEE International Conference on Multimedia and Expo. Amsterdam, The Netherlands, July. https://doi.org/10.1109/ICME.2005.1521350.

Clarke, Hugh Archibald. 1898. *A System of Harmony.* 1st ed.

———. 1880. *Harmony on the Inductive Method.* 1st ed.

Coon, Oscar. 1883. *Harmony and Instrumentation.* 1st ed.

Crawford, Tim, and Richard Lewis. 2016. "Review: Music Encoding Initiative." *Journal of the American Musicological Society* 69 (1): 273–285. ISSN: 0003-0139. https://doi.org/10.1525/jams.2016.69.1.273.

Cuthbert, Michael Scott, and Christopher Ariza. 2010. "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." *Proceedings of the 11th International Society for Music Information Retrieval Conference.*

Cutter, Benjamin. 1899. *Exercises in Harmony.* 1st ed.

———. 1902. *Harmonic Analysis: A Course in the Analysis of the Chords and of the Non-Harmonic Tones to Be Found in Music, Classic and Modern.* 1st ed.

Dahl, George E., Tara N. Sainath, and Geoffrey E. Hinton. 2013. "Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout." In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing,* 8609–8613. IEEE.

Delbouys, Rémi, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, and Manuel Moussallam. 2018. "Music Mood Detection Based on Audio and Lyrics with Deep Neural Net." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 370–375. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492427. https://doi.org/10.5281/zenodo.1492427.

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

Devaney, Johanna, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. 2015. "Theme and Variation Encodings with Roman Numerals (TAVERN): A New Data Set for Symbolic Music Analysis." In *Proceedings of the 16th International Society for Music Information Retrieval Conference,* 728–734. Málaga, Spain: ISMIR, October. https://doi.org/10.5281/zenodo.1417497. https://doi.org/10.5281/zenodo.1417497.

Deveaux, Orpha-F. 1919. *Les principes de l'harmonie.* 1st ed.

Dieleman, Sander, Philemon Brakel, and Benjamin Schrauwen. 2011. "Audio-Based Music Classification with a Pretrained Convolutional Network." In *Proceedings of the 12th International Society for Music Information Retrieval Conference,* 669–674. Miami, FL, USA: ISMIR, October. https://doi.org/10.5281/zenodo.1415188. https://doi.org/10.5281/zenodo.1415188.

Dreyer, Ernst-Jürgen. 1977. *Entwurf einer zusammenhängenden Harmonielehre.* 1st ed.

Dubois, Théodore. 1921. *Traité d'harmonie théorique et pratique.* 1st ed.

Durand, Emile. 1881. *Traité complet d'harmonie théorique et pratique.* 1st ed.

Emery, Stephen. 1879. *Elements of Harmony.* 1st ed.

Eyben, Florian, Sebastian Böck, Björn W. Schuller, and Alex Graves. 2010. "Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks." In *Proceedings of the 11th International Society for Music Information Retrieval Conference,* 589–594. Utrecht, The Netherlands: ISMIR, August. https://doi.org/10.5281/zenodo.1417131. https://doi.org/10.5281/zenodo.1417131.

Eyken, Heinrich van. 1911. *Harmonielehre.* 1st ed.

Faraldo, Ángel, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. 2016. "Key Estimation in Electronic Dance Music." In *Proceedings of the 38th European Conference on Information Retrieval,* 335–347. Padua, Italy: Springer.

Feisthauer, Laurent, Louis Bigo, Mathieu Giraud, and Florence Levé. 2020. "Estimating Keys and Modulations in Musical Pieces." In *Sound and Music Computing Conference (SMC 2020).* Torino, Italy: Simone Spagnol / Andrea Valle, June. https://hal.archives-ouvertes.fr/hal-02886399.

Fétis, François-Joseph. 1844. *Traité complet de la théorie et de la pratique de l'harmonie.* 2nd ed.

Foote, Arthur. 1919. *Modulation and Related Harmonic Questions.* 1st ed.

Foote, Arthur, and Walter R. Spalding. 1905. *Modern Harmony in Its Theory and Practice.* 1st ed.

Forte, Allen. 1979. *Tonal Harmony in Concept and Practice.* 3rd ed.

Fowles, Ernest. 1918. *Harmony in Pianoforte-Study.* 1st ed.

Gardner, Carl E. 1912. *Essentials of Music Theory Elementary.* 1st ed.

Gilson, Paul. 1914. *Etude sur les intervalles diatoniques et chromatiques.* 1st ed.

———. 1919. *Traité d'harmonie.* 1st ed.

Gkiokas, Aggelos, and Vassilios Katsouros. 2017. "Convolutional Neural Networks for Real-Time Beat Tracking: A Dancing Robot Application." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 286–293. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1417737. https://doi.org/10.5281/zenodo.1417737.

Gladstone, Francis Edward. 1908. *A Manual of Harmony for Schools.* 1st ed.

———. 1898. *Five-Part Harmony.* 1st ed.

Goetschius, Percy. 1892. *The Theory and Practice of Tone-Relations.* 1st ed.

Goldman, Richard Franko. 1965. *Harmony in Western Music.* 1st ed.

Gómez, Emilia, and Perfecto Herrera. 2004. "Estimating the Tonality of Polyphonic Audio Files: Cognitive Versus Machine Learning Modelling Strategies." In *Proceedings of the 5th International Conference on Music Information Retrieval.* Barcelona, Spain: ISMIR, October. https://doi.org/10.5281/zenodo.1418007. https://doi.org/10.5281/zenodo.1418007.

Good, Michael. 2001. "MusicXML for Notation and Analysis." *The virtual score: Representation, retrieval, restoration* 12 (113): 160.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* Adaptive computation and machine learning. Cambridge, MA: The MIT Press. ISBN: 978-0-262-03561-3.

Goodrich, Alfred John. 1893. *Goodrich's Analytical Harmony.* 1st ed.

Gotham, Mark, Dmitri Tymoczko, and Michael Cuthbert. 2019. "The RomanText Format: A Flexible and Standard Method for Representing Roman Numerial Analyses." In *Proceedings of the 20th International Society for Music Information Retrieval Conference,* 123–129. Delft, The Netherlands: ISMIR, November. https://doi.org/10.5281/zenodo.3527756. https://doi.org/10.5281/zenodo.3527756.

Gotham, Mark Robert Haigh, and Peter Jonas. 2022. "The Openscore Lieder Corpus." In *Proceedings of the Music Encoding Conference.*

Grave, Floyd, and Margaret Grave. 1988. *In Praise of Harmony: The Teachings of Abbé Georg Joseph Vogler.* Lincoln, NE: University of Nebraska Press. ISBN: 0-8032-2128-2 978-0-8032-2128-4.

Grill, Thomas, and Jan Schlüter. 2015. "Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations." In *Proceedings of the 16th International Society for Music Information Retrieval Conference,* 531–537. Málaga, Spain: ISMIR, October. https://doi.org/10.5281/zenodo.1417461. https://doi.org/10.5281/zenodo.1417461.

Gururani, Siddharth, Cameron Summers, and Alexander Lerch. 2018. "Instrument Activity Detection in Polyphonic Music Using Deep Neural Networks." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 569–576. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492479. https://doi.org/10.5281/zenodo.1492479.

Hába, Alois. 1927. *Neue Harmonielehre.* 1st ed.

Halm, August. 1900. *Harmonielehre.* 1st ed.

Ham, Albert. 1919. *The Rudiments of Music and Elementary Harmony.* 1st ed.

Hamilton, James Alexander. 1840. *A Catechism of the Rudiments of Harmony and Thorough Bass.* 12th ed.

Hankinson, Andrew, Perry Roland, and Ichiro Fujinaga. 2011. "The Music Encoding Initiative as a Document- Encoding Framework." In *Proceedings of the 12th International Society for Music Information Retrieval Conference,* 293–298. Miami, FL: ISMIR, October. https://doi.org/10.5281/zenodo.1417609. https://doi.org/10.5281/zenodo.1417609.

Harasim, Daniel, Timothy O'Donnell, and Martin Rohrmeier. 2019. "Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony." In *Proceedings of the 20th International Society for Music Information Retrieval Conference,* 335–342. Delft, The Netherlands, November.

Harasim, Daniel, Martin Rohrmeier, and Timothy J. O'Donnell. 2018. "A Generalized Parsing Framework for Generative Models of Harmonic Syntax." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 152–159. Paris, France, September.

Harte, Christopher, Mark Sandler, and Martin Gasser. 2006. "Detecting Harmonic Change in Musical Audio." In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia,* 21–26. AMCMM '06. New York, NY: ACM. ISBN: 978-1-59593-501-4, accessed August 6, 2019. https://doi.org/10.1145/1178723.1178727. http://doi.acm.org/10.1145/1178723.1178727.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 770–778.

Heacox, Arthur E. 1917. *Keyboard Training in Harmony.* 1st ed.

Heacox, Arthur E. 1907. *Lessons of Harmony: Complete. Parts I and Ii.* 1st ed.

Hentschel, Johannes, Andrew McLeod, Fabian Moss, Markus Neuwirth, and Martin Rohrmeier. 2022. "Towards a Unified Model of Chords in Western Harmony." In *Music Encoding Conference.*

Hentschel, Johannes, Markus Neuwirth, and Martin Rohrmeier. 2021. "The Annotated Mozart Sonatas: Score, Harmony, and Cadence." *Transactions of the International Society for Music Information Retrieval* 4 (1): 67–80. ISSN: 2514-3298. https://doi.org/10.5334/tismir.63. http://transactions.ismir.net/articles/10.5334/tismir.63/.

Hindemith, Paul. 1943. *A Concentrated Course in Traditional Harmony.* 2nd ed.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. 2006. "A Fast Learning Algorithm for Deep Belief Nets." *Neural Comput.* 18 (7): 1527–1554. ISSN: 0899-7667. https://doi.org/10.1162/neco.2006.18.7.1527. http://dx.doi.org/10.1162/neco.2006.18.7.1527.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9, no. 8 (November 1, 1997): 1735–1780. ISSN: 0899-7667, accessed September 23, 2019. https://doi.org/10.1162/neco.1997.9.8.1735. https://doi.org/10.1162/neco.1997.9.8.1735.

Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. "Densely Connected Convolutional Networks." In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2261–2269. https://doi.org/10.1109/CVPR.2017.243.

Huang, Po-Sen, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. 2014. "Singing-Voice Separation from Monaural Recordings Using Deep Recurrent Neural Networks." In *Proceedings of the 15th International Society for Music Information Retrieval Conference,* 477–482. Taipei, Taiwan: ISMIR, October. https://doi.org/10.5281/zenodo.1415678. https://doi.org/10.5281/zenodo.1415678.

Hull, Arthur Eaglefield. 1915. *Modern Harmony: Its Explanation and Application.* 1st ed.

Humphrey, E. J., and J. P. Bello. 2012. "Rethinking Automatic Chord Recognition with Convolutional Neural Networks," 2:357–362. Proceedings of the 2012 11th International Conference on Machine Learning and Applications. https://doi.org/10.1109/ICMLA.2012.220.

Huron, David. 1994. *The Humdrum Toolkit: Reference Manual.* Center for Computer Assisted Research in the Humanities.

———. 2016. *Voice Leading: The Science Behind the Musical Art.* Cambridge, MA: MIT Press. ISBN: 978-0-262-03485-2.

Hyer, Brian. 2002. “Tonality.” In *The Cambridge History of Western Music Theory,* edited by Thomas Christensen, 726–752. The Cambridge History of Music. Cambridge, UK: Cambridge University Press. ISBN: 978-0-521-68698-3, accessed August 20, 2021. https://doi.org/10.1017/CHOL9780521623711.025. https://www.cambridge.org/core/books/cambridge-history-of-western-music-theory/tonality/8CA2CCF5615C16D2DA7E24C5DC1259CB.

Illescas, Plácido R, David Rizo, and José Manuel Inesta Quereda. 2007. “Harmonic, Melodic, and Functional Automatic Analysis.” In *International Computer Music Conference.*

Izmirli, Özgür. 2007. “Localized Key Finding from Audio Using Nonnegative Matrix Factorization for Segmentation.” In *Proceedings of the 8th International Conference on Music Information Retrieval,* 195–200. Vienna, Austria: ISMIR, September. https://doi.org/10.5281/zenodo.1417197. https://doi.org/10.5281/zenodo.1417197.

Izmirli, Özgür, and Semih Bilgen. 1994. “Recognition of Musical Tonality from Sound Input.” In *Proceedings of MELECON ’94. Mediterranean Electrotechnical Conference,* 269–271. Proceedings of MELECON ’94. Mediterranean Electrotechnical Conference. https://doi.org/10.1109/MELCON.1994.381110.

Jacobs, Robert Louis. 1958. *Harmony for the Listener.* 1st ed.

Jadassohn, Salomon. 1890. *Die Kunst zu Modulieren und zu Präludieren.* 1st ed.

———. 1883. *Lehrbuch der Harmonie.* 1st ed.

Jones, Robert Gomer. 1939. *Harmony and Its Contrapuntal Treatment.* 1st ed.

Ju, Yaolong. 2021. “Addressing Ambiguity in Supervised Machine Learning: A Case Study on Automatic Chord Labelling.” PhD diss., McGill University.

Ju, Yaolong, Samuel Howes, Cory McKay, Nathaniel Condit-Schultz, Jorge Calvo-Zaragoza, and Ichiro Fujinaga. 2019. “An Interactive Workflow for Generating Chord Labels for Homorhythmic Music in Symbolic Formats.” In *Proceedings of the 20th International Society for Music Information Retrieval Conference,* 862–869. Delft, The Netherlands: ISMIR, November. https://doi.org/10.5281/zenodo.3527950. https://doi.org/10.5281/zenodo.3527950.

Kallenberg, Siegfried Garibaldi. 1913. *Musikalische Kompositionsformen.* 1st ed.

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

Kirnberger, Johann Philipp. 1774. *Die Kunst des Reinen Satzes in der Musik.* 1st ed.

Kistler, Cyrill. 1879. *Harmonielehre für Lehrer und Lernende.* 1st ed.

Kitson, Charles Hebert. 1920. *Elementary Harmony.* 1st ed.

Klatte, Wilhelm. 1922. *Grundlagen des Mehrstimmigen Satzes (harmonielehre).* 1st ed.

Klauser, Julius. 1909. *The Nature of Music; Original Harmony in One Voice by Julius Klauser.* 1st ed.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. *Jupyter Notebooks-a Publishing Format for Reproducible Computational Workflows.* Vol. 2016.

Knorr, Iwan. 1921. *Aufgaben für den Unterricht in der Harmonielehre.* 4th ed.

Koch, Friedrich E. 1920. *Der Aufbau der Kadenz und Anderes.* 1st ed.

Koechlin, Charles. 1928. *Traité de l'harmonie.* 1st ed.

Korzeniowski, Filip, and Gerhard Widmer. 2018. "Genre-Agnostic Key Classification with Convolutional Neural Networks." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 264–270. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492399. https://doi.org/10.5281/zenodo.1492399.

Kostka, Stefan, and Dorothy Payne. 2008. *Tonal Harmony.* Boston, MA: McGraw-Hill Education. ISBN: 978-0-07-340135-5.

Kostka, Stefan M., and Dorothy Payne. 1984. *Tonal Harmony: With an Introduction to Twentieth-Century Music.* 1st ed.

Krebs, Florian, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. 2016. "Downbeat Tracking Using Beat Synchronous Features with Recurrent Neural Networks." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 129–135. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1417819. https://doi.org/10.5281/zenodo.1417819.

Krehl, Stephan. 1928. *Theorie der Tonkunst und Kompositionlehre.* 4th ed.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Everest Hinton. 2012. "Imagenet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems 25,* edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 1097–1105. Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Krumhansl, Carol L. 1990. *Cognitive Foundations of Musical Pitch.* Cognitive Foundations of Musical Pitch. New York, NY: Oxford University Press. ISBN: 978-0-19-505475-0 978-0-19-514836-7.

Krumhansl, Carol L., and Edward J. Kessler. 1982. "Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys." *Psychological Review* 89 (4): 334–368. ISSN: 1939-1471(Electronic),0033-295X(Print). https://doi.org/10.1037/0033-295X.89.4.334.

Krumhansl, Carol L., and Roger N. Shepard. 1979. "Quantification of the Hierarchy of Tonal Functions Within a Diatonic Context." *Journal of Experimental Psychology: Human Perception and Performance* 5 (4): 579–594. ISSN: 1939-1277(Electronic),0096-1523(Print). https://doi.org/10.1037/0096-1523.5.4.579.

Laitz, Steven Geoffrey, and Christopher A. Barlette. 2010. *Graduate Review of Tonal Theory.* 1st ed.

Lavignac, Albert. 1909. *Cours d'harmonie, théorique et pratique.* 1st ed.

Leavitt, Helen Sewall. 1916. *Practical Lesson Plans in Harmony.* 1st ed.

LeCun, Yann. 1989. "Generalization and Network Design Strategies." In *Connectionism in Perspective,* edited by R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels. Elsevier.

LeCun, Yann, Larry Jackel, Bernard Boser, John Denker, Hans Graf, Isabelle Guyon, Don Henderson, Richard Howard, and Wayne Hubbard. 1989. "Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning." *IEEE Communications Magazine* 27, no. 11 (November): 41–46. ISSN: 0163-6804, 1558-1896. https://doi.org/10.1109/35.41400.

Lee, Kyogu, and Malcolm Slaney. 2007. "A Unified System for Chord Transcription and Key Extraction Using Hidden Markov Models." In *Proceedings of the 8th International Conference on Music Information Retrieval,* 245–250. Vienna, Austria: ISMIR, September. https://doi.org/10.5281/zenodo.1415208. https://doi.org/10.5281/zenodo.1415208.

Leman, Marc. 1992. "Een model van toonsemantiek: naar een theorie en discipline van de muzikale verbeelding." PhD diss., Ghent University.

Lenormand, René. 1913. *Étude sur l'harmonie moderne.* 1st ed.

Lerdahl, Fred. 2005. *Tonal Pitch Space.* Oxford University Press. ISBN: 978-0-19-987037-0, accessed October 8, 2019. http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780195178296.001.0001/acprof-9780195178296.

Lester, Joel. 1982. *Harmony in Tonal Music.* 1st ed.

Levy, Ernst. 1985. *A Theory of Harmony.* 1st ed.

Liang, Feynman T., Mark Gotham, Matthew Johnson, and Jamie Shotton. 2017. "Automatic Stylistic Composition of Bach Chorales with Deep LSTM." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 449–456. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1416208. https://doi.org/10.5281/zenodo.1416208.

Lim, Hyungui, Seungyeon Rhyu, and Kyogu Lee. 2017. "Chord Generation from Symbolic Melody Using BLSTM Networks." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 621–627. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1417327. https://doi.org/10.5281/zenodo.1417327.

Liu, I.-Ting, and Richard Randall. 2016. "Predicting Missing Music Components with Bidirectional Long Short-Term Memory Neural Networks." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 225–231. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1417239. https://doi.org/10.5281/zenodo.1417239.

Lobe, Johann Christian. 1850. *Lehrbuch der Musikalischen Komposition.* 1st ed.

Loewengard, Max Julius. 1908. *Lehrbuch der Harmonie.* 4th ed.

Logier, Johann Bernhard. 1827. *System der Musik-Wissenschaft und der Praktischen Composition.* 1st ed.

Longuet-Higgins, Hugh Christopher. 1971. "On Interpreting Bach." *Machine intelligence* 6.

———. 1976. "Perception of Melodies." *Nature* 263, no. 5579 (October): 646–653. ISSN: 1476-4687, accessed November 14, 2019. https://doi.org/10.1038/263646a0. https://www.nature.com/articles/263646a0.

Lostanlen, Vincent, and Carmine-Emanuele Cella. 2016. "Deep Convolutional Networks on the Pitch Spiral for Music Instrument Recognition." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 612–618. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1416928. https://doi.org/10.5281/zenodo.1416928.

Louis, Rudolf, and Ludwig Thuille. 1907. *Harmonielehre.* 7th ed.

Macpherson, Charles Stewart. 1920. *Melody and Harmony: A Treatise for the Teacher and the Student.* 1st ed.

Magalhães, José Pedro, and W. Bas de Haas. 2011. "Functional Modelling of Musical Harmony an Experience Report." 156, *ACM SIGPLAN Notices* 46 (9): 156–162. ISSN: 0362-1340. https://doi.org/10.1145/2034574.2034797.

Mangold, Carl. 1883. *Harmony.* 1st ed.

Mauch, Matthias, and Simon Dixon. 2010a. "Approximate Note Transcription for the Improved Identification of Difficult Chords." In *Proceedings of the 11th International Society for Music Information Retrieval Conference.* Utrecht, The Netherlands.

———. 2010b. "Simultaneous Estimation of Chords and Musical Context from Audio." *IEEE Transactions on Audio, Speech, and Language Processing* 18 (6): 1280–1289. ISSN: 1558-7916. https://doi.org/10.1109/TASL.2009.2032947.

Maxwell, H. John. 1984. "An Artificial Intelligence Approach to Computer-Implemented Analysis of Harmony in Tonal Music." PhD diss., Indiana University. https://books.google.ca/books?id=mTtrHwAACAAJ.

———. 1992. "An Expert System for Harmonizing Analysis of Tonal Music." In *Understanding Music with AI: Perspectives on Music Cognition,* 334–353. Cambridge, MA: MIT Press. ISBN: 0-262-52170-9.

McConathy, Osbourne. 1927. *An Approach to Harmony.* 1st ed.

McCulloch, Warren S., and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics* 5 (4): 115–133. ISSN: 1522-9602, accessed November 6, 2019. https://doi.org/10.1007/BF02478259. https://doi.org/10.1007/BF02478259.

McEnnis, Daniel, Cory McKay, Ichiro Fujinaga, and Philippe Depalle. 2005. "JAudio: An Feature Extraction Library." In *Proceedings of the 6th International Conference on Music Information Retrieval,* 600–603. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1416648. https://doi.org/10.5281/zenodo.1416648.

McKinney, Wes, et al. 2011. "pandas: A Foundational Python Library for Data Analysis and Statistics." Publisher: Seattle, *Python for High Performance and Scientific Computing* 14 (9): 1–9.

McLeod, Andrew, and Martin A. Rohrmeier. 2021. "A Modular System for the Harmonic Analysis of Musical Scores Using a Large Vocabulary." In *Proceedings of the 22nd International Society for Music Information Retrieval Conference,* 435–442. Online: ISMIR, November. https://doi.org/10.5281/zenodo.5655391. https://doi.org/10.5281/zenodo.5655391.

Mearns, Lesley, Emmanouil Benetos, and Simon Dixon. 2011. "Automatically Detecting Key Modulations in J. S. Bach Chorale Recordings." In *Proceedings of the 8th Sound and Music Computing Conference,* 25–32.

Meister, Johann Georg. 1852. *Vollständige Harmonie-Und Generalbasslehre Und Einleitung Zur Composition.* 1st ed.

Meredith, David. 2005. “Comparing Pitch Spelling Algorithms on a Large Corpus of Tonal Music.” In *Computer Music Modeling and Retrieval,* edited by Uffe Kock Wiil, 173–192. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-31807-1.

———. 2003. “Pitch Spelling Algorithms.” In *Proceedings of the Fifth Triennial ESCOM Conference,* 204–207. Hannover, Germany.

———. 2006. “The PS13 Pitch Spelling Algorithm.” *Journal of New Music Research* 35 (2): 121–159.

Meyes, Richard, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. 2019. “Ablation Studies to Uncover Structure of Learned Representations in Artificial Neural Networks.” In *Proceedings on the International Conference on Artificial Intelligence (ICAI),* 185–191. The Steering Committee of The World Congress in Computer Science, Computer …

Micchi, Gianluca, Mark Gotham, and Mathieu Giraud. 2020. “Not All Roads Lead to Rome: Pitch Representation and Model Architecture for Automatic Harmonic Analysis.” *Transactions of the International Society for Music Information Retrieval* 3:42–54. https://doi.org/10.5334/tismir.45.

Micchi, Gianluca, Katerina Kosta, Gabriele Medeot, and Pierre Chanquion. 2021. “A Deep Learning Method for Enforcing Coherence in Automatic Chord Recognition.” In *Proceedings of the 22nd International Society for Music Information Retrieval Conference,* 443–451. Online: ISMIR, November. https://doi.org/10.5281/zenodo.5624539. https://doi.org/10.5281/zenodo.5624539.

Mickelsen, William. 1977. *Hugo Riemann’s Theory of Harmony.* 1st ed.

Minsky, Marvin, and Seymour A. Papert. 1972. *Perceptrons: An Introduction to Computational Geometry.* 2nd ed. Cambridge, MA: The MIT Press. ISBN: 978-0-262-63022-1 978-0-262-13043-1.

Mirka, Danuta. 2015. “The Mystery of the Cadential Six-Four.” In *What is a Cadence? Theoretical and Analytical Perspectives on Cadences in the Classical Repertoire,* edited by Markus Neuwirth and Pieter Bergé, 157–184. Leuven University Press, May. https://eprints.soton.ac.uk/377412/.

Miron, Marius, Jordi Janer, and Emilia Gómez. 2017. “Monaural Score-Informed Source Separation for Classical Music Using Convolutional Neural Networks.” In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 55–62. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1416498. https://doi.org/10.5281/zenodo.1416498.

Mitchell, William John. 1965. *Elementary Harmony.* 3rd ed.

Mokrejs, John. 1913. *Lessons in Harmony.* 1st ed.

Molitor, Gregor. 1913. *Die Diatonisch-Rhythmische Harmonisation der Gregorianischen Choralmelodien.* 1st ed.

Moog, Robert Arthur. 1986. "MIDI: Musical Instrument Digital Interface." *Journal of the Audio Engineering Society* 34 (5): 394–404.

Morris, Reginald Owen. 1931. *Foundations of Practical Harmony & Counterpoint.* 2nd ed.

———. 1946. *The Oxford Harmony, Vol. 1.* 1st ed.

Moss, Fabian C., Markus Neuwirth, and Martin Rohrmeier. 2022. "The Line of Fifths and the Co-Evolution of Tonal Pitch-Classes." Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17459 *Journal of Mathematics and Music* 0 (0): 1–25. https://doi.org/10.1080/17459737.2022.2044927. https://doi.org/10.1080/17459737.2022.2044927.

Motte, Diether de la. 1978. *Harmonielehre.* 1st ed.

Müller, Meinard. 2015. "Music Representations." In *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications,* 1–37. Cham: Springer International Publishing. ISBN: 978-3-319-21945-5. https://doi.org/10.1007/978-3-319-21945-5_1. https://doi.org/10.1007/978-3-319-21945-5_1.

Murphy, Howard Ansley, and Edwin John Stringham. 1951. *Creative Harmony and Musicianship.* 1st ed.

Nápoles López, Néstor. 2017. "Automatic Harmonic Analysis of Classical String Quartets from Symbolic Score." Masters Thesis, Universitat Pompeu Fabra, December. https://doi.org/10.5281/zenodo.1095617. https://doi.org/10.5281/zenodo.1095617.

Nápoles López, Néstor, Claire Arthur, and Ichiro Fujinaga. 2019. "Key-Finding Based on a Hidden Markov Model and Key Profiles." In *Proceedings of the 6th International Conference on Digital Libraries for Musicology.* New York, NY: ACM.

Nápoles López, Néstor, Laurent Feisthauer, Florence Levé, and Ichiro Fujinaga. 2020. "On Local Keys, Modulations, and Tonicizations: A Dataset and Methodology for Evaluating Changes of Key." In *Proceedings of the 7th International Conference on Digital Libraries for Musicology,* 18–26. New York, NY: Association for Computing Machinery. ISBN: 978-1-4503-8760-6. https://doi.org/10.1145/3424911.3425515.

Nápoles López, Néstor, and Ichiro Fujinaga. 2020a. "Harmalysis: A Language for the Annotation of Roman Numerals in Symbolic Music Representations." In *Proceedings of the Music Encoding Conference.* Boston, MA: MEC.

Nápoles López, Néstor, and Ichiro Fujinaga. 2020b. "Harmonic Reductions as a Strategy for Creative Data Augmentation." In *Late-Breaking Demo at 21st International Society for Music Information Retrieval Conference.* Montreal, Canada, October.

Nápoles López, Néstor, Mark Gotham, and Ichiro Fujinaga. 2021. "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." In *Proceedings of the 22nd International Society for Music Information Retrieval Conference,* 404–411. November.

Nápoles López, Néstor, Gabriel Vigliensoni, and Ichiro Fujinaga. 2018. "Encoding Matters." In *Proceedings of the 5th International Conference on Digital Libraries for Musicology,* 69–73. New York, NY: ACM. ISBN: 978-1-4503-6522-2, accessed September 26, 2019. https://doi.org/10.1145/3273024.3273027.

———. 2019. "The Effects of Translation Between Symbolic Music Formats: A Case-Study with Humdrum, Lilypond, MEI, and MusicXML." In *Poster at the Music Encoding Conference.* Event-. Vienna, Austria.

Neuwirth, Markus, Daniel Harasim, Fabian C. Moss, and Martin Rohrmeier. 2018. "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets." *Frontiers in Digital Humanities* 5. ISSN: 2297-2668, accessed August 5, 2019. https://doi.org/10.3389/fdigh.2018.00016. https://www.frontiersin.org/articles/10.3389/fdigh.2018.00016/full.

Ninov, Dimitar. 2016. "Functional Nature of the Cadential Six-Four." Publisher: University of Ljubljana, Faculty of Arts, *Muzikoloski zbornik* 52 (1): 73.

Noland, Katy C., and Mark B. Sandler. 2006. "Key Estimation Using a Hidden Markov Model." In *Proceedings of the 7th International Society for Music Information Retrieval Conference,* 121–126. Victoria, Canada.

Norris, Homer Albert. 1894. *Practical Harmony: A Comprehensive System of Musical Theory on a French Basis.* 1st ed.

Oakey, George. 1884. *Text Book of Harmony.* 7th ed.

Oettingen, Arthur von. 1866. *Harmoniesystem in Dualer Entwicklung.* 1st ed.

Oliphant, Travis E. 2006. *A Guide to Numpy.* Vol. 1. Trelgol Publishing USA.

Orem, Preston Ware. 1916. *Harmony Book for Beginners.* 1st ed.

Ottman, Robert W. 1961a. *Advanced Harmony: Theory and Practice.* 1st ed.

———. 1961b. *Elementary Harmony: Theory and Practice.* 1st ed.

Ouseley, Frederick Arthur Gore. 1868. *A Treatise on Harmony.* 1st ed.

Pacha, Alexander, and Jorge Calvo-Zaragoza. 2018. “Optical Music Recognition in Mensural Notation with Region-Based Convolutional Neural Networks.” In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 240–247. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492393. https://doi.org/10.5281/zenodo.1492393.

Papadopoulos, Hélène, and Geoffroy Peeters. 2009. “Local Key Estimation Based on Harmonic and Metric Structures.” In *Proceedings of the 12th Int. Conference on Digital Audio Effects (DAFx-09),* 408–415. Como, Italy, September. Accessed August 5, 2019. https://hal.archives-ouvertes.fr/hal-00511452.

———. 2008. “Simultaneous Estimation of Chord Progression and Downbeats from an Audio File,” 121–124. ISSN: 1520-6149, 2379-190X, Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing. March. https://doi.org/10.1109/ICASSP.2008.4517561.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* https://doi.org/10.48550/ARXIV.1912.01703. https://arxiv.org/abs/1912.01703.

Pauwels, Johan, and Jean-Pierre Martens. 2014. “Combining Musicological Knowledge About Chords and Keys in a Simultaneous Chord and Local Key Estimation System.” *Journal of New Music Research* 43, no. 3 (July 3, 2014): 318–330. ISSN: 0929-8215, accessed August 4, 2019. https://doi.org/10.1080/09298215.2014.917684. https://doi.org/10.1080/09298215.2014.917684.

Pauwels, Johan, Ken O’Hanlon, Emilia Gómez, and Mark Brian Sandler. 2019. “20 Years of Automatic Chord Recognition from Audio.” In *International Society for Music Information Retrieval (ISMIR).* Delft, The Netherlands, April 11, 2019. http://hdl.handle.net/10230/42773.

Peeters, Geoffroy. 2006. “Chroma-Based Estimation of Musical Key from Audio- Signal Analysis.” In *Proceedings of the 7th International Conference on Music Information Retrieval,* 115–120. Victoria, Canada: ISMIR, October. https://doi.org/10.5281/zenodo.1416420. https://doi.org/10.5281/zenodo.1416420.

Piston, Walter. 1941. *Harmony.* 1st ed.

Pons, Jordi, Rong Gong, and Xavier Serra. 2017. “Score-Informed Syllable Segmentation for a Cappella Singing Voice with Convolutional Neural Networks.” In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 383–389. Suzhou, China: ISMIR, October. https://doi.org/10.5281/zenodo.1415632. https://doi.org/10.5281/zenodo.1415632.

Prout, Ebenezer. 1889. *Harmony Its Theory and Practice.* 5th ed.

Pugin, Laurent, Rodolfo Zitellini, and Perry Roland. 2014. "Verovio: A Library for Engraving Mei Music Notation into Svg." In *Proceedings of the 15th International Society for Music Information Retrieval Conference,* 107–112. Taipei, Taiwan: ISMIR, October. https://doi.org/10.5281/zenodo.1417589. https://doi.org/10.5281/zenodo.1417589.

Purwins, Hendrik, Benjamin Blankertz, and Klaus Obermayer. 2000. "A New Method for Tracking Modulations in Tonal Music in Audio Data Format," 6:270–275. ISSN: 1098-7576, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. https://doi.org/10.1109/IJCNN.2000.859408.

Quick, Donya. 2016. "Learning Production Probabilities for Musical Grammars." *Journal of New Music Research* 45 (4): 295–313.

Rabiner, Lawrence R. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77 (2): 257–286. https://doi.org/10.1109/5.18626.

Raphael, Christopher, and Joshua Stoddard. 2004. "Functional Harmonic Analysis Using Probabilistic Models." *Computer Music Journal* 28 (3): 45–52. ISSN: 0148-9267, accessed August 7, 2019. https://doi.org/10.1162/0148926041790676. https://doi.org/10.1162/0148926041790676.

———. 2003. "Harmonic Analysis with Probabilistic Graphical Models." In *ISMIR*.

Reber, Henri. 1862. *Traité d'harmonie.* 1st ed.

Reger, Max. 1904. *Supplement to the Theory of Modulation.* Leipzig: C. F. Kahnt Nachfolger.

Renggli, Cédric, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlas, Wentao Wu, and Ce Zhang. 2021. "A Data Quality-Driven View of MLops." *CoRR* abs/2102.07750. arXiv: 2102.07750. https://arxiv.org/abs/2102.07750.

Richter, Ernst Friedrich. 1860. *Lehrbuch der Harmonie.* 3rd ed.

Riemann, Hugo. 1902. *Grosse Kompositionslehre.* 1st ed.

———. 1913. *Handbuch der Harmonie-und Modulationslehre.* 8th ed.

———. 1890. *Katechismus der Harmonielehre (Theoretisch und Praktisch).* 1st ed.

———. 1883. *Neue Schule der Melodik.* 1st ed.

———. 1887. *Systematische Modulationslehre.* 1st ed.

Rigaud, François, and Mathieu Radenen. 2016. "Singing Voice Melody Transcription Using Deep Neural Networks." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 737–743. New York, NY: ISMIR, August. https://doi.org/10.5281/zenodo.1418051. https://doi.org/10.5281/zenodo.1418051.

Rimski-Korsakov, Nikolay. 1886. *Practical Manual of Harmony.* A. Büttner, St. Petersburg.

Rimsky-Korsakov, Nikolay, Joseph Achron, and Nicholas Hopkins. 2005. *Practical Manual of Harmony.* OCLC: 60523181. New York, NY: C. Fischer. ISBN: 978-0-8258-5699-0.

Robinson, Franklin. 1918. *Aural Harmony.* 1st ed.

Rocher, Thomas, Matthias Robine, Pierre Hanna, and Laurent Oudre. 2010. "Concurrent Estimation of Chords and Keys from Audio." In *Proceedings of the 11th International Society for Music Information Retrieval Conference,* 141–146. Utrecht, The Netherlands: ISMIR, August. https://doi.org/10.5281/zenodo.1417485. https://doi.org/10.5281/zenodo.1417485.

Rohrmeier, Martin. 2007. "Modelling Dynamics of Key Induction in Harmony Progressions." In *Proceedings of the 4th Sound and Music Computing Conference,* 8.

———. 2011. "Towards a Generative Syntax of Tonal Harmony." Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17459737.2011.573676, *Journal of Mathematics and Music* 5 (1): 35–53. https://doi.org/10.1080/17459737.2011.573676. https://doi.org/10.1080/17459737.2011.573676.

Rohrmeier, Martin, and Ian Cross. 2008. "Statistical Properties of Tonal Harmony in Bach's Chorales." In *Proceedings of the 10th International Conference on Music Perception and Cognition,* 6:123–1319. Hokkaido University Sapporo, Japan.

Roig-Francoli, Miguel A. 2011. *Harmony in Context.* 1st ed.

Roland, Perry. 2002. "The Music Encoding Initiative (MEI)." In *Proceedings of the First International Conference on Musical Applications Using XML,* 55–59.

Rosenblatt, F. 1958. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* 65 (6): 386–408. ISSN: 1939-1471(Electronic),0033-295X(Print). https://doi.org/10.1037/h0042519.

Rossum, Guido van, et al. 2007. "Python Programming Language." In *USENIX annual technical conference,* 41:1–36. Issue: 1. Santa Clara, CA.

Ruder, Sebastian. 2017. "An Overview of Multi-Task Learning in Deep Neural Networks." *arXiv preprint arXiv:1706.05098.*

Rumelhart, David E., Geoffrey E. Hinton, Ronald J. Williams, et al. 1988. "Learning Representations by Back-Propagating Errors." *Cognitive modeling* 5 (3): 1.

Saint-Saëns, Camille. 1885. *Harmonie et Mélodie.* 1st ed.

Sansa Llovich, Jordi. 2013. "Quintas Y Octavas Prohibidas En El Periodo Modal-Tonal." PhD diss., Universitat Autònoma de Barcelona. https://ddd.uab.cat/record/113130.

Sapp, Craig Stuart. 2011. "Computational Methods for the Analysis of Musical Structure." PhD diss., Stanford University.

———. 2001. "Harmonic Visualizations of Tonal Music." In *Proceedings of the International Computer Music Conference,* 1:419–422. Havana, Cuba.

———. 2005. "Online Database of Scores in the Humdrum File Format." In *Proceedings of the 6th International Conference on Music Information Retrieval,* 664–665. London, United Kingdom: ISMIR, September. https://doi.org/10.5281/zenodo.1417281. https://doi.org/10.5281/zenodo.1417281.

———. 2009. "tsroot." Manual page. http://extras.humdrum.org/man/tsroot/.

Sarnecki, Mark. 2010. *Harmony.* 2nd ed.

Scheidt, Walter. 1975. *Naturkundliche Harmonielehre.* 1st ed.

Schenker, Heinrich. 1935. *Der Freie Satz.* 1st ed.

———. 1921. *Der Tonwille.*

———. 1906. *Neue Musikalische Theorien und Phantasien: Harmonielehre.* 1st ed.

———. 1922. *Neue Musikalische Theorien und Phantasien: Kontrapunkt, Zweiter Teil.* 1st ed.

Schoenberg, Arnold. 1967. *Fundamentals of Music Composition.* 1st ed.

———. 1922. *Harmonielehre.* 3rd ed.

———. 1969. *Structural Functions of Harmony.* 1st ed.

Scholes, Percy Alfred. 1924. *The Beginner's Guide to Harmony.* 2nd ed.

Schreiber, Hendrik, and Meinard Müller. 2018. "A Single-Step Approach to Musical Tempo Estimation Using a Convolutional Neural Network." In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 98–105. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492353. https://doi.org/10.5281/zenodo.1492353.

———. 2019. "Musical Tempo and Key Estimation Using Convolutional Neural Networks with Directional Filters." In *Proceedings of the 16th Sound and Music Computing Conference.* Málaga, Spain.

Schuster, Mike, and Kuldip Paliwal. 1997. “Bidirectional Recurrent Neural Networks.” *IEEE Transactions on Signal Processing* 45 (11): 2673–2681. https://doi.org/10.1109/78.650093.

Sears, David, Filip Korzeniowski, and Gerhard Widmer. 2018. “Evaluating Language Models of Tonal Harmony.” In *Proceedings of the 19th International Society for Music Information Retrieval Conference,* 211–217. Paris, France: ISMIR, September. https://doi.org/10.5281/zenodo.1492385. https://doi.org/10.5281/zenodo.1492385.

Sechter, Simon. 1853. *Die Grundsätze der Musikalischen Komposition.* 1st ed.

Shepard, Frank Hartson. 1896. *Harmony Simplified.* 5th ed.

———. 1889. *How to Modulate.* 2nd ed.

Shinn, Frederick. 1904. *A Method of Teaching Harmony.* 1st ed.

Siegmeister, Elie. 1965. *Harmony and Melody.* 1st ed.

Sigtia, Siddharth, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S. d’Avila Garcez, and Simon Dixon. 2014. “An RNN-Based Music Language Model for Improving Automatic Music Transcription.” In *Proceedings of the 15th International Society for Music Information Retrieval Conference,* 53–58. Taipei, Taiwan: ISMIR, October. https://doi.org/10.5281/zenodo.1416792. https://doi.org/10.5281/zenodo.1416792.

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

Smith, Dave, and Chet Wood. 1981. “The Universal Synthesizer Interface.” In *Audio Engineering Society Convention 70.* Audio Engineering Society.

Southall, Carl, Ryan Stables, and Jason Hockman. 2017. “Automatic Drum Transcription for Polyphonic Recordings Using Soft Attention Mechanisms and Convolutional Neural Networks.” In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 606–612. Suzhou, China.

———. 2016. “Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks.” In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 591–597. New York City, NY.

Southard, Lucien H. 1855. *Course of Harmony: Being a Manual of Instruction in the Principles of Thorough-Bass and Harmony.* 1st ed.

Spencer, Charles Child. 1854. *A Rudimentary and Practical Treatise on Music.* 4th ed.

Spencer, Stanhope Reid. 1915. *Harmony.* 1st ed.

Stoddard, Joshua, Christopher Raphael, and Paul E Utgoff. 2004. "Well-Tempered Spelling: A Key-Invariant Pitch Spelling Algorithm." In *Proceedings of the 5th International Society for Music Information Retrieval Conference,* 6. Barcelona, Spain.

Swain, Joseph Peter. 2002. *Harmonic Rhythm.* 1st ed.

Tchaikovsky, Pyotr Ilyich. 1872. *Guide to the Practical Study of Harmony.* Moscow: P. Jurgenson.

Temperley, David. 2002. "A Bayesian Approach to Key-Finding." In *Music and Artificial Intelligence,* edited by Christina Anagnostopoulou, Miguel Ferrand, and Alan Smaill, 195–206. Lecture Notes in Computer Science. Edinburgh, Scotland: Springer Berlin Heidelberg. ISBN: 978-3-540-45722-0.

———. 2009. "A Unified Probabilistic Model for Polyphonic Music Analysis." Publisher: Taylor & Francis, *Journal of New Music Research* 38 (1): 3–18.

———. 1997. "An Algorithm for Harmonic Analysis." *Music Perception* 15 (1): 31–68.

———. 2004. *The Cognition of Basic Musical Structures.* 1st ed. Cambridge, MA: MIT Press. ISBN: 978-0-262-70105-1 978-0-262-20134-6.

———. 2000. "The Line of Fifths." *Music Analysis* 19 (3): 289–319.

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

Temperley, David, and Elizabeth West Marvin. 2008. "Pitch-Class Distribution and the Identification of Key." *Music Perception: An Interdisciplinary Journal* 25 (3): 193–212.

Temperley, David, and Daniel Sleator. 1999. "Modeling Meter and Harmony: A Preference-Rule Approach." *Computer Music Journal* 23 (1): 10–27.

Teodoru, Gabi, and Christopher Raphael. 2007. "Pitch Spelling with Conditionally Independent Voices." In *Proceedings of the 8th International Conference on Music Information Retrieval,* 201–206. Vienna, Austria.

Tiersch, Otto. 1874. *Elementarbuch der Musikalischen Harmonie-und Modulationslehre.* 1st ed.

———. 1868. *System und Methode der Harmonielehre.* 1st ed.

Tischler, Hans. 1964. *Practical Harmony.* 1st ed.

Toutant, William. 1985. *Functional Harmony.* 1st ed.

Tracy, James. 1878. *Theory and Rudimental Harmony.* 1st ed.

Tunley, David. 1984. *Harmony in Action.* 1st ed.

Ulehla, Ludmila. 1966. *Contemporary Harmony.* 1st ed.

Ullrich, Karen, Jan Schlüter, and Thomas Grill. 2014. "Boundary Detection in Music Structure Analysis Using Convolutional Neural Networks." In *Proceedings of the 15th International Society for Music Information Retrieval Conference,* 417–422. Taipei, Taiwan.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems 30.* Long Beach, CA.

Vinée, Anselme. 1909. *Principes du système musical et de l'harmonie théorique et appliquée.* 1st ed.

Vivier, Albert-Joseph. 1890. *Traité complet d'harmonie, théorique, pratique, vocale & instrumentale.* 5th ed.

Vogl, Richard, Matthias Dorfer, and Peter Knees. 2016. "Recurrent Neural Networks for Drum Transcription." In *Proceedings of the 17th International Society for Music Information Retrieval Conference,* 730–736. New York, NY.

Vogl, Richard, Matthias Dorfer, Gerhard Widmer, and Peter Knees. 2017. "Drum Transcription Via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 150–157. Suzhou, China.

Vogler, Georg Joseph. 1778. *Gründe der Kuhrpfälzischen Tonschule.* 1st ed.

———. 1802. *Handbuch zur Harmonielehre und für den Generalbaß.* 1st ed.

Volckmar, Wilhelm Valentin. 1860. *Harmonielehre: Zunächst zum Gebrauch für Schullehrer-Seminarien.* 1st ed.

Vos, Piet G., and Erwin W. van Geenen. 1996. "A Parallel-Processing Key-Finding Model." *Music Perception: An Interdisciplinary Journal* 14 (2): 185–223.

Wang, Ziyu, Dingsu Wang, Yixiao Zhang, and Gus Xia. 2020. "Learning Interpretable Representation for Controllable Polyphonic Music Generation." In *Proceedings of the 21st International Society for Music Information Retrieval Conference,* 662–669. Montreal, Canada.

Wason, Robert Wesley. 1985. *Viennese Harmonic Theory from Albrechtsberger to Schenker and Schoenberg.* UMI Research Press.

Watt, Henry J. 1919. *The Foundations of Music.* 1st ed.

Weber, Gottfried. 1818. *Versuch einer geordneten Theorie der Tonsetzkunst.* 1st ed. Vol. 2.

Wedge, George Anson. 1930. *Applied Harmony, a Text-Book.* 1st ed.

———. 1924. *Keyboard Harmony: A Practical Application of Music Theory, Including the Study of Melody Harmonization, Broken Chords and Arpeggios, Modulation and Improvisation.* G. Schirmer, Incorporated.

Weiß, Christof, and Julian Habryka. 2014. "Chroma-Based Scale Matching for Audio Tonality Analysis." In *Proceedings of the 9th Conference on Interdisciplinary Musicology,* 168–173.

Wel, Eelco van der, and Karen Ullrich. 2017. "Optical Music Recognition with Convolutional Sequence-to-Sequence Models." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 731–737. Suzhou, China, October.

Werbos, Paul John. 1990. "Backpropagation Through Time: What It Does and How to Do It." *Proceedings of the IEEE* 78 (10): 1550–1560. https://doi.org/10.1109/5.58337.

———. 1988. "Generalization of Backpropagation with Application to a Recurrent Gas Market Model." *Neural Networks* 1 (4): 339–356. ISSN: 0893-6080.

Werker, Wilhelm. 1898. *Die Theorie der Tonalität.* 1st ed.

White, William Alfred. 1911. *Harmonic Part-Writing.* 1st ed.

Widrow, Bernard, and Marcian E. Hoff. 1960. *Adaptive Switching Circuits.* Stanford University.

Winograd, Terry. 1968. "Linguistics and the Computer Analysis of Tonal Harmony." *Journal of Music Theory* 12 (1): 2–49.

Wu, Shangda, Yue Yang, Zhaowen Wang, Xiaobing Li, and Maosong Sun. 2021. "Melody Harmonization with Controllable Harmonic Rhythm." *Computing Research Repository* abs/2112.11122. https://arxiv.org/abs/2112.11122.

Ycart, Adrien, and Emmanouil Benetos. 2017. "A Study on LSTM Networks for Polyphonic Music Sequence Modelling." In *Proceedings of the 18th International Society for Music Information Retrieval Conference,* 421–427. Suzhou, China, October.

York, Francis Lodowick. 1909. *A Practical Introduction to Composition: Harmony Simplified.* 4th ed.

Zaharia, Matei, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, et al. 2018. "Accelerating the Machine Learning Lifecycle with MLflow." *IEEE Data Engineering Bulletin* 41 (4): 39–45.