Université de Lille

UNIVERSITÉ DE LILLE
École Graduée MADIS-631

THÈSE

pour obtenir le grade de :
Docteur de l'Université de Lille

dans la spécialité Informatique

par

Alexandre D'HOOGE

Assisting Western Popular Music Guitar Practice and Tablature Composition with Machine Learning

Assister la Pratique de la Guitare et la Composition de Tablatures en Musiques Actuelles Occidentales par Apprentissage Automatique

---

Thèse soutenue le 9 octobre 2025 devant le jury composé de :

Gérard ASSAYAG	Directeur de Recherche, IRCAM	(Rapporteur)
Romain MICHON	Chargé de Recherche (HDR), INRIA Lyon	(Rapporteur)
Rémi BARDENET	Directeur de Recherche CNRS, CRIStAL	(Président du Jury)
Isabel BARBANCHO	Professeure des Universités, Universidad de Málaga	(Examinatrice)
Dorien HERREMANS	Professeure Associée, AMAAI, Singapore University of Technology and Design	(Examinatrice)
Louis BIGO	Professeur des Universités, Bordeaux INP, Université de Bordeaux, LaBRI	(Co-Directeur)
Mathieu GIRAUD	Directeur de Recherche CNRS, CRIStAL	(Co-Directeur)
Ken DÉGUERNEL	Chargé de Recherche CNRS, CRIStAL	(Co-Encadrant)

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Univ. Lille, UMR 9189 CRIStAL, F-59000 Lille, France
CNRS, UMR 9189, F-59000 Lille, France
Centrale Lille, F-59000 Lille, France

Made with the novathesis template (Lourenço 2021).

" Crazy, but that's how it goes
Millions of people living as foes"

— Bob Daisley, Ozzy Osbourne, Randy Rhoads,
Crazy Train (Blizzard of Oz)

À mes parents.

Acknowledgements

I was expecting my PhD to be a harsh and lonely journey. Harsh it was for sure, but it was far from lonely, and I am thankful to all the great persons that made my thesis all the richer.

First of all, I would like to thank my reviewers, Gérard Assayag and Romain Michon, for their time spent thoroughly reading this thesis, as well as their feedback and interest in my work. Thank you also to my examiners, Isabel Barbancho, Rémi Bardenet, and Dorien Herremans, for insightful and thought-provoking discussions.
Thank you to Jean-Louis Giavitto for agreeing to be in my first year's *CSI* and helping me refine the directions my research took.

I am especially grateful to my supervisors, for their support and trust in my work, and helping me improve and build trust in my own skills as a researcher.

Merci Ken pour ton regard critique (mais bienveillant !) sur mon travail qui m'a permis de consolider sereinement articles et présentations. Merci également pour toutes ces très riches discussions sur la musique, les jeux vidéo, la politique, ..., qui ont alimenté ma curiosité et m'ont aussi fait grandir humainement.

Merci Mathieu de m'avoir accompagné pour que je m'investisse dans de nouveaux projets, dans ma recherche comme dans la vie du labo. Tu as rendu mon expérience de thèse bien plus riche.

Merci Louis de m'avoir accordé ta confiance pour cette thèse, pour ta présence bienveillante et ton écoute de mes opinions, questionnements, difficultés et inquiétudes. Merci d'avoir su m'accompagner et m'encadrer comme tu l'as fait.

Thanks to colleagues, interns and collaborators from Algomus and SCRIME, for the great time spent together. Thanks Alex, Annick, Baptiste, Florence, Francesco, Joseph, Léo, Louis C, Manu, Olivier, Patrice, Quentin, Rui, Vanessa Nina, Viet-Toan, Yann, and Zakaria! It was a pleasure coming to work thanks to you all.

Merci aux collègues du labo et de la FST : Abbas, Jean-Christophe, Jean-Stéphane, Justine et Laurentiu. C'était un plaisir de travailler et/ou échanger avec vous.

Merci Gilles et Yohann pour votre confiance. Je suis très fier d'avoir pu participer à l'aventure du Guitar Social Club.

Merci à Arobas Music de m'avoir ouvert leurs portes, et en particulier à Nicolas pour sa transparence et disponibilité pour discuter de Guitar Pro.

Merci bien sûr aux ami·es du club tricot/crochet. Je suis particulièrement reconnaissant pour Caleb, Clémence, Enzo, Karim, Madeleine, Marie-Eva, Oliver et Sara et pour notre amitié malgré tout le temps que je passais loin de Lille.

While it was several years back, I am very grateful to Preeti Rao for welcoming me into her lab in IITB during Summer 2019. This internship could be considered the beginning of my career as a researcher in MIR! I also want to thank Krishna with whom I've worked there, for being so helpful and for his great pedagogy.

I was also very lucky to meet many awesome humans throughout my PhD, remotely or at conferences, and am very grateful for their kindness and friendship. Thanks to Pierluigi for getting in touch with us and for the nice collaboration we had together, it really boosted my confidence!

I want to thank Pedro S, who I looked up to throughout my thesis, for being such a kind human. I loved our discussions as fellow guitarists and metalheads and really appreciated how you managed to build a community around guitar tab research 🤘
Thanks also to Drew, Mathieu, Ruby and Xavier from QMUL for great times at ISMIR and SMC.

During my PhD, I spent 3 months at the Music Technology Group in Barcelona for a research visit, and I first want to thank Xavier Serra for welcoming me in the lab. It was an amazing time and I'm so grateful for all the close friends I made during my stay. For great times in and out of the lab, many thanks to Adithi, Alia, Andrea, Andrés, Anmol, Behzad, Benno, Błażej, Genís, Guillem, Hyon, Jorge, Laura, Luis, Michael, Pablo, Pedro, Penny, Raquel, Roser, Seva, Spencer, Tinke, Tom, Valentina, Xavier and many more!

Merci aux ami·es d'ATIAM pour les moments passés avant et pendant la thèse. Merci à la Brigade des Tubes et Audrey en particulier, merci Julia d'avoir été ma camarade de solfège, et merci Alexis pour notre amitié malgré la distance et les années.

Merci à Antoine, Gaël, Théo, Yan, pour notre groupe *de musique* qui aura accompagné une bonne partie de cette thèse. J'inclus bien sûr Camille dans ces remerciements au groupe, je chéris tous les moments passés avec vous, même si je ne suis jamais dispo.

Un immense merci aux ami·es de l'ENS (et affilié·es !), pour les rires, soirées et moments passés ensemble. Je veux particulièrement remercier ma famille de cœur, Spammens, pour la richesse de nos relations individuelles et de groupe, et pour tout le bonheur qu'iels m'ont apporté depuis qu'on se connait. Un merci supplémentaire à Elric, Leela et Gaspard pour nos séances communes de rédaction qui ont grandement participé au maintien de ma santé mentale en cette fin de thèse.

Pour finir, je veux remercier ma famille pour leur soutien tout au long de ma vie, depuis le début de ma pratique musicale et tout au long de mes études. Merci Éléonore, Frédéric, Papa et Maman, je ne serais pas où j'en suis aujourd'hui sans vous. Merci aussi à Anne et Jean-François de m'avoir permis de réaliser mon stage d'observation au LAC, je peux maintenant affirmer que ce moment m'a poussé à me diriger vers la recherche.

Enfin, merci Aura pour ta présence et ton soutien, dans mes études, mon travail, et ma vie.

Du fond du cœur,

Merci.

Abstract

Guitarists who play Western Popular Music (WPM) are usually little trained in music theory, a great part of their learning being autonomous and informal. To learn new songs or share them to other guitarists, they will commonly resort to audio recordings and tablatures. When it comes to composing new songs, it is usually conducted directly with the guitar through jamming and experimenting, alone or in a band. This thesis aims at exploring new computational methods that could assist guitarists both in the learning and compositional phases.

Learning assistance is motivated by the large quantity of guitar resources available online, and aims at helping guitarists navigate them to ease informal learning. A first algorithmic approach was proposed to make song recommendations to learners that are conditioned by difficulty ratings of the songs and the learner's level estimate. A second approach proposes musical features to automatically estimate the difficulty of bass and guitar tablatures on multiple criteria, while guaranteeing explainability and interpretability. Both approaches are based on new datasets that are or will be shared openly to the research community.

Normally, tablatures of composed songs are only written in a later stage to transcribe the songs, for memorisation and communication purposes. Other contributions of this thesis propose ways to assist with composition, the objective being to motivate guitarists to use tablatures during the composition process by augmenting tablature notation software with Machine Learning (ML) functionalities. A first model permits suggesting where to add *bends* in a tablature, in order to make it more idiomatic of WPM guitar playing. The ML model was integrated in an online tablature notation software to illustrate how it could fit guitarist-composers' workflow. Other contributions focus on guitar accompaniment parts. A model for suggesting a guitar chord position, given the previous one, is introduced and tested with a set of newly introduced quantitative metrics. This work was used as a first step that led to the development of a transformer and a rule-based model that generate possible continuations of an existing bar of rhythm guitar, in tablature format. Both models were evaluated quantitatively, but also qualitatively by external participants through an online survey. Finally, another transformer model is proposed for generating bass tablatures, given an existing rhythm guitar track, to assist guitarist-composers who are not bass players.

Keywords: Guitar Tablatures, Western Popular Music, Music Information Retrieval, Computer Music, Music Learning, Machine Learning

Résumé

Les guitaristes jouant des Musiques Actuelles Occidentales sont généralement peu formés en théorie musicale, une grande partie de leur apprentissage étant autonome et informel. Pour apprendre de nouveaux morceaux ou les partager avec d'autres guitaristes, ils ont souvent recours à des enregistrements audio et à des tablatures. Quant à la composition de nouveaux morceaux, celle-ci est généralement réalisée directement sur la guitare, en *jammant* et expérimentant, seul ou en groupe. Cette thèse vise à explorer de nouvelles méthodes algorithmiques susceptibles d'aider les guitaristes lors des phases d'apprentissage et de composition.

L'assistance à l'apprentissage est motivée par la quantité importante de contenu guitaristique disponible en ligne et vise à aider les guitaristes à naviguer ce contenu afin de faciliter l'apprentissage informel. Une première approche est proposée pour recommander des morceaux aux apprenants, en prenant en compte une estimation de leur niveau et de la difficulté des morceaux. Une deuxième approche propose des descripteurs musicaux permettant d'estimer automatiquement la difficulté de tablatures de guitare et de basse selon plusieurs critères, tout en garantissant l'explicabilité et l'interprétabilité des résultats. Les deux approches reposent sur de nouvelles bases de données, déjà partagées ou en cours de partage avec la communauté scientifique.

Habituellement, une tablature d'une composition musicale n'est écrite qu'après la phase de création musicale, afin de transcrire les morceaux à des fins de mémorisation et de communication. D'autres contributions de cette thèse s'intéressent à l'assistance à la composition, avec pour objectif d'inciter les guitaristes à composer à l'aide de tablatures, en enrichissant les logiciels de notation par des fonctionnalités d'apprentissage automatique. Un premier modèle permet de suggérer l'ajout de notes tirées (*bends*) dans une tablature, afin de la rendre plus idiomatique. Ce modèle a été déployé dans un logiciel de notation de tablatures en ligne afin d'illustrer son intégration possible dans le travail des guitaristes-compositeurs. D'autres contributions se concentrent sur les parties d'accompagnement à la guitare. Un modèle de suggestion de positions d'accords, basé sur la position précédente, est proposé et évalué à l'aide de nouvelles mesures qualitatives. Ce travail a constitué une première étape ayant conduit au développement d'un modèle *transformer* ainsi que d'un modèle déterministe, permettant de générer des continuations possibles d'une mesure de guitare rythmique, au format tablature. Ces deux modèles ont été évalués quantitativement, mais aussi qualitativement par des participants externes via une enquête en ligne. Enfin, un autre *transformer* est proposé pour générer des tablatures de basse à partir d'une piste de guitare rythmique existante, afin d'assister les guitaristes-compositeurs qui ne sont pas bassistes.

Mots-clés : Tablatures de Guitare, Musiques Actuelles Occidentales, Informatique Musicale, Musique Assistée par Ordinateur, Apprentissage Musical, Apprentissage Automatique

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence *(pp. 2, 3, 5, 6, 25–27, 58, 61, 62, 64, 181, 183, 185, 217–219)*
AMT Automatic Music Transcription *(pp. 46, 47, 50, 54, 55, 61, 184)*
ANN Artificial Neural Network *(p. 34)*

BCE Binary Cross-Entropy *(pp. 30, 131)*
BERT Bidirectional Encoder Representations from Transformers *(pp. 42, 61, 122)*
BPM Beats Per Minute *(pp. 70, 74)*

CPU Central Processing Unit *(pp. 131, 154, 214)*
CRediT Contributor Role Taxonomy *(pp. 7, 8)*

DL Deep Learning *(pp. 4, 5, 27, 28, 31, 34, 37, 49, 51, 61, 64, 98, 102, 152–155, 157, 160, 213, 215)*

FPO Floating-Point Operation *(pp. 213–215)*

GELU Gaussian Error Linear Unit *(pp. 37, 38)*
GNB Gaussian Naive Bayes *(pp. 34, 95, 97–100, 102)*
GPT Generative Pretrained Transformer *(pp. 42, 63, 143)*
GPU Graphical Processing Unit *(pp. 153, 154, 174, 183, 214, 216)*
GRU Gated Recurrent Unit *(pp. 39, 40)*
GSC Guitar Social Club *(pp. 69, 73, 75, 80, 82, 84, 217)*

LLM Large Language Model *(pp. 27, 42)*
LSTM Long Short-Term Memory *(pp. xi, 39, 40, 61)*

MEI Music Encoding Initiative *(p. 17)*
MIR Music Information Retrieval *(pp. 1, 4, 5, 12, 25, 30, 45–47, 49, 50, 62, 102, 213, 219)*
ML Machine Learning *(pp. 1, 3–6, 25, 27–35, 50, 80, 85, 95, 109, 110, 112, 181)*

MLP MultiLayer Perceptron *(pp. xi, 34–38, 41, 42, 105, 120, 123, 130, 141)*
MSB mySongBook *(pp. 47–49, 113, 114, 132)*
MSE Mean Squared Error *(pp. 30, 97)*
MTG Music Technology Group *(pp. 85, 86)*

NLP Natural Language Processing *(pp. 2, 40, 42, 61, 215)*
NN Neural Network *(pp. 4, 27, 34, 37, 215)*
NPM Notes Per Minute *(pp. 70, 71, 81)*

ReLU Rectified Linear Unit *(pp. 37, 38)*
RNN Recurrent Neural Network *(pp. 38–41, 49, 50, 122)*

TDP Thermal Design Power *(p. 214)*
TPD Tablature Performance Difficulty *(pp. 86, 87)*

WPM Western Popular Music *(pp. 1, 3–5, 15, 17–20, 22, 46–49, 51, 52, 55, 57, 60, 64, 69, 75, 92, 106, 107, 109, 113, 128, 130, 132, 142, 145–147, 157, 167, 176, 181, 184, 222)*

Foreword

Context of this Thesis

The research presented in this thesis is motivated by the goal of elaborating computational tools to assist guitar practice and composition. In the Western Popular Music (WPM) repertoire, guitarists commonly learn music autonomously, with a limited use of music theory training (L. Green 2002). In that context, tablature notation (Figure A) is favoured by guitarists for its visual relationship with the guitar fretboard (Figure B) that facilitates its interpretation. In a tablature, each horizontal line represents a *string*, and numbers denote the position on the fretboard, delimited by *frets*, that should be pressed to play the desired notes. Tablature notation is prescriptive and represent gestures, which facilitates its use by guitarists who do not know how to read standard notation nor where each note can be played on the fretboard. While tablature also includes rhythm notation (Figure A), guitarists can play without knowing how to read rhythm, as they will learn most songs by listening to their original recordings (L. Green 2002). Tablatures are complementary to recordings and allow guitarists to know precisely how a song is played. They are used for learning, memorisation and communication purposes. Because tablature is the main type of notation used by guitarists in WPM, and because it conveys more performance information than standard notation (Figure C), tablature is the main notation type used throughout this thesis. In this work, tablatures and guitar songs are studied through statistical analyses and Machine Learning (ML) methods for providing musicians with automatic creative suggestions and assist their practice.

This thesis belongs to the broad field of Music Information Retrieval (MIR) (Burgoyne et al. 2015; Downie 2003), an interdisciplinary field that makes use of computer science for the analysis of musical data and the automation of music-related tasks, like chord recognition or music style transfer for instance. Within the MIR field, this thesis focuses on symbolic music data (Le et al. 2024), i.e. music represented as a sequence of symbols like a score or a tablature, as opposed to audio data. Using symbolic data to study music has two main advantages. First, music scores are high-level representations that directly reflect the musical content of a piece, compared to audio files that needs to be

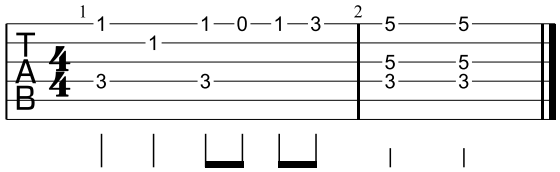


Figure A: Tablature notation.

Figure B: Picture of an electric guitar, by obBilder. The guitar is turned so that the strings are in the same order as the tablature above.

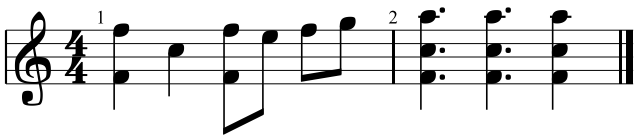


Figure C: Standard Notation.

processed to identify what notes are played and when. Symbolic music is also one of the main means of communication between musicians. By studying and possibly generating symbolic music, we benefit from a representation that is intelligible and usable by musicians directly. Second, studying symbolic music also has the advantage of enabling the use of techniques from the sub-fields of Natural Language Processing (NLP) that study text, thanks to parallels that can be drawn between symbolic music and written languages (Le et al. 2024). For instance, the revolutionary transformer network from NLP also has greatly improved symbolic music processing and generation (Y.-S. Huang et al. 2020; Z. Wang et al. 2021).

This thesis builds upon previous work conducted within the Algomus team. Cournut et al. (2020) introduced a parser for Guitar Pro files, the current prevalent digital format for tablatures. This parser is used throughout this thesis as it permits further studies like Cournut et al. (2021) that identified which guitar chord positions are used the most. It was also used in Régnier et al. (2021) that introduced an Artificial Intelligence (AI) method to identify whether a guitar part is accompaniment or not. This work will be particularly useful to this thesis as it allows to identify data subsets when developing new AI models that focus on melodies or accompaniment.

Objectives and Motivation

This thesis' contributions serve two main objectives: assisting guitar learning, and easing the composition process within tablature notation software.

Websites like Songsterr or Ultimate Guitar offer guitar players access to millions of songs' tablatures and chord progression aligned with lyrics. Helping learners navigate those resources calls for recommendation systems but, unlike traditional song recommendation based on personal taste or similar users (Born et al. 2021; Zeng et al. 2024), recommendations to guitar students should factor learning difficulty. Some systems like Barthet et al. (2011) are explicitly designed for learning, but do not take the difficulty of the songs or the level of the learner into account when making suggestions. An objective of this thesis is to propose a way to recommend guitar learners with new songs that are of an appropriate difficulty, given their current mastery of the instrument. Such a system requires songs' difficulty to be rated, which can be done through manual ratings or ML methods. However, while such ML methods exist for piano music, difficulty analysis of guitar songs was only conducted on chord progressions of accompaniment tracks (Vélez Vásquez et al. 2023). To address this limitation, another objective of this thesis is to propose an automatic model for estimating the difficulty of a guitar tablature, in an explainable and interpretable fashion.

The second main objective of this thesis is to research ways to make tablatures a part of the composition process for WPM guitarists, rather than only a means of transcription. Contributions related to that objective are at the heart of the ANR TABASCO project, within which this thesis played a significant role. Preliminary interviews conducted by Baptiste Bacot (musicologist involved in the project) with guitarist-composers showed that tablature notation software can be used for composition but that additional algorithmic features could make the process richer and easier for a number of guitarists. Some of the features of interest expressed by the interviewees relate to ideation and analysis. For instance, some composers are interested in automatic tools that would identify scales they can experiment with, or models that suggest drum and bass tracks (Bacot et al. 2024). Observation of how the notation software is used by composers also highlighted some behaviours that could be automated by the notation software, like the heavy use of the copy-pasting feature when writing accompaniment parts. This thesis presents AI models that could assist composers when writing tablatures, some focusing on lead guitar (melodic tracks) and others on rhythm guitar (accompaniment tracks). As will be discussed further in chapter 3, while guitar tablatures have already been studied in research, most AI methods focus on the generation of tablatures that might be performed directly by guitarists. Conversely, this thesis explicitly aims at assisting guitarists in specific tasks of the composition process, using tablatures as a common means of communication.

Outline of this Thesis

This thesis is organised in four main parts: an introductory one (part I), two for the contributions on guitar learning (part II) and tablature composition (part III), and a concluding one (part IV).

The introduction (part I) begins with a presentation of the musical background required for a full understanding of this thesis. Chapter 1 presents tablature notation from a historical perspective and how tablatures can now be encoded digitally. The rest of the chapter discusses how WPM guitar players practise their instrument and what role tablature plays, especially during the composition phase.

Chapter 2 provides explanations of the ML methods used throughout this thesis like decision trees or Naive Bayes models. Special attention is given to Neural Networks (NNs) and Deep Learning (DL) methods, as multiple architectures are used in the contributions, from simple perceptrons to large transformers.

Finally, chapter 3 presents existing work related to this thesis' contributions. A significant contribution of this chapter is a review of which data has been used in the literature on guitar and tablatures, and which WPM guitar datasets are now publicly available for research. We also present guitar-related MIR tasks, with a focus on symbolic methods that use tablatures.

The first part of the contributions (part II) focuses on methods to assist guitar learners in their practice. Chapter 4, presents a song recommendation system that takes the learners' level into account. This work is part of an industrial collaboration and benefits from a proprietary dataset of WPM guitar accompaniment songs rated across several difficulty criteria. Previous research already studied ways to assist guitar players in their learning (Ariga et al. 2017b; B. Wang et al. 2021), but none explicitly measures the level of the guitarists to make informed suggestions. This chapter proposes a way to model the level of a guitar player on multiple musical dimensions, and use that level estimate to recommend songs of an appropriate difficulty to keep the learners engaged. While the full corpus is proprietary, a subset of the data is released, as well as all the code of this recommendation system.

Chapter 5, presents a new approach to automatically analyse the difficulty of tablatures. While existing work studies the playability of tablatures (Ariga et al. 2017a), the presented criteria only include playing speed and finger stretching of chords, which can definitely correlate with difficulty but also miss other musical dimensions. Vélez Vásquez et al. (2023) include other difficulty criteria defined from discussions with guitar teachers but only study chord progressions of WPM guitar accompaniment tracks. This chapter aims at bridging the gap in the current literature by defining and testing various musical features for automatically analysing the difficulty of tablatures. This work is based on a newly-gathered dataset of bass and guitar tablatures given a global rating of difficulty by amateur WPM guitar players.

The second part of the contributions (part III) presents new AI models destined to assist guitarist-composers in writing tablatures, models that could ultimately be included in tablature notation software. Chapter 6 presents a new task and a corresponding model for rendering tablatures more idiomatic by suggesting where to add playing techniques. Playing techniques are an important part of guitar playing in WPM and include various articulations like *glissandi* (slides) or *legato* (hammer-on/pull-off), for instance. Techniques can be notated precisely in tablatures and can thus be studied automatically from tablature datasets. This chapter focuses on *bends*, a characteristic technique of WPM guitar that allows to change the pitch of a note continuously, never studied in MIR previously. We propose a detailed taxonomy to represent the variety of bend types, study their usage in a large corpus of WPM guitar tablatures, and design a model that automatically suggests where bends could be added in a tablature. This task of playing techniques suggestion is thought to help guitar players who might base their work on tablatures automatically arranged from other instruments (Edwards et al. 2024), as such tablatures might lack expressiveness and idiomaticity of WPM guitar playing. Suggesting playing techniques can also be a way for composers to vary the melodies they compose and explore playing styles they might not practise usually.

Chapter 7 focuses on rhythm guitar (accompaniment guitar tracks) and studies guitar chord positions. Since strings on a guitar are tuned in perfect fourth intervals (except for a major third between the G and B strings), the same note can be played in multiple locations on the fretboard, and chords can likewise be played in multiple positions. Guitar chords positions have already been studied in the literature, to analyse their statistical usage (Cournut et al. 2021) or provide users with all possible positions for a chord (Wortman et al. 2021). However, previous work on guitar consider chords as individual units and does not take the context they are used in into account. This chapter introduces a new approach to guitar chord position suggestion by taking the previous chord position into account to guarantee some continuity of musical characteristics. The objective of this chord position suggestion task is to help guitar players navigate the range of positions they can use and motivate the use of new and varied positions. Several musical dimensions are defined to capture a range of timbral and playability characteristics from symbolic data and used to measure the validity of the ML approach presented.

Chapter 8 extends rhythm guitar writing assistance by studying the generation of tablature excerpts that could be continuations of an existing measure. This study adapts existing work on tablature generation (Y.-H. Chen et al. 2020; McVicar et al. 2015; Sarmento et al. 2021) by focusing on rhythm guitar and continuations of a prompt rather than free generation. To make the approach modular and enhance control possibilities, we propose generating "picking patterns" that encode information of the picking hand. Those patterns can then be combined with chord positions to obtain guitar tablatures. To evaluate the performance of the models designed, we conduct a subjective evaluation through an online questionnaire and gather feedback that supports the idea that an approach that imitates copy-pasting might be favoured from more creative DL models.

Finally, Chapter 9 proposes to tackle a task expressed by several composers during interviews (Bacot et al. 2024): generating accompanying bass tablatures. While generating bass tablature was also studied in the literature (Sarmento et al. 2023a), no research proposed to condition that generation on existing tracks, except in the audio domain (Grachten et al. 2020; Nistal et al. 2024). This chapter presents a transformer model that successfully generates bass tablatures that follows an existing rhythm guitar track, and a qualitative analysis of the results is also proposed.

To conclude, Part IV ends this thesis with the perspectives envisioned, ranging from improvements on the contributions presented to new multimodal research. In the appendices, we present additional reflections upon the AI methods presented in the contributions and the impact they might have on musicians, the legality of using copyrighted data for training AI models, and the energy cost of conducting a thesis that uses ML methods. These discussions are presented as appendices because they can be read independently, but they are nonetheless an integral part of this thesis.

Publications

You will find below a list of publications resulting from this thesis work. Note that Guilluy et al. (2025) is not presented in this thesis, because the topic studied is too far from the other contributions. Because I have not been equally involved in all publications, I detail my contributions with the Contributor Role Taxonomy (CRediT).[1] I also presented my work to fellow researchers at DMRN in 2023,[2] at the ICCARE day on *Partition et Numérique* in 2025[3] and the 23rd IASPM International Conference in 2025.[4] All publications are available on HAL.[5]

(D'Hooge et al. 2023a) Alexandre D'Hooge, Louis Bigo, Ken Déguernel. "Modeling Bends in Popular Music Guitar Tablatures", *Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR)*, 2023.
— CRediT: Conceptualisation, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualisation, Writing.

(D'Hooge et al. 2024a) Alexandre D'Hooge, Louis Bigo, Ken Déguernel, Nicolas Martin. "Guitar Chord Diagram Suggestion for Western Popular Music", *Proceedings of the 21st Sound and Music Computing Conference (SMC)*, 2024.
— CRediT: Conceptualisation, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualisation, Writing.

(Bontempi et al. 2024) Pierluigi Bontempi, Daniele Manerba, Alexandre D'Hooge, Sergio Canazza. "From MIDI to Rich Tablatures: an Automatic Generative System incorporating Lead Guitarists' Fingering and Stylistic choices", *Proceedings of the 21st Sound and Music Computing Conference (SMC)*, 2024.
— CRediT: Data curation, Formal analysis, Software, Visualisation, Writing.

(Ramoneda et al. 2024b) Pedro Ramoneda, Vsevolod Eremenko, Alexandre D'Hooge, Emilia Parada-Cabaleiro, Xavier Serra. "Towards Explainable and Interpretable Musical Difficulty Extimation: a Parameter-efficient Approach", *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.
— CRediT: Formal analysis, Software, Validation, Visualisation, Writing.

[1] https://credit.niso.org/, accessed in June 2025.
[2] https://www.qmul.ac.uk/dmrn/dmrn18/, accessed in June 2025.
[3] https://cmbv.fr/fr/evenements/iccare-partition-et-numerique, accessed in June 2025.
[4] https://iaspm-paris2025.sciencesconf.org/?forward-action=index&forward-controller=index&lang=en, accessed in June 2025.
[5] https://hal.science/search/index/?q=*&authIdPerson_i=1265306, accessed in June 2025

(D'Hooge et al. 2024b) Alexandre D'Hooge, Mathieu Giraud, Yohann Abbou, Gilles Guillemain. "Suggestions Pédagogiques Personnalisées pour la Guitare", *Actes des Journées d'Informatique Musicale (JIM)*, 2024.
— CRediT: Conceptualisation, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualisation, Writing.

(Guilluy et al. 2025) Quentin Guilluy, Anis Farji, Joao Fernandes, Mathieu Giraud, Alexandre D'Hooge, Emmanuel Leguy. "Vers une taxonomie et une analyse des gestes guitaristiques dans le Brutal Death Metal", *Actes des Journées d'Informatique Musicale (JIM)*, 2025.
— CRediT: Methodology, Supervision, Validation, Writing (Review and Editing).

(Hassein-Bey et al. 2025) Zakaria Hassein-Bey, Yohann Abbou, Alexandre D'Hooge, Mathieu Giraud, Gilles Guillemain, Aurélien Jeanneau. "What song now? Personalized Rhythm Guitar Learning in Western Popular Music", *Proceedings of the 26th International Society for Music Information Retrieval Conference (ISMIR)*, 2025.
— CRediT: Conceptualisation, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing.

(Anoufa et al. 2025) Olivier Anoufa, Alexandre D'Hooge, Ken Déguernel. "Conditional Generation of Bass Guitar Tablature for Guitar Accompaniment in Western Popular Music", *Proceedings of the AI Music Creativity Conference (AIMC)*, 2025.
— CRediT: Conceptualisation, Formal analysis, Methodology, Software, Supervision, Validation, Visualisation, Writing.

Part I

Introduction

1

Musical Background

"Most musicologists are scared of tablature."

Griffiths (2021)

Contents

Tablature notation can be looked down upon in some musical contexts (Balman 2020). It is however the main means of communication for guitarists and bassists who play Western Popular Music and, consequently, the core music notation type used in this thesis. This chapter aims at introducing the musical concepts required to understand the contributions that are presented later on. Overall, this thesis does not involve complex music theory concepts, even though basic knowledge of chords and rhythm can be necessary. However, guitar practice in Western Popular Music and the usage of tablatures are discussed extensively.

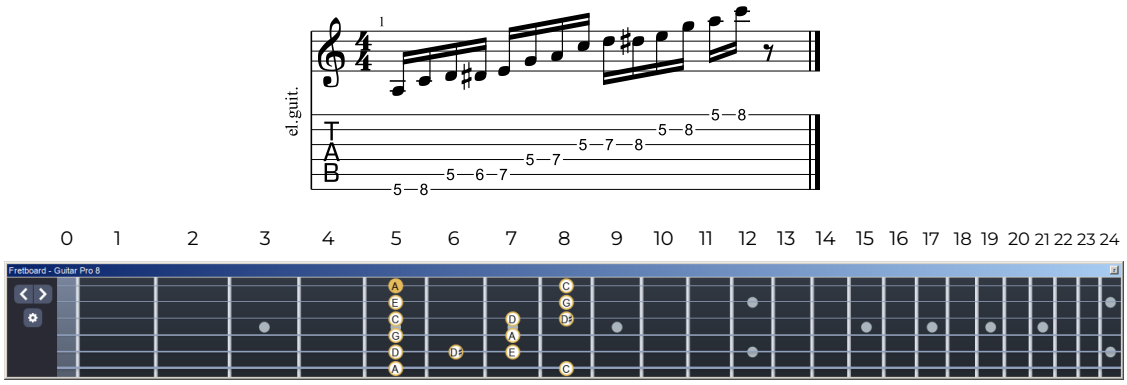


Figure 1.1: Modern tablature notation and the corresponding fretboard view (in Guitar Pro). The horizontal lines of the tablature are the strings of the guitar, the number denote the frets (horizontal dimension on the fretboard shown).

1.1 Guitar Tablatures

Griffiths, in his talk at the Annual Conference of the Musicological Society of Australia in 2021 (Griffiths 2021),[1] makes a case for tablature notation that he deems not considered enough in musicological research. When claiming that musicologists are scared of tablature, he defends that the lack of research on tablatures might be due to their absence in most music curriculums and a slow but steady historical bias that drove tablature out of the scope of interest. This observation could also be made regarding MIR research: for the 2000-2020 period, the ISMIR paper explorer only returns 6 papers mentioning tablature in their title, among which 3 study guitar tablatures (instead of lute).[2] The situation slightly changed in the recent years as will be discussed later, for now, this section provides a broad introduction on tablatures, starting with a brief historical overview before diving deeper into how tablatures are currently used in popular music.

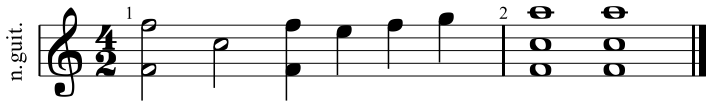
1.1.1 From Lute Music to Digital Tablature Notation Software

The tablature is a form of notation that arose in Western music at least in the 14th century (Dart et al. 2001). It was used for keyboard and lute music alike (among other instruments) and has several differences from staff notation. In keyboard tablatures, each note is written using a unique letter, sometimes in addition to a staff (Apel 1942). Tablatures for stringed instruments usually feature horizontal lines that represent the strings/courses of the instrument and individual notes are denoted using letters[3] (French tablatures) or numbers (Italian or Spanish tablatures). In both cases, the character indicates on which fret a finger should be pressed to obtain the desired pitch, going incrementally from the open string (0 or a) to higher frets (Figure 1.1).

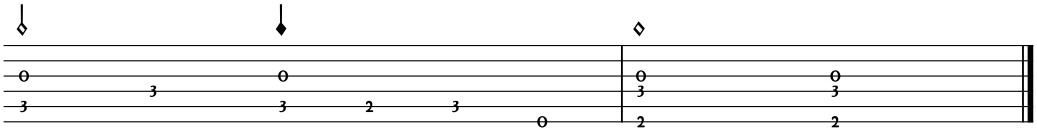
[1] My thanks to Quentin Guilluy for introducing me to Griffiths' work.

[2] https://ismir-explorer.ai.ovgu.de/app/?#tab, accessed in June 2025.

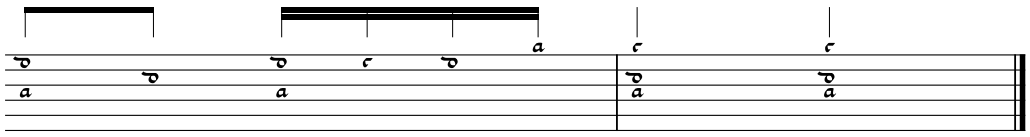
[3] Those letters are unrelated to English note names and are simply used to count frets, with *a* representing the 0th fret (open string), *b* the 1st fret, etc.



(a) Standard/Staff Notation



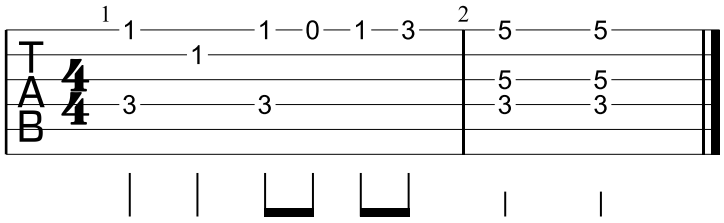
(b) Italian Lute Tablature



(c) French Lute Tablature

```
e|-1-----1--0--1--3--|-5--5--|
B|----1--------------|-------|
G|-------------------|-5--5--|
D|-3-----3-----------|-3--3--|
A|-------------------|-------|
E|-------------------|-------|
```

(d) ASCII Tablature



(e) Modern Tablature Notation

Figure 1.2: Different types of music notation of the same musical excerpt (with minor adaptations of the rhythm notation). The first three are reproduced from Dalitz et al. (2013).

Most tablatures would also include some kind of rhythm notation above the *staff*, usually using symbols similar to their mensural notation counterparts (Figure 1.2).

Why were tablatures used? For the keyboard, they were a way to indicate the bass notes efficiently, below the melody written in standard notation (Apel 1942, p. 23). Polyphonic music could also be entirely written with tablatures, as it allowed for a compact and easily printable representation (Apel 1942, p. 32). For the lute, the tablature is more detailed and serve a different purpose. As put in Apel 1942, p. 55:

> “
>
> If we conceive notation as a link connecting the writer of a composition with its performer, i.e., as an expedient showing the player or singer the tones which the composer wants him to produce, then we must realize that, generally speaking, there is a direct and an indirect way to achieve this goal. In a notation representing the latter method, the player is referred to his instrument through the medium of numerous elements of a distinctly intellectual character, such as pitch, intervals, tonality, accidentals, scales and many other such points. In a notation representing the direct method, however, his fingers are referred immediately to the technical devices of his instrument, the keys, frets, strings, holes, etc. In German terminology, these two species are distinguished as *Tonschrift* and *Griffschrift*, terms which may be conveniently translated ‘pitch notation’ and ‘finger notation.’
>
> ”

Tablature undoubtedly belong to the category of finger notation and, because it enables a direct mapping between the transcription and the instrument, it is accessible to players with little musical theory knowledge. This notation that favours performance might explain the popularity of the lute (Apel 1942; Griffiths 2021, pp. 55, 4), and maybe even the current popularity of guitar where tablature notation is appreciated for similar reasons (L. Green 2002).

Back to the lute, tablature use decreased rapidly in the 18th century when staff notation became more widespread and tablatures started being seen as less “serious” (Griffiths 2021), while instruments that used tablatures became less played outside of popular music or switched to staff notation. Tablature notation could have died out, if it was not for its renewed popularity in the 20th century for transcribing electric bass and guitar parts in Western popular music (Navarret 2013, 40, Figure 1.2e). Tablatures – with some additional information – even went back to being considered a “serious” notation system, with the inclusion of electric guitars in Contemporary Classical Music (Laliberté 2010).

This thesis focuses on *modern* guitar tablatures, like the ones presented Figure 1.1 and Figure 1.2e, especially in its digital formats. Like lute tablatures, modern guitar tablatures use horizontal lines to represent the strings, and fret positions are written with numbers.

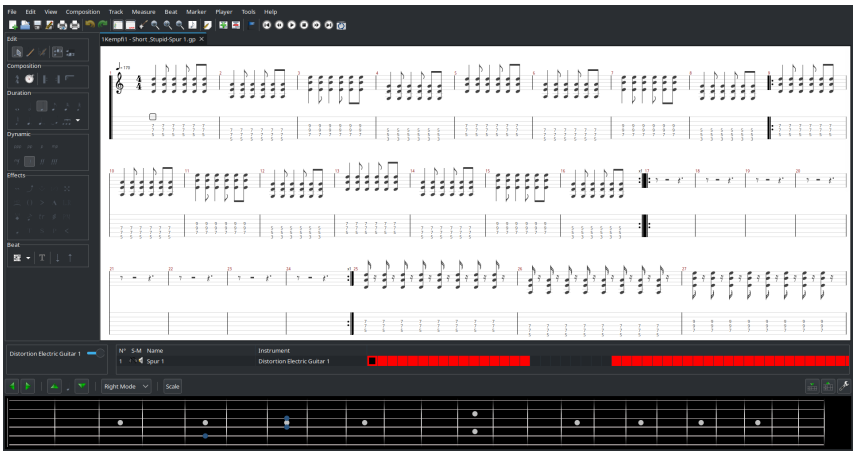
They are also used for detailed gesture annotation, like expressive playing techniques that are a strong aspect of guitar playing in popular music. Tablature songbooks are fairly popular since the late 20th century, and tablatures can also be found in specialised magazines (Navarret 2013, pp. 40–41). While these commercial options exist, open access to free tablatures developed with the Internet. Pictures of tablatures could be shared, but they were soon replaced with dedicated digital formats and software, as presented hereafter.

1.1.2 Tablature Digital Formats

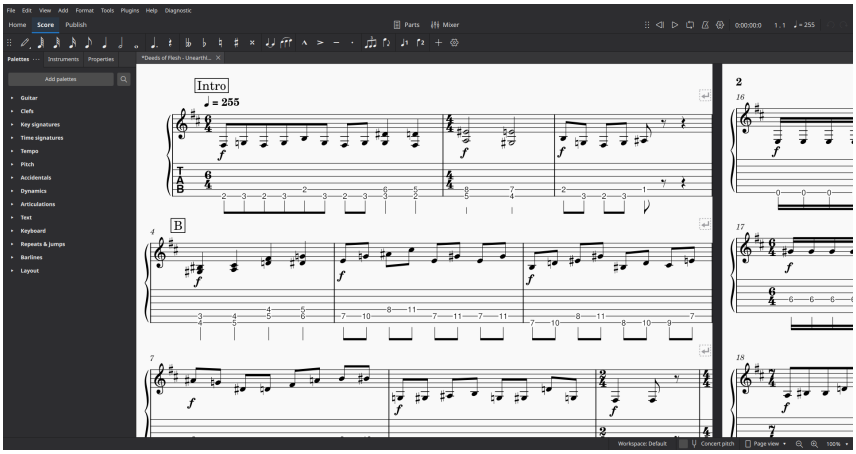
One type of tablature that is particularly suitable for online sharing is *ASCII tablature*, i.e. tablatures that only use simple digital characters and symbols (Figure 1.2d). This constrains the quantity of information that can be represented, for instance rhythm details are always omitted even if they can be implied by the number of hyphens used between notes. Those limitations do not impact the popularity of the format, with websites like www.ultimate-guitar.com and 911tabs.com gathering *millions* of ASCII tablatures (Barthet et al. 2011). The main advantage of using ASCII characters to write tablatures is that they can be written with any writing software. Nonetheless, digital tools dedicated to writing tablatures soon came about, to attain cleaner scores comparable to the professionnaly engraved ones found in songbooks. *MuseScore*[4] (Figure 1.3b), an open-source notation software, added tablature support in its version `2.0.0`, in March 2015. *TuxGuitar*[5] (Figure 1.3a) is another open-source software that is focused almost exclusively on tablature engraving. The two previous tools are often presented as free and open-source alternatives to another software which can be considered as the leader of digital tablature engraving, at least for WPM: *Guitar Pro* (Figure 1.3c). Guitar Pro is a proprietary software developed by the company *Arobas Music*,[6] funded in 1997 and based in Lille, France. While it is hard to be certain that it is the most used software for tablature edition, it was downloaded over 15 million times since its creation (Arobas-Music 2025) and the `.gp` file format is the standard format on the biggest tablature sharing websites that are *Ultimate Guitar* and *Songsterr*.

We discussed previously how the use of tablatures quickly evolved to adapt to the Internet era. This evolution was also witnessed for other types of sheet music, with huge classical music editors/publishers like Hal Leonard or Theodore Presser now also offering most of their catalogue in digital format (Kim 2025; Narici 2025). The generalisation of digital formats called for standardised encodings that would allow sharing, parsing and using digital music more easily. For tablatures, ASCII tabs might be considered as the first tab encoding. However, while it partly standardised how to write and share tablatures, it is hard to parse automatically (though not impossible, Barthet et al. 2011).

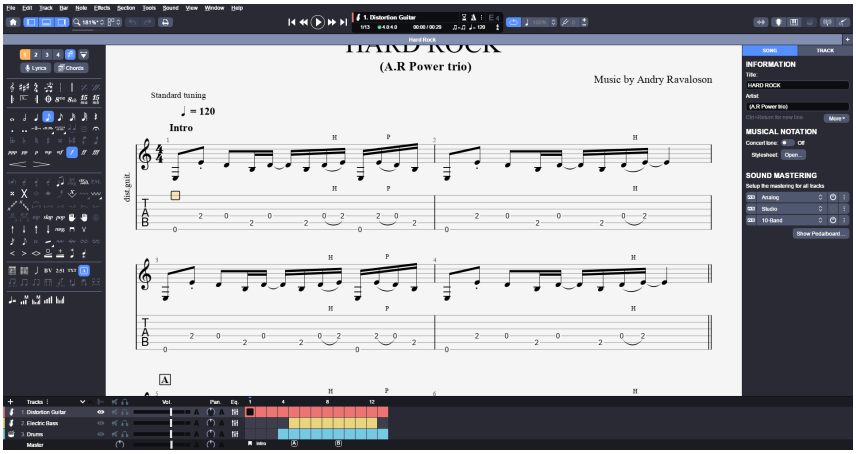
[4] https://musescore.org/en, accessed in June 2025.
[5] https://github.com/helge17/tuxguitar/, accessed in June 2025.
[6] https://www.guitar-pro.com/fr/c/11-arobas-music, accessed in June 2025.



(a) TuxGuitar



(b) Musescore



(c) Guitar Pro

Figure 1.3: User interfaces of three music notation programs with tablature support.

It also has the disadvantage to bind musical content and visualisation together, i.e. it is not possible to alter the design of the tablature independently of the musical content written. More extensive standards came about in the early 2000s, like *MusicXML*, first released in 2005 and managed by a private company (Recordare LLC) until 2015 when it was transferred authority to the W3C. MusicXML included tablatures from the start and now supports them in an extensive way.[7] Another well-known standard is the Music Encoding Initiative (MEI), which released its first version in 2010. It included basic guitar notation features, and improved with time to manage more complex and detailed tablatures.[8] However, like MusicXML, the adoption of the MEI standard is rather slow when it comes to tablature. It is yet to be supported by Guitar Pro, for instance. Since the Arobas Music company started Guitar Pro in 1997, they came up with their own notation standard for tablatures. The Guitar Pro format evolved through the years and software versions (`.gp3`, `.gp4`, `.gp5`, `.gpx`, `.gp`), with a particularly big difference after the 5th version, that limits its support in other notation software. The latest format is still human and machine-readable and has a detailed support of most possible guitar ornaments and techniques. Those formats are illustrated along with a simple tablature example Figure 1.4.

In this thesis, to benefit from the exhaustiveness of the format and to limit conversion issues, we use Guitar Pro files that are later parsed in Python either with the `music21` library (Cuthbert et al. 2010) – thanks to the work of Cournut et al. (2020) – or the `pyguitarpro` library.[9]

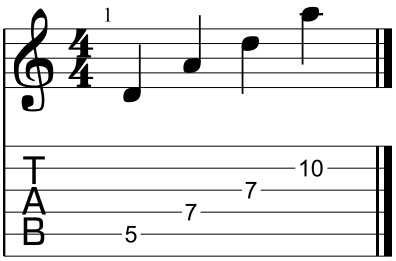
1.2 Modern Guitar Practice

In this thesis, we focus on *Western Popular Music* (abbreviated as WPM from now on). It is an abstract concept, so we will define it through a brief summary of Middleton et al. (2001). Note that, because of this summarising, not all criteria presented hereafter will apply to all kinds of popular music. We also insist on using the *Western* adjective for it, acknowledging the fact that this music derives from Western music tradition and that popular music can be different depending on the tradition it stems from.
Popular music has wide appeal and it developed strongly through the advent of new technologies in music. WPM can be viewed as a genre on its own, since it is assumed that most popular songs would have common characteristics like their audience, their duration, etc. However, WPM can still be divided into multiple genres and subgenres like “background” or “dance” music. Those genre categories are usually not set in stone and depend on context, a more objective common ground for WPM is *form*. For example, most lyrics-based popular music would contain verses separated by a repeating chorus.

[7] https://www.w3.org/2021/06/musicxml40/tutorial/tablature/, accessed in June 2025.
[8] https://music-encoding.org/guidelines/v5/content/tablature.html, accessed in June 2025.
[9] https://pyguitarpro.readthedocs.io/en/stable/, accessed in June 2025.

```
<Notes>
<Note id="0">
<InstrumentArticulation>0</InstrumentArticulation>
<Properties>
<Property name="ConcertPitch">
<Pitch><Step>D</Step><Accidental></Accidental><Octave>4</Octave></Pitch></Property><Property name="Fret">
<Fret>5</Fret>
</Property>
<Property name="Midi">
<Number>50</Number>
</Property>
<Property name="String">
<String>1</String>
</Property>
<Property name="TransposedPitch">
<Pitch><Step>D</Step><Accidental></Accidental><Octave>5</Octave></Pitch></Property></Properties>
</Note>
```

a) .gp



b) MusicXML

```
<note>
 <pitch>
  <step>D</step>
  <octave>3</octave>
 </pitch>
 <duration>1</duration>
 <voice>5</voice>
 <type>quarter</type>
 <stem>up</stem>
 <staff>2</staff>
 <notations>
  <technical>
   <string>5</string>
   <fret>5</fret>
  </technical>
 </notations>
</note>
<note>
 <pitch>
  <step>A</step>
  <octave>3</octave>
 </pitch>
 <duration>1</duration>
 <voice>5</voice>
 <type>quarter</type>
 <stem>up</stem>
 <staff>2</staff>
 <notations>
  <technical>
   <string>4</string>
   <fret>7</fret>
  </technical>
 </notations>
</note>
```

c) MEI

```
<staff n="2">
   <layer n="5">
      <tabGrp dur.ppq="1" dur="4">
         <tabDurSym />
         <note tab.fret="5" tab.course="5" />
      </tabGrp>
      <tabGrp dur.ppq="1" dur="4">
         <tabDurSym />
         <note tab.fret="7" tab.course="4" />
      </tabGrp>
      <tabGrp dur.ppq="1" dur="4">
         <tabDurSym />
         <note tab.fret="7" tab.course="3" />
      </tabGrp>
      <tabGrp dur.ppq="1" dur="4">
         <tabDurSym />
         <note tab.fret="10" tab.course="2" />
      </tabGrp>
   </layer>
</staff>
```

Figure 1.4: Excerpts of internal file representations for the same tablature excerpt. Note that only the first note is represented for the `.gp` file format (a), and only the first two notes for MusicXML (b).

Likewise, instruments would often play repetitive structures over any multiple of 4 bars (Bimbot et al. 2016) on common harmonic sequences. Finally, WPM can be defined by the variety of styles it encompasses, most if not all of them deriving from blues and early jazz music. To cite a few: Rock, Funk, Metal, Hip-Hop, Electronic Dance Music…In this thesis, all styles are not represented equally since we focus on guitar tablatures, so songs without a guitar are not studied. Besides, tablature notation is not used consistently across WPM sub-styles. For instance, rock and metal music are overrepresented because they are mostly notated and shared through tablature notation, unlike jazz music that uses it to a much lesser extent compared to chord progressions and standard notation for melodies. Likewise, styles of WPM that do not often feature guitars in their songs are not studied as much in this thesis, like most kinds of rap or electronic music.

Nonetheless, WPM is more than a musical category, as it also implies ways of listening, learning, practising and creating music (L. Green 2002). We discuss its characteristics more precisely when it comes to guitar music in the following sections.

1.2.1 On Tablature Usage

Tablature notation focuses on *gesture* (Apel 1942). The first gesture that it encompasses is simply to specify which string-fret pair is required to play a desired note. This disambiguation is needed because the same note can be played at several positions on the guitar. For instance, with a guitar in standard tuning – (E_2, A_2, D_3, G_3, B_3, E_4), also called E-standard tuning – the A_3 can be played on: i) the 5th string, 12th fret; ii) the 4th string, 7th fret; iii) the 3rd string, 2nd fret .[10] On electric guitar, because it is easy to play over the 12th fret, we could even include iv) 6th string, 17th fret.

However, as G. Merchi beautifully put it:

> “
>
> If one objects that [the tablature] is required to denote positions, I would answer that the violin and the cello etc. do not have tablatures and that the guitar needs it even less because it has frets.[11]
>
> ”

We will not answer this claim on the historical grounds of lute tablature since it was already done in the literature (Dart et al. 2001; Griffiths 2021), we will however try to answer as if that comment was made today regarding modern-days guitar tablatures. A partial answer was already given by showing that a single note can be played on multiple positions of a guitar's fretboard. In that case, standard notation would not provide enough information to play the song exactly as intended. This is especially problematic in rock and metal sub-styles where authenticity is a core value (McKinna 2014; Somdahl-Sands et al. 2015).

The lack of gesture information in standard notation has two main implications: firstly, reading expected positions from a tablature should guarantee a certain level of playability, whereas deriving positions from staff notation might lead beginner guitarists to play a song in a sub-optimal way; secondly, on what string and fret a note is played on guitar has a noticeable impact on timbre. The most striking example is how an open string (playing a string without pressing any fret) has very different sound qualities – longer resonance and more high harmonic content – from the same note played on a thicker string. Both typical use-cases mentioned here are mostly aimed at beginner guitar players, it is true that most guitar players should eventually learn to derive the *correct* way to play a song from staff notation, in particular if they are helped by a teacher. But, as it so happens, the practice of guitar in WPM is precisely based on autonomous learning (Navarret 2013, pp. 55–56), even though popular music is increasingly being considered in music schools'

[10] In guitar lingo, the 1st string is the thinnest and highest sounding, i.e. the E_4, and the 6th is the thickest/lowest in pitch: the E_2. It might also be worth reminding that each fret allows to play one semitone higher than the previous one.

[11] Quote taken from Merchi (1700–1799), translated from French. More context and the original French text can be found in Bosseur (2005) or Navarret (2013, p. 40).

curriculums (L. Green 2002, p. 127). More precisely, learning guitar for popular musicians is heavily based on *aurality*, i.e. listening and then copying songs (L. Green 2002, pp. 60–61). Based on this observation, one can also understand why tablatures do not need to be precise when it comes to rhythm information (like ASCII tabs), since songs and their rhythm are learned by listening to the original recordings. The situation evolved slightly because tablature notation software now usually feature a *playback* function that is considered very important by guitarists (Bacot et al. 2024). With such a feature, it is necessary for the rhythm to be transcribed precisely, so that the digital tablature can replace actual recordings when practising. Nonetheless, modern tablature notation allows much more than simply specifying where to play notes on the fretboard. It allows extensive writing of *playing techniques*, an example of all those techniques and how they are engraved is shown Figure 1.5. This notation is not entirely standardised and slight variations can still be observed even if they are all somewhat similar (Gelling 2003; Laliberté 2010). The importance of guitar techniques can also be seen from the fact that 14 pages from the Guitar Pro User Guide are dedicated to them (*Guitar Pro8 User Guide* 2025, pp. 113–127) or that the website Ultimate-Guitar have a dedicated section in its ASCII Tablature guide (*Ultimate Guitar Tablature Guide* 2025). One could also have a look to musicological research like (Gomez 2016, pp. 17–50) or (Norton 2008, pp. 23–33) as well as guitar educational books (Gelling 2003) to observe how important those techniques are in guitar playing.

1.2.2 Composing Tablatures?

This section's title is a question, because composing with/on guitar tablatures is not a common activity. Indeed, as said in Navarret (2013, p. 47): "for most popular music of the 20th century, the score is not a prerequisite to creation but rather an artefact drawn from it.",[12] an observation that was also made for Rock'n'Roll in (Bertrand 1998). This claim is also supported by Tobias (2013) on WPM in general. Through field work with high-school students creating popular music, Tobias observed that the students never referred to their work as composing but rather as songwriting, or even "recording", or "jamming". The students created their song dynamically, updating or recording new tracks throughout production and, in addition, none used actual scores to discuss and prepare their music, all discussions being based on what was shown in the interface of the Digital Audio Workstation. Besides, while "[m]usic education literature has traditionally framed recording as an act of preservation" (Tobias 2013), recording is a key part of the creative process in popular music. Recordings are also the main artefacts used by popular musicians when learning (L. Green 2002; Navarret 2013). Those observations have direct implications on tablature usage. A tablature is not designed to be written then directly performed on stage, but rather *may* come at a posterior time for memorising and sharing purposes. In that case, it does not really make sense talking about tablature composition for WPM. However, guitar players can be composing *with* tablatures.

[12] Translated from French.

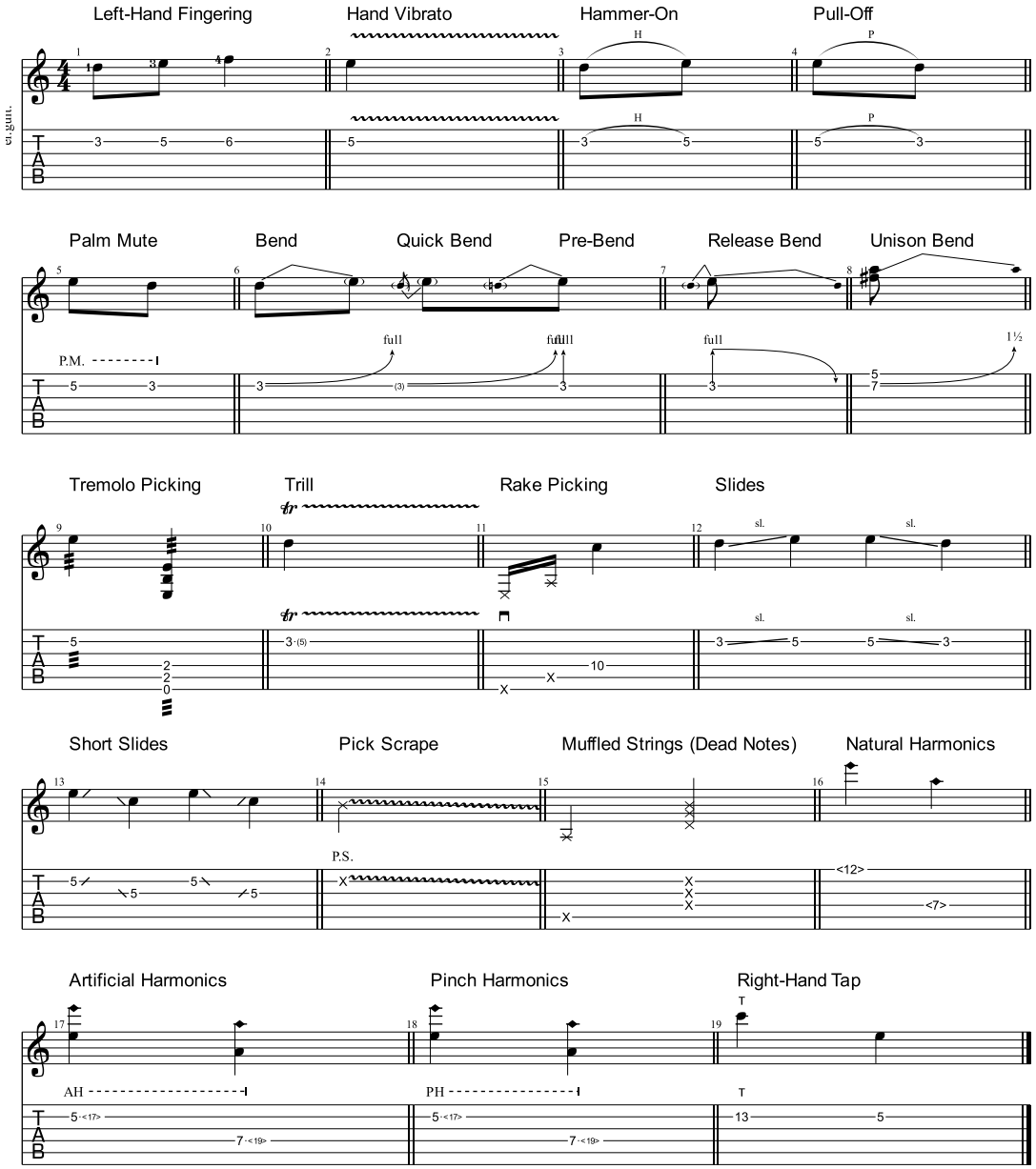
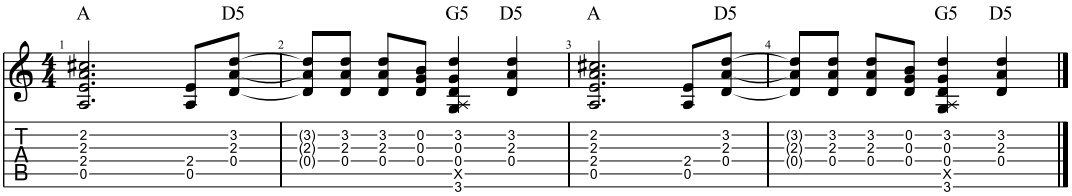
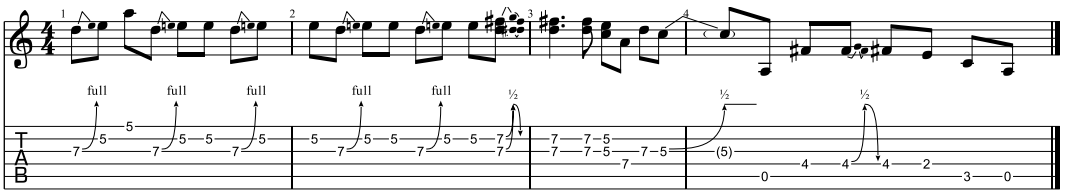


Figure 1.5: Some guitar techniques and their notation as found in AC/DC 2008. Some symbols have been changed when being reproduced in Guitar Pro.



(a) Chorus excerpt.



(b) Solo excerpt.

Figure 1.6: Tablature excerpts from *Highway to Hell* by the band AC/DC, written by Bon Scott, Angus Young and Malcolm Young. Transcription taken from DadaGP(uncredited).

Even though composition usually happens while testing ideas with the guitar or during band practice for instance, the process of writing down a guitar track on tablature can be part of the compositional process. It was indeed shown that digital software, through their affordance, have an impact on songwriting (Peterson 2009; Tobias 2013). This is also the case for guitar tablature notation, as was shown in Bacot et al. (2024). Through interviews with guitar composers, Bacot et al. highlight how some features of the notation software (Guitar Pro in that instance), have a direct impact on the creation process. For instance the simple fact of being able to loop a section and listen to it with the playback function allows the guitarist to transcribe an accompaniment section and then improvise melodic licks over it until a satisfying one is attained.

In this thesis, we therefore consider tablatures as both composition artefacts and an iterative notation format that can be enhanced digitally with co-creative tools. The tablature as an artefact allows us to analyse how guitar tablatures are composed and to derive musicological knowledge on guitar practice in WPM. At the same time, the notation of tablature itself can be improved through AI models. The objective being that, if well integrated in the existing musicians' workflow, those improvements could make tablature a part of the composition process in WPM.

1.2.3 Rhythm and Lead Guitar

Like other polyphonic instruments, the guitar can play multiple roles in bands and ensembles. Overall, a guitar player has two main ways of playing: they can play melodic lines, or play chords for accompanying other melodic instruments like a singing voice. In WPM, those two roles would be called *Lead* or *Rhythm* guitar, respectively (Nemeroff

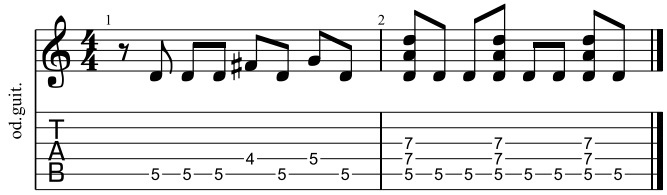


Figure 1.7: Main riff of *The Number of the Beast*, played by Dave Murray, Iron Maiden. Transcription made manually.

2024). In bands with two guitar players, there will usually be one *lead guitarist* and one *rhythm guitarist*. Famous examples would be: Angus Young (Lead) and Malcolm Young (Rhythm) from *AC/DC*, Rodrigo (Lead) and Gabriela (Rhythm) from *Rodrigo y Gabriela* or Lita Ford (mostly Lead) and Joan Jett (mostly Rhythm) from *The Runaways* (among other bands). A first issue starts to arise: the role of playing lead or rhythm guitar can evolve and is not always set in stone. In *The Runaways* for instance, Lita Ford and Joan Jett both played both roles at some point in the band's history. Likewise, James Hetfield from *Metallica* is a famous rhythm guitarist (Coffman 2023), but Hetfield will also play solos at time and thus endorse what is considered the lead guitarist role.

In fact, it makes little sense trying to assign permanently a single role to a guitar player or even to a guitar track inside a song. As an example, we can have a look at the two tablature excerpts from Figure 1.6. Both are played by the same guitarist – Angus Young – at different moments of the song *Highway to Hell*. Figure 1.6a is typical rhythm guitar, as the guitar only plays chords, underlining the harmonic progression that supports the vocals during the chorus. On the contrary, Figure 1.6b is a lead guitar section, taken from the solo of the song.

This simple example is one of many, and supports the hypothesis of Régnier et al. (2021) – which we also adopt in this thesis – that labelling a tablature as lead or rhythm guitar should be done at a lower level than the track. As proposed in the aforementioned paper, we consider that each bar of a tablature can be classified between rhythm and lead guitar (automatically if needed) and will use such bar-level labelling in the rest of this thesis. Distinguishing types of guitar playing is required when developing tools for assisting creativity because guitar composers may encounter different issues when composing accompaniment parts or melodic tracks. One should note, however, that while those categorisations are understandable and necessary to guitar players to communicate, the line between the two can be blurry at times. Riffs, are a good example of tricky excerpts, because they will usually be both rhythmic and melodic (Figure 1.7).

2

Machine Learning Models used in this Thesis

"As our understanding of computers continues to mature, it seems inevitable that machine learning will play an increasingly central role in computer science and computer technology."

Mitchell (1997)

Contents

A major part of MIR research now consists in using Machine Learning methods to study musical data, for tasks spanning from computational musicology to text-to-music generation. This thesis is no exception, and this chapter provides theoretical explanations of the Artificial Intelligence methods and core Machine Learning concepts used in the contributions presented parts II and III.

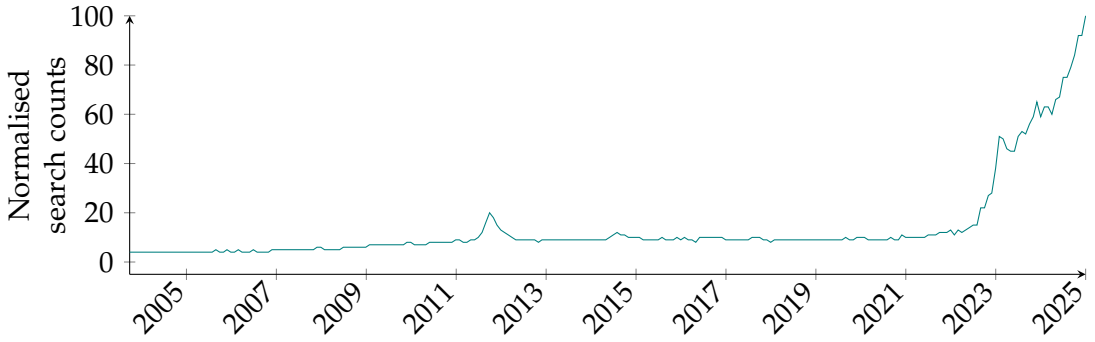


Figure 2.1: Normalised and monthly average of the number of queries including the word "AI" on Google. Data from Google Trends. The spike around 2012 *might* be due to the release of AlexNet (Krizhevsky et al. 2017, the model is from 2012 even though this paper came much later), while the latest increase roughly corresponds to the public release of ChatGPT (OpenAI 2025).

2.1 About Artificial Intelligence

What is Artificial Intelligence? Artificial Intelligence (AI) is now a term that is widely used in society even though its actual definition is rarely considered. The Cambridge English Dictionary[1] now defines Artificial Intelligence as:

Definition (Artificial Intelligence) *The use or study of computer systems or machines that have some of the qualities that the human brain has, such as the ability to interpret and produce language in a way that seems human, recognise or create images, solve problems, and learn from data supplied to them.* 🕮

Interestingly, this definition is more specific than the one of *Intelligence*, from the same dictionary:

Definition (Intelligence) *The ability to learn, understand, and make judgements or have opinions that are based on reason.* 🕮

The core idea behind Artificial Intelligence is therefore not to make a machine *intelligent* in the human sense – something that is sometimes called *Artificial General Intelligence* (Goertzel 2014) – but rather to design a computer program that tackles tasks commonly done through human thinking like patterns recognition, language understanding, image or music creation, etc.[2] Those tasks however benefit from high computational power that allows to tackle problems that can be hard to solve for humans, like clustering samples in more than 3 dimensions for instance. However, the AI field tends to use terms that humanise the algorithms used (Cooper et al. 2023), and some models are designed to pass

[1] https://dictionary.cambridge.org/dictionary/english/artificial-intelligence

[2] Robotics might be considered a field of artificial intelligence with that definition. While some aspects of robotics might be based on AI algorithms like computer vision, the field is broader and many aspects do not intersect with AI (Garcia et al. 2007).

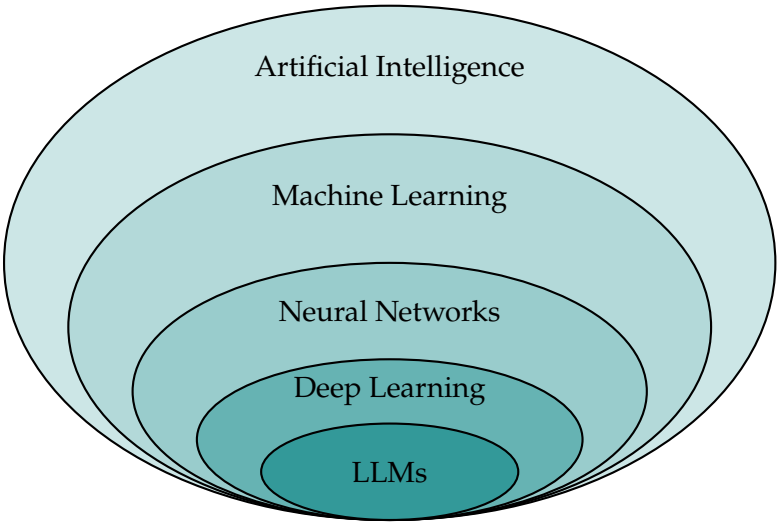


Figure 2.2: Artificial Intelligence and the sub-categories it encompasses.

as human. The original Turing test,[3] for instance, was actually designed to observe if a computer could pass as human to an external observer through a textual conversation, with the objective to answer the broad question "Can machines think?" (Turing 1950).

AI sub-types in research AI is a very prevalent term in society since the public release of *ChatGPT* in November 2022 (see Figure 2.1). Other AI-powered chatbots, image and music generators were released around that time or before, but none seem to have generated that much public discourse. However, Artificial Intelligence it is not a new field of research at all. Artificial Intelligence exists as a research field since the 1950s, and it was applied early to the music field, with for instance Hiller et al. using a computer to generate the "Illiac Suite" consisting of four movements for a string quartet[4] (Hiller et al. 1979, pp. 152–164). Rather than defining AI on its own, we choose to define it by describing how it is usually implemented, including in this thesis, i.e. with Machine Learning (ML) methods. The term Machine Learning was coined in Samuel (1959), where the author devises an algorithm to play checkers that expand its graph of known moves when playing games. Within ML, multiple sub-categories can be identified. The main one would be Neural Networks (NNs), a category of programs originally designed as a numerical model of biological neurones. Those NNs, when stacked together, become *Deep Neural Networks* and hereby define the category of Deep Learning (DL). Finally, among DL techniques, recent architectures that spurred interest in AI like ChatGPT or DeepSeek are dubbed Large Language Models (LLMs).

A summary of those categorisations of AI techniques is given Figure 2.2, and theoretical explanations of those techniques are provided in the following sections.

[3] This test is mentioned for its historical value, but should not be considered as a state-of-the-art evaluation of AI systems (Pease et al. 2011).

[4] The first movement can be found at: https://youtu.be/n0njBFLQSk8?si=ZvClccIjiPu851Xl (accessed in July 2025).

2.2 Machine Learning Methods

In this section, we introduce considerations and definitions that apply to the entirety of ML methods. After presenting common ML tasks, learning paradigms and evaluations metrics, we discuss what features are and how they are commonly defined and used in ML. We then present non DL-based ML models used in this thesis: decisions trees and Naive Bayes models. The decision tree is used in chapter 6 to suggest where to add bends in a tablature without playing techniques. The Naive Bayes model is used in chapter 5 to predict the difficulty of a tablature from a set of input features.

2.2.1 General Considerations

Let us start with a definition of Machine Learning:

Definition 2.1 (Machine Learning Mitchell 1997, p. 2) *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure M, if its performance at tasks in T, as measured by M, improves with experience E.*

While we will try to limit "metaphorical anthropomorphism" as much as possible (Cooper et al. 2023), the term "learning" is used in this chapter in the sense of definition 2.1. The tasks T that can be tackled by ML methods are usually split in two categories: *Classification* and *Regression*.

Definition 2.2 (Classification Goodfellow et al. 2016, p. 100) *In this type of task, the computer program is asked to specify which of N categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \rightarrow \{1, \ldots, N\}$. When $y = f(\mathbf{x})$, the model assigns an input described by vector $\mathbf{x}$ to a category identified by numeric code y.*

Definition 2.3 (Regression Goodfellow et al. 2016, p. 101) *In this type of task, the computer program is asked to predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$.*

ML methods aim at obtaining a function $\hat{f}$ that replicates the desired processing function f on task T. The experience E will necessarily be based on a dataset X, and the designed method should ultimately guarantee $\hat{f}(x) \simeq f(x), \forall x \in X$. Different learning paradigms can arise from this initial statement:

Definition 2.4 (Supervised Learning) *Each element x of the dataset is associated with an expected output y, also called "label" or "target" (Goodfellow et al. 2016, p. 105). In that situation, the ML method has a defined objective and is trained to output $\hat{y} = \underset{y \in Y}{\text{argmax}} P(y|x)$ i.e. the target that is the most likely given the existing (x, y) pairs.*

Definition 2.5 (Unsupervised Learning) *In an unsupervised learning paradigm, there is no existing data Y that is expected. This includes tasks where no definite output is known, like clustering data samples or compressing information.*

Obtaining labels for supervised learning is expensive, and might be unrealistic for large datasets. For instance, most of this thesis required processing thousands of tablatures, and manually annotating them would have proven tedious and extremely time-consuming. Unsupervised learning is nonetheless not entirely satisfactory, because it limits the tasks to ones where a possible *ground-truth* is not needed. An intermediate method is a refinement of unsupervised learning, called self-supervised learning, which is the main learning paradigm used throughout this thesis.

Definition 2.6 (Self-Supervised Learning de Sa (1993)) *Machine Learning paradigm where a model produces its own output labels Y from the input dataset X, either through transformations or combinations of data modalities. The simplest example might be models for lossless compression (autoencoders), where the expected output is the input reproduced identically.*

It is also possible to train a model on unlabelled data by exploiting information of the performance measure M.

Definition 2.7 (Reinforcement Learning Goodfellow et al. 2016, p. 106) *Machine Learning paradigm where the algorithm interacts with an environment, creating a feedback loop between the system and the experiments. The model learns iteratively what actions lead to the best result, according to a performance measure M. This paradigm does not require any labels but only a definite measure of the model's performance. Typical applications are on training models for playing games.*

For all those learning paradigms, the initial phase of fitting the $\hat{f}$ function using the available data is called *training*. A common issue that can arise when training a ML model is that of *overfitting*.

Definition 2.8 (Overfitting Briot et al. 2020, p. 269) *The situation for a machine learning algorithm when the model learnt is well fit to the training data but not to the evaluation data. This means the inability of the model to generalise well.*

In music-related tasks, overfitting can be a pernicious issue, often hard to detect. For instance, music style classifiers models can be biased by characteristics as subtle as the way a song is mastered[5] (Sturm 2012).

One of the most common ways to limit overfitting and weak generalisation of ML models is to have three different subsets of data for each of three phases: *train*, *validation* and *test*. The train set will be used to fit the model until a satisfactory performance is reached with $\hat{f}$. The validation set is used throughout the training phase to measure when the model starts overfitting. Because this data is not used directly during training, a decrease in the performance measure M for the validation set suggests that the model is starting to overfit the training data and lose generalisation properties. The test set should not be used during training at all, as it is data that is set apart from the start to ultimately

[5]Mastering is the action of modifying the frequency spectrum of an audio file to ensure it maintains auditory qualities on different loudspeakers.

measure the performance of the model. Ideally, test data should be taken from another source than train and validation data to limit internal biases, though this is not always possible. Once the model is trained, it can be used to process new unknown data samples, and this procedure is dubbed *inference*.

Performance measures. The performance measure M required for a ML method (definition 2.1) depends on the task category at hand. We focus here on measures for supervised and self-supervised learning paradigms, where performance measures can be readily defined. Let us write the expected results as $\mathbf{Y}$, and the predictions of the ML model as $\widehat{\mathbf{Y}}$. In regression tasks, the Mean Squared Error (MSE) can be used, defined as:

$$\text{MSE}\left(\mathbf{Y}, \widehat{\mathbf{Y}}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{2.1}$$

For classification tasks, all values from $\mathbf{Y}$ and $\widehat{\mathbf{Y}}$ are expected to be between 0 and 1, and the *Binary Cross-Entropy (BCE) Loss* is one of the most common performance measure:

$$\text{BCE}\left(\mathbf{Y}, \widehat{\mathbf{Y}}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right) \tag{2.2}$$

While this loss is for binary classification problems (only 2 classes possible), it can be generalised to any number of classes. Technically, the two measures presented above are *loss functions*, the closer they are to 0, the better. A performance measure derived from those would need to be their reciprocal, so that a better performing model has a higher measure value.

Performance Measures for music-related tasks. The two measures presented before are generic metrics that can be used to fit an ML model in an agnostic manner to the task at hand. However, musical tasks might need more complex measures that analyse the musical content in some way. In the next chapters, we will present new metrics designed for the musical tasks studied, other metrics related to the field of MIR can be found in Raffel et al. (2014). Some tasks are however not straightforward to evaluate, especially when some kind of *creativity* is expected from the model. In such situations, thoroughly thought-out quantitative metrics might be useful, but they should be complemented by appropriate qualitative human evaluations (Jordanous 2012).

2.2.2 Features in Machine Learning

A *feature*, sometimes also called a *descriptor*, refers to a "piece of information included in the representation of a [data item]" (Goodfellow et al. 2016, p. 3).

Feature Engineering consists in manually preprocessing data to extract relevant feature representations. For music-related tasks for instance, the input data can be a tablature in *Guitar Pro* format, and the features can represent information about the chords played

in each bar (Régnier et al. 2021). Examples of features that can be extracted from audio signals are reported in Peeters (2004). On symbolic music data, aspects like functional harmony (Gotham et al. 2023) or texture (Giraud et al. 2014) can also be analysed and used as features for further numerical studies. One of the main limitations of feature engineering is that it usually requires extensive domain knowledge for the task at hand. Selecting appropriate features and pre-processing the data to extract them can be time-consuming for the researcher during the selection process. However, manually selecting high-level features ensures that the ML models used downstream benefit from a meaningful data representation. Pre-processing data in such a manner allows to reduce the dimensionality of the data by removing information unrelated to the task at hand, e.g. if one studies the chord progression of rock songs, they can remove melodic and rhythmic information. In this thesis, manual feature engineering is used to study which features allow to understand when bends are used in guitar tablature (chapter 6) or what makes a tablature difficult to play (chapter 5). Features are also used to determine the level of a guitar player and what songs they should learn with their current expertise (chapter 4).

To simplify the process of feature selection, some ML works permit automatically selecting intermediate representations and features for a complex task. This approach, dubbed *feature learning* or *representation learning*, can be relevant for studying music, because songs contain complex interleaved information such as chords, melodies and rhythm that interact with one another simultaneously but also through time. While fields like representation learning specifically study how to efficiently extract features from the input data, this action is also done intrinsically by most DL models, where information is condensed as it goes deeper in the network. Those approaches, while efficient when no features can be defined beforehand (because they are too complex to define or do not capture the aspects studied), require heavy computational loads because they use larger models and repeat the feature extraction process every time a sample is passed even if it is known. We implement such a DL model in chapter 9, where a BiLSTM model (see section 2.3) computes an intermediate representation that is used as input by a transformer model (ibid).

2.2.3 On Interpretability and Explainability

Using features in ML has the advantage to embed high-level knowledge in representations that can be used by digital models. These features can allow for more understandable results as they often directly derive from human representations. However, an ML model is not necessarily *interpretable* and *explainable*. Let us use the definitions provided in Dib (2024).

Definition (Explainability) *The explainability of a machine learning model f can be defined as its ability to provide understandable, clear, and accessible explanations to human users about the underlying reasons for specific predictions generated by the model. Formally, f is considered explainable if $\forall(x, y) \in X \times Y$, the user can understand how the output y is derived from the input*

x. In other words, explainability aims to identify and make understandable the specific aspects of this observation that influenced the model's decision. 📖

Definition (Interpretability) *The interpretability of a machine learning model f can be defined as its ability to be understood and interpreted in its entirety, allowing human users to grasp the overall functioning of the model, including its internal components, relationships between variables, and decision-making process. Formally, f is considered interpretable if $\forall(x, y) \in X \times Y$, the decision-making process $Y = f(X)$ is understandable by users. In other words, interpretability seeks to unveil the internal aspects of the model that led to the final decision, providing a more holistic view of its decision-making process.* 📖

The deeper and more complex an ML model gets, the less interpretable it becomes, as outputs may be influenced by unsuspected factors (Sturm 2017). Besides, not all models are explainable. Even a transformer network can technically be dissected to analyse what drove the predictions as it is done in the original paper (Vaswani et al. 2017), but those analyses can soon be intractable with deep and complex models. While techniques like LIME (Ribeiro et al. 2016) and SHAP (Lundberg et al. 2017) aims at making any prediction explainable, they are not applicable to all network architectures. Overall, to guarantee the interpretability and explainability of an ML method, one should consider using simple models, like the ones presented hereafter.

2.2.4 Decision Trees

In chapter 6, we use a decision tree to suggest where to add playing techniques in a melody written in tablature notation. Decision trees are a category of ML models that represent a succession of choices as a tree and can be used for classification tasks. A mock example of a decision tree for determining which style of *metal* a song belongs to is provided Figure 2.3. The *root* of the tree is its starting point, at the top, and it gradually splits at new *nodes*. If a node is the last of its branch (i.e. it is not split further), it is called a *leaf* (or a terminal node/subset). The splits are defined by logical equations on features of the input data like, in the example provided, the lyrical content and instrumentation used. If nodes can only be split in two branches, we talk about a *binary decision tree*.

While a decision tree can be made manually in simple cases, it can also be built automatically from available data in a supervised learning paradigm. Algorithms that automatically build decision trees from data are available in dedicated Python libraries such as `scikit-learn` (Pedregosa et al. 2011). In essence, the goal of those algorithms is to find the best splits so that each leaf contains only samples belonging to a single class of the input dataset (even though the same class can be found on several leaves). The core process is a function that ensures that the *purity* – the class homogeneity of data samples in a node – of the new nodes is higher than the parent node. A common metric to measure the *impurity* (which has to be minimised) is the *Gini* criterion (Breiman et al. 1984):

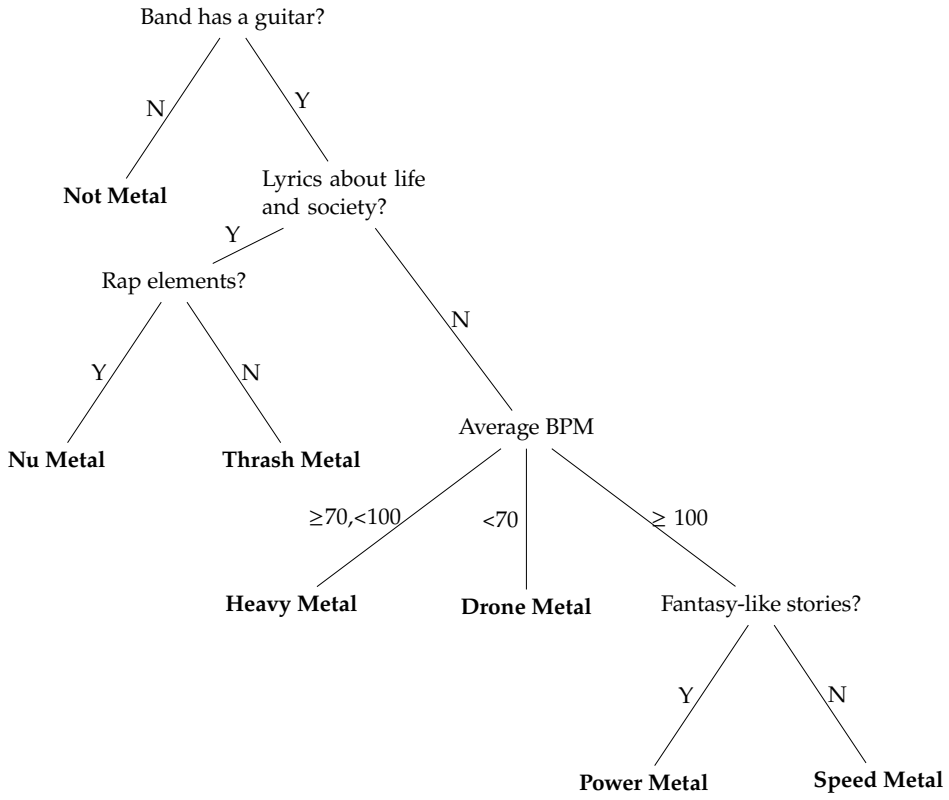


Figure 2.3: Dummy decision tree to disambiguate metal styles for a given song, leaves are written in bold, Y/N stands for Yes/No. This is of course an *oversimplification* and is not intended as a valid description of metal styles whatsoever.

$$I(n) = \sum_{c \in C} p_{nc}(1 - p_{nc}). \tag{Gini}$$

where n is the node studied, C the set of possible classes, and p_{nc} is the proportion of samples belonging to class c on node n. Once a decision tree has been fitted on training data, it can be used for inference on new data by simply following the decision boundaries to move from one branch to the next. The class assigned to the new sample is then the final leaf attained by the model.

2.2.5 Naive Bayes Models

Another common ML model for classification is the Naive Bayes Classifier, which we use in chapter 5 to predict the difficulty class of bass and guitar tablatures. As the name suggests, those techniques are based on the *Bayes' theorem*:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \text{ if } P(B) \neq 0. \tag{2.3}$$

Naive Bayes can be used to build *classifiers* that should predict one of M output classes $C_j, j \in [\![1, M]\!]$, given a set of N input features $F_i, i \in [\![1, N]\!]$. In that case, the Bayes'

Theorem can be rewritten as follows:

$$P(C_j|F_1, \ldots .F_N) = \frac{P(C_j)P(F_1, \ldots, F_N|C_j)}{P(F_1, \ldots, F_N)}. \tag{2.4}$$

For classification, the model is expected to find $\widehat{C_j}$:

$$\widehat{C_j} = \underset{j \in [\![1,M]\!]}{\operatorname{argmax}} P(C_j|F_1, \ldots, F_N). \tag{2.5}$$

Now comes the assumption that make those classifiers "naive": all input features are considered independent from one another (they only depend of the class C_j). While this assumption can be unrealistic, it allows to simplify $P(F_1, \ldots, F_N|C_j)$ greatly (using the probability chain rule):

$$P(F_1, \ldots, F_N|C_j) = \prod_{i=1}^{N} P(F_i|C_j). \tag{2.6}$$

We then obtain the final equation Naive Bayes models aim at optimising:

$$\widehat{C_j} = \underset{j \in [\![1,M]\!]}{\operatorname{argmax}} \quad P(C_j) \prod_{i=1}^{N} P(F_i|C_j). \tag{2.7}$$

Naive Bayes classifiers are fitted in a supervised fashion. Several options are available to model the *a priori* probabilities $P(C_j)$ and the *likelihoods* $P(F_i|C_j)$. The $P(C_j)$ are usually a uniform distribution based on the observed class frequencies in the training dataset. For continuous feature values, likelihoods are often estimated using a Gaussian Naive Bayes (GNB) model that assumes a normal distribution of the values for each input feature, where the mean and standard deviation are estimated from the data. The GNB is the version used in chapter 5.

2.3 Deep Learning Methods

This section aims at introducing neural networks and the Deep Learning models used in future chapters. In chapter 7, a MultiLayer Perceptron is used to suggest guitar positions, while chapters 8 and 9 use transformer networks to generate rhythm guitar or bass tablatures, respectively. For a more detailed presentation of those networks, and others, one can also read Briot et al. (2020), Goodfellow et al. (2016), and Le Cun (2019).

2.3.1 Neural Networks and Perceptrons

Deep Learning methods are a specific part of Machine Learning, but they are also a subcategory of Neural Networks (NNs), sometimes also called Artificial Neural Networks. The adjective *neural* comes from the fact that the first research on such methods tried to numerically replicate the functioning of neurons. The first mathematical modelling of binary artificial neurons can be found in McCulloch et al. (1943) where a number of inputs

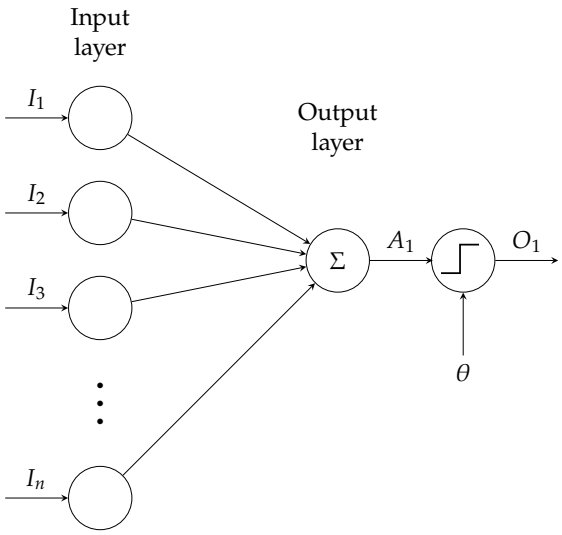


Figure 2.4: An artificial neuron as introduced in McCulloch et al. (1943). The binary inputs are summed into an activation value, which is further converted by a step function of threshold θ.

goes into a "cell" and are summed, activating the input (1) if a threshold is reached, or no activation (0) otherwise. A graphical representation of such an artificial neuron is proposed Figure 2.4.

This model, though simple, was further adapted in Rosenblatt (1958) where the authors softens some of the original assumptions. The main modification is that the binary input values now go through a *weighted* sum instead of a standard one. The other important improvement is that Rosenblatt suggests updating those weights dynamically on known data, effectively conducting supervised learning. He called this model the *Perceptron*, of which we propose a representation in Figure 2.5.

Like ML models, neural networks are designed to find an estimate $\hat{f}$ of a function $f : x \mapsto y, \forall (x, y) \in X \times Y$. With a perceptron, $\hat{f}$ is defined as:

$$\hat{f} : x \mapsto H\left(\sum_{i=1}^{n} w_i.x_i - \theta\right). \tag{2.8}$$

where H is the Heaviside step function, θ the decision threshold, w_i the weights of the network and x_i the elements of the input vector. However, the perceptron is limited in the functions it can model, because it is entirely linear. For this reason, a perceptron can only be used on data that is *linearly separable*. More precisely, if a perceptron has n different inputs, it can only define a hyperplane in this n-dimensional space, i.e. a boundary defined by $n - 1$ parameters. This might not be an issue in some cases, but imitating the logical exclusive OR (XOR) is a simple example of the limitations of perceptrons (Minsky et al. 1969). To address this issue and a few others, the MultiLayer Perceptron (MLP) architecture introduces the following main improvements:

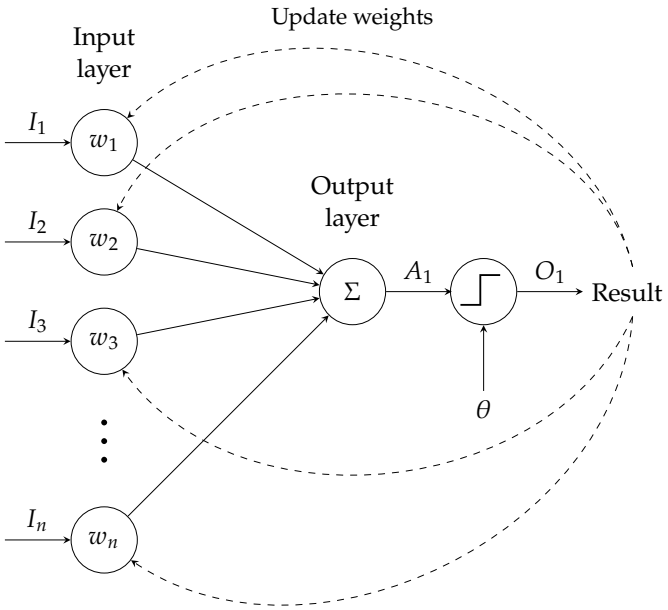


Figure 2.5: The perceptron (Rosenblatt 1958).

- The inputs and outputs are no longer binary and can be real values;
- Hidden layers can be added between the input and output layers, with a varying number of weights. Those layers allow to add *depth* to the perceptron and thus model more complex functions, each layer multiplying the previous layer's outputs by new weights;
- All layers can include a *bias*, a weight that is not dependent on the input data;
- All layers but the input are followed by a *differentiable* (required for training) *activation function*;
- The weights are updated based on training data using *backpropagation*.

The final MLP with all those modifications is presented Figure 2.6, and the newly introduced activation functions and backpropagation principle are introduced hereafter.

2.3.1.1 Activation Functions

The perceptron can only process linearly separable problems. Even with more weights and layers, an MLP has the same constraints if no non-linear operations are introduced in the processing. The activation functions (denoted with φ in Figure 2.6) are used to add non-linearity into the neural network, thus allowing it to model non-linear functions. Four common activation functions are represented Figure 2.7. Because an MLP can output continuous values, it might also be desirable to enforce the model to only return values

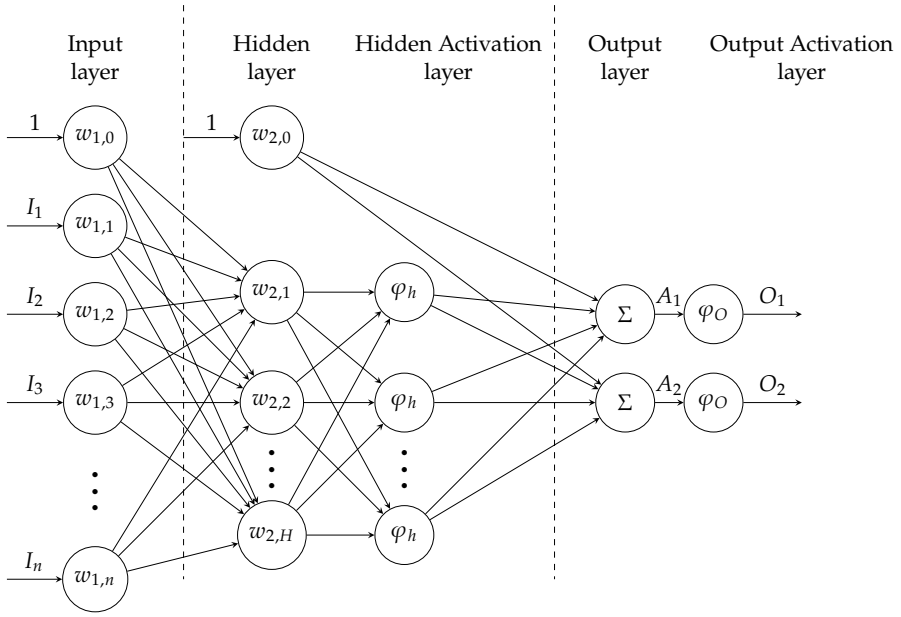


Figure 2.6: A Multilayer Perceptron with n inputs and one input *bias* coefficient $w_{1,0}$. It also has one hidden layer of size H with a bias coefficient $w_{2,0}$ and an activation function φ_h. The final activation function is φ_O. Because each element of each layer is connected to all elements of the next, we call this a Feed-Forward NN.

between a small subset of $\mathbb{R}$, to use the output as a class probability for instance. This is usually done with the *sigmoid* or the *tanh* function which map $\mathbb{R}$ to $[0, 1]$ or $[-1, 1]$, respectively. Activation functions can also be used after hidden layers to add even more non-linearity within the model. The previously mentioned functions can be used, but functions that limit the amount of negative values can be used as well, like the Rectified Linear Unit (ReLU) or the Gaussian Error Linear Unit (GELU).

Activation functions could theoretically be any function that is beneficial to the model's purpose. However, those functions need to be *differentiable* to be compatible with the back-propagation training technique, presented below.

2.3.1.2 Loss Functions and Backpropagation

When stacking layers in models, like in an MLP, we enter the realm of Deep Learning. In a deep model with a lot of connections, updating weights to better reproduce a desired function can be complicated. The solution was found in using *backpropagation*, usually implemented through *gradient descent*, to propagate errors backwards in a neural network and update its weights accordingly. To put it simply, gradient descent consists in finding the direction of the steepest slope of a loss function, and updating the weights in that direction. Let us call this loss function $\mathcal{L}$, it has to be minimised to reach the best performance of the model. We want to find the optimal weight configuration $\mathbf{w}^*$ on a dataset $(\mathbf{X}, \mathbf{Y})$:

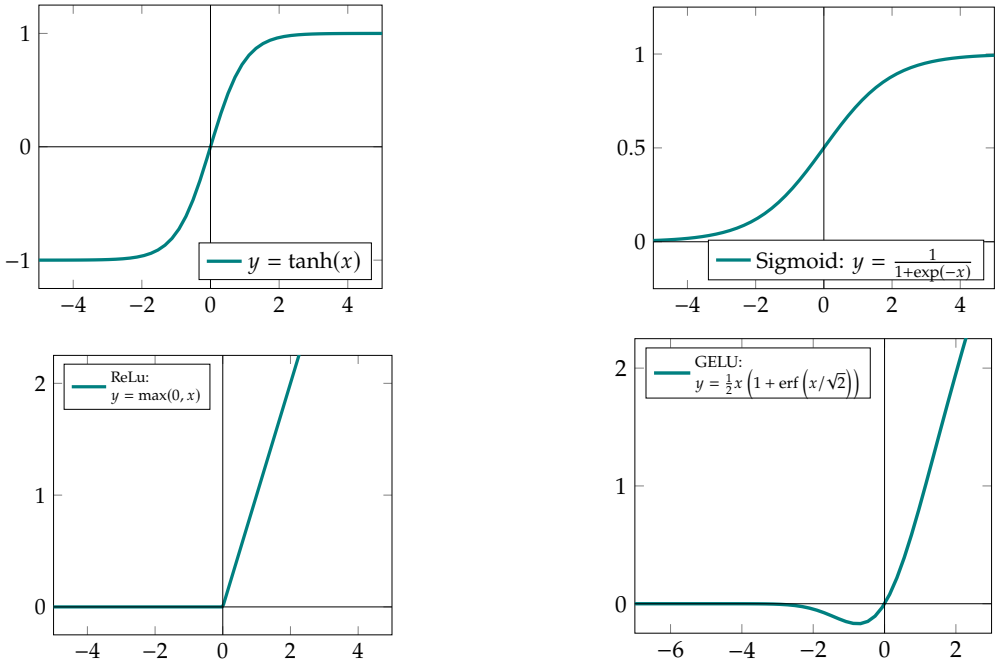


Figure 2.7: Activation functions, from left to right and top to bottom: tanh, sigmoid, ReLU and GELU.

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}\left(\hat{f}(\mathbf{w}, \mathbf{X}, \mathbf{Y})\right). \tag{2.9}$$

To do so, we compute the partial derivative of the loss function with respect to each weight w_i of the model and use it to update each weight of the model:

$$w_i = w_i - \lambda \frac{\partial \mathcal{L}\left(\hat{f}(\mathbf{w}, \mathbf{X}, \mathbf{Y})\right)}{\partial w_i} \tag{2.10}$$

Because we want to minimise the loss value, we subtract the derivate to *descend* the gradient slope when updating the weights. The λ parameter is called the *learning rate* and controls the amount of each weight's update. This core principle can be improved in many ways, for example by adding momentum to the gradient descent (Kingma et al. 2015), learning rate scheduling (Alonso-Jiménez et al. 2023), etc.

2.3.2 Recurrent Neural Networks

In MLPs, each layer is entirely connected to the previous one, and only processes information from that previous layer. Those types of networks are qualified as "Feed-Forward". However, there are situations where the sequential nature of the input data is important, like when processing text, or *music*, for instance. For those situations, a Recurrent Neural Network (RNN) should be used (Rumelhart et al. 1986). Compared to a regular neural network, an RNN will also include a cyclic connection that allows past information of a sequence to be retained (Figure 2.8). The exact mechanism for retaining that information

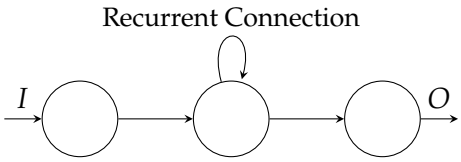


Figure 2.8: A simple RNN with one recurrent connection.

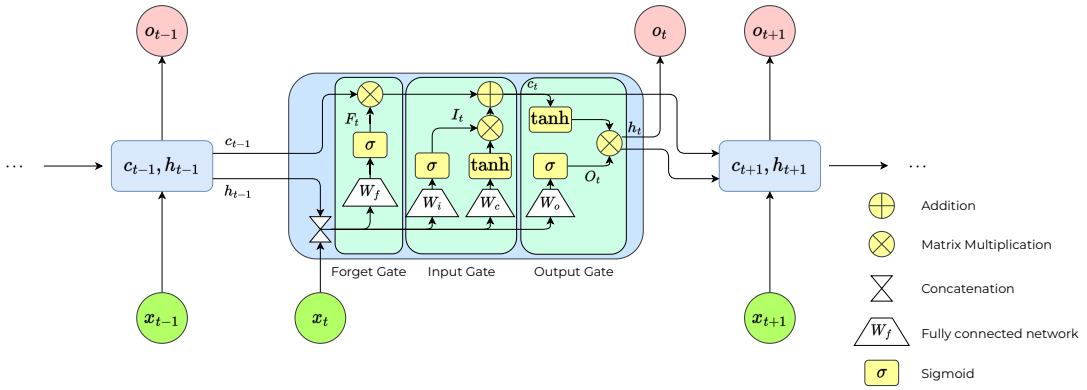


Figure 2.9: An LSTM cell and its connections to the previous and next ones. Original figure by Ixnay (Wikimedia Commons).

can differ, two common architectures are discussed hereafter: the Long Short-Term Memory (LSTM) network and the Gated Recurrent Unit (GRU). Both architectures are called *gated* RNNs and have parts charged in retaining information from processing one element of a sequence to the next.

Data Pre-processing. RNNs cannot use textual or musical data as input directly two pre-processing steps are required: i) the files need to be organised into sequences of multiple chunks, ii) each chunk itself has to be represented as a vector of float values. Those two pre-processing steps are called *tokenisation* and *embeddings* computation, respectively. The tokenisation consists in splitting the sequence into unique elements of a vocabulary. For text, it could be characters, words or phonemes. For symbolic music, it will depend on the task at hand, but most tokenisation schemes will represent the data at the note level (Le et al. 2024). Once the data can be represented as a sequence of vocabulary elements, it is multiplied by a matrix which weights are learnt during training to represent sequence elements in an embedding space. In language processing, embeddings inform a model on what words have similar or opposite meanings. Likewise in music, embeddings are used to locate tokens in a multi-dimensional space in a way that informs the model on their inter-relations.

LSTM The Long Short-Term Memory network has been introduced in Hochreiter et al. (1997) and one LSTM cell is represented Figure 2.9. An LSTM network is made of multiple LSTM units connected to one another and gets its name from the fact that it retain long-term and short-term memory in two separate variables. At its core, an LSTM unit contains

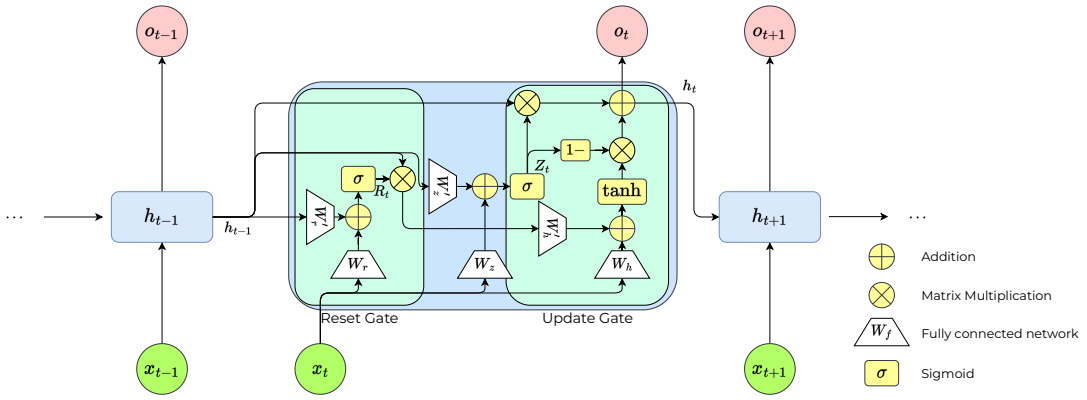


Figure 2.10: One Gated Recurrent Unit and the previous and next units. Original figure by Ixnay (Wikimedia Commons).

a cell state c_t that acts as the long-term memory, and a hidden state h_t that serves as the output at each time step as well as the short-term memory. To control how information is retained from one unit to the next, the LSTM uses three main gates: the forget gate, the input gate, and the output gate. Each gate multiply its weights with the input, and add a bias coefficient, all parameters are updated in the training process. The forget gate controls how much of the long-term memory persists within the LSTM unit, given the current input x_t and the past hidden state h_{t-1}. The input gate controls how much of the short-term memory (related to the current input x_t) should be stored in the cell state (the long-term memory). Finally, the output gate controls what part of the updated long-term memory (the cell state) should be copied to the hidden state and sent to the next unit.[6]

GRU The Gated Recurrent Unit was introduced in Cho et al. (2014) and is very similar to the LSTM architecture (Figure 2.10). Now the short and long-term memories are mixed in a single memory variable, and the unit only has a hidden state h_t that is sent to the next unit. The Forget and Input Gate are merged into an *Update Gate* that controls how much information from the previous hidden state should be carried over. The *Reset Gate* is used to control the balance between the input data and the hidden state that are transmitted to the output. Because the GRU has one less gate, it requires a little less operations that an LSTM unit, while having similar performance.

Finally, both architectures (and RNNs in general) can be made bidirectional by duplicating the model but changing the direction of the memory propagation. Such models allow to capture more complex dependencies and are called BiLSTMs or BiGRUs. Bidirectional RNNs can better process sequences where the end of the input has an impact on the beginning of the output. An NLP example is a model designed to switch a sentence from the active form to the passive form: "The cat ate the mouse" → "The mouse was eaten by the cat".

[6]For a short but detailed explanation, we recommend watching https://www.youtube.com/watch?v=YCzL96nL7j0. Accessed in June 2025.

2.3.3 Attention-Based Models

The RNNs allow to process data sequences and have had a significant impact on research in language and music. However, those networks have two major limitations: i) the data needs to be processed sequentially, which means that the training process cannot be parallelised, increasing the time required for training; ii) the memory is maintained throughout the sequence and gradually fades out, which means that the end of the sequence will only be weakly linked to the beginning. Both issues were addressed by the introduction of the *transformer* network (Figure 2.13), first presented in Vaswani et al. (2017). This section introduces the principles of a transformer network, more detailed technical explanations can be found in the original paper or Alammar (2018).

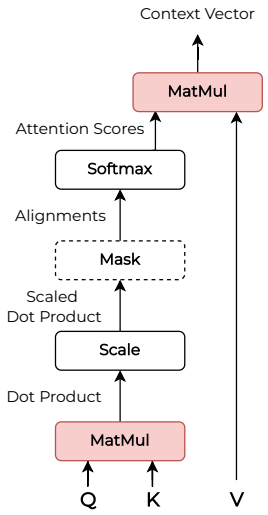


Figure 2.11: Internal principle of the self-attention mechanism in transformer networks. Masking is optional. Original graph by dvgodoy.

Like RNNs, a transformer network requires the input to be tokenised and represented in embeddings. In addition to the regular embeddings, transformers need a last *positional embedding* (sometimes also called encoding). This additional information is needed for the transformer to retain the position of the tokens in the sequence, since it processes tokens from the sequence in parallel. In the original paper, it was based on sinusoidal functions which frequencies depend on the position of the token of the sequence. However, in newer approaches, the positional embeddings are obtained by training a custom MLP model. Now that the input data is converted to vectors, it can be processed by the model.

The core principle of a transformer network for processing token sequences is *self-attention*. It allows the model to link all elements of the input and output sequences to one another within and across sequences. The computational process is represented Figure 2.11. The Q, K and V variables are called *Query, Key,* and *Value* and are all obtained by processing a token of the sequence with a weight matrix (one different for each variable). The intuition behind those variables is that the query represents a reference token that will be compared to another token, represented by the key, for every token of the sequence (including itself). The value is an additional learnable representation of a token that determines the final information that is preserved by the model. The matrices for Q, K and V are trained to obtain what could be compared to new embeddings. "MatMul" is short for Matrix Multiplication and the mask is an optional binary mask to ignore some tokens that should not be seen. For instance, it might be necessary to mask future tokens when training, to imitate the situation when the transformer generates tokens (and thus does not have access to future data). Finally, the result is passed to a *SoftMax* function, defined

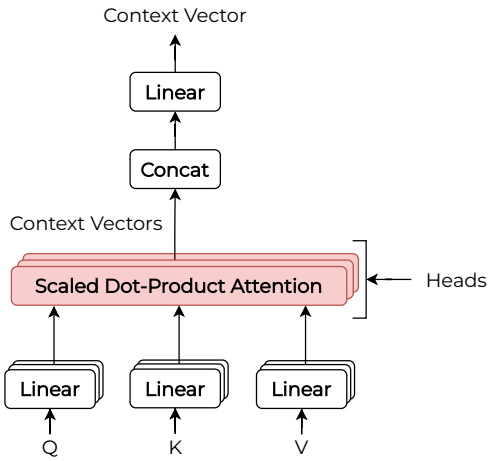


Figure 2.12: Attention with multiple attention heads. Original graph by dvgodoy.

below, that maps a vector of real values to $[0, 1]$, ensuring that the maximum value of the original vector is closest to 1 in the output.

$$\text{SoftMax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{x_j \in \mathbf{x}} e^{x_j}} \tag{2.11}$$

After this step, we get what are called the attention scores that represent what tokens of the sequence the query should most attend to. The final result is obtained from multiplying the *value* (which is updated during training) by those attention scores.

Finally, it is worth noting that everything can be done in parallel when computing attention. All tokens from a sequence can be used as the query at once, and a transformer can even have multiple *attention heads* to allow the model to attend to the sequence in multiple ways at once (Figure 2.12). An encoder and a decoder network can then be built by stacking self-attention layers, connected by MLPs. The encoder and decoder have their own self-attention layers to attend to the input and output sequences respectively, but the decoder also has a *cross-attention* layer that works identically to self-attention, except that it allows to link the decoder's output with the encoder's input (Figure 2.13).

The success of the transformer model on NLP tasks soon motivated its use in music, and the development of new models. Interestingly, both the encoder and the decoder of the transformer model can be used separately and perform well on their own sets of tasks. The encoder used alone and pretrained with masked tokens is usually called Bidirectional Encoder Representations from Transformers (BERT) and is appropriate for music understanding and analysis tasks (Devlin et al. 2019). It will generally be used as a pre-trained model and complemented by a smaller model that benefits from the learned representation and is fine-tuned for solving a new downstream task. Conversely, the decoder part trained for next-token prediction is well-suited for generation tasks and is dubbed Generative Pretrained Transformer (GPT), which is the basis of most LLMs like ChatGPT (Radford et al. 2018).

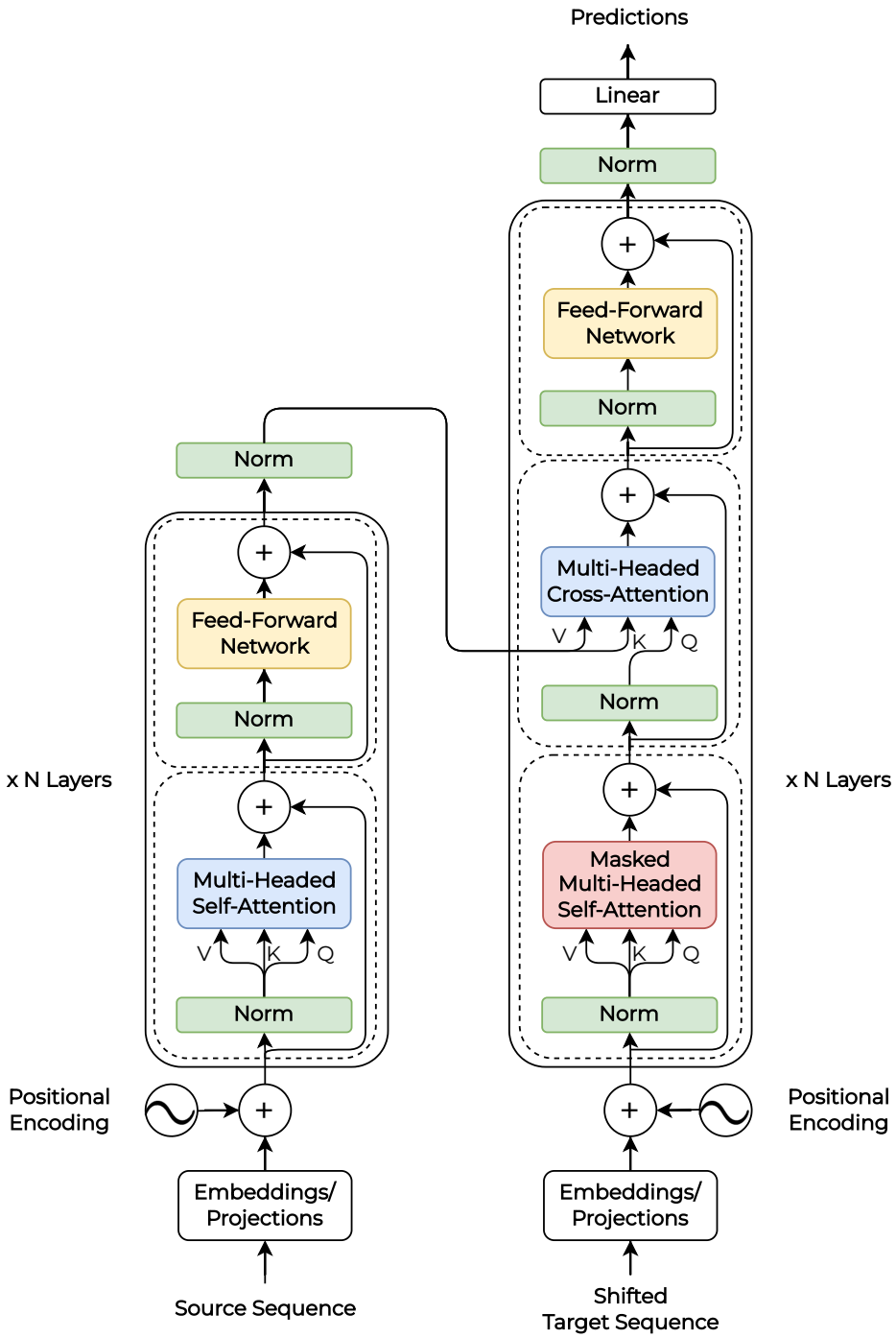


Figure 2.13: Diagram of the full Transformer Architecture. The encoder is on the left, the decoder on the right. Original graph by dvgodoy.

3

State of the Art

"Everything anyone ever makes is inspired by what's in their head – what they've played or read or encountered or thought a lot about."

Cavanagh (2019)

Contents

While the `mir-datasets` repository gathers 238 datasets[1] for MIR research, only 6 datasets mention guitar explicitly in their content, and only one of them (DadaGP, Sarmento et al. (2021)) is not designed for transcription and contains tablatures instead of audio recordings. This does not mean, however, that there have been no research on guitar tablatures since the first ISMIR conference in 2000, as this chapter first discusses. Then, following the categorisation of the contributions of this thesis (parts II and III), this chapter includes two main parts focusing on computational tools to assist tablature composition (section 3.2) and guitar pedagogy (section 3.3).

[1] https://github.com/ismir/mir-datasets, accessed in June 2025.

3.1 Tablature Data in MIR Research

Guitar in WPM is relatively little studied by the MIR community, especially when compared to other instruments like the piano, or other musical styles like Western classical music. Indeed, since the first ISMIR edition in 2000 up to 2024, 39 papers mention "guitar" in their title or abstract, versus 148 for "piano".[2] Tablatures were even less studied with only 9 ISMIR papers during the same period. In this section, we discuss how guitar was studied since the early 2000s, and how tablature data fit into that research. We then present two major tablature datasets we used in this work: mySongBook and DadaGP. After discussing other tablature datasets also available to researchers, we present possible digital representations of tablatures, and how tablatures have been studied from a computational musicology perspective.

3.1.1 Tablature Datasets and Representations

Guitar and Tablature Data in MIR Research The rarity of MIR research on (guitar) tablature might be both the origin and the consequence of a lack of appropriate datasets freely available to researchers. Indeed, guitar has not been represented in *publicly available* research datasets until fairly recently, its earliest inclusion being in audio datasets for chord recognition (Bosch et al. 2012), sound effects detection (Stein et al. 2010) or transcription (Kehling et al. 2014). There have been research on guitar and even tablatures before the 2010s, but it usually came with a reduced dataset assembled specifically for the question under scrutiny. The data was kept internal, and papers would provide information to varying extents for other researchers to reproduce the dataset on their own (see Table 3.2 at the end of this section). For instance, one of the earliest work of research on computer science applied to guitar music, Sayegh (1989), was almost entirely theoretical, supporting the proposed approach through simple musical examples like scales. Other works used audio data that was not shared publicly, for chord recognition for instance (Cabral et al. 2005; Zhang et al. 2008) or instrument classification (Eichner et al. 2006; Hamel et al. 2009). Using internal datasets was common practice and is still encountered in some recent works. In Table 3.2, we present research that specifically studied *guitar tablatures*, but with data not released publicly. We report on the amount of information they provide regarding the data used, to reflect on the actual reproducibility of the presented approach.

The first guitar *tablature* datasets publicly and openly available to researchers are technically datasets for Automatic Music Transcription (AMT). The first one might be the *IDMT-SMT Guitar set* (Kehling et al. 2014), which was followed by other datasets like *GuitarSet* (Xi et al. 2018), EGDB (Y.-H. Chen et al. 2022), or GAPS (Riley et al. 2024a). Because those datasets are designed for AMT, they contain audio data aligned with the

[2] I am indebted to Dinh-Viet-Toan Le for sharing with me the paper analysis tool he created for his survey (Le et al. 2024).

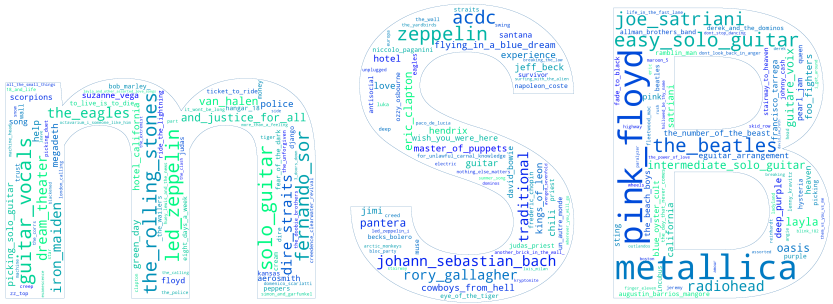


Figure 3.1: Wordcloud representation of the words contained in the filenames of the mySongBook database.

corresponding transcription, usually in tablature format. For this reason, they could technically be used by researchers to study tablatures. However, the studies would be limited because of the musical examples recorded for those datasets. For instance, GuitarSet contains recordings and transcriptions of guitarists playing three different chord progressions in five different styles. Likewise EGDB recordings are based on tablatures of "solos, arpeggios and comping in various genres". GAPS is an exception as is contains 300 performances of Western classical guitar pieces, with the corresponding tablatures. Nevertheless, open tablature datasets not specifically designed for AMT but general MIR tasks were also released. In Table 3.3, we present tablature datasets openly available to researchers and the papers that introduced them. The datasets used in this thesis, *mySongBook* and *DadaGP*, are discussed more extensively hereafter.

3.1.2 mySongBook

Most early work of this thesis has been conducted with the *mySongBook (MSB)* database. An excerpt of this database, property of the *Arobas Music* company, was gracefully shared to our research team thanks to the initiative of Louis Bigo who got in touch with the company, also based in *Lille*.

MSB is a tablature catalogue[3] that contains professionally transcribed tablatures in a variety of musical styles, like WPM in general but also Western classical music (see Figure 3.1). The full database grows weekly with new transcriptions, the version available in the team dates back to 2017 and contains 2115 guitar tablature files in the latest Guitar Pro format at the time: `.gp`. Those tablatures are transcriptions of songs from 308 different artists (with some songs being traditional music from various cultures, or educational content that have no clear artists identified). There are 1054 songs in total, out of which 304 are re-arrangements either to adapt a song to guitar, or to reproduce a live performance of it. Those arrangements can be the only occurrence of a song in the dataset, or another version of a song already transcribed, as is the case for 110 arrangements. For instance, a guitar tablature might be a transcription of the original version of a song, while

[3] https://www.guitar-pro.com/tabs, accessed in April 2025.

another tablature is a transcription of a specific live performance.

Being proprietary, the dataset cannot be shared with our research community. However, analyses and other musical descriptors were released as the *Algomus* team worked with MSB, at a time when no other large tablature datasets were available. More precisely, Cournut et al. (2021) shared the vector representations of the 1000 most common chords found in the MSB dataset, available at www.algomus.fr/data. The dataset was also used in the team to analyse *rhythm and lead guitar* tablatures (subsection 1.2.3) and train a model to automatically identify rhythm guitar bars (Régnier et al. 2021). Feature values computed on 102 guitar tablatures with manual annotations of bars that are considered rhythm guitar were also shared publicly at the previous link. Finally, a parser converting `.gp` files to `music21` Python objects (Cuthbert et al. 2010) was also released in Cournut et al. (2020).

Strengths and Weaknesses. The MSB database has allowed the *Algomus* team to conduct research on guitar tablatures at a time where guitar tablatures datasets were not adapted to computational musicology research (cf. Table 3.3). This dataset is particularly useful for its quality, since it comes from professional transcribers who ensured that the tablatures produced are as close as possible to the original songs. The tablatures are also very detailed, including extensive guitar techniques notation, structure markings, preconfigured effect pedals, etc. that could be lacking or even missing at times from amateur transcriptions on websites like Ultimate Guitar. Nonetheless, this dataset is relatively small, with 2115 tablatures, which is bigger than most tablature collections used in the literature before (Table 3.2), but still not enough for most modern Deep Learning models to solve complex tasks on guitar music data. Besides, the proprietary nature of MSB was incompatible with the good practices of open science we tried to follow during this thesis, sharing partial data or intermediate representations hindering actual reproducibility. These limitations justified the gradual switch from MSB to the DadaGP dataset, released in 2021.

3.1.3 The DadaGP Dataset

An important milestone for guitar tablature research was the release of the DadaGP dataset in Sarmento et al. (2021) that provided researchers with more than 25 000 tablatures of WPM. This dataset results of the joint efforts of the Dadabots duo[4] and Pedro Sarmento along with colleagues and supervisors. Tablatures were obtained from publicly available tab sharing websites. This dataset is described to a great extent in the original paper and Sarmento (2024) so we will only provide a brief overview of its content and the research it enabled.

[4] https://www.youtube.com/watch?v=JF2p0Hlg_5U, accessed in June 2025.

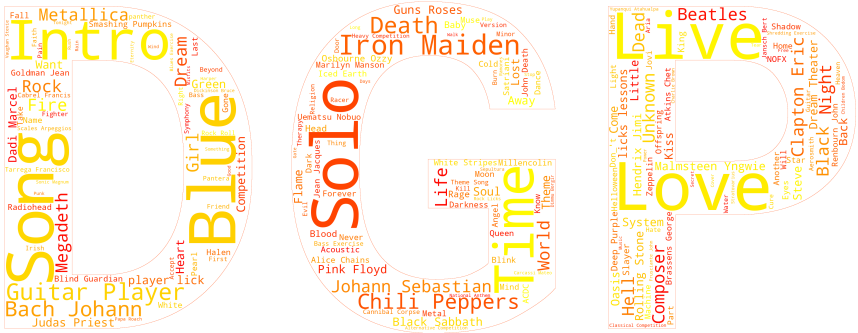


Figure 3.2: Wordcloud representation of the words contained in the filenames of the DadaGP dataset. This word cloud contains more structure information like "Song", "Intro", "Solo" compared to MSB because it contains multiple duplicate tablatures of some songs where only a part was transcribed.

DadaGP contains 26 181 songs composed by 882 different artists that span 739 musical styles.[5] While most songs can be considered rock or metal music in a broad sense, the dataset also covers WPM widely, with funk, pop, or even EDM songs (see Figure 3.2). All songs are in the Guitar Pro file format, the specific version can differ (included versions are `.gp3`, `.gp4`, and `.gp5`) but all can be converted to the token format presented in Sarmento et al. (2021) using the `pyguitarpro` Python library[6] to use them as input to RNNs (section 2.3.2). Songs may contain multiple instrument tracks and are not restricted to guitars, even though guitar is the most represented instrument (Table 3.1). Other instruments are for instance bass guitars, drums, or "lead" and "pad" instruments playing melodic or ambient lines, respectively. The dataset also includes songs in various guitar tunings, and songs with 7-string guitars or 5/6-string bass guitars. Finally, while most songs use a $\frac{4}{4}$ time signature, songs can be partly or entirely in other time signatures like $\frac{2}{4}$, $\frac{3}{4}$, or $\frac{6}{8}$, to a greater extent than MSB.

Strengths and Weaknesses. The DadaGP dataset is extremely varied, even with its bias towards rock and metal songs. Its large size and availability for research (upon request, no license specified) allows the MIR community to further study guitar tablatures, even with large DL models. This dataset also comes with a dedicated tokenisation format that allows using it and studying tablatures even without using the original Guitar Pro files. The main weakness of this dataset, compared to MSB, is that the transcriptions were made and shared freely by guitar enthusiasts. Without denying the importance and selflessness

[5] Obtained from the Spotify Web API, querying by artist and song title. Note that each song can have multiple style tags, 4 on average in DadaGP.

[6] https://github.com/Perlence/PyGuitarPro, accessed on April 2025. The library cannot process any Guitar Pro file in a version newer than 5.

Table 3.1: Number of tracks for each instrument in DadaGP. Note that each file can contain multiple instrument tracks. Lead is for "instruments with sharp attacks, e.g. piano" while pad is used for "instruments used more ambiently, like a choir or a string ensemble" (Sarmento et al. 2021).

Instrument	Number of Tracks	Ratio
Bass	14 941	17%
Drums	14 248	16%
Clean Guitar	16 190	18%
Distorted Guitar	28 003	32%
"Lead"	11 048	13%
Pads	3 358	4%
Total	87 788	

of those transcribers, it is worth noting that most are not professional transcribers, causing the quality of the tablatures to vary greatly.

Finally, it is important to acknowledge the impact the DadaGP dataset has had on MIR research. Many papers built upon the original generative transformer trained on the dataset and presented in Sarmento et al. (2021) to adapt it to a variety of situations (Adkins et al. 2023; Cui et al. 2024; Loth et al. 2023; Sarmento et al. 2023a; b). In addition, this dataset also has been used in other MIR subfields like MIDI-to-Tab conversion (Edwards et al. 2024) or AMT (Cwitkowitz et al. 2022). The dataset was also adopted by the community, since it has now been used in other works without the original authors (Pedroza et al. 2024; Zang et al. 2024).

3.1.4 Digital Representations of Tablatures

In subsection 1.1.2, we discussed the different file formats that can be used to store guitar tablatures. Those files are designed to encode the data for digital notation software, and sometimes mix information regarding the musical content with display parameters. For tablatures to be analysed automatically or used for training ML models, specific digital encodings are required. However, encodings already defined for music using standard notation cannot be used directly, as they would lose all tablature-specific information, like where notes are played on the fretboard or when playing techniques are used. For instance, Cournut et al. (2020) introduce different encodings for chords in tablatures like the *relative* encoding where frets are counted from the first played and not the first on the fretboard. With that encoding, they show how chords can easily be played as "boxes" translated around the fretboard. Similarly, ASCII tabs, because they are only lightly structured text files, also need suitable encodings to be used. Several representations were tested in Eldby (2021), and a representation where the tab is sent column-by-column turned out the best for an RNN to process.

Finally, tablatures can also be *tokenised* to be processed by transformer models. The two main tokenisation schemes that currently proved their efficiency for tablature generation

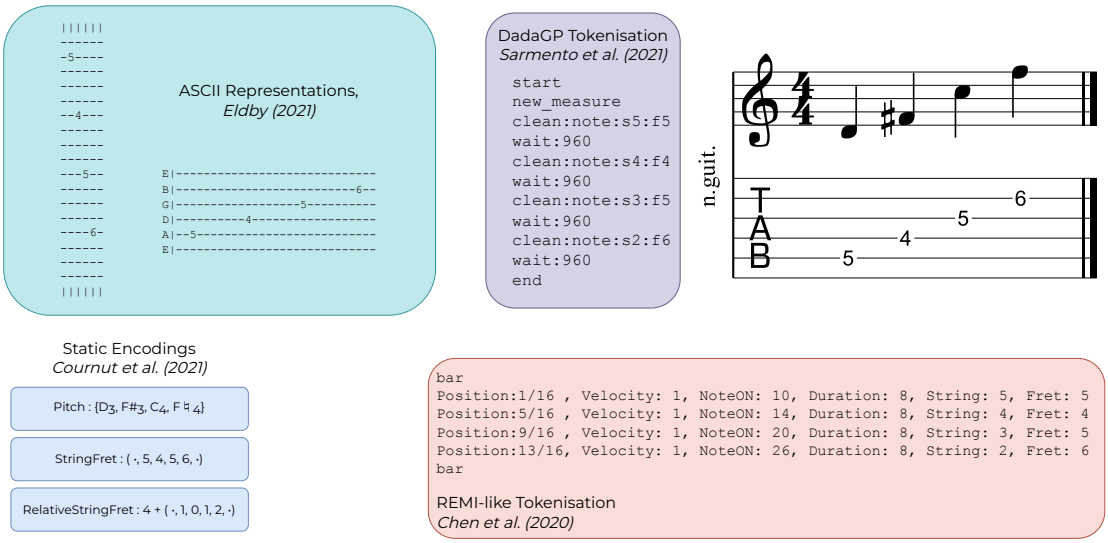


Figure 3.3: One bar of guitar tablature and its possible representations.

are from Y.-H. Chen et al. (2020) and Sarmento et al. (2021). In the former, each note is represented by an onset value, a duration, and a string-fret pair. In the latter, a note is a single token that can be preceded or followed by `wait` tokens that allow to infer the onsets and durations. Choosing a tokenisation scheme is usually a matter of balancing its efficiency (number of tokens required to represent a given musical content) and its robustness (token sequences easily reproducible by a model and usable even with missing tokens). An example of all encodings presented in this section is provided Figure 3.3.

3.1.5 Tablatures in Computational Musicology

Guitar scores and tablatures can be used as training data for generative DL models, but they can also be considered as artefacts of WPM practice and composition in computational musicology analyses. In Koozin (2011), the author shows how tablatures, because they can represent *gesture*, allow for deeper and more meaningful interpretations of guitar practice in WPM. Besides, the author shows how the guitar *affordance* – only noticeable in tablature notation – impact composition, a concept also discussed by Yim (2011). Many researchers also studied the playing styles of famous guitarists through analyses of transcriptions of their solos, with tablature notation. Ferretti (2016), for instance, analyses solos of Eric Clapton, David Gilmour, Jimi Hendrix, B.B. King, and Eddie Van Halen. Ferretti represents the solos as directed graphs and observes that the playing styles of the guitarists studied have noticeable differences in their network representations. In Das et al. (2018), the authors study solos of Clapton, Gilmour, Hendrix, and Mark Knopfler with zero and first-order Markov chains in which they also identify global trends on musical characteristics that can be linked to playing styles. Likewise, Sarmento et al. (2023b) analyse solos of Gilmour, Hendrix, Steve Vai, and Yngwie Malmsteen from the DadaGP

dataset and train a classifier that recognises guitar players from tablatures with an accuracy of 89%. In particular, they analyse the use of playing techniques by each guitarist and notice clear style tendencies. For instance, Steve Vai uses tapping techniques more often than the other guitarists, and Jimi Hendrix often plays on the E♭ minor pentatonic scale. Tablatures can also be used to study guitar practice in WPM: since tabs allow to disambiguate where to play notes and chords on the fretboard, they can be used to observe what chord positions are most used by guitar players (Cournut et al. 2021). Tablatures can also be studied in Optical Music Recognition research, like Ma et al. (2023) who aim at automatically distinguishing tablatures from numbered notation (jiǎnpǔ notation, Yang et al. (2025)).

Table 3.2: Research works on guitar tablatures and the data specifically gathered for them.

Paper	Task	Data Description	Source of Data	Format	Availability/Reproducibility	License
Dahia et al. 2004	Rhythm guitar accompaniment generation	21 2-bars rhythmic patterns for Bossa Nova guitar	Expert knowledge and literature on Bossa-Nova analysis	N/A	Exact patterns are unknown and cannot be retrieved with the information provided.	N/A
D. Tuohy et al. 2006	Arrangement of standard notation scores in tablature notation	2 movements from Western classical music	Official published scores	Unknown	Original publications for the scores are provided.	N/A
McVicar et al. 2014a; b; 2015	Guitar tablature generation	50 popular tablatures of songs from 5 "well-known guitarists"	www.gprotab.net	GuitarPro	Song titles available, but the exact tablature versions are not.	N/A
Ferretti 2016	Guitar solos modelling and analysis	Tablatures for solos of "Important Guitarists" (unknown dataset size)	AZ Guitar Tabs and Ultimate Guitar	MusicXML	Selected solos are unknown.	N/A
Ariga et al. 2017a	Tablature arrangement from polyphonic audio	Western classical guitar tablatures (unknown dataset size)	Classtab	Plain text	Selected songs are unknown.	N/A
Ariga et al. 2017b	Suggest personalised exercises to practise chords and songs	727 chord progressions from the Billboard dataset (Burgoyne et al. 2011) and possible guitar positions	Billboard dataset and expert guitarists	Plain text	The chord fingerings used to play the songs are not shared.	CC0 for the Billboard dataset.

Mistler 2017	Arrange standard notation scores into tablatures	100 tablatures of "popular" songs	Ultimate Guitar	Guitar Pro	Exact tabs are unknown.	N/A
Das et al. 2018	Analyse and classify rock guitarists from their solos	20 guitar solos from Clapton, Gilmour, Hendrix and Knopfler, each.	Ultimate Guitar	Guitar Pro	Song titles are not provided.	N/A
Y.-H. Chen et al. 2020	Generation of fingerstyle guitar tablatures	333 fingerstyle guitar tablatures	Unknown	MIDI	Song titles are not provided.	N/A
Eldby 2021	Generation of Western classical guitar tablatures	3060 Western classical guitar tablatures	Classtab	Plain text	The exact files or song titles are not provided.	Unspecified
Ma et al. 2023	Classification of tablature and numbered notation	1986 tablature images and 5856 numbered musical notation images	Unknown	Unknown	No information provided.	N/A
Vélez Vásquez et al. 2023	Estimation of guitar chords playing difficulty	200 songs from the Billboard dataset (Burgoyne et al. 2011) with playability annotations of the chord sequences	Subset of the Billboard Dataset and expert guitarists	`.txt` or `.lab`	The songs and the annotations are shared on Github, but chord fingerings are not provided.	Unspecified
Riley et al. 2024b	Automatic Music Transcription of Jazz guitar songs	4 hours of audio with professional transcriptions	Francois Leduc commercial catalogue	GuitarPro for the tablatures, audio format unspecified	Full track list is provided.	N/A

Table 3.3: Guitar tablature datasets available for research, top-half contains datasets for Automatic Music Transcription (AMT), while the bottom-half shows general purpose datasets.

Dataset	Data Description	Source	Format	Availability/Reproducibility	License
Guitarset (2018)	3 hours of hexaphonic (each guitar string is recorded individually) recordings of chord progressions with their corresponding transcriptions (30 excerpts played by six guitarists).	In-house recordings	JAMS/WAV (Humphrey et al. 2014)	https://zenodo.org/records/3371780, accessed in April 2025.	MIT
EGDB (2022)	2 hours of hexaphonic recordings of tablatures of solos, arpeggios, and comping (240 files).	In-house recordings	MIDI/WAV	https://ss12f32v.github.io/Guitar-Transcription/, accessed in April 2025.	Unspecified
GAPS (2024)	14 hours of Western classical guitar recordings and their corresponding tablatures (300 songs).	ClassClef free scores and matching Youtube videos	MusicXML/audio format unknown	https://zenodo.org/records/13962272, accessed in April 2025.	CC BY-NC-SA
DadaGP (2021)	26 181 amateur transcriptions of WPM songs.	Unknown	GuitarPro	Available for research on demand.	Unspecified
mySongBook	2 115 professionally-transcribed guitar tablatures	Arobas Music Company	GuitarPro	Parts or features of the tabs have been released, but the full dataset is unavailable.	Unspecified
AnimeTAB (2022)	412 tablatures of fingerstyle guitar arrangements of *anime* songs	"open-source scores on the Internet"	MusicXML	https://github.com/amamiya-yuuko/AnimeTAB, accessed in April 2025.	CC BY-NC

Cunha et al. 2018	342 Blues Licks of 1 or 2 bars duration.	Specialised music books	MusicXML	http://dorienherremans.com/guitar_licks_dataset, accessed in April 2025	Unspecified
Keating et al. 2024	3 698 voice-leading examples of Jazz chord progressions and the corresponding guitar fingerings	Educational book and manual transcriptions	CSV files	https://github.com/mbkeating/AIMC, accessed in April 2025	Unspecified

3.2 Assisted Guitar Composition and Tablature Generation

The objective of the TABASCO project, pursued in a part of this thesis's contributions (part III), is to assist guitar players when they compose WPM. The choice was made to focus on using tablatures for assisting composition, which includes our work in the broader field of symbolic music research. In this section, we begin by introducing music generation as a whole, before focusing on tablatures and how they have been studied in automatic arrangement systems as well as generative models. Finally, we discuss how such AI models can fit in a musician's pipeline when performing or composing.

3.2.1 Music Generation in the Audio and Symbolic Domains

Tablature generation is a subfield of symbolic music generation which is itself a subfield of music generation, a field that can considered at least a few centuries old. For instance, *musical dice games* were popular in Western Europe in the 18th century (Nierhaus 2009), the principle being that a music could be assembled from a numbered list of excerpts by rolling dice, anyone could then generate music "randomly" from the originally composed options. In general, music generation approaches can belong to three main approaches (not exclusively). Firstly, some models aim at assisting persons who are not musically trained in creating music. Musical dice games could be considered to belong to that category since they allow to generate new music, without knowing how to compose. Users still need to know how to read and play music nonetheless, since the result of the game is a score. More modern approaches that generate full audio, like Jukebox (Dhariwal et al. 2020) from lyrics and style information can also be helpful to laypersons. Recently, commercial models like SUNO[7] or Udio,[8] or research works like MusicLM (Agostinelli et al. 2023), can generate songs from a simple text prompt, allowing anyone to create music from their language (unless they are from the global South Choudhury (2023)). Secondly, some works focus on studying creativity or whether machines can be creative (Colton et al. 2012). The Dadabots duo for instance, aims at "eliminating humans from music"[9] (Windsor 2021) and develop models that generate metal music indefinitely. Chemla–Romeu-Santos et al. (2022) model and study ways of making a music generation model more creative by diverging from the training set distribution. In Barenboim et al. (2024), the authors analyse the latent space of MusicVAE (Roberts et al. 2019) and observe that some specific latent coefficient correlate with measures related to rhythm and pitch in music. Likewise, Cádiz et al. (2021) conduct case studies on two models and show that, through the ability to extrapolate from what is available in the training data, those models could be considered creative. Finally, some generative music models focus on assisting artists. This category is often interwoven with the previous one, where models are used in a co-creative fashion. The co-creativity can be in real-time, like with systems based

[7] https://suno.com/home, accessed in June 2025.
[8] https://www.udio.com/, accessed in June 2025.
[9] https://www.youtube.com/@dadabots_, accessed in June 2025.

on the OMax paradigm (Assayag et al. 2006) such as the DYCI2 agents (Nika et al. 2017). Artists can also rely upon AI models for ideation, this is for instance how the Algomus team and Sébastien Gulluni composed a song for the 2021 AI song contest (Déguernel et al. 2022), an approach followed by most of the contestants of the 2020 edition as well (C.-Z. A. Huang et al. 2020). More models from all categories can be found in surveys on generative music (Briot et al. 2019; Herremans et al. 2017; Ji et al. 2023; Le et al. 2024). Our work belongs primarily to the last category since our objective is to assist guitarists in composing with tablatures. Because we use such a notation system, our work also belongs to the category of *symbolic music* models. Scores are commonly used as a means of communication between musicians in Western music, classical music composers being used to compose directly with scores (Leech-Wilkinson 2012), while popular musicians will learn to play songs from their recordings, but also through scores and tablatures (L. Green 2002). Using models that generate scores is thus a way to assist composers without affecting their workflow. We pursue this section with a description of common tasks in symbolic music generation, before discussing tablature generation research.

Tasks in Symbolic Music Generation Symbolic music generation can be tackled in many ways, a common approach is to implement models that realise *arrangements*. An arrangement is defined in the Harvard Music Dictionary (Randel 2003) as:

Definition (Arrangement) *The adaptation of a composition for a medium different from that for which it was originally written, so made that the musical substance remains essentially unchanged.* 📖

A specific type of arrangement is for instance orchestration, i.e. adapting a score (piano for instance) to an orchestra, and it can be tackled by symbolic generation models (Maccarini 2024). Arrangements can also be used to change the difficulty of an existing score, like Gover et al. (2022) that propose a model to automatically simplify a piano score for beginners. Arrangement is a specific type of *conditional generation*, where conditioning data can be anything that drives the model towards a desired output (Briot et al. 2019). The conditioning can for example be a prompt that needs to be continued, the Music Transformer (C.-Z. A. Huang et al. 2019) can for instance generate continuations of piano MIDI files while maintaining consistency and being more varied than the vanilla transformer, thanks to a relative attention mechanism (see chapter 9 for more explanations of this mechanism). Conditioning values can also be used to inform the model on desired characteristics for the generated output. Ens et al. (2020) present a model that can generate MIDI files with precise controls over the instruments present, as well as note density, S.-L. Wu et al. (2023) even allow rhythm and polyphony controls to be time-varying. Another conditional generation approach often studied to assist composers is *inpainting* (term derived from image generation research) which consists in generating musical content based on past *and* future context (K. Chen et al. 2020; Hadjeres et al. 2021). A derived approach

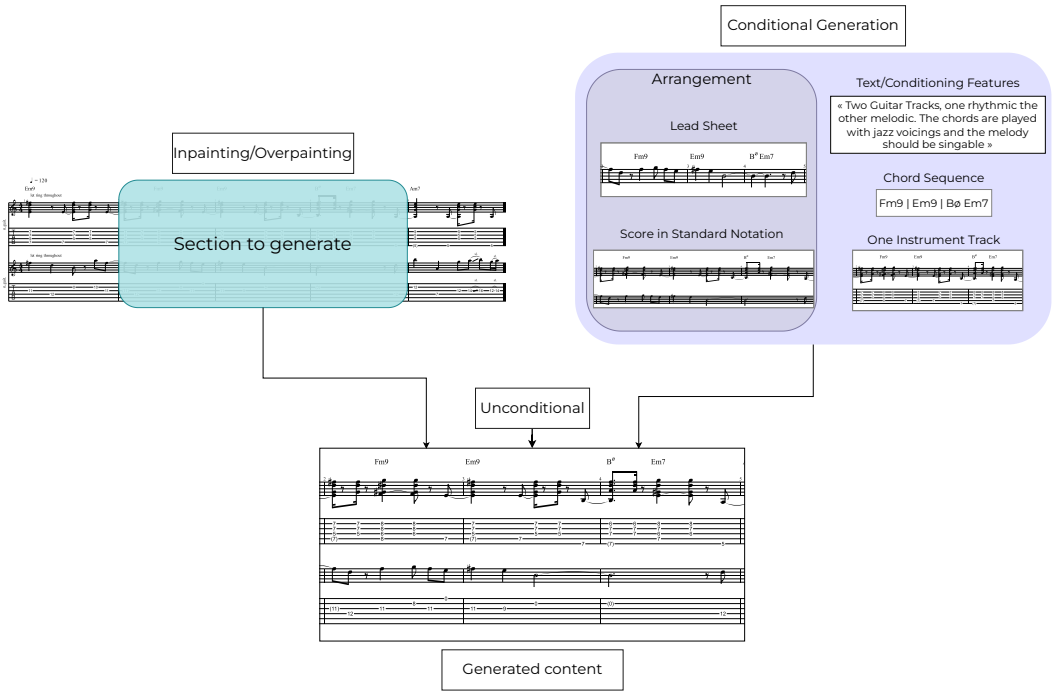


Figure 3.4: Summary of the different generation types for symbolic music. Only a few selected examples are given for the conditional generation but the conditioning could be in almost any shape or form.

is *overpainting* (Row et al. 2023), which complements inpainting with the constraint of keeping the original content recognisable within the newly generated one.

Conversely to conditional generation, generation can be *unconditional* or *free*. In that situation there is little to no control on what the model generates. Many models will support both free and conditional generation by design. For instance, transformer-based models can generate either the continuation of a prompt, or generate an entire sequence freely, as illustrated by the Pop Music Transformer (Y.-S. Huang et al. 2020).

The presented tasks are summarised Figure 3.4, a more extensive presentation of symbolic music generation models can be found in Ji et al. (2023) and Le et al. (2024).

3.2.2 Tablatures in Symbolic Music Generation

We previously introduced symbolic music generation and the different tasks it encompasses. Tablature, as a specific type of symbolic notation, can also be generated in many different ways. Generating tablatures is specific in the way it can include complex playing techniques, and the string-fret positions generated are expected to be playable. Dahia et al. (2004) presents a rule-based and case-based reasoning method for choosing rhythmic bossa nova guitar patterns. As is often the case, rules are obtained from domain experts and allow to automatically filter patterns from a selected database. A k-nearest neighbours algorithm is then applied for case-based reasoning, selecting the best match on a set of musical features. Guitar tabs can also be generated using integer programming.

Cunha et al. (2018) present a method for generating blues guitar solos, by defining transition costs and classifying existing licks before finding an optimal path between them. Machine learning methods can also be used, like in McVicar et al. (2014a), b, 2015. In those papers, the authors use a variety of data-driven methods to generate guitar tablatures. Notes and their position on the fretboard are sampled from a training set, using *n-gram modelling* and a random-walk to obtain the final tablatures. In McVicar et al. (2014a), the random walk generation using those n-grams is improved by adding an additional component in the probability computation that favours chord tones, to ensure that the solos generated match the underlying chord progression. Fretting-hand techniques are also added in a post-processing step to make the final tablature more idiomatic.

Other tablature generation models are based on the Transformer-XL model introduced in Dai et al. (2019) and applied to music in Y.-S. Huang et al. (2020). Compared to a vanilla transformer, the Transformer-XL model uses a recurrence mechanism which consists in splitting the input sequence into segments, and the latent value obtained from processing a segment is reused when processing the next segment. This modification allows to maintain consistency over longer token sequences. The model also uses relative positional encoding, which informs the model of the relative distances between each token, instead of their position in the sequence, ensuring better consistency between tokens. This network is for example used in Y.-H. Chen et al. (2020) to generate fingerstyle guitar tablatures. They implement "groove" controls that inform the model on the expected rhythmic density when generating tablatures. The Transformer-XL is also the model used in Sarmento et al. (2021) and all papers that built upon the DadaGP model and dataset (Loth et al. 2023; Sarmento et al. 2023a; b). In the original DadaGP paper, Sarmento et al. prove that the model can be trained on the DadaGP dataset to generate realistic guitar tablatures, as well as other instrument tracks. In Sarmento et al. (2023a), they experiment with different prompting strategies to assess the controllability of the Transformer-XL model, and in particular control which instruments should be included in the generation. Because those controls did not successfully remove unwanted instruments, in Loth et al. (2023) and Sarmento et al. (2023b), they retrain the model on progressive metal tablatures or tablatures of famous guitarists, to generate new tablatures similar to their training set. Playing techniques are usually included in those models' generations, either focusing on specific techniques like for *fingerstyle* guitar (Y.-H. Chen et al. 2020), or techniques common to WPM guitar playing (Sarmento et al. 2021). Because those techniques are included in the tokens used by those models, they are generated automatically and do not need to be added in a post-processing step.

3.2.3 Automatic Tablature Arrangement

Tablature, like other kind of musical notation, can be the objective of arrangement models. Most research focuses on converting a score in standard notation (or a MIDI file) into a *playable* tablature, but there are also studies on arranging a tablature from one style to

another.

Arrangement of existing Western classical music scores to playable guitar tablatures was studied in D. Tuohy et al. (2006), 2005 using *genetic algorithms* (Eiben et al. 2010). In such algorithms, a population of individuals – here, *valid tablatures* – is evaluated by a fitness function – a measure of the tablature's playability – until a satisfactory score is attained. Individuals are used to create a new generation by optionally mixing genes – *parts of the tablature* – and possibly undergoing a mutation – *randomly changing the fingering of a chord*. A similar optimisation-based approach is used in Sakai et al. (2024) where, given a lead-sheet and a set of chords, a fingerstyle guitar arrangement in tablature format is generated. Required values for initial and transition probabilities are defined through expert knowledge. Conversely, Mistler (2017) uses a DL-based approach. Mistler trains an LSTM network to obtain a possible tablature from a new unseen score in standard notation, but all musical constraints are learned from existing data. Using a tablature as input, Zhuang (2023) adapts the CycleGAN (Zhu et al. 2017) architecture from image processing to music for converting classical guitar tabs to Blues tabs. In Kaliakatsos-Papakostas et al. (2022), the authors propose to use a Convolutional Neural Network to arrange a MIDI file into a guitar tablature. To do so, they process time-frames sequentially and use previously generated tablature frames as additional input to improve the playability of the generated tablatures. Other advanced DL architectures from NLP can be used, like in Edwards et al. (2024) where a BERT model is used to automatically find a possible tablature to play a MIDI file. Finally, arrangement systems can even generate playable tablatures directly from audio, as studied in Ariga et al. (2017a). Though related to AMT, the difference with this approach is that the original audio is not necessarily a guitar recording.

3.2.4 Tablature Generation Models for Co-Creativity

We discussed in section 1.2.2 that tablatures are usually not at the heart of the composition process. Could new AI methods change that fact? The previous section introduced research works that generates guitar tablatures in various fashions, but most systems propose free generation, i.e. generating content with little to no control for the user. Even models with more control usually generate full songs and authors do not discuss how their tool would fit in a musician's workflow. In those situations, songs are *artefacts* that are generated or used for training, and how those models could be used for co-creation is little discussed (Jordanous 2016). A few exceptions exist and explicitly present tablature generation as a co-creativity task. Adkins et al. (2023), for instance, aims at generating loopable content for live coding performances. While the proposed model was not tested in live contexts, generated loops were evaluated on their internal consistency and loop seamlessness through an online survey, and the generated loops were rated fairly close to human-composed ones. The approach is further extended in Cui et al. (2024) where the authors try adding *emotion control* to the generated loops. Even though the user study does not strongly support the efficiency of this control – attributing emotions to music

is dependent on context and personal experience (Schubert 2013) – its addition shows a desire to get closer to the way musicians can imagine compositions (Déguernel et al. 2023).

Another interesting example is how Loth et al. (2023) used the DadaGP dataset and the corresponding transformer model to compose a progressive metal song from scratch. The model generated suggestions in tablature format, that were selected, practised on guitar, and recorded by the authors to obtain a fully produced progressive metal song. In Loth et al. (2023), they detail the process of making the model and song and reflect upon this interaction with an AI model. For instance, they observe that the model can successfully maintain consistency throughout the song, by repeating an early riff at the end of a new section. However, some parts of the generated tablatures were too hard to play as is (even by seasoned guitarists) and had to be adapted to be playable. Finally, it is also worth noting that the authors used the same approach to record two fully-fledged songs for the *AI song contest* in 2023 and 2024.[10]

3.3 Computer Assisted Guitar Education

A part of this thesis focuses on assisting guitarists in their learning process. The corresponding contributions are discussed in part II, and this section presents existing work in computer assisted guitar education. MIR research on guitar education includes two main fields, score performance difficulty estimation and song recommendation for learning. Both fields are driven by the amount of guitar educational content that can be found online, as estimating the difficulty of a song or recommending appropriate tablatures can help students navigate those resources (Figure 3.5).

3.3.1 Music Difficulty Estimation

Difficulty is a subjective aspect in essence, as what students find difficult in a song will depend on their background and past experience. Regardless of that subjectivity, music teachers aim at recommending content that favour a smooth and rewarding learning process (McQueen et al. 2018; Nielsen et al. 2023). Difficulty estimation tools could facilitate this recommendation process by automatically processing large song catalogues, and be useful to both teachers and self-taught learners. This task has been little studied in the MIR community, maybe partly because of a lack of appropriate datasets, especially for instruments other than piano.

The earliest work might be Sébastien et al. (2012), in which the authors introduce seven criteria, like playing speed or hand displacement, to automatically evaluate the difficulty of piano scores. The methods attains over 50% agreement with three piano teachers when rating songs "representative of a classical piano cursus in a French music conservatory",

[10] https://www.aisongcontest.com/participants-2023/hel9000 (2023), https://www.aisongcontest.com/participants-2024/hel9000 (2024). Links accessed in April 2025.

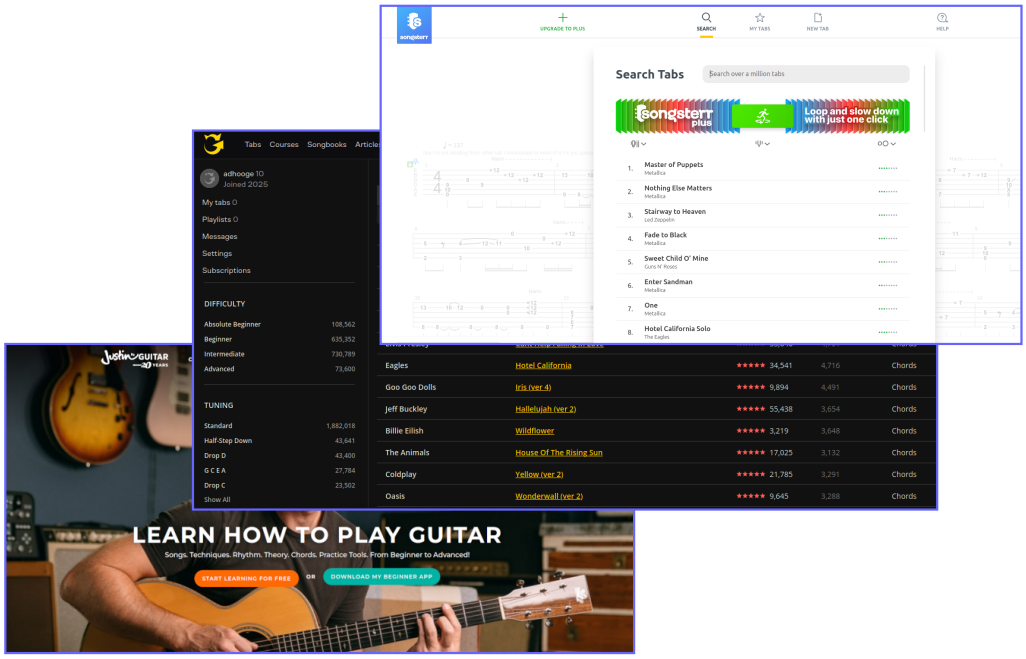


Figure 3.5: Screenshots of three websites with guitar-related resources. From left to right: JustinGuitar (Educational website with 7 complete courses and over 700 song tutorials), Ultimate Guitar (over 2 million tabs), and Songsterr (over a million tabs).

on a difficulty scale with three levels. A similar study is done in Chiu et al. (2012) with a partly different set of eight features on a dataset of over 300 MIDI files from online piano catalogues.[11] The final regression model reaches a coefficient of determination $R^2 \simeq .4$. In Ramoneda et al. (2022), 2024a, the authors introduce two new datasets for piano difficulty estimation: Béla Bartók's *Mikrokosmos*, and difficulty-classified scores of Henle's catalogue.[12] Ramoneda et al. (2022) present a DeepGRU model that attain over 60% accuracy when classifying scores in three difficulty levels. In Ramoneda et al. (2024a), the balanced accuracy of the model drops to 39.5% on the new dataset, but the classification is now between 9 difficulty levels. Ramoneda et al. (2023) further show that their deep learning based approach can also work directly from score images, reaching 40.3 % balanced accuracy with a new GPT-based model trained and evaluated on multiple piano datasets combined.

When it comes to guitar, difficulty can be taken into account during tablature generation. Ariga et al. (2017a) defines an arrangement model that generates a tablature from an audio recording, with explicit difficulty controls that control movement speed and chord fingering complexity. In Vélez Vásquez et al. (2023), the authors focus on analysing the difficulty of playing chords in accompaniment guitar tracks. To do so, they interviewed

[11] https://www.pianostreet.com/ and https://www.8notes.com/, accessed in June 2025.
[12] https://www.henle.de/en/, accessed in June 2025.

guitar teachers and built a *rubric* of criteria for rating the difficulty of guitar accompaniment songs. Criteria like *chord fingering difficulty* or *strumming complexity*[13] are then used to train different AI models to reproduce ratings produced by experts on a public dataset of WPM songs. Apart from this last work, we could not find additional research on difficulty estimation for guitar in WPM, highlighting a gap in the state-of-the-art.

3.3.2 Games and AI-Models for Learning and Teaching Guitar

Assisted guitar pedagogy research in the literature focuses either on recommending learners with songs, or helping beginners in their practice. Song recommendation is a vast research field in itself. Common approaches include collaborative filtering, where similar users are grouped together, or content-based filtering based on manually defined features or automatic representations extracted by DL models (Paul et al. 2020; Zeng et al. 2024). However, recommending songs to learners (instead of listeners) require taking specific musical aspects into account, like the difficulty of the songs or the techniques they involve for instance. Because new skills need to be learned gradually, recommending songs to learners is more of a digital pedagogy task than a typical music recommendation one. Indeed, learning specific chords or playing techniques can be prerequisites to some songs, and the learning order can therefore not be random. Ideas can be drawn from research on e-learning for instance (Hafsa et al. 2022), but they will not be discussed further in this thesis. Approaches like collaborative filtering can still be applied for recommending new songs to learn, but they require access to large amounts of students data that are hard to come by, except for large music learning platforms like Yousician[14] (Müllerschön et al. 2025).

Directly recommendings songs is however not the only way to assist guitarists in learning and practising their instrument. For instance, because of the large quantity of tablatures available online, any algorithm helping to navigate the vast amount of data can be useful. This is studied in Macrae et al. (2011), where the authors propose a way to automatically mine tablatures on the Internet and assess their quality to provide learners with appropriate resources for learning a desired song. A similar system is studied in Barthet et al. (2011), where video tutorials are selected from YouTube with their matching chord sequence and tablature. B. Wang et al. (2021) proposes a comparable tool, where existing educational videos are augmented with sound processing tools to check that the user plays the song correctly, similarly to what can be seen is commercial applications like Yousician (Yue et al. 2025).

Another way to help beginner guitarists is to accompany them in learning their first chords. This is precisely the goal of Ariga et al. (2017b) that proposes a complete software to guide beginners learning guitar chords while also suggesting songs that will put the

[13] Criteria called "right-hand complexity" in the original paper, renamed to account for left-handed guitarists.

[14] https://yousician.com/, accessed in November 2025.

most common chords in context. Focusing on a similar task, Wortman et al. (2021) dynamically find chord fingerings based on a variety of constraints, including maximum finger stretches, to adapt to any guitar player. Another music performance assistance tool is studied in Keating et al. (2024), that introduces a method to suggest chord fingerings that comply with voice-leading constraints for playing jazz songs. While not tested in actual performance and learning contexts, the authors assessed the validity of the suggestions and observed that it might indeed be a good starting point for beginners.

Video games can also help learning music. While playing *Guitar Hero* is very different from actual guitar playing (Arsenault 2008), *Rocksmith* is played with an actual electric guitar or bass plugged to the computer or console. Interfaces of both games along with the Guitar Hero controller are shown Figure 3.6. With such a game, players can really learn to play guitar or might be motivated to practice by the *fun* nature of the task, as compared to taking lessons (Havre et al. 2019). Games can also create communities on a variety of platforms, which has an impact on users engagement and improved learners investment in Rocksmith (Rodriguez et al. 2020). Other *serious games* were developed to help learning guitar: Skreinig et al. (2022), 2023 use Augmented Reality to help guitar players visualise chord shapes on the fretboard, thus helping beginners remembering how to play the chords they learned.

Figure 3.6: Top: The Guitar Hero game interface (screenshot taken from Arsenault 2008) and the Guitar Hero controller (picture from ICGAMES); Bottom: The Rocksmith game interface (screenshot by firefang9212). In Guitar Hero, each column has coloured dots falling down that corresponds to one of the five buttons on the guitar controller, the player then have to synchronously tap on the "pick" with their other hand. In Rocksmith, the display resembles a tablature that would have been flipped upside down (the lowest string is on top), each string is colour-coded.

Part II

Assisted Guitar Pedagogy through Automated Difficulty Estimation

4

Difficulty Adjusted Guitar Song Suggestion

"I couldn't play normally […] so I just started tapping because I couldn't do anything else well enough to get through the things I wanted to learn."

Sarah Longfield, in Larson (2018)

Contents

Guitar learners have access to millions of tablatures or songbooks online, but have no direct way of knowing what their current level allows them to practise without facing discouraging difficulty. In this chapter, we present our research on recommending guitar learners with new songs to practise, based on their current level. To do so, we propose a simple algorithmic approach to estimate the level of learners on different musical dimensions like rhythm patterns and chord complexity, and suggest songs with similar difficulty levels. This work was made in collaboration with the Guitar Social Club (GSC) start-up[1] on their proprietary dataset of WPM accompaniment guitar songs, rated on several difficulty criteria by our partnering guitar teacher. The research presented in this chapter has been published in the following papers:

"Suggestions Pédagogiques Personnalisées pour la Guitare"
Alexandre D'Hooge, Mathieu Giraud, Yohann Abbou, Gilles Guillemain.
Actes des Journées d'Informatique Musicale (JIM), 2024. (D'Hooge et al. 2024b)

"What Song Now? Personalized Rhythm Guitar Learning in Western Popular Music"
Zakaria Hassein-Bey, Yohann Abbou, Alexandre D'Hooge, Mathieu Giraud, Gilles Guillemain, Aurélien Jeanneau.
Proc. of the 26th Int. Society for Music Information Retrieval Conf. (ISMIR), 2025.
(Hassein-Bey et al. 2025)

[1] https://guitarsocialclub.com/, accessed in June 2025.

4.1 What makes learning new songs difficult?

Students will find new songs difficult to learn for different reasons, depending on their past experience. In this section, we propose a formulation of difficulty for accompaniment guitar parts, based on the experience of a collaborating WPM guitar teacher. We also present a way of sub-dividing songs, to imitate how guitar teachers might adapt their lessons to their students.

4.1.1 Difficulty Criteria and Exercises

The skill level of a learner is not well-defined when using a single value. A guitarist may, for instance, know many chords including complex ones, but struggle with using them in context with advanced rhythmic patterns. Conversely, another guitarist may only know basic chords but switch between them effortlessly, even while playing complex rhythms. Vélez Vásquez et al. (2023) acknowledge those possibilities by identifying several difficulty criteria: the intrinsic complexity of a chord based on its fingering, the rarity of a chord, how the chord is strummed and, within a chord sequence, the repetitiveness and speed of chord transitions. They obtained those criteria through discussions with guitar teachers and "expert guitarists", and used them in a later step to annotate a subset of the Billboard dataset (Burgoyne et al. 2011).

The criteria we propose in this chapter are also based on discussions with a guitar teacher but even though many criteria are similar, some differences arose, especially when it comes to the relative importance of each aspect. It is also worth noting that our work focuses on recommending appropriate songs to learners, while Vélez Vásquez et al. (2023) study the playability of guitar songs regardless of a learner's level. In the rest of this section, we present those criteria as they are defined and used by our partnering guitar teacher to annotate the difficulty of the corpus used in this chapter.

Chord Complexity (Table 4.1a) The difficulty of playing chords can be estimated using several indicators: the number of fingers required, the fret span, the use of open or muted strings, and the overall hand positioning. Besides, transitioning from chords can also be complex depending on which fingers are used in each chord. All those aspects are taken into account by the guitar teacher when rating chord complexity ℓ_c. Example of typical chords expected in each level are shared Table 4.1a.

Playing Speed (Table 4.1b) Considering *tempo*, the amount of Beats Per Minute (BPM) alone is not sufficient to characterise tempo complexity. For instance, a song at 70 Beats Per Minute (BPM) where the guitarist primarily plays sixteenth notes can present a similar level of rhythmic challenge as a song at 140 BPM with mostly eighth notes. In both cases, the effective playing rate is comparable, even if the perceived speed is different for the player and the listener (Elowsson et al. 2013). To better capture this aspect, we consider the number of Notes Per Minute (NPM). In the previously mentioned examples, both

would result in approximately 280 NPM. The number of NPM is converted to a grade ℓ_T using the boundaries defined Table 4.1b.

Rhythm Difficulty (Table 4.1c) *Rhythmic patterns* also play a significant role in determining difficulty, in particular their regularity. Interestingly, rhythmic patterns were assigned a weight of 0 in Vélez Vásquez et al. (2023) for difficulty estimation. While they state that rhythmic difficulty contributed little extra information and could be overlooked, we believe it can be due to a bias of their dataset. For this reason, we include a rhythm difficulty criterion ℓ_R to ensure the robustness of our approach, and patterns are rated according to the categories defined Table 4.1c.

Global Rating (Table 4.1d) In addition to the previous criteria, each song is rated *as a whole* to reflect the overall difficulty as perceived by the guitar teacher, especially when multiple criteria interact (Table 4.1d). This global rating ℓ_G summarises multiple difficulty dimensions, something that could not be formalised simply with the criteria presented before. Adding explicit criteria that could replace this global difficulty rating is an area of improvement considered for future work.

Amount of Exercises When working with a teacher, students are often given specific *exercises* to target challenging aspects of a song. For instance, one should probably start by learning the chords used in the song and practise transitioning from one to the next. If the song uses a peculiar rhythm, it might also require dedicated exercises to gradually reach the desired complexity. As a result, the difficulty of learning a new song also depends on the number and complexity of new exercises it introduces, beyond those the student has already mastered. The amount of exercises to learn is denoted by ℓ_E and exercises used in the corpus are presented in section 4.2. For our recommendation system, we assume that an exercise is either known or not, as a binary value. We acknowledge the limitations of this approach and consider implementing automatic evaluation systems in the future (Eremenko et al. 2020) to verify to what extent the exercises are mastered. Currently, an exercise can be declared as cleared by a student, and we trust that they would change their declaration if they realise that they still need to practise the exercise more.

Additional Criteria Finally, other discrete criteria can be implemented, for example to favour songs within a given style or within user preferences. Such criteria can be defined simply as:

$$C_k(g, v) = \begin{cases} 0 & \text{when the criteria is met,} \\ 1 & \text{otherwise.} \end{cases}$$

Table 4.1: Criteria for assessing song/part/version difficulty in guitar playing. From top to bottom and from left to right: (a) Chords. (b) Notes Per Minute (NPM). (c) Rhythmic Patterns. (d) Global difficulty. Learning times may vary considerably among students.

(a)

ℓ_C	**Chord complexity**	**Chord examples**
0-2	Basic open chords, simple voicings, few fingers	Em, E, Am, A7sus4
3-4	Open chords, light inversions, more open positions	C, D7, Gsus4, Fadd9
5-6	Common barre chords, moderate stretches	F, Dmin7, G9, Bbadd6
7-8	More barre positions, less intuitive grips	EM9, F#m, C#m/G#, Eb
9-10	Advanced voicings, large stretches, full fretboard usage	Fm7, Abm, C#/G, Esus4/B

(b)

ℓ_T	**Tempo Complexity**
0-2	Slow (60-200 NPM)
3-4	Medium (200-400)
5-6	Fast (400-600)
7-8	High-speed (600-800)
9-10	Extreme (>800)

(c)

ℓ_R	**Rhythmic Patterns**
0-2	Mostly half and quarter notes
3-4	Mostly eights, some sixteenths
5-6	Sixteenths with regular patterns
7-8	16th with irregular patterns, polyrhythms
9-10	Elaborated polyrhythms, odd-time signatures

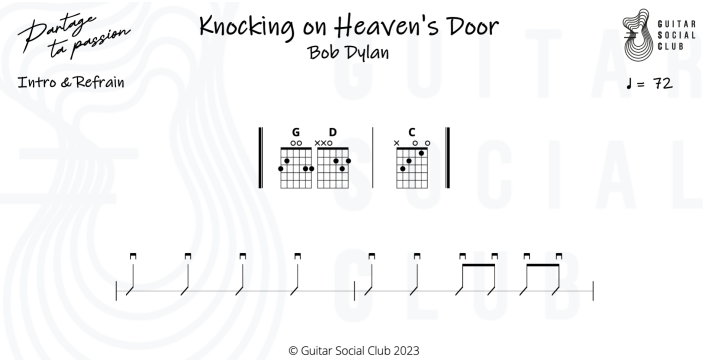
(d)

ℓ_G	**Typical time**	**Global Difficulty**
0-2	T0	Absolute beginner: basic open chords, simple strumming, slow tempo
3-4	T0 + 6 m.	Beginner: basic open chords, simple strumming, slow tempo
5-6	T0 + 3 y.	Intermediate: barre chords, basic rhythm variations, moderate tempo
7-8	T0 + 5 y.	Confirmed: complex transitions, intricate rhythms, faster playability
9-10	T0 + 8 y.	Advanced: mastery of multiple styles, technical proficiency

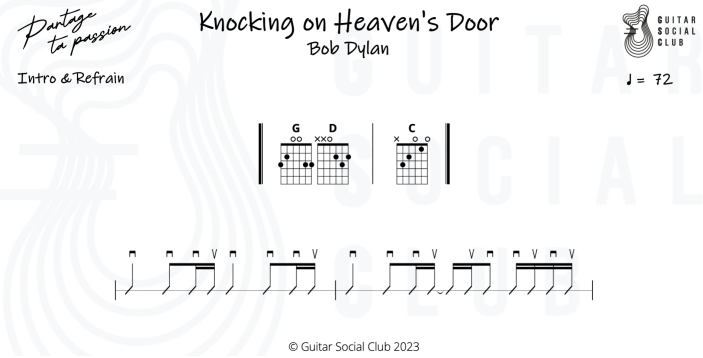
4.1.2 Songs, Parts, and Versions

Even with several criteria, considering the difficulty of a song as a fixed set of measures is not realistic, because difficulty is not uniform throughout a piece. To address this issue, we propose splitting *songs* into their *parts*, such as Intro, Verse, Chorus, Bridge, or Outro. The segmentation of a song into parts is both musical and technical: since parts are often homogeneous from a guitaristic perspective, it is possible to uniformly rate their difficulty and pedagogical interest for a given person. R.E.M.'s *Losing my Religion*, for example, has a Bridge that is more technical and challenging than the song's verses because of the fast-paced chord changes (Table 4.2). Indeed, the Verse is mostly composed of the simple sequence of chords A minor-E minor, making it accessible to a guitar beginner. The Bridge, however, contains the chords G, F, D minor, and A minor, and transitions between them. The F chord is particularly complex for beginners because it has to be played with a *barré*. This technique requires pressing down multiple strings with a single finger (usually the index) and is notoriously difficult to beginners.[2] However, what does it truly mean to "play *Losing my Religion*"? Some advanced players may aim to master the full tablature as performed by the band, while for other guitarists, strumming the chords of the verse with

[2] https://www.youtube.com/shorts/2MhIUwz2-S8, accessed in June 2025.



(a) Version (d): simplified rhythm (SR).



(b) Version (e): original rhythm (OR).

Figure 4.1: Chords and rhythm patterns for *Knocking on Heaven's Doors* in versions **d** (top) and **e** (bottom). Screenshots from the GSC application.

a simplified rhythm to be able to sing along can already be a meaningful achievement. With that observation, we propose that parts of a song could have multiple *versions*, each with possible simplifications that reflect different ways of playing the original part. This idea aims at reproducing the way a guitar teacher might prepare an easier arrangement of a song for their student to practise at their current level. As illustrated Table 4.2 for *Knocking on Heaven's Doors*, we define 5 different possible versions. Versions **a**/**b**/**c** are introductory arrangements where the chords are played on whole, half or quarter notes respectively. Version **d** is close to the original song but features a Simplified Rhythm (SR), defined by the guitar teacher to reduce the complexity of the most challenging patterns. Version **e** presents the Original Rhythm (OR), exactly reproducing the version performed by the original band. Example of those last two versions are presented Figure 4.1.

To summarise, we consider *songs*, that are split into *parts*, and each part can exist in multiple *versions*. Different versions of the same part can have a modified and simplified rhythm, but also other changes like different chord fingerings. One can see from Table 4.2 that not all songs exist in all 5 possible versions. The first reason for that is purely practical: each version of a part of a song is equivalent to multiple videos in the GSC app, and recording all possible videos was too time-consuming. Given this limitation, songs that are most likely to be played by beginners (based on the teacher's experience) exist in all versions. However, complex songs like Metallica's *Fade to Black* would be hard to simplify

and are reserved to more advanced players. The Intro part, for instance, is based on fast arpeggios that would be hard to simplify in a meaningful way (i.e. the song might no longer be recognisable).

Table 4.2: Ratings for selected versions and parts from three songs. Versions are possible simplifications of a part, the original version being (e) (Original Rhythm, OR), down to (a) where all chords are played with whole notes. This is specified for instance by "𝅝/72" which means that this version is played with whole notes (𝅝) at 72 BPM.

1. Knocking on Heaven's Door (Bob Dylan, 1973)

	Version a						Version b						Version c					
			ℓ_C	ℓ_R	ℓ_T	ℓ_G			ℓ_C	ℓ_R	ℓ_T	ℓ_G			ℓ_C	ℓ_R	ℓ_T	ℓ_G
Intro/Chorus	1.1.a	𝅝 / 120	1	0	0	0	1.1.b	𝅗𝅥 / 90	1	1	1	0	1.1.c	♩ / 120	1	1	1	1
Verse	2.2.a	𝅝 / 120	3	0	0	1	2.2.b	𝅗𝅥 / 120	3	1	1	2	2.2.c	♩ / 120	3	1	1	2
Full Song	1.a	𝅝 / 72	3	0	0	1	1.b	𝅗𝅥 / 72	3	1	1	2	1.c	♩ / 72	3	1	1	2

1. Knocking on Heaven's Door (Bob Dylan, 1973) (Continued)

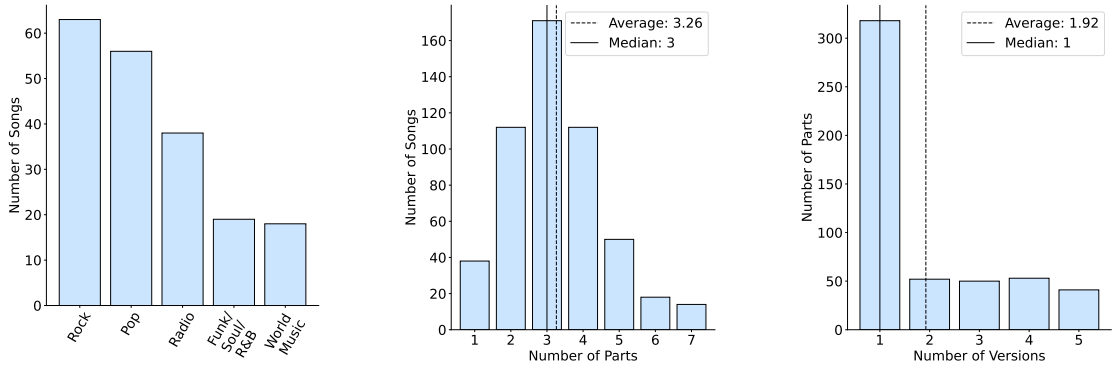
	Version d (SR)						Version e (OR)					
			ℓ_C	ℓ_R	ℓ_T	ℓ_G			ℓ_C	ℓ_R	ℓ_T	ℓ_G
Intro / Chorus	1.1.d	♪ / 72	1	2	2	1	1.1.e	♪ / 72	1	2	2	1
Verse	1.2.d	♪ / 72	3	2	2	2	1.2.e	♪ / 72	3	2	2	3
Full Song	1.d	♪ / 72	3	2	2	2	1.e	♪ / 72	3	2	2	3

65. Losing my Religion (R.E.M., 1991)

	Version c						Version d (SR)						Version e (OR)					
			ℓ_C	ℓ_R	ℓ_T	ℓ_G			ℓ_C	ℓ_R	ℓ_T	ℓ_G			ℓ_C	ℓ_R	ℓ_T	ℓ_G
Intro/Chorus	65.1.c	♩/ 120	4	1	1	3	65.1.d	𝅘𝅥𝅮/ 90	4	2	3	3	65.1.e	𝅘𝅥𝅮/ 125	4	2	4	4
Verse	65.2.c	♩/ 120	2	1	1	2	65.2.d	𝅘𝅥𝅮/ 90	2	2	3	2	65.2.e	𝅘𝅥𝅮/ 125	4	2	1	2
Bridge	65.3.c	♩/ 120	4	1	1	2	65.3.d	𝅘𝅥𝅮/ 90	4	2	1	2	65.3.e	𝅘𝅥𝅮/ 125	4	2	1	2
Full Song	65.c	♩/ 125	4	1	1	3	65.d	𝅘𝅥𝅮/ 125	4	2	3	3	65.e	𝅘𝅥𝅮/ 125	4	2	4	4

55. Fade To Black (Metallica, 1984)

	Version e (OR)					
			ℓ_C	ℓ_R	ℓ_T	ℓ_G
Intro	55.1.e	𝅘𝅥𝅮/ 58	2	3	3	4
Bridge 0	55.2.e	𝅘𝅥𝅮/ 90	9	3	3	8
Verse	55.3.e	♩/ 120	3	3	3	5
Chorus	55.4.e	𝅘𝅥𝅯/ 58	3	3	6	6
Bridge 1	55.5.e	𝅘𝅥𝅯/ 58	3	3	7	7
Bridge 2	55.6.e	𝅘𝅥𝅯/ 58	3	3	7	6
Outro	55.7.e	SR / 58	4	4	5	5
Full Song	55.e	OR / 58	9	4	7	8



(a) (b) (c)

Figure 4.2: From left to right: Number of songs per styles; distribution of the number of parts per songs; distribution of the number of versions per songs.

4.2 Difficulty-Annotated Corpus

The corpus, assembled for the GSC application, contains 195 songs. These songs come from the WPM guitar accompaniment repertoire, with the list curated based on the experience of our collaborating guitar teacher, who selected pieces frequently requested by students of various ages, genders, and musical preferences. The songs are categorised into five musical styles (Figure 4.2a): Rock (32%), Pop (29%), Funk/Soul/R&B (10%), World Music (9%), and Radio (20%).[3] Expanding the corpus and increasing the representation of under-represented styles is an area for future improvement. The songs contain between 1 and 7 parts, with most having 2 to 4, totalling 514 parts (Figure 4.2b). The parts have in average 1.9 versions, totalling 989 versions (Figure 4.2c). Finally, in the commercial application, the corpus includes 3248 *exercises* accompanied by dedicated videos. These exercises focus on practising chord transitions at varying speeds. Each exercise was manually linked to specific versions/parts/songs of the dataset. In the GSC app, each part's version corresponds to multiple practice videos and each video is accompanied by a simple music sheet reminding the chord fingerings and the rhythm for strumming (Figure 4.1). Those elements are proprietary, as well as the difficulty-annotated songs, but a subset of the annotations is released under an OdBL license[4] to foster research in difficulty-informed song recommendation.

The difficulty annotations were manually realised by our collaborating guitar teacher and are checked against Table 4.1 to limit inconsistencies. Table 4.2 shows the ratings assigned by the guitar teacher for three songs of increasing difficulty, and Figure 4.3 the distribution of ratings in the corpus for all versions **a, b, c, d, e**. Except for rhythm-related criteria, where moderate and slow values predominate, the distribution of the ratings is mostly centred around intermediate values. This observation suggests that the corpus is

[3] Radio music includes artists or songs that do not clearly belong in other styles but still are popular music of some sort. Examples of "Radio" songs are *Yellow* by Coldplay, or *Englishman in New York* by Sting.

[4] Data available at https://gitlab.com/algomus.fr/guitor/, accessed in June 2025.

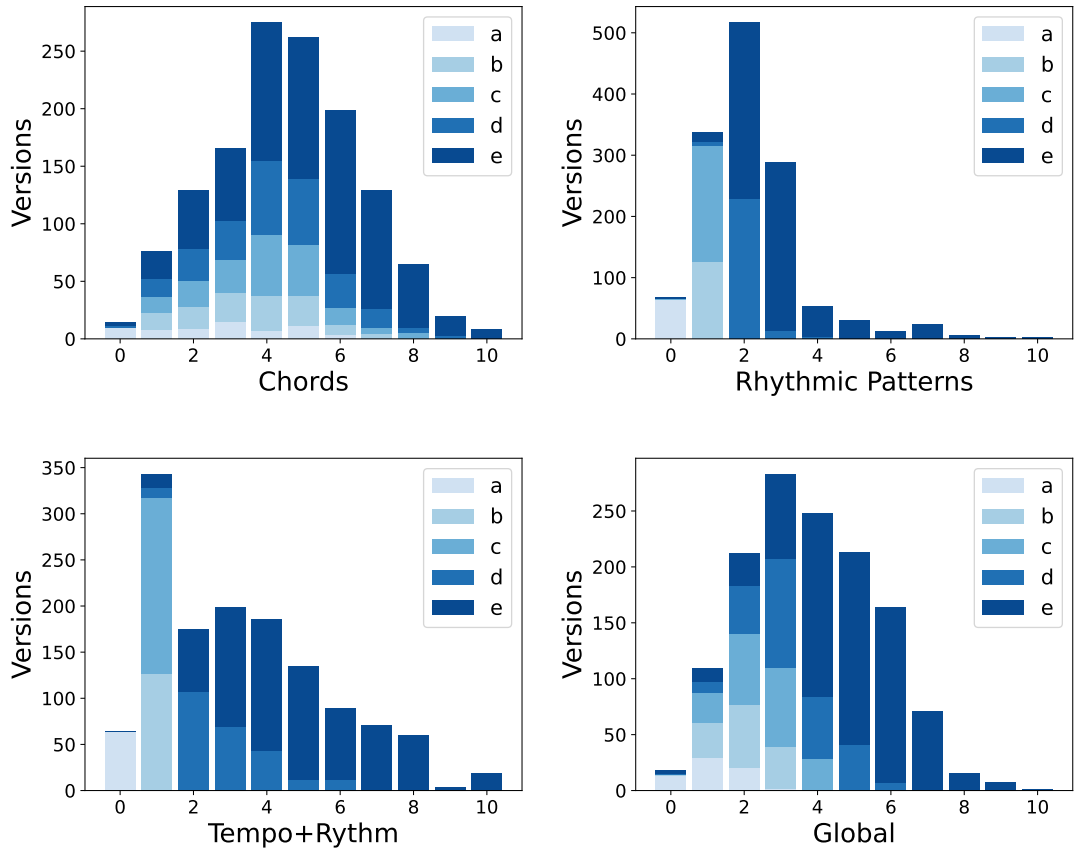


Figure 4.3: Distribution of the 989 versions among the four dimensions. Versions range from **a** (easiest, chords played with whole notes) to **e** (most difficult, original rhythm of the part).

well distributed across levels and could likely benefit guitarists of all levels. As expected as well, more complex versions (**d, e**) are often rated with high difficulty levels, while **a** versions are rarely rated over 3 out of 10.

4.3 A Model for Personalised Suggestion

Now that we have introduced our data and the difficulty annotations, we formalise the song recommendation method we implemented. Each song is attributed levels on the criteria described in section 4.1. Likewise, each learner is attributed levels on the same criteria, based on the songs they already practised. It is then possible to compare the level of a learner with the ratings of a part's version, to estimate the learning difficulty and therefore make appropriate recommendations. Ultimately, the model does not recommend a single song but rather provides the user with several options of varying difficulty to give them agency in their musical practice.

4.3.1 Definitions

We assume that we have sets of S songs $\mathcal{S}$, P parts $\mathcal{P}$, V versions $\mathcal{V}$, and E exercises $\mathcal{E}$. Each version v is linked to one part p, and each part p is linked to one song s. Each version v is also linked to a set of exercises $\varepsilon(v) \subset \mathcal{E}$.

$$\mathcal{S} = \{s_1, s_2, ..., s_S\}, \tag{4.1}$$

$$\mathcal{P} = \{p_1, p_2, ..., p_P\}, \tag{4.2}$$

$$\mathcal{V} = \{v_1, v_2, ..., v_V\}, \tag{4.3}$$

$$\mathcal{E} = \{e_1, e_2, ..., e_E\}. \tag{4.4}$$

Let us consider that a guitarist g already learned a subset $\mathcal{V}_g \subset \mathcal{V}$ of parts' versions available in the corpus. We assume that it implies they already master the corresponding exercises:

$$\left(\mathcal{E}_g = \bigcup_{v \in \mathcal{V}_g} \varepsilon(v)\right) \subset \mathcal{E}. \tag{4.5}$$

Given that information, which version should the model recommend to the learner? To provide a personalised pedagogical suggestion, our approach is to consider *multiple criteria* to model the difficulty of a version and to estimate the guitarist's skill levels. This allows us to model the pedagogical value of a version for each learner.

Modelling a Version's Difficulty Each version's learning difficulty is evaluated according to K criteria:

$$\{\ell_1, \ell_2, \ldots, \ell_k, \ldots, \ell_K\}.$$

Each criterion has a value ℓ_k ranging between 0 and 10, where 0 indicates a version that a beginner can play and 10 is for an experienced guitarist. In this chapter, we consider $K = 5$ criteria, but any number could be used if other difficulty annotations were retrieved. The criteria we use are the 5 described section 4.1: chord complexity, playing speed, rhythm difficulty, global rating, and amount of new exercises required.

4.3.2 Estimating a Learner's Skill Level

How can we describe the skill level of a guitarist g across the considered criteria? For each criterion ℓ_k we retrieve, among the versions already practised by the guitarist, the list $\mathcal{V}_g^{k[N]} \subset \mathcal{V}_g$ of the N most difficult versions according to that criterion. The skill level $\ell_k(g)$ of the guitarist in that criterion is then the average difficulty of these N versions:

$$\ell_k(g) = \frac{1}{N} \sum_{v \in \mathcal{V}_g^{k[N]}} \ell_k(v). \tag{4.6}$$

For example, with $N = 4$, a guitarist g who knows perfectly *Knocking on Heaven's Door* (all parts, version **e**) but only the easiest version (**c**) of *Losing my Religion* i.e. using the ids from Table 4.2:

$$\mathcal{V}_g = [1.1.e, 1.2.e, 1.e, 65.1.c, 65.2.c, 65.3.c, 65.c] \tag{4.7}$$

would have the following skill levels:

$$\begin{aligned} \ell_C(g) &= 3.75 \qquad (3+4+4+4)/4 \\ \ell_R(g) &= 1.75 \qquad (2+2+2+1)/4 \\ \ell_T(g) &= 1.75 \qquad (2+2+2+1)/4 \\ \ell_G(g) &= 3.00 \qquad (3+3+3+3)/4 \end{aligned}$$

In the actual recommendation system, the levels of each guitarist are estimated by selecting $N = 15$ versions among the ones they practised, which is equivalent to 6 songs on average. Preliminary experiments showed that it allowed to smooth high values as desired, because being able to play one difficult song is not equivalent to mastering the corresponding level of difficulty. Conversely, higher N values tended to underestimate the learners' level by including too many easy versions that advanced guitarists also know how to play.

4.3.3 Personalised Version Suggestions

Once the learner's skill levels are estimated, they can be compared to the ratings of a version to assess the learning difficulty the guitarist would face. A *challenge fit* $\mathcal{F}_g(v)$ is computed for a guitarist g wishing to learn a version v. It is calculated as a weighted sum of the absolute values of K *challenge criteria* $C_k(g, v)$, each weighted by a coefficient α_k:

$$\mathcal{F}_g(v) = \sum_{k=1}^{K} \alpha_k \left| C_k(g, v) \right| \tag{4.8}$$

Challenge for a Difficulty Criterion. For all difficulty criteria except the number of exercises, the challenge is modelled using an exponential function with a scaling coefficient β_k and a bias to control the target difficulty γ_k:

$$C_k(g, v) = \exp\left(\beta_k \left[\ell_k(v) - \ell_k(g) - \gamma_k\right]\right) - 1 \tag{4.9}$$

When $\gamma_k = 0$, then the value $C_k(g, v)$ equals 0 when $\ell_k(p) = \ell_k(g)$, meaning the version exactly matches the guitarist's level. When γ_k is strictly positive (resp. negative), we want the guitarist to target a higher (resp. lower) difficulty than their current level. The exponential function allows to penalise versions that are harder than what γ_k permits. This function also has the advantage to be differentiable, which permits training the coefficients using gradient descent, something that is considered for future work.

Table 4.3: Coefficients for each criterion. The γ values were set increasing on the 4 difficulty tiers. Tier 2 is meant to be almost the estimated level of the guitarist.

	α	β	γ (by challenge)			
			1	2	3	4
Chords	0.5	1	−0.5	0	1	3
Rhythmic Patterns	0.5	1.5	−0.5	0	1	3
Tempo/Impact .	0.5	1.5	−0.5	0	1	3
Global	0.8	0.3	−0.5	0.5	1	3
Number of exercises	2	0.8	2	3	5	7

Challenge for the New Exercises. We call $N_E(g, v)$ the *new* exercises required to learn version v. In our corpus, exercises allow students to learn new chords and practise chord transitions required for the corresponding version. For this reason, some exercises can be requirements for multiple parts' versions, we therefore only count exercises not already practised by the learners as contributions to the learning difficulty. Let γ_E be the number of new exercises to practise, the challenge value is then defined similarly as before:

$$C_E(g, v) = \exp\left(\beta_E \left[N_E(g, v) - \gamma_E\right]\right) - 1 \tag{4.10}$$

Difficulty Tiers The recommendation system eventually proposes four difficulty tiers. We chose to have multiple tiers with 3-4 versions recommended in each level so that learners have some agency in their practice, and to acknowledge the fact that some recommendations might not be interesting to the students for various reasons. The recommended versions $\widehat{v}$ are the ones with the best (i.e. lowest) challenge fit, for each tier D:

$$\widehat{v} = \underset{v \in \mathcal{V}}{\operatorname{argmin}}\, \mathcal{F}_g(v) \tag{4.11}$$

The selection procedure is repeated until each difficulty tier is populated with the best possible recommendations. For controlling the desired difficulty level, each tier is defined by its own α_k, β_k and γ_k parameters. Tier D_1 suggests versions that should be easy to the learner (γ_k negative), Tier D_2 suggests versions at the user's level ($\gamma_k \approx 0$), and Tiers D_3 and D_4 suggest progressively harder parts (γ_k positive). The system's γ_k parameters were chosen to target different difficulty levels and to control the number of new exercises introduced, allowing learners flexibility in shaping their learning paths (see Table 4.3). Note that the number of exercises that can be considered is never 0 ($\gamma_k \geq 2$) because each version has always a few specific exercises that will necessarily be new. The coefficients α_k and β_k were iteratively refined during the evaluation process (see Section 4.4). The relatively small amount of data available, combined with the intention to maintain a very simple and interpretable model, led us to favour manual tuning over automated optimisation techniques, even if we acknowledge that it might introduce bias and overfitting.

Future work will benefit from users' feedback in the GSC application that could be used to update the coefficients through ML techniques.

4.4 Evaluation

With our partnering guitar teacher, we created eight test profiles $G1 - G8$, representing guitarist with different skill levels, spanning from absolute beginners to advanced players. The profiles define songs that are considered already learned by the guitarists, as well as *wish* songs that they desire to learn specifically. The test profiles each focus on different styles, to also reflect different affinities. For each guitarist g, the model suggests between 10 and 12 versions (of parts or full songs) with, for each recommended version v, a suggested difficulty tier $\phi(v, g) \in [1, 2, 3, 4]$. Our partner guitar teacher evaluated the relevance of these suggestions, providing for each recommendation v for a guitarist g an *expected difficulty tier* $\psi(v, g) \in [-\infty, 1, 2, 3, 4, +\infty]$, where $-\infty$ means that the version is too simple for g and should never have been proposed, and conversely with $+\infty$, that it is way too difficult. Intermediate values are used to denote the right difficulty tier for the recommendation, according to the guitar teacher. We acknowledge the limitation of evaluating the predictions by only one person. Repeating this evaluation process with multiple teachers will be studied in future work once the GSC app reaches a stable state.

Figure 4.4 shows the distribution of $\Delta(v) = \phi(v) - \psi(v)$ on the 90 suggestions for various models (10-12 per test profile). 86% of the suggestions are considered appropriate by the teacher, that is with $|\Delta| \leq 1$. Ablation models, in which certain criteria are omitted, still achieve solid results, with correct average prediction rates ranging from 60% to 86%. This highlights the robustness of the proposed approach but also the significance of individual criteria, particularly the assessed overall difficulty. Indeed, removing the global difficulty rating results in the worst evaluation results. This supports the idea that this criterion represents information that is not included in other criteria, further motivating the need for future studies on what musical dimensions could replace this rating. Besides we notice that the test profiles do not react homogeneously to criterion ablation. For instance, removing chords rating improves the performance on G3, but the results plummet if the global difficulty is removed. Conversely, results for G4 stays fairly consistent with at least 60% of correctly recommended songs regardless of the ablation. Overall, the observed errors mostly concern songs that are either too simple or too complex, even within challenge levels 1 and 4, respectively. Future work will focus on studying these edge cases further to improve the model's accuracy and relevance. We will also need to clarify if the differences between profiles when removing criteria are due to the profile definitions favouring some criterion over others, or bias in the evaluation process.

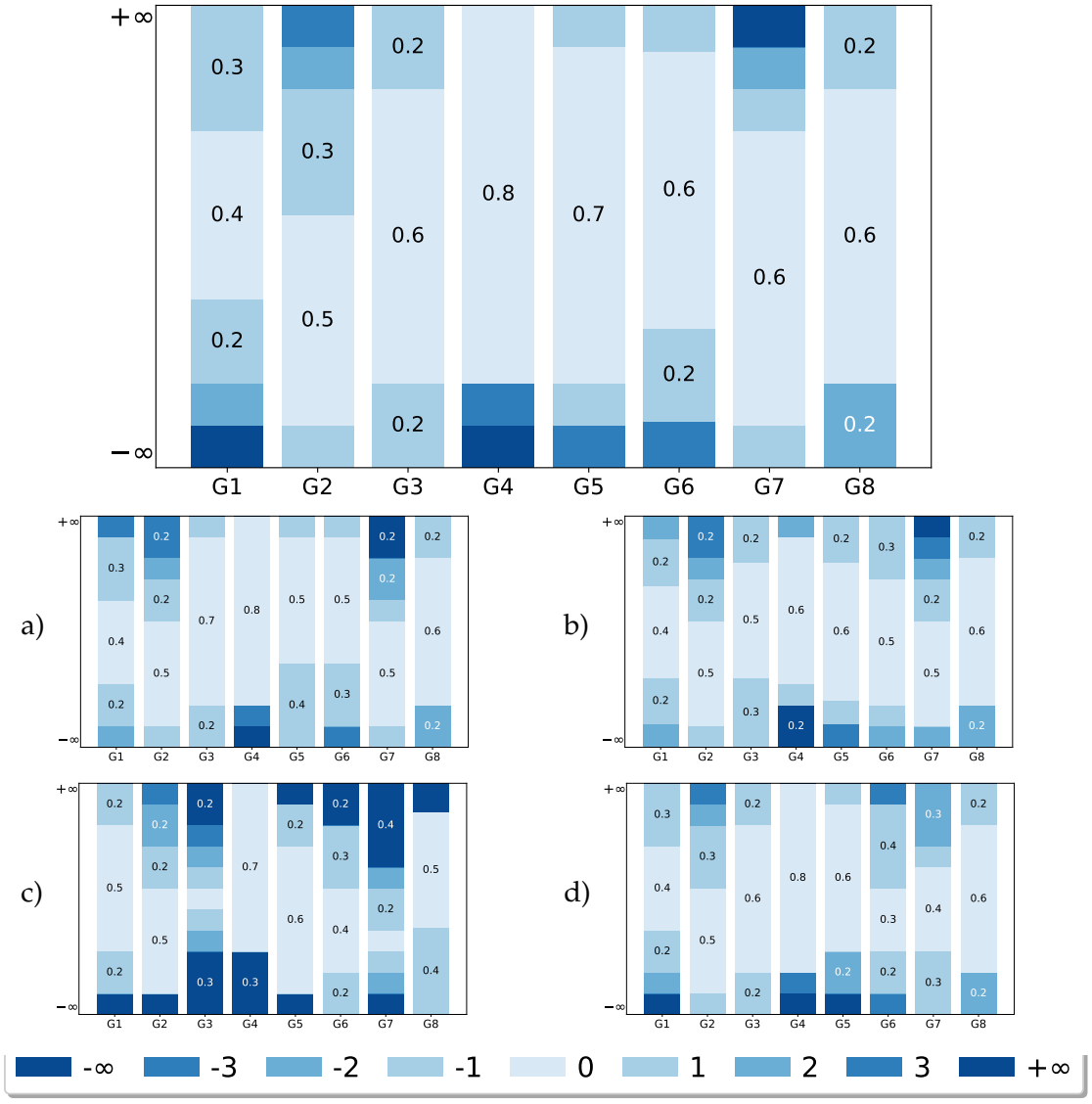


Figure 4.4: Distribution of the deviations between the predicted difficulty tier and the expert annotation on 90+ recommendations from 8 test profiles. (Top) Full model. On average, 61% of the model's suggestions perfectly match the difficulty tier from the expert (0), and 25% deviate of at most 1 tier (+1 or −1). (Bottom) Ablation models, ignoring a) chords, b) NPM and Rhythmic Patterns, c) global difficulty, or d) number of exercices.

4.5 Discussions and Conclusion

In this chapter, we presented a recommendation system that estimates the level of learners across different criteria. That level is then compared with the ratings of parts' versions in our proprietary corpus to make difficulty-informed suggestions. However, throughout this chapter, we identified limitations and areas of improvement for future work. In this section, we discuss those perspectives further, and reflect upon this industrial collaboration.

4.5.1 Limitations and Perspectives

Pitfalls of the current approach This recommendation system was created jointly with a partnering guitar teacher to draw from their expert knowledge. However, basing all our approach on the feedback of a single expert is likely to introduce bias and even overfitting to a kind of guitar teaching. This is currently a contextual limitation of this research, as it originates from a collaboration with industrial partners that currently limit the possibilities to open the data to people outside of the project. Opening the annotation data to other teachers would however allow to make the recommendations more robust and study possible discrepancies between teaching approaches. Another key limitation of the work presented in this chapter is also that the evaluation procedure is based on a reduced number of recommendations analysed by the same guitar teacher. While the different test profiles are designed to make the evaluation process systematic and representative of different learners' backgrounds, the corpus is not covered entirely and we do not dispose of an automatic way of exploiting this evaluation to update the coefficients (and it would likely overfit our data). Because the system is currently deployed in the GSC application, users feedback and usage data will become available. This data will allow to research reinforcement learning approaches to gradually update the coefficients (X. Wang et al. 2024). Strategies from common recommender systems (Zeng et al. 2024) like collaborative filtering will also become usable and might improve suggestions by grouping similar user profiles. Finally, the application currently trusts users' about their mastery of the exercises they practised through self-assessment feedback. Implementing automatic assessment algorithms (Eremenko et al. 2020) could ensure that guitarists do not move on to the next step before really mastering the required techniques, to reduce possible frustration and demotivation (Margoudi et al. 2016).

Suggesting learning paths The suggestion model presented in this chapter aims at evaluating the guitarist level at a discrete time t, and suggests them appropriate songs to practise at a time $t + 1$. While we showed that such suggestions are already satisfactory, they lack temporal consistency. With our current system, if a specific song the learner wishes to practise is too difficult, it might never appear in the recommendations. A guitar teacher would be able to elaborate a learning path to allow the student to gradually reach the required level, but our model has no such planning capabilities. To address

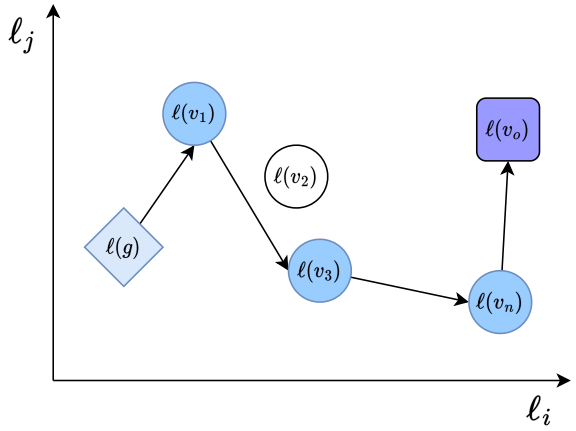


Figure 4.5: Mock representation of a learning path. We use two dimensions for clarity but our current suggestion algorithm uses 5 criteria for computation.

this limitation, Aurélien Jeanneau worked as a research intern, under the supervision of Mathieu Giraud and me, to study possible ways to suggest learning paths to learners. The studied approach consists in representing the corpus as a graph, by defining a distance between the learner's level and the difficulty-annotated versions. This distance is defined as the fit value (eq. (4.8)) and allows to locate every version in a K dimensional space. Because the objective is, given the current level of the learner, to prepare a learning path towards a desired version, the task is akin to the *shortest path problem*. However, given the musical nature of our research, additional properties are expected from the paths, like stylistic consistency to some extent, and limiting difficulty peaks as much as possible. For this reason, the problem is most likely NP-complete, and finding the exact solution is "prohibitively time-consuming" (Vazirani 2003). Aurélien Jeanneau has thus identified a variant of the A* search algorithm (Hart et al. 1968) as a promising way of finding the best learning path. When a guitarist g has the objective to learn the version v_o, the algorithm explores the possible versions v on the path and minimises the quantity:

$$a(v) = \mathcal{F}_g(v) + h(v, v_o) \tag{4.12}$$

The fit value $\mathcal{F}_g(v)$ (eq. (4.8)) is used as the distance in our space, and h defines a heuristic that measures the distance from v to the goal v_o. In our case, this heuristic is the fit value $\mathcal{F}_{g*}(v_o)$, with $g*$ the guitarist profile if he had practised the version v. In other words, the heuristic simply measures if practising v would reduce the distance to v_o. Because minimising $a(v)$ is equivalent to minimising the sum of the distance and the heuristic, the best path will not necessarily be a "straight line" towards the objective. A mock representation of a path is provided Figure 4.5. Experiments yielded results similar to what is shown, i.e. that the best path tends to include versions that will improve the learner's level towards the difficulty of v_o on a single criterion, keeping other criterion at low values. This behaviour is desirable in our situation because learning a song that is more difficult on a single aspect will be easier that increasing difficulty on all dimensions

at once. Experiments are still ongoing for the search of learning paths, but high importance is given to the speed and efficiency of the final algorithm. Current implementations use a beam search variant of A* (Furcy et al. 2005) to reduce the set of versions that are explored at each step of the path.

Perspectives of Future Analyses Finally, this collaboration with GSC could be a great opportunity to study how guitarists learn new songs, and what approaches work best. One of the first thing that could be studied is whether dividing the corpus in parts and versions is beneficial to the learning process. While our partnering teacher's experience suggests that beginner guitarists tend to learn only specific parts of some songs (a well-known riff, the chorus to sing along), usage data from the app would allow us to verify that assumption. Further longitudinal studies could also be imagined, to determine how learners use the app over a period of several months, and what progress they made.

4.5.2 Conducting Public Research within an Industrial Collaboration

This chapter presented work based on an industrial collaboration. GSC being a start-up that was created around the time we started working together, I got the opportunity to see it grow and evolve with time, and to be involved in some of the decisions along the way. Despite the commercial nature of this collaboration, we comply with our duty of conducting open research by releasing some of the code and the data openly, as detailed below.

Commercial application. The commercial application Guitar Social Club, already deployed,[5] provides guitarists with a comprehensive environment centred around guitar playing, with more than 3200 instructional videos. Users can indicate their preferred styles, the songs they already know, and the songs they wish to learn. The "full model", that is the model described here enhanced with additional criteria,[6] then recommends parts' versions to practise. Guitarists then select the piece they want to work on and progress through a series of exercises by watching the corresponding videos that guide them through the learning process.

Open-source and open-data components. Developed in Python, the code for the recommendation model described in this chapter is available under the LGPLv3+ license at https://gitlab.com/algomus.fr/guitor/. Although the complete dataset is proprietary, we release a subset of this data under the open OdBL license, covering difficulty metadata for 41 songs, 111 parts, and 337 versions, as well as 73 expected difficulty tier evaluations. This public dataset is available on the same git repository.

[5] www.guitarsocialclub.com, accessed in June 2025.

[6] The main addition is an automated global rating that does a weighted combination of the manual chord, rhythm and other playing annotations.

5

Features for Automatic Difficulty Assessment of Tablatures

"I was at 200 [bpm], let's say that was difficult."

Petrucci (1995)

Contents

As discussed in the previous chapter, the difficulty of performing a piece of music involves a variety of musical aspects. Can a Machine Learning method automatically analyse the difficulty of a tablature on multiple criteria? In this chapter, we present a feature engineering approach for difficulty analysis, trained and tested on a newly-gathered dataset of difficulty-annotated guitar and bass tablatures. This work was initiated during a research stay at the Music Technology Group (MTG) in Barcelona (Spain), in collaboration with Pedro Ramoneda and Vsevolod Eremenko. Part of this work has been published in:

"Towards Explainable and Interpretable Musical Difficulty Estimation: a Parameter-efficient Approach"

Pedro Ramoneda, Vsevolod Eremenko, Alexandre D'Hooge, Emilia Parada-Cabaleiro, Xavier Serra.

Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR), 2024.

(Ramoneda et al. 2024b)

5.1 Objectives

The previous chapter discusses how to suggest new songs to guitar learners that are adapted to their current level, measuring several difficulty aspects of guitar playing. However, such suggestions require guitar teachers to manually annotate songs on each difficulty criterion. That analysis is time-consuming and has to be conducted by expert teachers. Guitar teachers might benefit from a system that can provide a first rating of songs they do not know, allowing them to browse faster through songs that might interest their students. Self-taught guitarists could also benefit from an automatic difficulty assessment tool as a way for them to navigate online resources more easily. Nevertheless, that difficulty analysis needs to be interpretable and accurate. Indeed, some guitar tablatures website have difficulty annotations: Songsterr automatically processes tablatures and rate them from 1 to 8;[1] Ultimate Guitar classifies songs in four difficulty categories.[2] But both websites do not share precisely how those ratings are obtained and, while they might help learners quickly find songs of a general level, guitarists cannot know why a song is difficult or not. For this reason, during this research stay at the Music Technology Group, we studied ways to automatically analyse the difficulty of tablatures in a way that is interpretable and might be beneficial to guitar players and researchers. To do so, we assemble a new dataset from an online community of guitar and bass players, propose features that could capture difficulty dimensions of tablatures, and conduct preliminary experiments to choose difficulty prediction models that would be interpretable, explainable, and usable.

5.2 The Tablature Performance Difficulty Dataset

To the best of our knowledge, no dataset of performance difficulty annotated guitar tablatures currently exist. For instance, the data we presented in the previous chapter and in D'Hooge et al. (2024b) and Hassein-Bey et al. (2025) has precise difficulty annotations on multiple dimensions but corresponds to chord progressions and not tablatures. Likewise, in Vélez Vásquez et al. (2023), the difficulty annotations were obtained on chord progressions with their corresponding positions. To study if difficulty can be studied from tablatures, we assembled a new dataset, as described below. This dataset will be a first step towards proposing more difficulty-annotated scores, as they can already exist for piano in Western classical music (Ramoneda et al. 2023; 2024a).

To protect the community from which the dataset was retrieved, we hereby call the dataset the Tablature Performance Difficulty (TPD) dataset and only share limited information about the data source. Legal aspects that motivate this decision further are discussed in appendix C. The TPD dataset will nonetheless be shared publicly and openly, once we publish our results.

[1] https://www.songsterr.com/, accessed in June 2025.
[2] https://www.ultimate-guitar.com/explore, accessed in June 2025.

Table 5.1: Grade ranking system used in the competition.

Category	Grade
Beginner	1-3
Intermediate	4-5
Advanced	6-7
Master	8-9
Expert	10

5.2.1 Data Source and Difficulty Ratings

The TPD dataset is based on a Competition that has been running in an online community for over 10 years. Community members who participate in the competition can join a *league* by evaluating their own level to enter one of the four categories: *Beginner*, *Intermediate*, *Advanced* or *Master*. Bass players have their own leagues while guitar players can enter both a *rhythm guitar* and a *lead guitar* one. Each week, a song is provided for every league at all levels. Those songs can be suggested by participants and are rated by "expert members" of the community before being presented in the competition. Around week 60, songs started being also graded from 1 to 10, to further classify them inside a category, as depicted Table 5.1. The Expert level (Table 5.1) is a special category that first appeared more than 2 years after the beginning of the competition (week 128) and only comes once a month. Members that tackle this level of challenge are expected to play particularly difficult songs, or all songs of an album in a row. The songs were transcribed by several creators, 370 in total. Some transcribers contributed with hundreds of tablatures, but most (66 %) got one or two transcriptions included in the competition.

5.2.2 Data Retrieval and Preparation

The organisation of the competition is completely open, most of it happening through online spreadsheets. We extracted all metadata available, in particular the difficulty ratings, and then retrieved the transcription files. While all are indexed through the same website, they are then stored on different cloud providers like Google Drive, Dropbox, Mega or Mediafire. To fetch the files from all those different sources, we made a scraping script for each website. When we retrieved the data in Spring 2024, approximately 530 weeks of competition had elapsed. However, this does not mean that we have access to 530 rated songs per league, because moderators of the competition can reuse songs selected in the past from time to time. With this in mind and discarding no longer available or invalid files, we end up with 1 293 Lead guitar files, 1 297 bass files and 973 rhythm guitar files. Songs were transcribed by 370 different creators and encompass 765 different artists. We represent the distribution of difficulty levels Figure 5.1. Technically, there should be the same amount of songs in all difficulty categories (except the Expert level), the differences observed are only due to missing files or the fact that some songs were selected in multiple weeks.

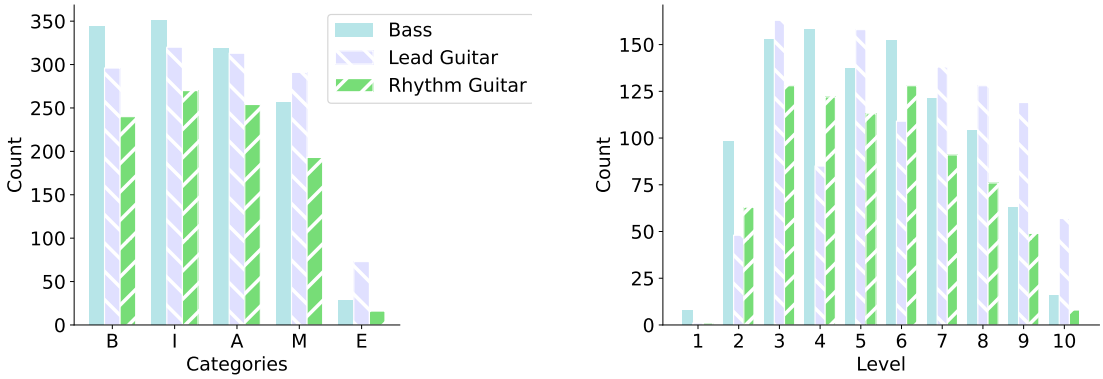


Figure 5.1: Distributions of files in each difficulty category/level

We can however make more observations on the distribution of songs across difficulty levels (from 1 to 10, compared to difficulty categories, from B to G). It appears that some difficulty levels are less used, like '1' in particular, questioning the validity of the defined difficulty range since it is not used uniformly by the competition's moderators. Differences in the relative distributions between leagues are also noticeable, though they might be due to "experts" of each league having different frames of reference.

5.2.3 Style Analysis

We hypothesise that the difficulty of interpreting a musical piece is correlated to its musical style, in the sense that songs of different styles will be difficult for different reasons. To verify this hypothesis, we need style information for the songs in the dataset. Unfortunately, artist and title metadata are prone to typos or other errors, or might even be of songs not represented on any standard metadata website (some creators sometimes add their own unpublished compositions to the competition). However, all song files contain a 30 s audio excerpt as a teaser. It is defined by the user and usually containing a recognisable part of the song like a riff or a chorus. We extract those and then use one of the MAEST pretrained deep neural networks (Alonso-Jiménez et al. 2023) to obtain the musical style estimate of each song using the Discogs styles taxonomy.[3]. The MAEST models are based on an audio transformer model trained to conduct style classification from audio, and evaluated on multiple automatic tagging datasets. We use the model trained on 30 seconds audio excerpts[4] and directly process the audio teasers after downsampling them to the expected 16 kHz. Results of this analysis are represented Figure 5.2.

[3] https://www.discogs.com/, accessed in July 2025.

[4] Directly available through *Huggingface* (https://huggingface.co/mtg-upf/discogs-maest-30s-pw-129e, accessed in June 2025).

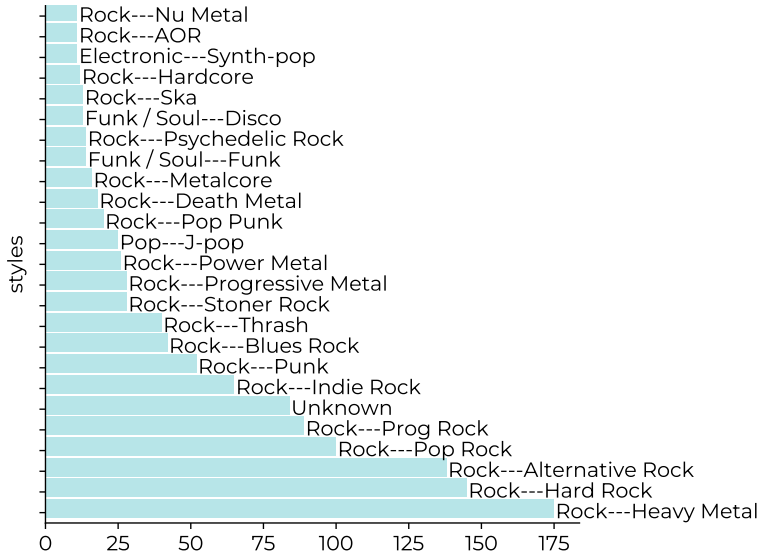


Figure 5.2: Counts of the styles detected in the dataset. Note that styles with less than 10 occurrences were removed for clarity.

While most of the songs are identified as Rock, we can make distinctions between sub-styles. With this style data now available, we can study if difficulty is correlated with style to uncover possible biases in our dataset. The dataset shows a clear over-representation of rock-inspired substyles (Figure 5.2), meaning that our approach might not generalise well to unknown styles. Unfortunately, 75 songs could not be classified because they did not have any (valid) audio teaser file, but manual exploration showed that most are also sub-styles of Rock, so they would not significantly change the conclusions made previously.

We conduct Kruskal-Wallis tests (Corder et al. 2009) to determine if the difficulty ratings distributions change when grouping songs by styles. The dependent numerical variable is the difficulty category, and the categorical independent variable is either the main style group (e.g. "Rock"), or the sub-style group (e.g. "Punk"). The null hypothesis is "there is no difference between the groups' median values". It can be rejected if the H-statistic is greater than χ^2_c and the p-value is lower than 0.05. Results for those tests are provided Table 5.2. It appears that a significant difference between the median ratings can only be observed when grouping songs by sub-styles for lead guitar. Overall, the results support the idea that musical style has little impact on the difficulty rating of a song, i.e. that progressive rock songs, for instance, are not rated consistently lower/higher than pop rock songs. Those observations are however not enough to definitely conclude that difficulty is not influenced by musical style, because we only analysed global ratings. Future work will focus on verifying whether songs of different styles are difficult for different reasons. The style analysis conducted in this section can nonetheless be useful when splitting the dataset, as ensuring that the proportions of each style is kept between splits might help the model generalise better to new data. Indeed, we expect that a model trained only on rock music would adapt poorly to unseen styles like Jazz, for example.

Table 5.2: Results of the Kruskal-Wallis tests comparing difficulty ratings by styles for each instrument classroom. The only statistically significant result is shown in bold.

Instrument	Style Type	H-statistic	χ^2_c	p-value
Bass	Main Style	12.30	22.36	0.42
	Sub-Style	125.72	137.70	0.16
Lead Guitar	Main Style	8.37	22.36	0.76
	Sub-Style	**170.29**	**136.59**	**0.000 2**
Rhythm Guitar	Main Style	6.84	21.03	0.81
	Sub-Style	107.31	122.11	0.22

5.3 Features for Playing Difficulty Estimation

To attain an explainable and interpretable difficulty estimation system, we rely upon extracting features that capture different dimensions of musical difficulty. Our choice of features is guided by existing knowledge, of previous research (Chiu et al. 2012) for piano score difficulty analysis, but also guitar schools syllabi (ABRSM 2021; RSL Awards 2023; Trinity College London 2017). Many of the features presented hereafter are interrelated, we use them to ensure that most aspects of performance difficulty are captured, and we analyse their correlation with difficulty ratings at the end of this section.

5.3.1 Instrument-agnostic Features

In Ramoneda et al. (2024b), we presented features that can be used to evaluate the difficulty of piano scores. Some of the features can however be used without any particular instrument in mind. The first four are taken from Chiu et al. (2012), while the last one was introduced with our paper:

- **Pitch Entropy.** The entropy of the observed pitch event, based on the probability to observe each pitch p within the set of all observed pitches $\mathcal{P}$:

$$-\sum_{p \in \mathcal{P}} P(p) \log_2 P(p) \tag{5.1}$$

  Pitch Entropy, as emphasised by Chiu et al. (2012), is particularly relevant in assessing musical difficulty. As Sayood (2018) discusses, there is a link between entropy of a task and the cognitive load it imposes on the performer, a concept that may also apply to music performance (Palmer 2006);

- **Pitch Range.** The distance between the lowest and highest pitches (numbering them as MIDI) in the score. Complex pieces might use a greater range of the pitches available;

- **Average Pitch.** The average MIDI pitch value in the score. While it is not a reason for a piece to be difficult, it might correlate with difficulty ratings, like Pitch Range;
- **Average IOI.** Average Inter Onset Interval. Average time in seconds between onsets of two consecutive pitch set events, measures the playing speed. With T_i denoting the ith onset time, it is defined as:

$$\frac{\sum_{1 \leq i \leq N^{\text{events}}-1} (T_{i+1} - T_i)}{N^{\text{events}} - 1}; \tag{5.2}$$

- **Pitch Set LZ.** Lempel-Ziv (LZ) complexity of the pitch-sets sequence. Pitch entropy measures the cognitive load of playing a sequence of pitches. However, music is often perceived in terms of larger structures like phrases and sections, not just isolated pitches, prompting us to seek a descriptor that captures the "repetitiveness" of music on a broader scale (Margulis 2014). To this end, we employ LZ-complexity, a measure of redundancy introduced by Ziv et al. (1978). In the context of music research, it was used for binary encoded rhythm analysis by Shmulevich et al. (2000). We apply LZ-complexity to sequence of pitch sets by identifying all subsequences of pitch sets that cannot be reproduced from preceding material through a recursive copying procedure. The number of such unique subsequences is defined as the LZ-complexity of the part. This approach allows us to assess the structural complexity and redundancy of a musical piece.

For this study on guitar difficulty prediction, we also added multiple other instrument-agnostic features to reduce the risk of missing some aspects of difficulty. The relevance of each feature will be tested afterwards nonetheless. Added features are:

- **Tempo.** The tempo of the song, in BPM. While the average IOI measures the playing speed, the tempo of the song is a general measure agnostic to the rhythms played;
- **Smallest Note Duration.** Represents a higher bound of the playing speed;
- **Number of Note Onsets.** The more notes a song have, the more mistakes might occur, so counting notes might be an appropriate proxy of difficulty;
- **Song Duration.** While the duration of a song does not necessarily reflects the difficulty of playing it, longer songs require musicians to be focused and to maintain the quality of their playing for a longer time;
- **Pitch Entropy Rate.** Computed like pitch entropy, but on transitions from one pitch to another. Denoting consecutive pitch pairs as $(p_{(t)}, p'_{(t-1)})$:

$$-\sum_{(p,p') \in \mathcal{P}^2} P(p_{(t)}, p'_{(t-1)}) \log_2 P(p_{(t)}, p'_{(t-1)}); \tag{5.3}$$

- **Syncopation.** Metric of the syncopation in each bar, based on Fitch et al. (2007) and Longuet-Higgins et al. (1984), onsets are given a weight and are considered syncopated if they break a level of a bar's "rhythm tree". The syncopation measures

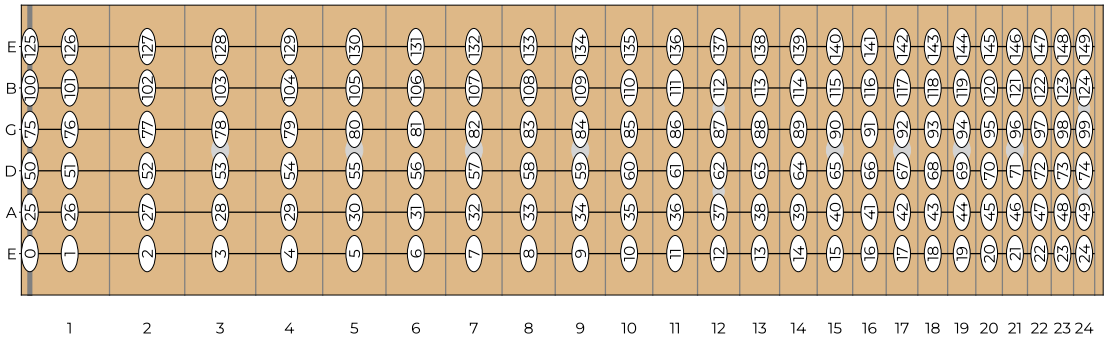


Figure 5.3: Numbering of all string-fret pairs on the fretboard.

can also consider 'salience', based on Sioros (2014, pp. 75–79), by taking tempo into account and assessing how it influences the perception of beats. Given the syncopation value of notes, we also compute syncopation entropy to determine how complex syncopation patterns can be;

- **Triplets and Sextuplets.** We also determine whether a song contains triplets or sextuplets as a boolean, to determine if those specific rhythms can correlate with difficulty;
- **Number of Different Note Durations.** A song with many different note durations might be more complex because the musician needs to be able to switch comfortably between rhythms.

5.3.2 Guitar-related Features

The features presented in the previous section can capture some aspects of musical difficulty, but guitar playing has specificities that call for dedicated features as well. Some are taken from Chiu et al. (2012) and adapted from piano to guitar, while others are drawn from Guitar WPM Syllabi (ABRSM 2021; RSL Awards 2023; Trinity College London 2017).

The first modification proposed is to adapt pitch measures by also applying them to sequences of string-fret, string, and fret sets data. The string-fret values could be compared to MIDI note numbers for positions on a fretboard, and are defined like shown Figure 5.3. We expect this representation of the fretboard to be beneficial when combined with entropy and LZ measures since it encodes additional positional information, compared to pitch data. Applying entropy measures to string numbers is also a way to encode the picking patterns used for playing a song, while the fret entropy encodes movements of the fretting hand. Other added features are listed below:

- **"Horizontal" Velocity.** Adapted from the right and left hand velocity of piano, the guitar implementation is derived from Hori et al. (2013) and reflects movement of the fretting hand along the fretboard (from low to high frets). With $F_{(t)}$ and $F_{(t-1)}$ the

lowest fretted[5] position of two consecutive onsets, and d_t the time interval between them:

$$\frac{1}{2d_t} \exp\left(-\frac{\left|F_{(t)} - F_{(t-1)}\right|}{d_t}\right); \tag{5.4}$$

- **Playing Techniques Ratio.** Notes with expressive playing techniques are extracted from the score and compared to the total amount of notes. A tablature with more playing techniques can be more complex to perform than one with only regular notes. Refer to chapter 6 for a presentation of the guitar techniques measured;
- **Chord Fingering Difficulty.** We compute the difficulty of playing chords with the metric defined in Wortman et al. (2021). The authors define acceptable distances between fingers based on bio-mechanical data, and use them to derive an "anatomical score" for any possible chord fingering. The implementation is presented in more details in chapter 7.

5.3.3 Analysing Feature Importance

To assess the relative importance of each feature, we first determine which ones are most correlated with the difficulty ratings. Because we deal with "heavily-tied rankings" (a lot of songs have the same ratings), we use the Kendall Tau's correlation measure in its τ_c version (Kendall 1945). Results for the most important features are presented in Table 5.3. The τ_c coefficient is a value between -1 and 1, -1 indicates a strong disagreement, 1 a strong agreement, and 0 an absence of correlation between the variables.

The first observation is that the LZ features are the most correlated with the difficulty ratings, for all leagues. It also seems that features based on pitch information or string-fret numbers are almost identically correlated to the ratings. While many of the most important features are similar between leagues, it also appears that the league type has an impact. For instance, the average chord difficulty is relatively important for rhythm guitar, while not that relevant for lead and bass. Likewise, the onsets entropy, which measures the entropy of regular and syncopated notes, is fairly correlated to bass difficulty, which seems consistent with how bass parts tend to use complex rhythms like in Funk or Progressive Metal. However, features that should not correlate with difficulty *a priori* are significant in all leagues, like pitch range or fret range. That observation could be surprising but may be explained by the fact that even if such values can *correlate* with the difficulty of a song, they are not necessarily the *cause* of the difficulty, something that only musicians can tell currently. For instance, a lead guitar part with a fast solo on high frets and a melody on the middle of the fretboard will have a wide fret range. The solo can add difficulty to the part, and is represented partly by an increase fret range.

[5] I.e. not an open string.

Table 5.3: τ_c correlation measures of the top 15 features for each league. Some alternate versions of entropy features are omitted for clarity.

(a) Lead Guitar

Feature	τ_c
String-Fret LZ	0.752
Pitch Set LZ	0.748
String-Fret Entropy Rate	0.654
Pitch Set Entropy Rate	0.652
Has Sextuplets?	0.562
Pitch Range	0.550
Fret Range	0.521
Has Triplets?	0.485
Smallest Note Duration	−0.454
Hammer-on/Pull-off Ratio	0.422
Num. of Different Note Durations	0.375
Max Horizontal Velocity	0.374
Song Duration	0.371
Playing Techniques Ratio	0.350
Average IOI	0.308

(b) Rhythm Guitar

Feature	τ_c
String-Fret LZ	0.664
Pitch Set LZ	0.663
String-Fret Entropy Rate	0.589
Pitch Set Entropy Rate	0.587
Smallest Note Duration	−0.409
Fret Range	0.357
Std. dev. Horizontal Velocity	0.350
Average Chord Difficulty	0.342
Pitch Range	0.334
Hammer-on/Pull-off Ratio	0.327
Song Duration	0.311
Mean Playing Speed	0.310
Playing Techniques Ratio	0.304
Num. of Different Note Durations	0.303
Max Horizontal Velocity	0.291

(c) Bass Guitar

Feature	τ_c
String-Fret LZ	0.614
Pitch Set LZ	0.613
String-Fret Entropy Rate	0.545
Pitch Set Entropy Rate	0.542
Pitch Range	0.498
Smallest Note Duration	−0.411
Fret Range	0.384
Std. dev. Horizontal Velocity	0.354
Syncopes Entropy	0.319
Mean Playing Speed	0.314
Has Triplets?	0.304
Max Horizontal Velocity	0.303
Num. of Different Note Durations	0.289
Playing Techniques Ratio	0.284
Average IOI Squared	0.274

The features least correlated to difficulty are also similar across leagues: features related to the BPM, time signature or key signature have little to no relation to difficulty ratings. Their low correlations with difficulty motivated us to discard them in upcoming experiments.

5.3.4 Defining Feature Groups

We defined several versions of the presented features to see if some implementations better captured aspects related to difficulty. We can also verify numerically which features are most intercorrelated to group them significantly. Those groups will be used in further studies to ensure that multiple difficulty dimensions are captured and do not limit the model to use features of a single group. To do so, we calculate conditional τ_c correlations for all feature pairs given a fixed difficulty level, and average the coefficients across

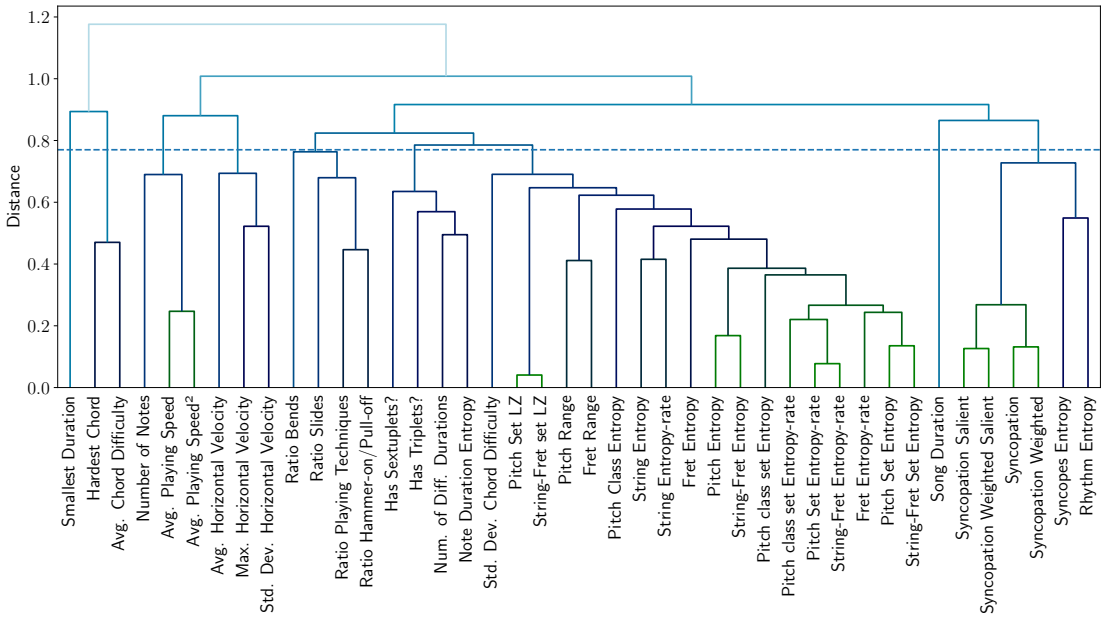


Figure 5.4: Dendogram for the hierarchical clustering of features. The horizontal dashline represents the group splitting threshold.

all difficulty levels and leagues. We then convert these coefficients into a distance matrix and apply hierarchical clustering based on average distance to identify clusters of correlated features (Figure 5.4). From the resulting dendrogram, we observe that the features most correlated with difficulty are also interrelated, like entropy or LZ features. We define a grouping threshold of 0.77 and manually adjust some of the groupings to better gather features representing similar musical dimensions. We end up with 7 groups, summarised in Table 5.4.

5.4 Machine Learning Models for Difficulty Analysis

Different models can be used to study the difficulty of scores in standard or tablature notation. In Ramoneda et al. (2024b), we introduced a new ML architecture dubbed RubricNet to analyse the difficulty of piano scores in an explainable and interpretable fashion, which could also be applied to tablatures. We also discuss using GNB models, for their lightness and fast training, that allowed us to quickly compare the efficiency of using different feature sets.

5.4.1 RubricNet

The "RubricNet" network, introduced in Ramoneda et al. (2024b), is shown in Figure 5.5. As its name suggests, it aims at showing results in a *rubric* (see for example Vélez Vásquez et al. (2023)), attributing scores to each aspect of difficulty. It takes as input a set of features, and outputs a probability estimating which difficulty level the song belongs to.

Table 5.4: Groups of features based on their mutual correlation.

Group	Features
Speed	Max. Horizontal Velocity Average IOI Average IOI Squared Smallest Duration Avg. Horizontal Velocity Std. Dev. Horizontal Velocity
Stamina	Number of Notes Played Song Duration
Structure/Repetition	Pitch Set LZ String-Fret Set LZ Pitch Set Entropy-rate String-Fret Entropy-rate String Entropy-rate Fret Entropy-rate
Technicity	Ratio of Expressive Techniques Ratio of Bends Ratio of Slides Ratio of Hammer-on/Pull-off Max. Chord Difficulty Avg. Chord Difficulty Std. Dev. Chord Difficulty
Rhythm	Syncopation Weighted/Salient Syncopes Entropy Onsets Entropy Num. of Different Durations Note Duration Entropy
Pitch and Position	Pitch Entropy Pitch Range Fret Range Pitch-set Entropy Pitch-class-set Entropy Pitch-class Entropy String-fret Entropy String-fret-set Entropy String Entropy Fret Entropy

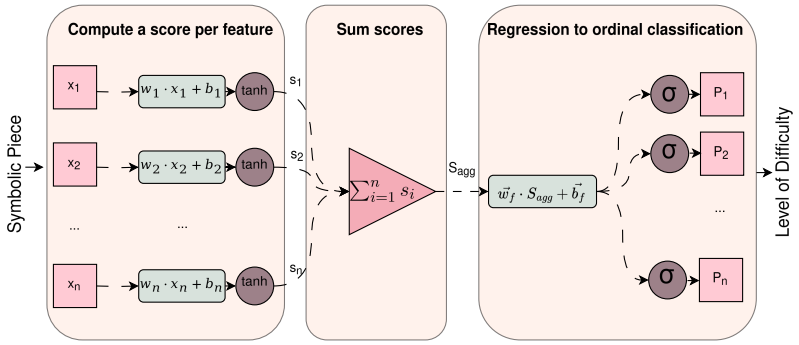


Figure 5.5: The RubricNet architecture. Figure from Ramoneda et al. (2024b).

It comprises a series of linear layers dedicated to process individual input descriptors, followed by a non-linear activation function (tanh).

Given a set of N input features, each feature x_i is first processed through its dedicated linear layer with weight w_i and bias b_i, followed by a hyperbolic tangent activation function to yield:

$$s_i = \tanh(w_i \cdot x_i + b_i) \tag{5.5}$$

where s_i represents the processed score for the i-th descriptor. Scores are then aggregated in a single score S_{agg}:

$$S_{agg} = \sum_{i=1}^{n} s_i \tag{5.6}$$

The aggregated score S_{agg} is then passed through a final linear layer to obtain the logits for the difficulty level prediction, which are mapped to probabilities with a sigmoid function:

$$\vec{P} = \sigma(S_{agg} \cdot \overrightarrow{w_f} + \overrightarrow{b_f}) \tag{5.7}$$

where σ denotes the sigmoid function, $\overrightarrow{w_f}$ and $\overrightarrow{b_f}$ are the weight and bias of the final linear layer, respectively.

Ordinal Optimisation This model applies an ordinal optimisation approach (Cheng et al. 2008), predicting ordered categorical outcomes, i.e. difficulty levels such as beginner (1), intermediate (2), and advanced (3), through logits. These logits, computed using a MSE loss, indicate the model's predictions on the ordinal scale. The difficulty level is then obtained as:

$$\max\{i \text{ where } P_i \geq 0.5 \text{ and } P_j \geq 0.5, \forall j < i\} \tag{5.8}$$

Advantages and Limitations Like a GNB model, RubricNet requires a small number of parameters: 2 for each input feature (a weight and a bias coefficient), and 2 for each difficulty level at the output stage. Since it is built as a neural network it can be trained with common gradient descent procedures. However, the experiments conducted on piano difficulty prediction showed that this model requires precise finetuning of the model's hyperparameters to ensure that a satisfying minimum is reached. That hyperparameter

search makes the training process computationally intensive, but does not change the fact that the resulting model is small and allows for fast inference. Besides, we showed in Ramoneda et al. (2024b) that this model performs at least as well and often better than deep learning models applied to the task of piano difficulty analysis in the past (Ramoneda et al. 2024a). Such results are encouraging since they suggest that a computationally efficient method might be a viable alternative to costly (in terms both of energy and data requirements) DL models. The RubricNet model is also explainable and interpretable, which is not the case of the DL methods studied in Ramoneda et al. (2024a), that are only partly explainable. Indeed, using a reduced set of musical features as input, rather than an entire tablature or piano score, allows to clearly identify the computational process that led to the final prediction.

5.4.2 Gaussian Naive Bayes

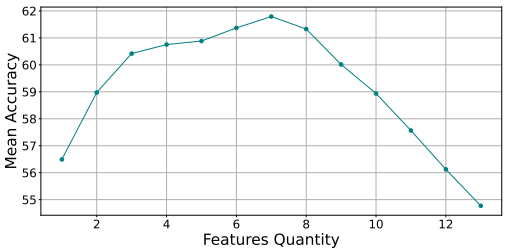
Naive Bayes models (introduced in chapter 2) are commonly implemented by assuming that the likelihoods follow a Gaussian distribution. Denoting the mean and the standard deviation by μ and σ respectively, the probability of observing the feature value x_i in class C_j is:

$$P(x_i|C_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \tag{5.9}$$

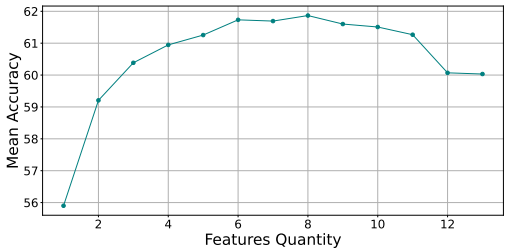
Advantages and Limitations The advantage of using such a system is that it is easy to fit, small, and efficient, because it only requires two parameters per input feature: a mean and a standard deviation. Besides, there are even techniques to update the parameters of a GNB model when additional data is used, without restarting the fitting process entirely (Chan et al. 1982). This update possibility is especially interesting in our case, since the competition goes on weekly and we might want to update the dataset on a yearly basis (or more often). However, the "Naive" Bayes models assume that all features are independent, which is rarely the case, as we studied in the previous sections. Besides, all features contribute equally to the output classification probability, which might not be desired in the case of music difficulty analysis. Fortunately, visualisations like a radar-plot (Figure 5.7c) can help better understand the internal computations of a GNB model.

5.4.3 Selecting a Subset of Features

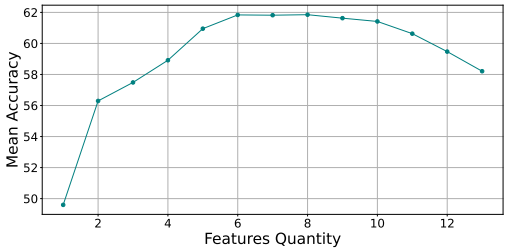
One might wonder why we group features, and not just use all of them. Taking advantage of the fast training of a GNB model, we use it to determine the best possible classification accuracy with a different number of features on 90% of our dataset. Those experiments showed that the best accuracy increases with the number of features until 7 before dropping with any additional feature. This trend is consistent for all instrument types (Figure 5.6). This observation motivates the fact that only a subset of features should be selected.



(a) Lead Guitar



(b) Rhythm Guitar



(c) Bass Guitar

Figure 5.6: Evolution of the mean accuracy with larger feature sets.

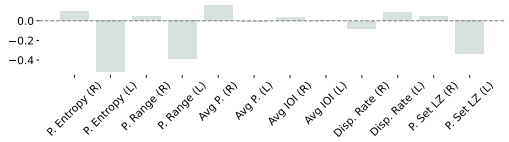
However, selecting the best features requires testing at least 100 000 combinations (if we assume we take one feature of each group defined in Table 5.4). Besides, other considerations for explainability and usability need to be taken into account, as discussed chapter 2. For instance, while a GNB model can reach more than 50% accuracy (with 5 possible difficulty classes, random chance would be at 20%) with a single LZ feature, users might be more interested in a model that analyses difficulty through multiple descriptors. In snippet 5.1, we show the best feature clusters for lead guitar, obtained by enforcing that at least one feature from each group (defined Table 5.4) is used before taking another from that group. Choosing which feature cluster to use is an open question, as accuracy only might not be an appropriate metric since performance vary by only a few percentage points between feature sets. One possibility might be to conduct an online study with users to ask them what clusters seem most appropriate to them, but this might depend on the visualisation chosen. This question therefore remains open and is left to future work.

5.4.4 Visualisations for Usability

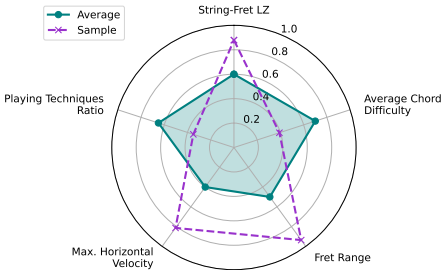
When analysing the difficulty of a musical score, we believe the system has to be *interpretable* and *explainable* (chapter 2). However, the system should also be usable i.e. easily accessible to end-users, regardless of their background, allowing them to effectively understand and interact with the model. If our system is destined to be used by any musicians, simply providing technical explanations of the results might not necessarily be enough. A common way of making a model's prediction explainable and useful is to implement some sort of visualisation (Chatti et al. 2024; Laato et al. 2022). Which visualisation would be the most appropriate for our task?

Descriptor Name	Score	Grade Divergence	Accumulative Score
Pitch Entropy	+0.79	-0.17	0.79
Pitch Range	+0.01	-0.27	0.8
Average Pitch	+0.03	-0.01	0.83
Average Ioi Seconds	+0.0	-0.03	0.83
Displacement Rate	+0.91	-0.05	1.74
Pitch Set Lz	+0.73	-0.21	2.47
Final Score	–	–	**4.89**

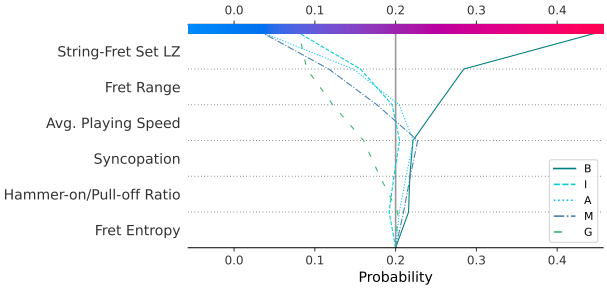
(a) Rubric visualisation



(b) Bar-plot visualisation



(c) Radar plot.



(d) SHAP values decision plot.

Figure 5.7: Possible visualisations for difficulty analysis. The first two are taken from Ramoneda et al. 2024b.

In Ramoneda et al. (2024b), we showed that predictions can be visualised through a rubric (Figure 5.7a) or a bar-plot (Figure 5.7b). Predictions from a GNB model can be easily represented by a "radar-plot" (Figure 5.7c). They can also, like predictions from other models, be represented through SHAP values (Lundberg et al. 2017) to show which features made one class be predicted instead of others (Figure 5.7d). All those visualisation possibilities are however affected by the features chosen for the prediction models to use. The features are, in that case, as important as the visualisation itself, because they define the information shared to the user. In that situation, a system that uses only one feature might perform well, but not be usable because it conveys little information on the rating obtained. Conversely, a model with over 10 features will be harder to comprehend in a glance and might not help users either. We showed previously that the best performance is attained with groups of 6 to 8 features, which might already be too much for users to understand easily. Choosing an appropriate feature set is thus not only a matter of performance, but also of explainability and usability, where even the name of the features can have an impact on the model's understanding by users. Those are open questions and we are considering asking musicians which visualisation styles they prefer in an online survey, as well as what feature sets seem most understandable to them.

Snippet 5.1 Best feature clusters for lead guitar difficulty estimation.

1. String-Fret Set LZ.
   Accuracy: 0.547;
2. String-Fret Set LZ, Fret Entropy.
   Accuracy: 0.556;
3. String-Fret Set LZ, Fret Entropy, Average IOI Squared.
   Accuracy: 0.584;
4. String-Fret Set LZ, Fret Entropy, Average IOI Squared, Average Chord Difficulty.
   Accuracy: 0.551;
5. String-Fret Set LZ, Fret Entropy, Average IOI, Fret Range, Weighted Syncopation.
   Accuracy: 0.560;
6. String-Fret Set LZ, Fret Entropy, Average IOI, Fret Range, Weighted Syncopation, Hammer-On/Pull-Off Ratio.
   Accuracy: 0.601;
7. String-Fret Set LZ, Fret Entropy, Average IOI, Average Chord Difficulty, Weighted Syncopation, Fret Range, Hammer-On/Pull-Off Ratio.
   Accuracy: 0.588;
8. String-Fret Set LZ, Fret Entropy, Average IOI, Average Chord Difficulty, Weighted Syncopation, Fret Range, Playing Techniques Ratio, Note Duration Entropy.
   Accuracy: 0.580;
9. String-Fret Set LZ, String-Fret Entropy, Average IOI, Average Chord Difficulty, Syncopation Entropy, Weighted Syncopation, Std. Dev. of Horizontal Velocity, Pitch Range, Playing Techniques Ratio.
   Accuracy: 0.556;
10. String-Fret Set LZ, String-Fret Set Entropy, Pitch Range, Hammer-On/Pull-Off Ratio, Average IOI, Weighted Syncopation, Max. Horizontal Velocity, Average Chord Difficulty, Number of Notes Played, Note Duration Entropy.
    Accuracy: 0.588.

5.5 Conclusion and Perspectives

In this chapter, we presented a new dataset for tablature difficulty analysis and musical features to address this task. We analysed those features extensively to determine how they correlate with actual difficulty ratings and among themselves. While DL models could be used for automatically analysing tablatures, we presented two interpretable and explainable alternative models. RubricNet was tested on piano score difficulty analysis and is a promising option for tablatures. A Gaussian Naive Bayes model could also be used to predict difficulty from musical features, and it was used to conduct preliminary analyses on the accuracy that can be reached from different feature clusters. Results suggest that not all features should be used at once, and that drawing features from a predefined group is a valid approach to select meaningful clusters with good performance. Questions subsist on how to choose between feature clusters with similar accuracy, and which visualisations would be appropriate to make the difficulty analysis system usable. Answering those questions would likely require conducting at least an online survey or even interviews with guitarists, and is considered for future work.

Assembling the tablature dataset was a difficult task because of all the different cloud providers used, but it proved a valuable resource that will hopefully help develop the field of tablature difficulty analysis (the dataset will be openly released after our first publication on the topic). Using such data however poses ethical questions, firstly because what the online community does is on the verge of illegality. Indeed, even though most transcriptions are custom-made, sharing them semi-publicly, with an audio recording, is not a usage permitted by Western copyright laws (see appendix C). Studying this data and using it for research thus poses the question of possibly shedding light on a community that currently benefits from being little known. We chose not to share any compromising information about the community that created the data in this chapter to avoid affecting it negatively in any way. The dataset will be shared publicly to the MIR community nonetheless, once the results presented in this chapter are published. Another questioning aspect is the scraping of the website's data. While used for research that will ultimately be as transparent as possible, exploiting the work of transcribers (without them knowing) is also morally questionable. Testing the models with the community might be the opportunity for them to share their thoughts on our preoccupations.

Part III

Assisting Guitar Tablature Composition

6

Modelling and Predicting Guitar Techniques: the Specific Case of Bends

"I don't think I should do big bends like that because it's going to sound like that era' – the boomer-ish sound."

Tim Henson, in Beato (2021)

Contents

In addition to rhythm and position information, tablatures commonly include a variety of annotations indicating the use of specific playing techniques. Guitarists are indeed accustomed to seeing tablatures with "slides", "tapping", or "bends", for instance. In this chapter, we introduce our approach for the analysis, modelling and suggestion of guitar techniques and in particular guitar bends. A guitarist composer could benefit from a system that suggests playing techniques as it would allow to render tablature arrangement of non-guitar songs more idiomatic to the guitar. To do so, we design features that capture multiple musical dimensions, and use them to train a decision tree and an MLP model for suggesting where to add bends in a tablature.

The research presented in this chapter has been published in:

"Modeling Bends in Popular Music Guitar Tablatures"
Alexandre D'Hooge, Louis Bigo, Ken Déguernel
Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR), 2023.
(D'Hooge et al. 2023a)

"From MIDI to Rich Tablatures: an Automatic Generative System incorporating Lead Guitarists' Fingering and Stylistic choices."

Pierluigi Bontempi, Daniele Manerba, Alexandre D'Hooge, Sergio Canazza.

Proc. of the 21st Sound and Music Computing Conf. (SMC), 2024. (Bontempi et al. 2024)

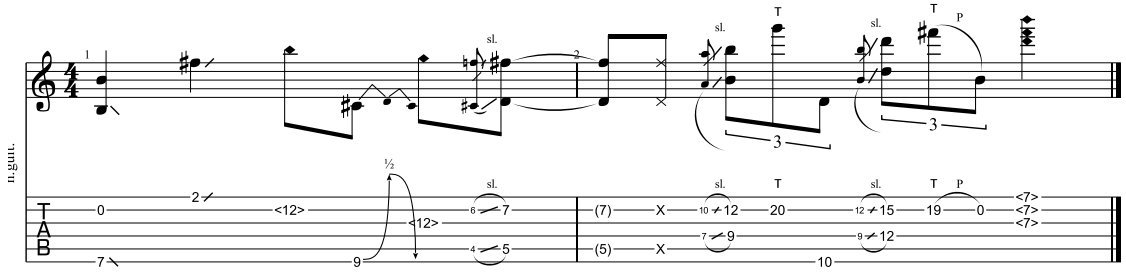


Figure 6.1: First two bars of *G.O.A.T.*, as played by Tim Henson from Polyphia. This excerpt contains slides, natural harmonics, bends, dead notes, tapping, and a pull-off.

6.1 Introduction

This section presents guitar playing techniques, with a focus on *bends*, as well as the objective and motivation for this work.

Guitar Playing Techniques WPM guitarists can use a variety of playing techniques on their instrument such as slides, bends, harmonics, tapping, etc. All techniques have corresponding notation symbols in tablatures, as illustrated Figure 6.1. Playing techniques can be classified using the two main categories defined in Bontempi (2025, p. 13), splitting techniques between *articulations* and *expressive techniques*. The former helps to transition smoothly from one note to another and are mostly used for playability reasons. The latter serve a musical purpose by adding expressive characteristics to the notes. Table 6.1 reports the techniques discussed further in this section, with descriptions from Bontempi (2025). In general, lead guitarists in WPM will use such techniques regularly, which is why any automatic system aiming at generating or arranging realistic and *idiomatic* guitar tablatures should include those techniques (Bontempi et al. 2024).

String Distribution of Playing Techniques This chapter focuses on lead guitar, because melodies and solos use many playing techniques, especially compared to rhythm guitar. As a first experiment, we conduct a statistical analysis of the playing techniques used in the mySongBook (MSB) dataset, after extracting a subset of all lead guitar tracks thanks to Régnier et al. (2021). Figure 6.2 shows the ratio of hammer-on, pull-off, vibrato and bends, with respect to the total amount of notes on each string. While hammer-on and pull-off are both techniques used for *legato*, their distribution across strings are noticeably different, suggesting that they are not used equivalently by guitarists.

Table 6.1: A description of some of the most common guitar playing techniques in WPM. Definitions are directly taken from Bontempi (2025, p. 13).

Technique	Type	Description
Hammer-on	Articulation	This technique involves using a finger from the fretting hand to hit the string from above, making it vibrate without being picked. Akin to a *legato* articulation.
Pull-off	Articulation	A finger of the fretting hand "pulls" the string, producing a note lower than the previous one without the need to pick the string. Can be considered the "opposite" of a hammer-on.
Bend	Expressive	Increases the pitch of a string-fret combination by pushing up or pulling down the string, perpendicular to the fretboard, with one or more fingers.
Vibrato	Expressive	Rapidly bending and releasing a string to rhythmically vary the pitch.

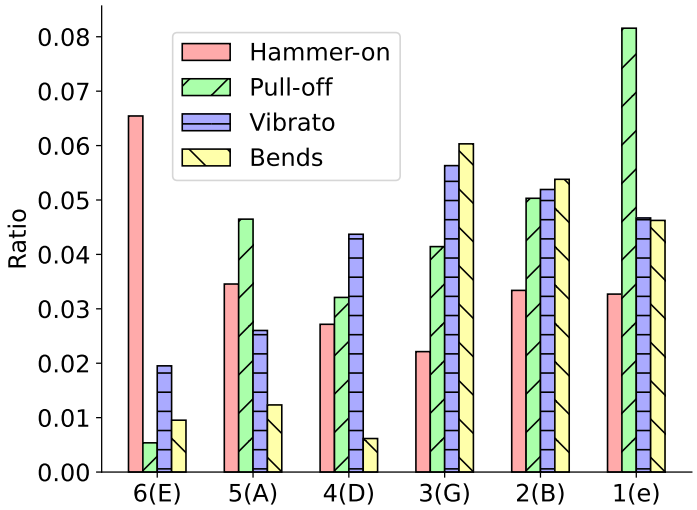


Figure 6.2: Ratio of the use of playing techniques to the total number of notes for each string. Strings are labelled according to guitar standard tuning, the 6th string being the low "E" and the 1st string the high "e".

It can also be observed that vibrato and bends are used with similar ratios on the three high strings. However, bends are rare on lower strings probably because thicker strings are harder to bend, while a vibrato is a smaller movement that is manageable even on stiffer strings. This simple analysis already shows that biomechanical affordance of the guitar can affect the use of playing techniques, especially for bends. This observation supports the idea that a feature-based approach could be used to suggest where to add playing techniques in a tablature in an idiomatic manner. The rest of this chapter focuses on suggesting bends, and they are presented in more details hereafter.

Figure 6.3: A bend being performed on guitar. Photo source: musicradar

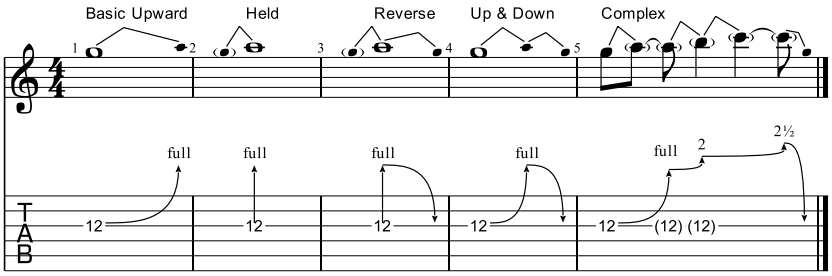


Figure 6.4: Five different bend types and their corresponding tablature notation in Guitar Pro. "full" denotes an amplitude of a whole step, numbers on the last bend are the amplitude in number of whole steps.

Bends The guitar, whether acoustic or electric, is (usually) a fretted instrument. This enforces the playing of discrete pitch values on a chromatic scale. This constraint can be beneficial, as it limits the risk of playing out-of-tune. However, it also prevents microtonal experiments or continuous pitch shifts, which can be a powerful means of musical expressiveness. To overcome this limitation, guitarists can use *bends*, a technique that consists in altering the string tension with their fretting hand (Grimes 2014) to reach a completely new pitch, up to several semitones higher (see Figure 6.3). We considered that bends were particularly interesting to study in the way they allow for continuous pitch variations. This specificity allows to play notes in a way that can sometimes be compared to singing (Kolb 2020), and multiple types of bends can be encountered.

While guitar players mostly agree over the existing types, the names used can differ. In this chapter, we consider the five bend types described in Gomez (2016): *Basic upward*, *Held*, *Reverse*, *Up & Down*, and *Complex* bends i.e. bends that do not belong to any of the previous categories. Figure 6.4 shows that writing bends in tablatures can be complex, but most notation software allows precise engraving (Figure 6.5). This can be linked to the prescriptive nature of tablature notation and how it conveys gesture to the guitarist. With arrows indicating the string movement, the guitar player only have to reproduce it with their fretting hand and can anticipate how the pitch will evolve without looking at the standard notation staff.

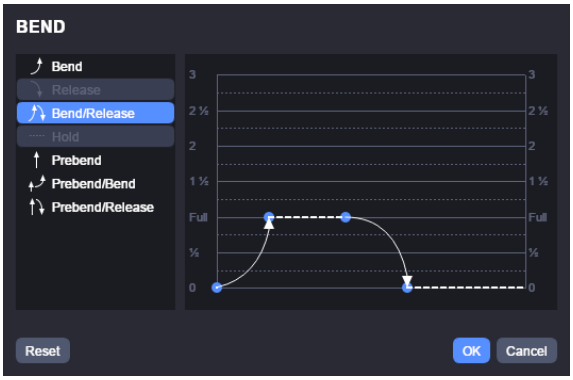


Figure 6.5: Screenshot of the bend notation window in Guitar Pro.

Motivation and Objective Deciding where and when to use bends is part of the idioms of guitar WPM playing and an important part of learning those styles. However, deciding when to use different bend types might not be as natural for a beginner than for an experimented guitarist. For instance, a guitarist playing a score that was composed for another instrument or a MIDI file arranged to a tablature (Edwards et al. 2024) may want to add expressiveness using bends, and could need help on deciding which notes to bend and which bend to use. In this chapter, we design an approach for suggesting guitar players with notes that should be bent in a tablature. With such a tool, it would for instance be possible to adapt a melody composed for a piano into an idiomatic and expressive guitar tablature.

6.2 Digital Representation of Bends

To formulate bend prediction as a machine learning task, we adopt a representation for bent notes consisting of four different labels and present it in this section. We also propose an approach to pre-process our data to obtain "bend-less" scores that will be used as input to a bend suggestion model. Finally, this section also presents the features designed for training an ML model for bend suggestion.

6.2.1 Labelling

In the introduction, we presented the different types of bends that can be encountered in guitar tablatures. Based on this taxonomy, we define 4 labels that represent the motion and current state of the played string for every note:

- $\varnothing$ — the string is not bent;
- $\uparrow$ — the string is bent, causing the pitch to go up;
- $\rightarrow$ — the string was bent previously and is plucked again in that state. The pitch is constant, but is not the one expected from the string-fret position;
- $\downarrow$ — the string was bent and is released, making the pitch go down accordingly.

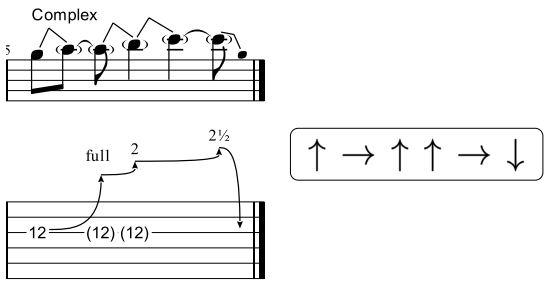


Figure 6.6: A complex bend and its corresponding labels. Each label corresponds to one note in the score (even if they are tied).

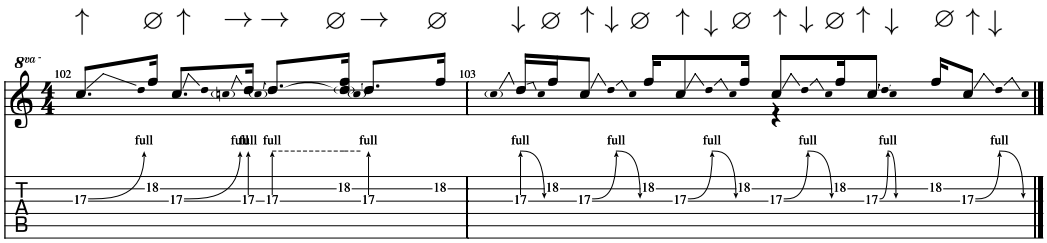


Figure 6.7: Excerpt from Lynyrd Skynyrd's *Free Bird* solo (written and played by Allen Collins). String movements are labelled above the standard notation staff.

We define those labels to circumvent the issue with *up & down* and *complex bends* that can be written in several different ways in tablature notation. Labelling notes with the associated string movements indeed permits representing bends accurately, without loss of generality, even for "complex bends" (Figure 6.6). Using these labels, all non-bent notes will be labelled ∅, *basic upward bends* are labelled ↑, *held bends* correspond to →, and *reverse bends* are ↓. To fit with this representation, *up & down bends* are split into two notes of equal duration (half the duration of the original note), the first one being labelled with ↑, and the second one with ↓. An example of such labelling on an actual guitar track is shown on top of Figure 6.7.

While bends can be of different amplitude, we do not include that information in our labelling. We do not aim at predicting amplitude in this work but only focus on predicting when bends occur. Similarly, we do not distinguish single notes from chords when predicting bends. This means that when the system predicts the presence of a bend in a chord, it does not specify on which string it occurs. The impact of this simplification is however minor, as 12% of the bends in our lead guitar dataset occur in chords.

6.2.2 Deriving a Bend-Less Score Simplification

Since our goal is to predict whether each note of the score is played with a bend, we must start from a "simplified" tablature that does not have any bend information, to use it as input to an ML model (Figure 6.8). Removing bend annotations from a tablature is however not a straightforward task since bending affects the pitch of the note performed.

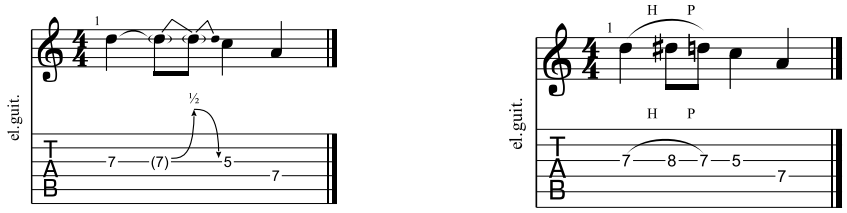


Figure 6.8: Example of the same melody played with (left) or without (right) bends.

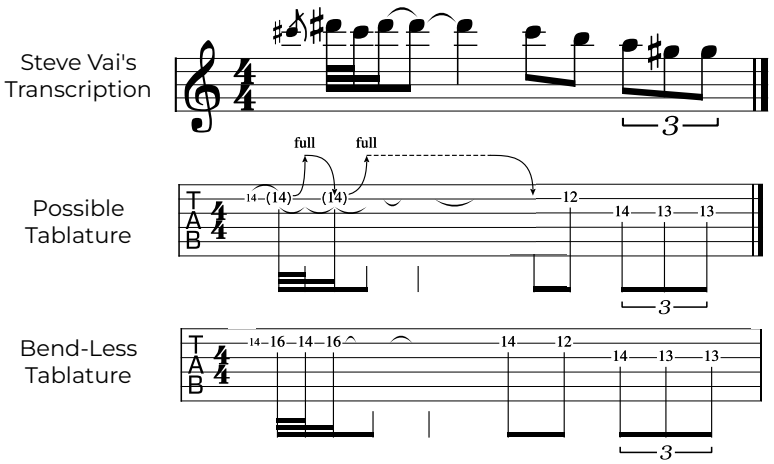


Figure 6.9: Excerpt from *Watermelon in Easter Hay*, composed and played by Frank Zappa, as transcribed by Steve Vai (Zappa 2017) in standard notation (top). Below are two possible tablature representations of this excerpt with (middle), or without (bottom) bends transcribed manually.

We design the following procedure when translating bent notes from the string-fret space to the pitch space:

- If a note is labelled by $\varnothing$, its pitch is directly obtained from the string-fret combination;
- Otherwise, the pitch is the one of the bend arrival note. In particular:
  - if the label is $\uparrow$ or $\rightarrow$, the arrival pitch is the pitch of the string-fret combination, to which the bend amplitude is added;
  - if the label is $\downarrow$, the arrival pitch is either the one corresponding to the string-fret position, if the string is released to its default state, or derived from the amplitude as before if the bend is partially released.

This approach is also the one chosen by Steve Vai when transcribing Frank Zappa's melodies to standard notation (Figure 6.9). The middle tablature in Figure 6.9 shows how this excerpt is actually played (based on a live performance) and the bottom tablature illustrates how the same excerpt might be played without bends. This is nonetheless an assumption because there is an uncertainty regarding where a bent note would be played

Table 6.2: List of the high-level features extracted from the note events. n denotes the current note index, and an exponent on a feature indicates on which neighbours it is computed. $k \in \{1, 2\}$ for features that are only computed on neighbours and not the current note, and $j \in [\![\mathrm{n}-2, \mathrm{n}+2]\!]$ (i.e. computed on all notes). If no exponent index is specified, the feature is only computed on the current note n.

Temporal	Duration
	Beat Strength
	Longer than previous
	Shorter than previous
	Same duration as previous
Pitch	Number of notes
	$\mathrm{Pitch}^{(j)}$
	$\mathrm{Pitch\ jump}^{(\mathrm{n}\pm k)}$
	Accidentals
	Pitch-class w.r.t scale root
Position	$\mathrm{Fret}^{(\mathrm{n}\pm k)}$
	$\mathrm{String}^{(\mathrm{n}\pm k)}$
	$\mathrm{Fret\ jump}^{(\mathrm{n}\pm 2)}$
	$\mathrm{String\ jump}^{(\mathrm{n}\pm 2)}$

on the fretboard without a bend (keeping its destination pitch but losing the technique). A guitar player might indeed choose to play a note on a higher string, if remaining on the same string called for an uncomfortably large hand span. Because deciding arbitrarily of a hand position could introduce bias into our model, we choose not to include any string-fret information concerning the current note in the proposed features for our bend suggestion task, as explained hereafter.

6.2.3 High-level Features for Bends Suggestion

To suggest what notes should be bent, we propose an intermediate representation as a set of high-level features, presented in Table 6.2. Using these features allow to extract musical data we expect to be correlated with bend usage rather than train a large neural network that would automatically extract intermediate features. Besides, choosing the features beforehand allows an ML model's predictions to be more explainable, as some architectures can be analysed to determine what input features contributed most to the results.

Some of the features used focus on the event under scrutiny, while others provide short-term context, both from the past and the future. Part of the features are derived from standard notation and convey *temporal* and *pitch* information, while others are related to *position* and the tablature space. If the studied event is a chord, the pitch, fret, and string values are averaged over all its constituting notes. While the average might seem an overly simple metric, experiments with other functionals such as *min*, *max* or *standard deviation* did not improve results. We discard any open strings for the fretboard features so that the average fret and string represent the actual position of the fretting hand. Apart

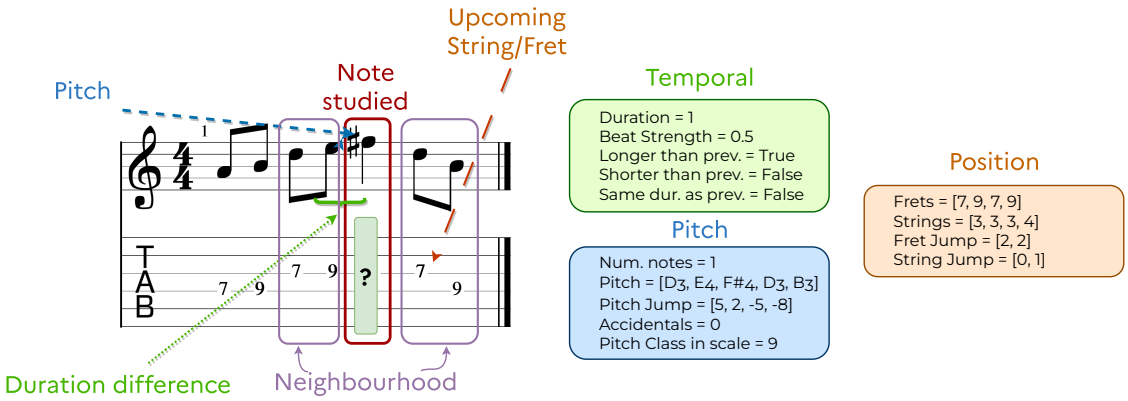


Figure 6.10: Illustration of the feature definitions, and values for the note studied.

from absolute/relative duration values, temporal features include the *beat strength*, which is a value between 0 and 1 suggesting how "strong" the beat of a note is. We obtain this value from the default implementation of the `music21` library (Cuthbert et al. 2010). These beat strength values have been designed for Western classical music, and therefore may be debatable for WPM. However, they are mostly used here to represent onset times independently of the time signature, while grouping notes that share rhythmic properties.

In addition to the features on the current note, we extract a context of two past and two future note events (Figure 6.10), as preliminary experiments did not show any benefits in using longer contexts. Additional boolean features are provided to recall if a neighbouring note event is missing, when a note is preceded or followed by a whole rest for instance. When a note is missing, all corresponding features are set to 0. From this context, we compute the pitch jump between neighbouring notes as well as the string and fret jumps when they are defined, i.e., not with respect to the current note (because we do not know where the guitarist would play the note if they were to bend it). We expect these features to help our algorithm derive the hand position on the fretboard. Furthermore, we add information about the key signature through the number of accidentals (positive for sharps, negative for flats). Fortunately, the key signature is transcribed properly within the MSB dataset, unlike DadaGP where most transcribers only use accidentals within the score and do not define a key signature for the piece.[1] From those accidentals, we derive the root note of the corresponding pentatonic minor scale (that scale encompassing much of guitar WPM Temperley 2018) and store the position of each note on this scale. For example, one sharp would make the root E (based on E minor, relative of G major), and an A would therefore be numbered 5 since it is 5 semitones above E.

[1] Amateur transcribers will usually only write the tablature notation, and the standard notation staff is filled automatically in most tablature notation software. Since they probably do not even look at it, they might not think about setting a key signature for the song that would make the score more readable (and allow for easier computational musicological analyses).

Table 6.3: Number of notes per label in our dataset.

∅	↑	→	↓	Total
123 231	9 627	1 270	3 314	137 442

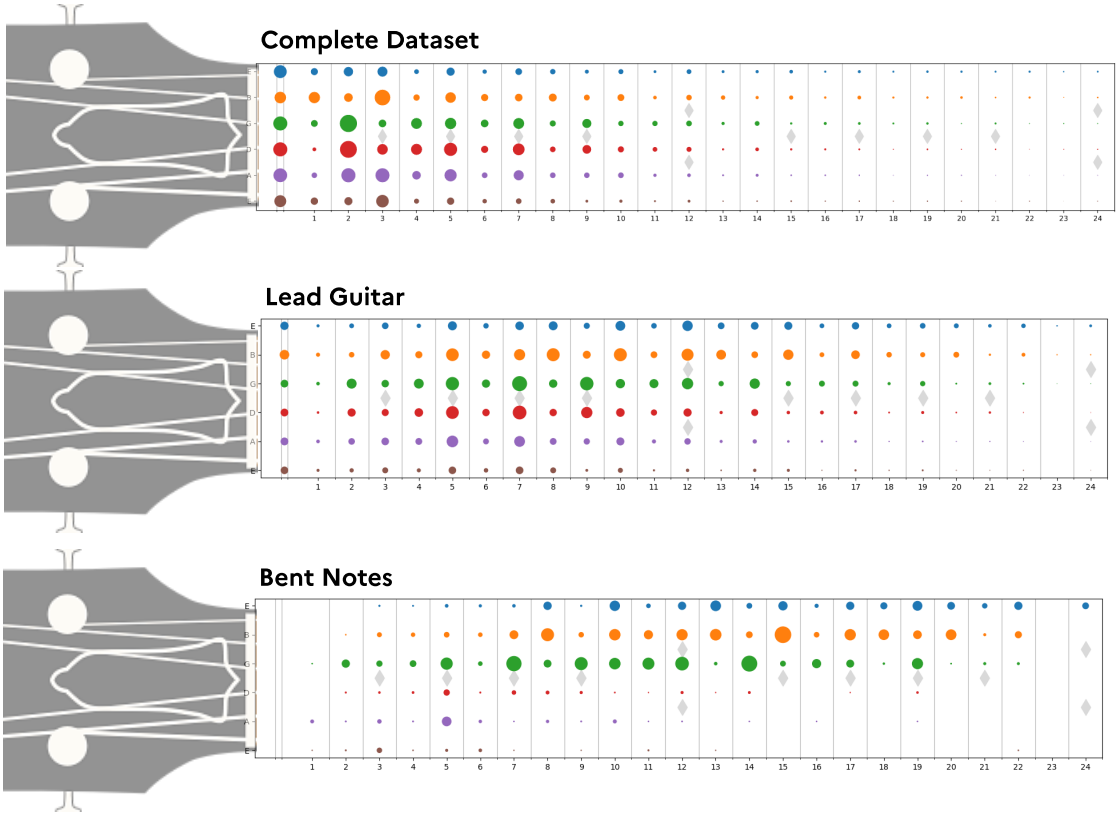
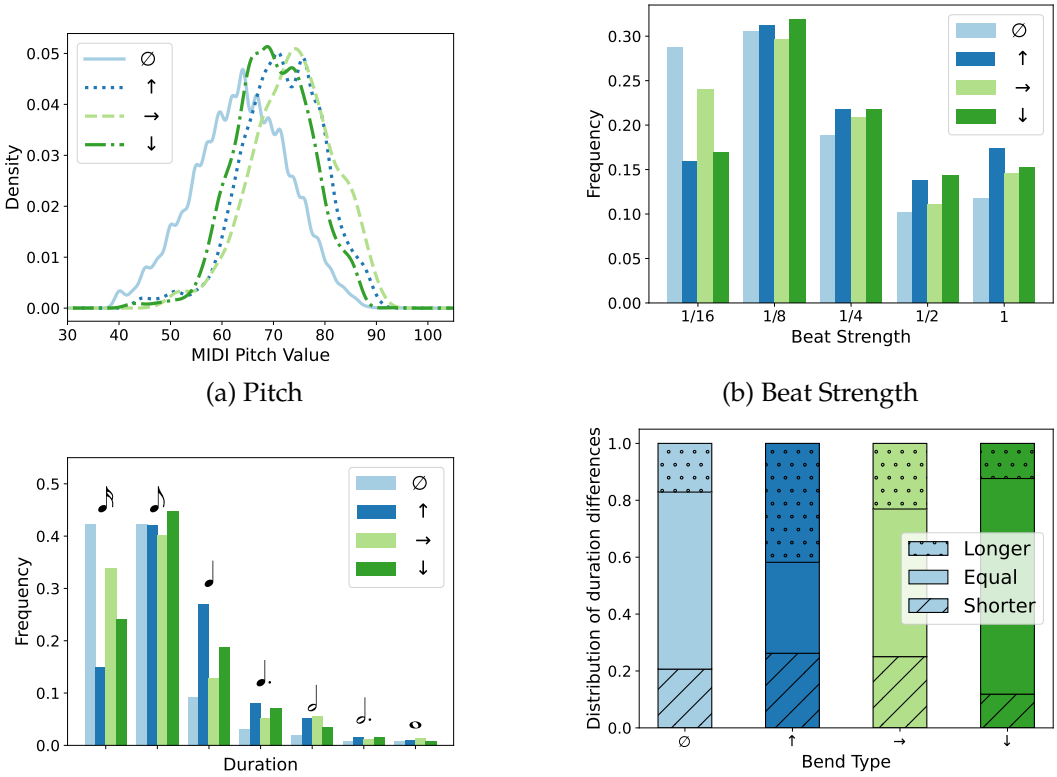


Figure 6.11: Normalised heatmaps of all notes in MSB (top), all lead guitar notes (middle), and bent notes among the lead notes (bottom). The bigger a circle is, the more notes are played on this location. Counts are normalised by the respective total amount of notes.

6.3 Bends Computational Analysis

All experiments have been conducted on the MSB dataset. A subset of 932 tracks estimated as *lead guitar* – totalling more than 130 000 notes – was extracted by applying the classification technique from Régnier et al. (2021). Our experiments focus on lead guitar parts because they tend to feature heavier use of playing techniques. In contrast with the whole corpus, which includes 2.5 % of bent notes, our lead guitar sub-corpus indeed contains 10% of bent notes. While this means that the classes (non-bent notes and bend types) are still highly unbalanced, focusing on lead guitar mitigates this imbalance.

Bend Features Distribution Table 6.3 reports the number of each type of bend in our corpus. The distribution of bent notes on the fretboard, compared to all notes, is shown in Figure 6.11. We confirm the trend from Figure 6.2 that most bends occur on the top 3



(a) Pitch

(b) Beat Strength

(c) Duration

(d) Duration difference with previous

Figure 6.12: Distribution of four of the extracted features, normalised per bend class.

strings, but also in the middle area of the fretboard. This observation differs from notes in general that are played on all strings, and especially on the two middle ones and around the 2nd fret for MSB as a whole and the 7th fret for the lead guitar subset. While it is possible that the obtained heatmaps are biased by an over-representation of certain key signatures in the dataset – 43% of the tracks are in G major/E minor or C major/A minor – this bias should affect all heatmaps equally, so their mutual comparison is still possible. Because bent notes are found on both higher strings and higher frets than all notes, their pitch is similarly higher on average, as it can be observed in Figure 6.12a.

The distribution of *beat strength* values is shown in Figure 6.12b. Because most beats and sub-beats in a measure have a beat strength of 0.25 or below, no label is mostly played on strong beats. An interesting result nonetheless is that ↑ and ↓ labels appear more often on stronger beats than ∅ and →. This apparent correlation of ↑ and ↓ labels with the meter might suggest a link between note expressiveness and accentuation in performance, which would need to be investigated further. In contrast, the → label is most often encountered on *weaker* beats. This observation can be linked to the fact that this technique is often used as a quick repetition of the previous note and will thus be played on the next offbeat, like in Figure 6.7.

The comparison of the duration of notes with or without bends (Figure 6.12c) confirms that ↑ and ↓ labels share some essential properties. Both labels have a proportionally

higher tendency to be found on notes with longer durations, even though eighth note is the most common duration for all classes. This figure also confirms that ∅ and → classes share some context properties. Figure 6.12d shows a strong tendency of ↑ labels to appear on notes with longer duration than their predecessor. This further supports the hypothesis that upward bends could be used to emphasise significant notes in a lead guitar part. This observation could also be related to Figure 6.12c and the substantial physical effort required to bend a string on short duration notes.

6.4 Bend Prediction Results

In this section, we present the performance of the decision tree trained to predict the labels of each note from the MSB lead guitar subset. We also present results of a feature importance analysis used to identify which features contribute most to the predictions of the model. Overall, we demonstrate the validity of our approach for suggesting bends, but observe surprisingly that features related to hand position contribute less to predictions than pitch and duration information.

6.4.1 Model Performance

We trained a decision tree (see section 2.2) to predict the bend label of a note from its feature representation. We choose this high-level approach to facilitate the interpretation of the results as well as the analysis of the contribution of the features. In addition to the elaboration of a predictive model, conducting our experiments in an explainable AI framework allows us to improve our understanding of the use of bends in this repertoire. Our classifier is trained on 75% of the dataset, and evaluated on the remaining 25%. To avoid some leakage from the training set to the test set, we ensure the split does not separate excerpts from a same track. We also ensure that each bend class is represented similarly in both sets. Duplicate feature vectors are removed track-wise to avoid overfitting due to repeated riffs/patterns. But, we acknowledge the fact that identical feature vectors can be found in different tracks and thus keep duplicates when found in different files. We use a decision tree with a Gini criterion and allow the creation of new branches until all final nodes were pure, i.e. with a single possible bend label in each leaf.

The confusion matrix of Figure 6.13 (left) shows the results of the multi-class classification on the joint prediction of all labels. Because the dataset is highly unbalanced, our model is naturally biased towards the ∅ label. However, it successfully identifies more than half of the ↑ and ↓ labels. Samples labelled as → are often misidentified as ↑ but this result still shows, presumably, that the model captures the difference between ∅ and → labels.

To reduce class imbalance, we tried applying SMOTE oversampling (Chawla et al. 2002) to the training data. This technique consists in taking samples of classes that are

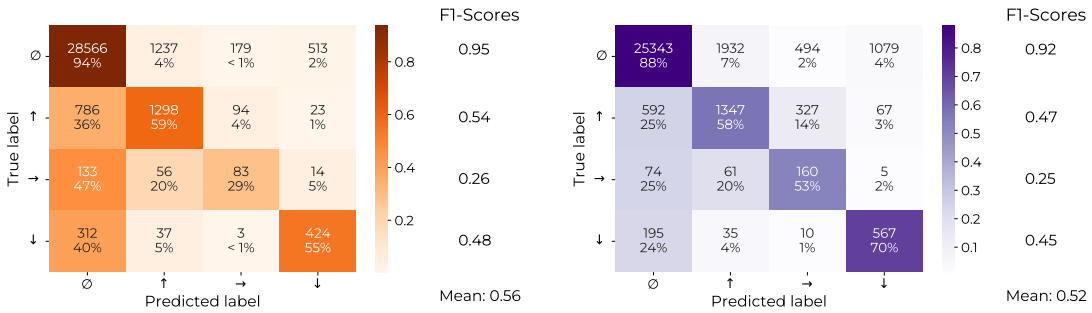


Figure 6.13: Confusion matrices obtained for classifying each note event to one of the bend class either without (left) or with (right) SMOTE oversampling. Those matrices are obtained on a split with average performance.

Table 6.4: Feature importance of the 8 most significant features for the decision tree. Standard deviation of any feature importance is never above 0.005.

Feature	Importance
Pitch	0.20
Pitch jump$^{(n+1)}$	0.17
Pitch jump$^{(n-1)}$	0.16
Duration	0.14
Same duration as previous	0.07
Fret jump$^{(n-2)}$	0.07
String$^{(n+1)}$	0.05
Pitch$^{(n+1)}$	0.05

under-represented and create new artificial samples in the neighbourhood of the real samples, until classes are balanced. We observed (Figure 6.13, right) that this oversampling technique doubles the number of correctly identified → notes and increases the ratio of well-classified ↓ notes by 15%pt. (percentage points). Nevertheless, ↑ notes *True Positives* (TP) ratio is about the same while the quantity of ↑ notes misidentified as → or ↓ increases. Similarly, TP ratio of ∅ notes drops by 7%pt., so 3 000 more notes are wrongly predicted as bent. Because we observed that bent notes are sparse in guitar tracks, we consider that *precision* is more important than *recall* and do not use any oversampling for the rest of our analysis.

6.4.2 Feature Importance

To assess the contribution of each feature, we conduct an *all bend* binary classification experiment where ↑, →, ↓ are merged into a single class, versus the ∅ class. Table 6.4 shows the importance of the eight most contributing features, computed using the random feature permutation technique (Breiman 2017) and monitoring its impact on the F_1-score of our model. Temporal and pitch features appear to have a higher impact on classification than position-related features, an observation confirmed by training the binary classifier

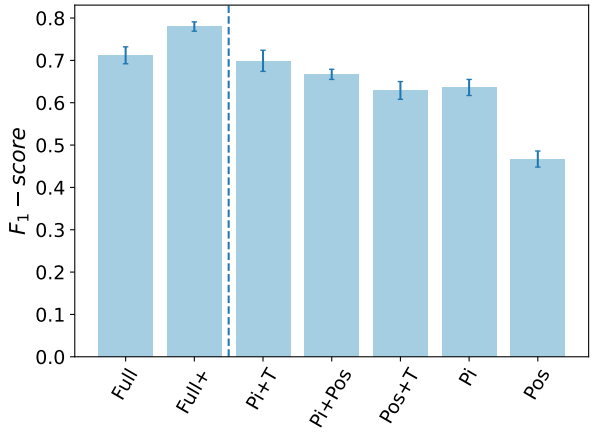


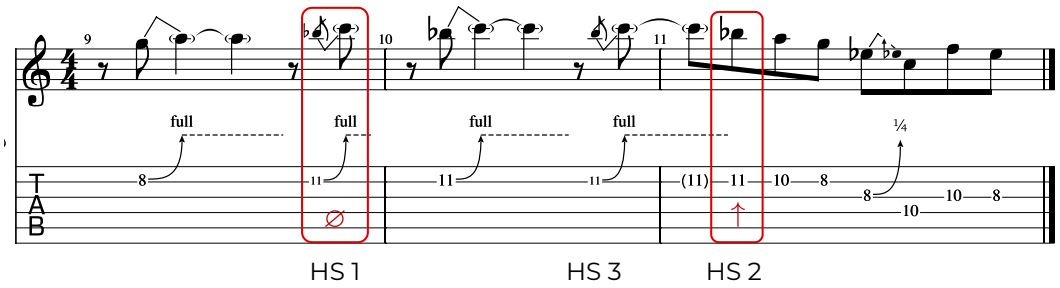
Figure 6.14: Average F_1-scores from 4 different train/test splits for the binary classification task. The leftmost part shows the performance of the decision tree trained on all features, with (Full+) or without (Full) SMOTE oversampling. The rightmost part corresponds to decision trees trained with a reduced set of features. *T* stands for temporal , *Pi* for pitch and *Pos* for position features.

on selected subsets of features. The results in Figure 6.14 indeed confirm the dominant influence of pitch features. However, adding gesture and temporal information improve the F_1-score by over 0.05. This result suggests that, while fret context contributes to induce bent notes, a large part of the prediction can be done from the strict pitch and rhythm content as it would be written in standard notation.

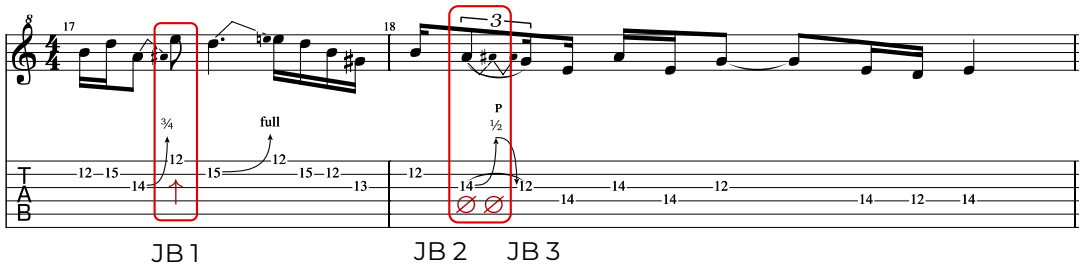
6.5 Prediction Analysis

In addition to the quantitative results presented in the last section, we present a qualitative analysis of selected predictions. Two examples were selected based on their representativeness of the errors commonly done by the model (considering the original tablature as the ground truth), and the decision path of the misclassified notes are studied to understand where the model diverged from the expected class.

Figure 6.15a shows one bent note wrongly classified as ∅ and, conversely, one non-bent note classified as ↑. Following the decision path provided by the decision tree (Figure 6.20), we can gain some insight on what feature differences have caused those wrong predictions. Both notes actually have more than half their decision path in common and split on their Pitch jump$^{(n+2)}$ value, their first divergence being due to future context. In particular, the second false prediction did not use any features related to past context. This might explain this error because the pitch could indeed be obtained by bending on the 10th fret by one semitone (an information that makes sense based on future context) but continuity with the previous notes called for playing the note without bend on the 11th fret (an information that should have been derived from past context). Another observation is that, in spite of similar context, the second bent note was misidentified whereas



(a) Excerpt from *Highway Star*, Deep Purple (composed and played by Ritchie Blackmore).

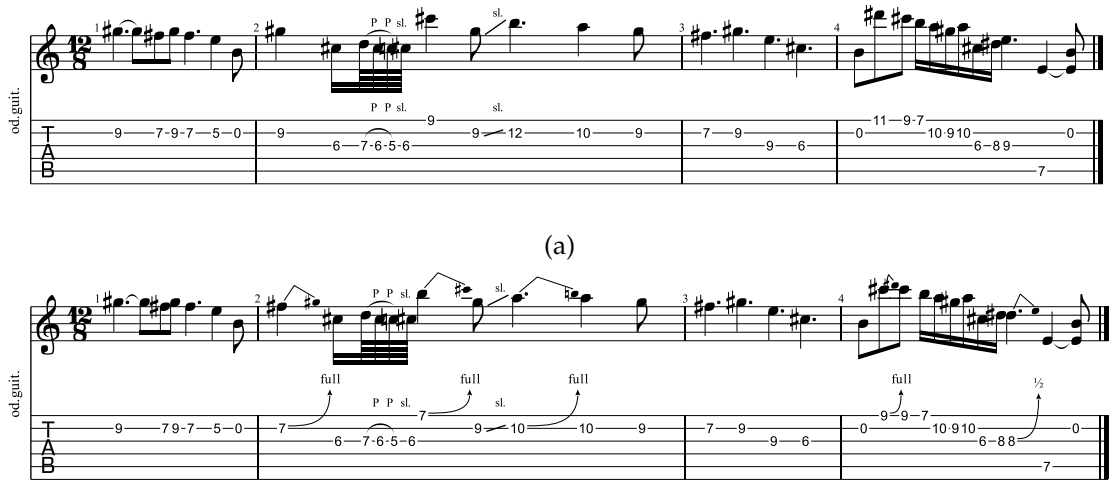


(b) Excerpt from *Jailbreak*, AC/DC (composed and played by Angus Young).

Figure 6.15: Examples of excerpts with predictions obtained with our Full Tree model. Predicted labels are shown in red. Only wrong predictions are shown for clarity. All other notes are labelled correctly. Decision paths for Highway Star (HS) and Jailbreak (JB) are shown Figure 6.20 and Figure 6.21, respectively.

the fourth bent note was not (Figure 6.20c). While those two notes look very similar at first glance, the latter has a longer duration because it is tied to the following eighth note, which illustrates the importance of the *duration* feature. An analysis of the decision paths indeed shows a divergence from the second decision rule, based on that feature. This highlights the presumably strong influence of rhythm in the classification of the first four bent notes, which bypasses pitch features.

Figure 6.15b shows a regular note wrongly tagged with a ↑ label (Figure 6.21a). The decision path for this prediction does not consider any feature related to the next note. It does however use many features concerning the second next note, which was correctly classified as ∅ , most likely because of its lower duration. The lack of information about the current note's position was probably critical in that case. Larger past context and fingering information might also have helped making the decision because the first note of the bar suggest putting the index finger on the 12th fret, which makes the bend very unlikely. The second error on that tablature is an *up & down bend* that was not identified, probably because of the short duration of the involved notes. Nevertheless, this example suggests that our method to obtain a bend-less transcription from an *up & down* bend might be detrimental to the algorithm performance. Indeed, our procedure has an impact on *duration* and *pitch jump*$^{(n\pm1)}$ which are among the most useful features to our algorithm. Despite those errors, we observe that our algorithm predicted correctly six bend labels in



(a)

(b)

Figure 6.16: Chopin's *Nocturne Op.9 No.2* arranged for guitar, without bends (top) or with bends suggested by our model (bottom).

the selected examples with a limited amount of false positives. These encouraging results suggest that our method could be used as a suggestion tool for the idiomatic use of bends.

Example of Suggested Bends for Non-Guitar Music To further demonstrate the capabilities of the model, we use it to suggest bends on a tablature arranged from non-guitar music. We start with the opening theme from *Nocturne Op.9 No.2* by Frédéric Chopin and a tablature arrangement is made manually (Figure 6.16a). Then, this arrangement is pre-processed to extract features for each note, and the trained decision tree is used to suggest where to add bends. Figure 6.16b shows the resulting tablatures, with bends added as per the model's suggestions. Because the model does not predict an amplitude for the bends, they were chosen manually and the fret positions updated accordingly. The resulting tablature is convincing and bends are located on notes where they can add both expressiveness and improve playability by reducing finger movement in some occasions.

6.6 Controllable Bends Suggestion

While the performance of the decision tree is satisfactory, the model does not allow to gradually add more bends to a tablature. Indeed, as we update the tree until all leaves are pure, decisions for each note are final and we lack a way to add more bends to a tablature gradually. For this reason, we experimented with replicating the experiments with an MLP (see section 2.3) so that the predictions can be between 0 and 1. While the predictions should not be considered equivalent to probabilities *a priori* (Guo et al. 2017), the fact that they are float values means it is possible to define a decision threshold to accept the prediction of the model anywhere between 0 and 1. In that way, a high threshold will only accept very confident predictions, which is equivalent to having a

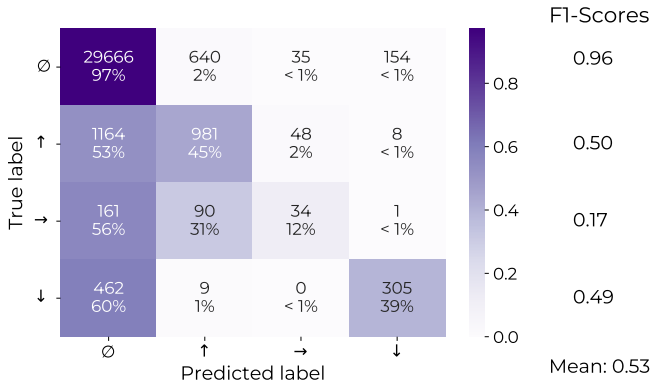


Figure 6.17: Confusion Matrix of the predictions of the MLP model on the same test set as the decision trees, with a threshold of 0.5.

small amount of bends due to class imbalance, while a very low threshold might add bends to almost all notes on the tablature.

We build an MLP model using `scikit-learn`, with 5 hidden layers of size

$$[50, 30, 20, 20, 10]$$

with a tanh activation function after each. The performance on the test set is slightly lower than the decision tree (Figure 6.17) but it would probably be possible to reach similar or even better performance by searching for better hyper-parameters. Because this was mainly intended as a proof-of-concept, we judged the performance sufficient.

In Figure 6.18, we show how bends are added when diminishing the decision threshold on the same input tablature. The main observation we want to share is that lowering the threshold increases the amount of bends as expected (even though the increase is not linear), but also that the bends are still added on meaningful notes that contribute to increasing the expressiveness of the melody. For instance, with a threshold of 0.3, bends are only suggested on long notes that begin sub-phrases of the riff. The last note being bent is not an ideal choice knowing that the 4 bars are looped, but given that the model did not have access to this future context, adding a bend on that spot allows to reduce finger movement efficiently. Besides, even with a very low threshold of 0.05, bends are still added on meaningful notes and playability is overall preserved. In particular, no bends are added on the final upward scale (except the last note), where bending a string would hinder playability.

6.7 Conclusions and Perspectives

In this chapter, we proposed a modelling strategy for guitar bends and discussed how these expressive playing techniques relate to both tablature and score content. Introducing a set of high-level features, we showed that a decision tree can successfully predict bend occurrences with satisfactory precision, in spite of the difficulty of the task due to the

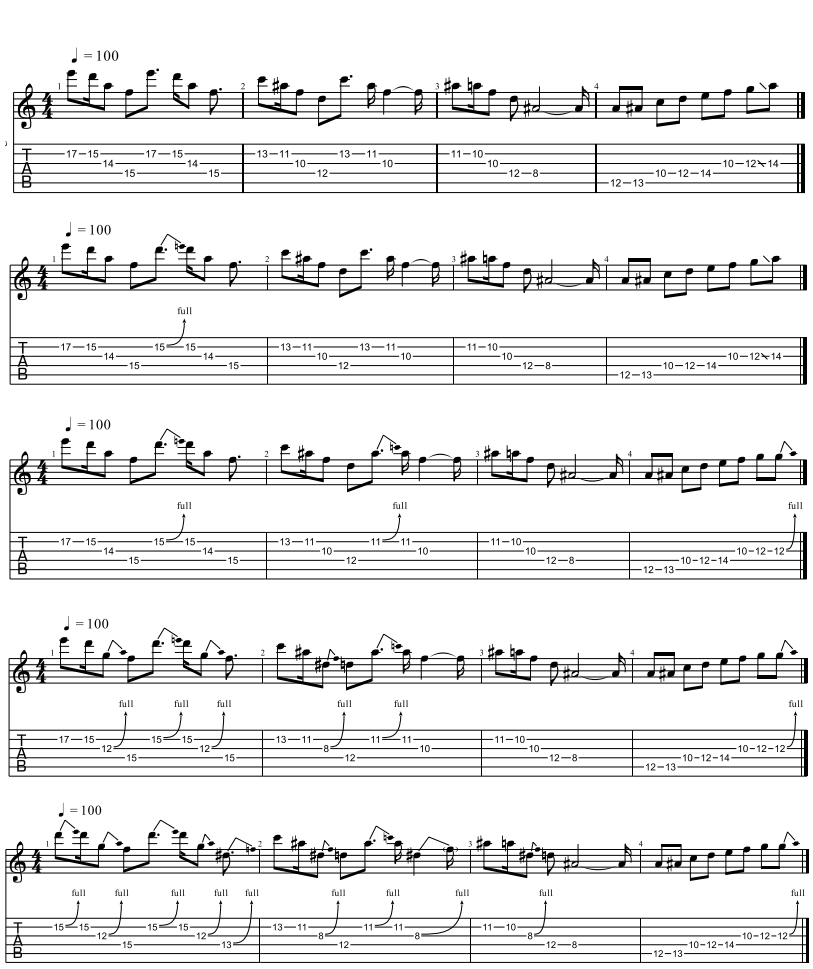


Figure 6.18: Tablature excerpt with bends suggested by the MLP model with different thresholds. Tablature arranged from the saxophone theme of *Careless Whisper*, George Michael.

low proportion of bent notes in guitar music. However, the low performance on predicting → labels suggests that our modelling choices could be improved and that held bends might not be considered as an expressive technique but rather another way of playing regular notes. It is also possible that our approach, because it is not based on an RNN, lack sequential information that would allow to predict → bends that are informed by past (and possibly future) bends. An advantage of our approach is the use of a lightweight and explainable algorithm, facilitating its use in an assisted-composition context. In future work, this approach could be extended to other guitar playing techniques, and might benefit from adding more context information like the chord being played over a bar, using rhythm guitar parts aligned with lead guitar, etc. While less explainable, it might also be worth using more complex models like BERT as it proved efficient on predicting tablature from standard notation (Edwards et al. 2024). Because bends are arguably more commonly performed with the ring finger and little finger than other fingers of the

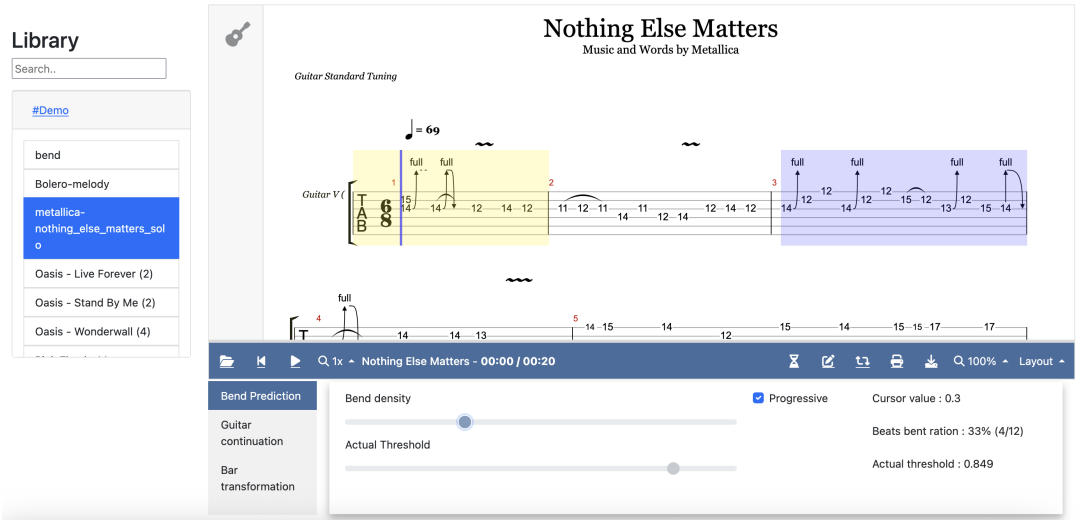


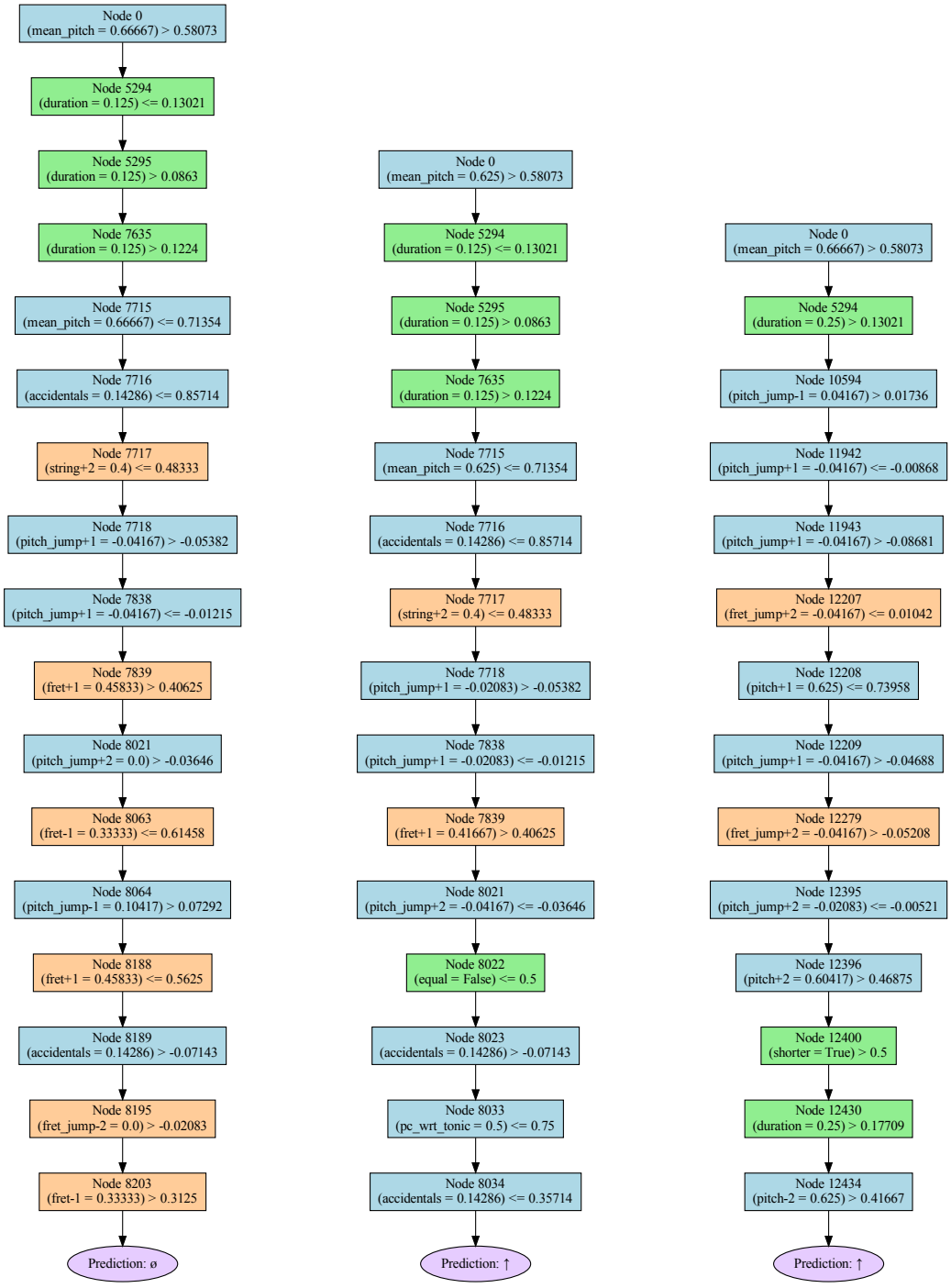
Figure 6.19: Screenshot of the demonstration interface made by Léo Dupouey in alphaTab. The first slider at the bottom controls the decision threshold for bend suggestion.

fretting-hand, combining our work with finger prediction technique (Hori 2021) might also improve prediction performance. Besides, our modelling strategy could also be used to study the playing style of specific guitarists, and evaluate the potential of bends for automatic guitarist identification (Sarmento et al. 2023b).

Finally, to better help tablature composers use this model and move from *prediction* to *suggestion*, we studied using an MLP model and controlling the output threshold to add bends gradually. For it to be really useful, however, it should be integrated within the tool already used by guitar composers like Guitar Pro. Léo Dupouey worked as a research engineer at LaBRI with that goal in mind, and integrated the bend prediction MLP in an online tablature visualisation website based on AlphaTab.[2] A screenshot of the system is shown Figure 6.19. Next steps should include interviewing guitar players and receive their feedback on this tool, as conducted by Baptiste Bacot in preliminary works (Bacot et al. 2024). Because the system aims at providing interactive suggestions, interviews would permit guitarists to express directly the strengths and weaknesses of the model. Adding support of other techniques and suggesting the amplitude of bends are likely to be among the first improvements requested.

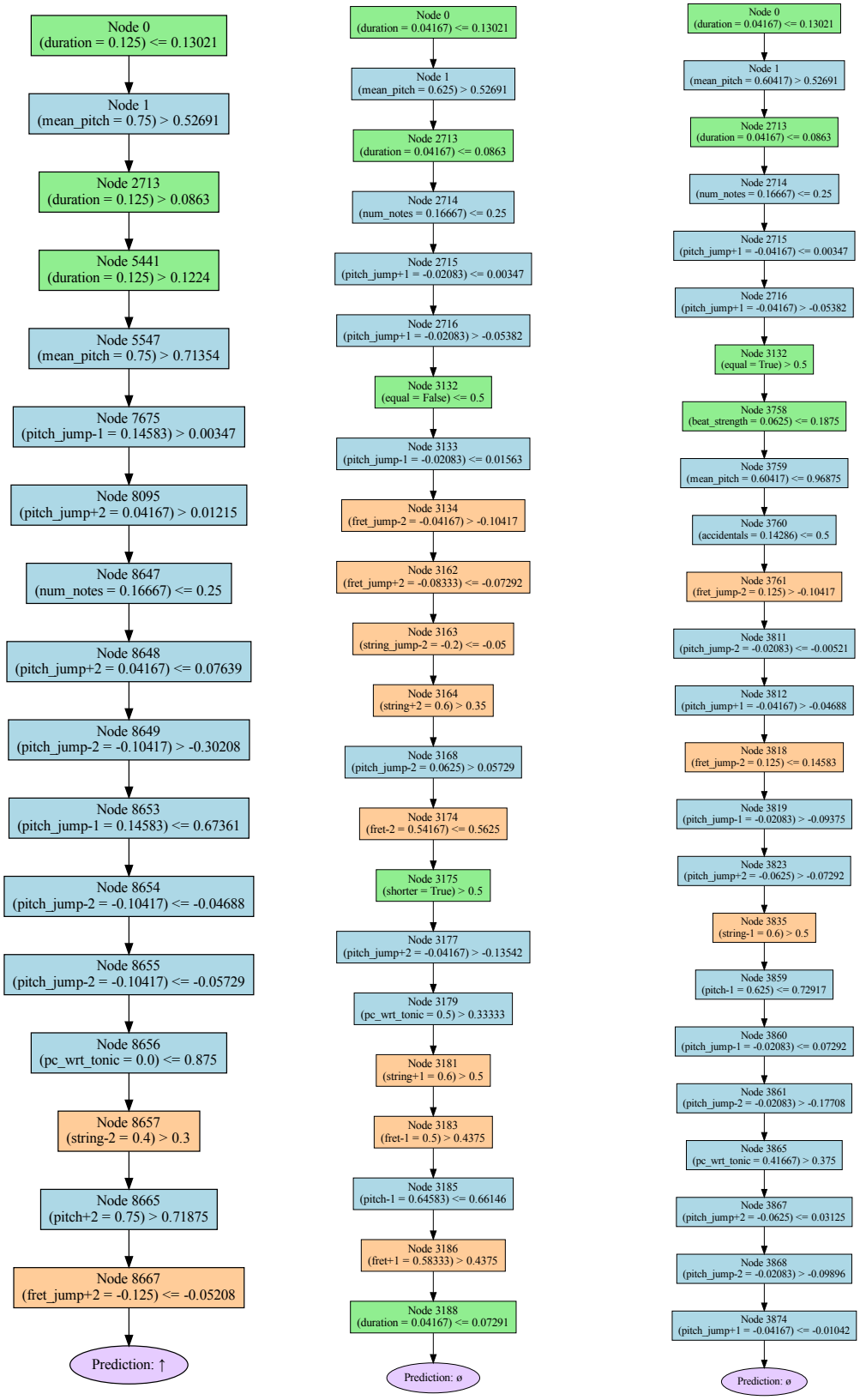
All code related to this chapter is made publicly available (parsing of `.gp` files, extraction of features, training, and evaluation of bend classification models) under a GPL-3.0 license. We also release the complete set of features extracted on each note of our corpus under an ODbL license at: http://algomus.fr/code/.

[2] https://alphatab.net/, accessed in June 2025.



(a) HS 1 (b) HS 2 (c) HS 3

Figure 6.20: Decision paths for the two wrong predictions in Highway Star (HS) as well as the correctly predicted fourth bend (rightmost path).



(a) JB 1 (b) JB 2 (c) JB 3

Figure 6.21: Decision paths for the three wrong predictions in Jailbreak (JB).

7

Guitar Chord Diagram Suggestion

"It's just a pretty chord you know, it's so versatile."

Tuck (2025)

Contents

Choosing a position on the fretboard to perform a chord, depending on context, requires specific skills for guitarists who play accompaniment parts, A C major chord can for instance be played, without inversions, in 5 different positions on a guitar in standard tuning,[1] each with different biomechanical and timbral characteristics. In this chapter, we propose a model that suggests a chord diagram,[2] given a chord label and the previous notated diagram. The contributions of this work are as follows: (i) a context-aware approach for guitar chord diagram suggestion; (ii) a set of metrics to assess performance in this task and characterise texture for guitar chord diagrams; (iii) openly released code and data for all of the above. The results of this chapter have been published in:

"Guitar Chord Diagram Suggestion for Western Popular Music"
Alexandre D'Hooge, Louis Bigo, Ken Déguernel, Nicolas Martin.
Proceedings of the 21st Sound and Music Computing Conference (SMC), 2024.
(D'Hooge et al. 2024a)

[1] https://www.scales-chords.com/chord/guitar/C, accessed in June 2025.

[2] In this chapter, we use chord diagram as a metonym to chord position to avoid any ambiguity from using the word "position".

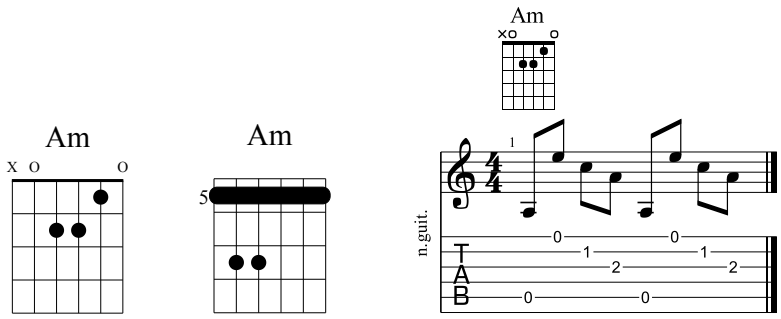


Figure 7.1: Two guitar chord diagrams for an A minor chord, and a diagram above a tablature. A number on the side of a diagram indicates the "root fret" of that diagram, i.e. the lowest fret used.

7.1 Suggesting Guitar Chord Diagrams

A large part of the WPM guitar repertoire is available as "songbooks" which contain lyrics of songs with the corresponding chords (Tostig 2025). However, like single notes, a chord can be played in multiple ways on a guitar's fretboard, each position having its own pitch, timbral and biomechanical specificities. To help guitarists use appropriate chord positions, songbooks will usually contain *chord diagrams*.

Definition (Chord Diagram, Polin 2022) *A chord diagram graphically conveys the section of the neck on which the chord is placed. In a diagram, each note fretted is represented by a dot [...]. The Xs and Os situated at the top of the neck show you if the string beside which the symbol appears should be played "open" or not.* 📖

Chord diagrams can also be used on top of tablature notation to help players know what fingerings to use at a glance, before reading the details of the tablature. An example of those *chord diagrams* is shown Figure 7.1. The first diagram presented contains both an open (`0`) and a muted (`x`) string and is presumably one of the most common positions used to perform an A minor chord. It is typically used by beginners for its simplicity, and in pop music which tends to favour the resonance of open strings. The latter is a *barre* chord and is harder to play, but its hand position can be shifted along the fretboard to directly play minor chords with a different root note. Chord positions can also be reduced to a text format, compatible with ASCII tablatures, with a number indicating which fret is played on each string. For instance, the chord diagrams from Figure 7.1 can respectively be annotated as `x.0.2.2.1.0` and `5.7.7.5.5.5`.

In this chapter, we propose the task of chord diagram suggestion to assist composition. The principle is to provide guitarists with diagrams suggestions so that they can explore new ways of playing chords, rather than use a reduced number of positions they already master. For the suggestions to be useful, we suggest taking into account the previous diagram (Figure 7.2). That way, we can ensure that two consecutive chords share similar musical characteristics and that the transition from one another is not too difficult.

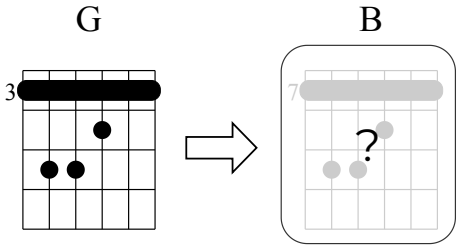


Figure 7.2: Summary of the diagram suggestion task. The label of the next chord as well as the previous diagram are known.

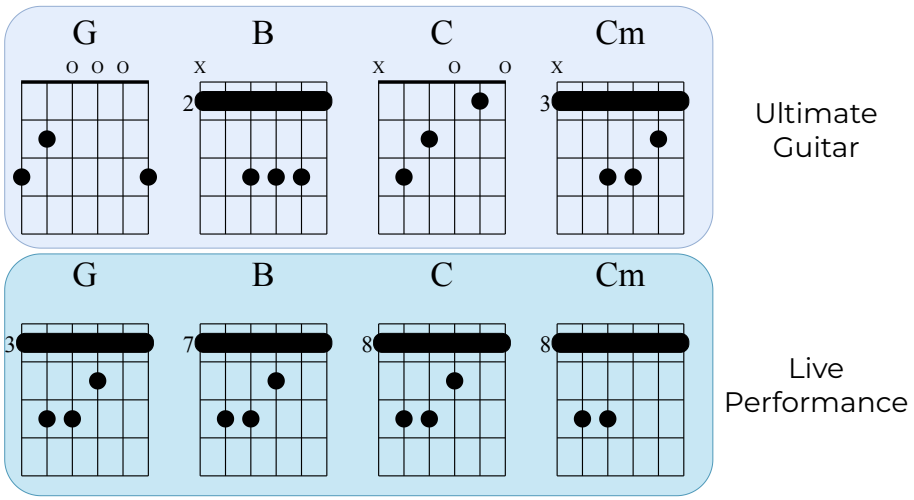


Figure 7.3: Diagrams presented for *Creep* by Radiohead, on Ultimate Guitar (top). Those are not the chords used by the guitarist in live performances (bottom).

Chord diagram research has so far been focusing on playability (Sawayama et al. 2006; Vélez Vásquez et al. 2023; Wortman et al. 2021) with limited ability to take into account the relationship between consecutive chords. There have been studies on generating chord sequences (Dalmazzo et al. 2024b), and voicings can also be generated (Dalmazzo et al. 2024a) or adapted to a new musical style (T.-P. Chen et al. 2020). However, while voicings specify how notes in a chord are organised, exact chord diagrams are not included in those pipelines. We can see similar problems in the suggestions of the most popular guitar learning services, which only provide catalogues of standard chord diagrams (Figure 7.3). While such an approach provides agency to the user, it might also drive beginners towards using the same diagrams over and over again. Besides, playing a song in the "wrong" way can be more complex, as in *Creep* all chords are played with the same *barré* shape moved along the fretboard which makes it easier playing the repetitive arpeggio pattern of the Intro.

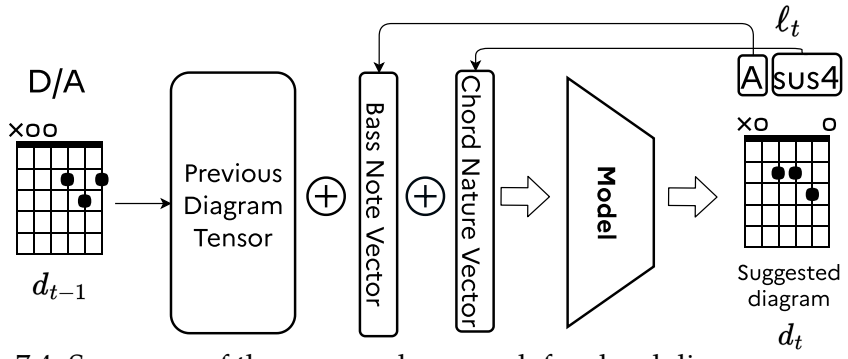


Figure 7.4: Summary of the proposed approach for chord diagram suggestion.

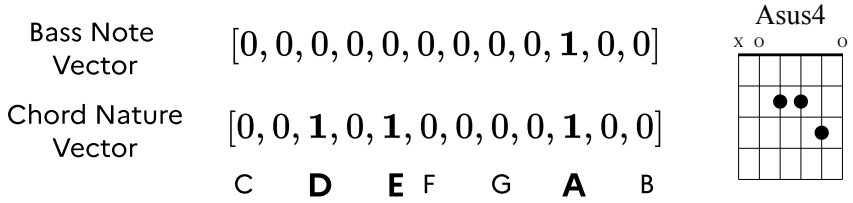


Figure 7.5: Computed vectors for an *Asus4* chord.

7.2 Methodology

In this chapter, we aim at suggesting a diagram d_t for a chord, based on its label ℓ_t and the previous diagram d_{t-1}. While one could imagine using more past information and looking at n chords in the past, this chapter present a first experiment in the task of diagram suggestion and aims at showing that looking at just one previous chord already improves the quality of the diagram suggested. In this section, we present the model we use for diagram suggestion, as well as the vector representation for the input data. We then present the final model architecture that was implemented.

7.2.1 Proposed Model

The task consists in finding the diagram which probability is highest in the provided context. To do so, we convert the chord labels and diagrams into vectors and train an MLP on predicting a chord diagram, given a chord label and the previous chord diagram. For each chord label, we first extract its bass note, which might be different from its root note in case of inverted chords (also called slash chords in WPM). This bass note information is then converted into a *one-hot* vector of size 12 (where enharmonic equivalents of the twelve-tone equal temperament are merged together) we call the *Bass Note Vector*. Then, the pitch-class content of the chord is converted into a *many-hot* vector of size 12: the *Chord Nature Vector* (see Figure 7.5). The complete pipeline we propose for the task of diagram suggestion is represented Figure 7.4.

The diagrams are converted into many-hot arrays of size $[6, 25+1]$, each row accounting for a string (6 on a standard guitar) and its 25 frets (counting the open string as a zeroth fret). An additional per-string coefficient is added to account for muted strings. This ensures that all vectors always have a non-zero value for a string, which ultimately permits normalising predictions and using them as probabilities. In summary, input data contains 180 values for the previous diagram d_{t-1} (6×26 values) and the new chord label ℓ_t (2×12 values), from which the bass note and chord nature vectors are computed. As a result, the model outputs 156 probabilities, 26 per string, for d_t.

Finally, to measure to which extent adding information about the previous diagram improves chord diagram suggestion, we define the baseline as the same model, but removing information about d_{t-1}. In that case, the baseline model takes only 24 values as input, but still returns 156.

7.2.2 Implementation Details

We use a fully connected neural network as an architecture (see section 2.3) for both the baseline and the full suggestion model. The hyperparameters of the two architectures were tuned through a grid search to maximise performance. We report here the final configurations that were obtained.

Baseline Model. The best performance of the baseline model was obtained with no hidden layer, i.e. a regular perceptron. This means that the baseline model is entirely represented by a matrix of its weights of size $(24, 156)$ and a bias vector of size $(1, 156)$, adding up to 3 900 parameters.

Full Model. The full model performed best with one hidden layer of size 150 with an identity activation function before the output layer. It therefore has two weight matrices of size $(180, 150)$ and $(150, 156)$, respectively, and bias vectors for each with 150 and 156 coefficients. This model therefore has 50 706 parameters.

Training. Performance during training is evaluated with a BCE loss on the output, obtained after a sigmoid activation function, and string-wise normalisation of the predictions (to enforce a single prediction per string). Training uses the Adam optimiser with $(\beta_1, \beta_2) = (0.9, 0.999)$, and a learning rate $\lambda = 0.001$. The training is stopped whenever the validation loss does not improve by at least $\delta = 0.001$ for two consecutive epochs. Full training of the model can be done in a few minutes on a standard laptop Central Processing Unit (CPU).

7.3 Data

This work on diagram suggestion marked a transition from using mySongBook to using DadaGP as a dataset. In this section, we describe how chord pairs were extracted from

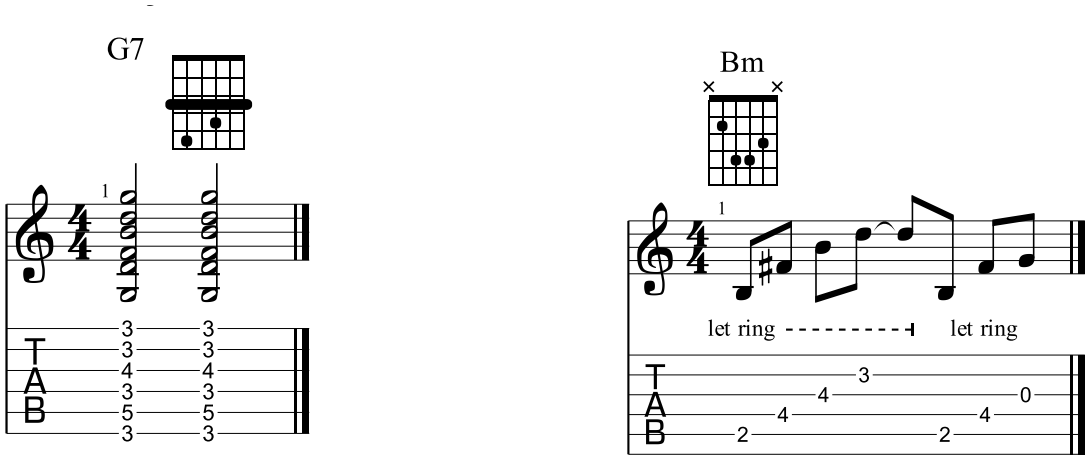
both datasets and conduct statistical analyses on the most common labels and diagrams encountered. We finally discuss the data augmentation strategy implemented to remove any bias towards specific key signatures.

7.3.1 Corpora

This study is conducted on both DadaGP and MSB. The `.gp` file format of these datasets allows including chord diagrams aligned with the tablature data. We only use tracks which contain chord diagram data, reducing the datasets to 2 766 (10 %) and 520 (25 %) tracks for DadaGP and MSB respectively. From these tracks, we extract chord pairs that occur within a 2 bars interval and finally keep one occurrence of each unique transition $(\ell_{t-1}, \mathrm{d}_{t-1}) \rightarrow (\ell_t, \mathrm{d}_t)$ per track. By doing so, we acknowledge the fact that a chord transition can be more common in a given repertoire and occur in many songs, but we remove duplicate transitions that are inherent to the repetitive nature of WPM. This processing step leaves us with 31 321 and 7 365 unique diagram transitions for DadaGP and MSB, respectively. While those numbers can already seem high, analyses of the datasets suggested that even more diagram pairs could have been extracted from the tablatures. The main limitation is that transcribers do not always take the time to explicitly engrave diagrams in the tablature they write. Indeed, a common observation on tablatures was that chord labels would be written, but the corresponding diagrams were not specified. The data could thus be greatly expanded by developing a way to derive chord diagrams from tablature excerpts. There might be situations where it is straightforward (Figure 7.6a) but there are also many times where it is not as clear (Figure 7.6b) because possible ornamentations or out-of-diagram notes can be used. It can also happen that the chords played on the guitar are "missing" some notes that are played on other instruments. Overall, musical context is key in determining what really is the correct chord diagram for an excerpt. While a human annotator might be able to decide what diagram is appropriate based on such context, the variety of situations that can occur make the automatic recognition of diagrams from tablatures a non-trivial task, we leave it for future work.

7.3.2 Statistical Analyses

In this subsection, we provide detailed statistics of the chords used in the datasets. This analysis aims at emphasising the bias towards most common diagrams and key signatures. Furthermore, because of the considered repertoire, and the large amount of pop, rock, and metal tracks, some chord types are more common than others. Nevertheless, we also want to underline that this bias is not at the expense of variety, even though it results in a highly unbalanced distribution of chords. A first dimension to analyse on chords is the root note they are built on. Figure 7.7 shows that the distributions of root notes are similar between datasets, with more than half the chords having *A*, *G*, *D*, *C* or *E* as root. Perspectives include comparing those distributions with chords used in WPM in general to determine whether this bias is specific to guitar or a bias of rock and metal songs.



(a) (b)

Figure 7.6: Two tablature excerpts where it would be straightforward (left) or ambiguous (right) to find the chord diagram from the notes played (and eventually the chord label). In the second example, more context would be required to confirm that this chord is a B minor and not a possible inversion of a Gmaj7, i.e. whether the 3rd string in the diagram should be with a note on the 4th fret or an open string.

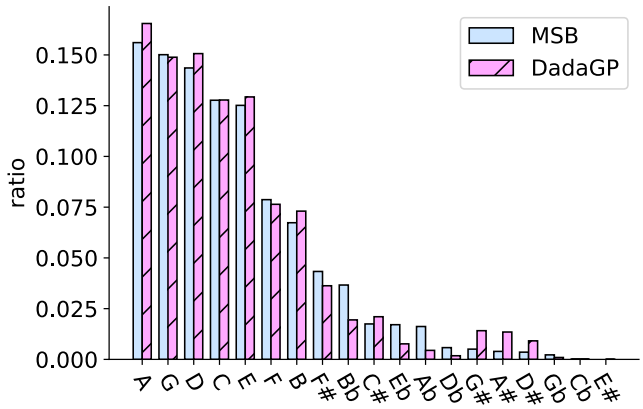


Figure 7.7: Distribution of the root notes of chords in both datasets.

After root notes, we can also compare chord natures. In both datasets, major chords (M) are the most common (Figure 7.8), followed by minor chords (m) and *power chords* (5). The tail of the distribution then includes more complex chords, like seventh, suspended or added-tone chords. The distribution of chord natures is in fact highly unbalanced towards the three first classes, which might make the suggestion of diagrams for less common chords a bigger challenge. Besides, this plot shows that notation between files can vary, MSB containing *7M* and *maj7* labels that are both representing major seventh chords. Those notation discrepancies are not an issue when we convert the chord labels to vectors, but they illustrate that identical chords can be labelled differently. It is worth noting that those labels distribution might not be entirely accurate, as it was observed (especially in the DadaGP dataset) that chords tend to be wrongly labelled. For instance, we observed

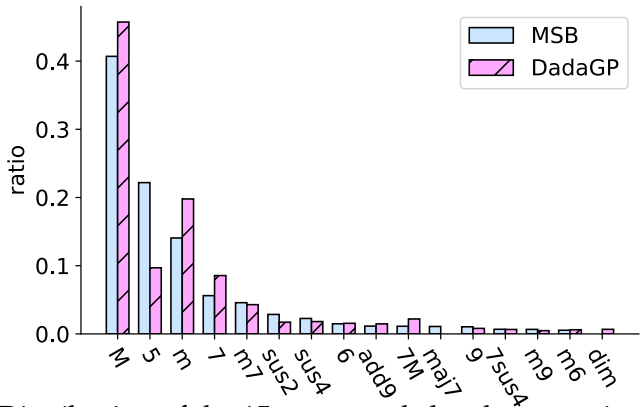


Figure 7.8: Distribution of the 15 most used chord natures in each dataset.

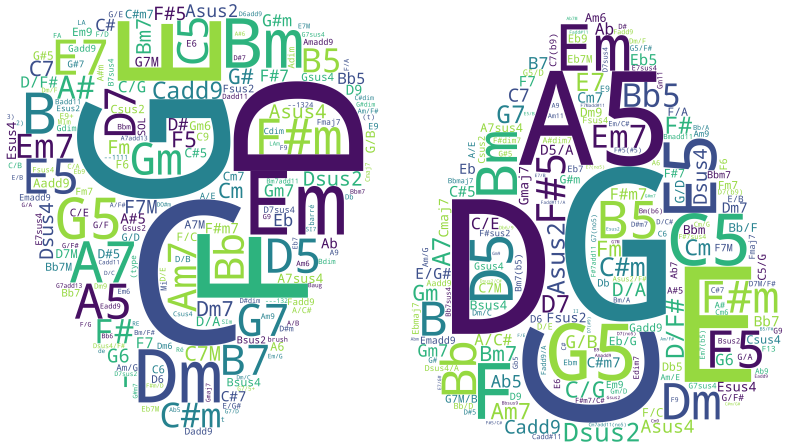


Figure 7.9: Word cloud representation of chord labels in DadaGP (left) and MSB (right)

in metal songs that power chords are often labelled as major chords, possibly because chords are considered to be power chords "by default" in this genre. We also noticed that slash chords were not always labelled with the correct bass note. In our work, we decided to trust chord labels nonetheless, because most are still correct, but future work could explicitly verify chord labels to make the system more robust to transcription errors.

Word cloud representations of the chord labels are also provided (Figure 7.9) to give an overview of the combined root notes and chord natures. Overall, one can observe that the comments made on root notes and chord natures separately still apply to complete chord labels.

While chord labels are already numerous and varied, even more variety comes with the different diagrams that can be used for each label. One can get a sense of it through the median number of diagrams per label: 11.5 for MSB and 24 for DadaGP. This observation illustrates the number of possibilities when suggesting a diagram for a chord, while also showing how a larger dataset comes with more variety. This variety of data is particularly noticeable on the most common chords, G Major having 33 and 108 different diagrams in MSB and DadaGP respectively. The variety of diagrams observed comes mostly from the

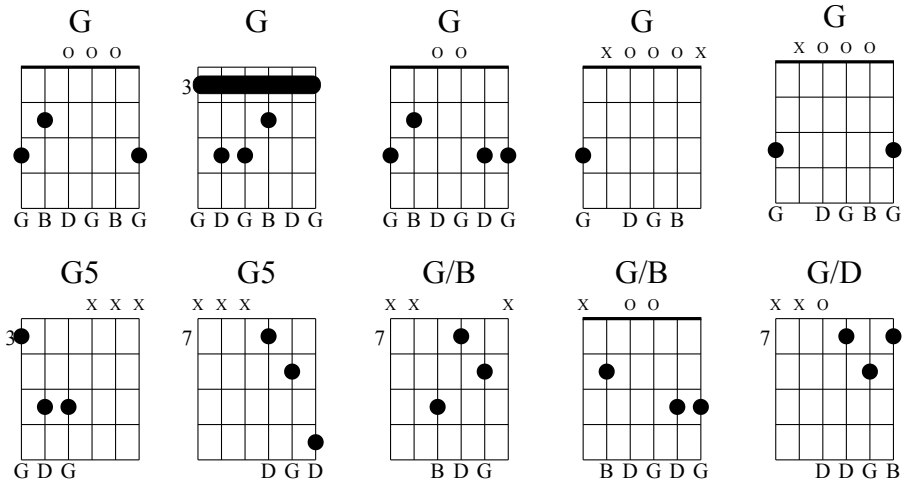


Figure 7.10: 5 most common diagrams for a G major chord in DadaGP (top), and 5 random diagrams of chords that are wrongly labelled as "G" (bottom). Below the diagrams are the notes played on each string. The G major chord is supposed to contain the notes G, B and D. The bottom row indicates the correct label of each chord.

fact that transcribers might write some strings as muted instead of open or played, which can yield different diagrams for the same hand position on the fretboard. The staggering amount of diagrams in DadaGP comes both from the same issue, worsened by the size of the corpus, and occasional incorrect labelling. The 5 most common G major diagrams from DadaGP, as well as 5 diagrams wrongly-labelled as "G" are presented Figure 7.10.

7.3.3 Data Augmentation Strategy

The previous statistical analyses showed that a wide variety of diagrams and labels are represented in both datasets. However, some chords are over-represented and might prevent the model to suggest less common diagrams. We want to improve the robustness of the proposed approach and reduce its bias towards key signatures that are more frequent in the datasets so that the model can suggest diagrams even for rare tonalities and chord natures. To do so, we apply the augmentation technique from McVicar et al. (2015). For each chord pair, both diagrams are shifted one fret down and the chord labels transposed one semitone down accordingly, until one of the diagrams contains an open string. Similarly, chord pairs are also shifted one fret up until reaching the 15th fret. This maximum is chosen based on the highest diagrams observed in data and to prevent the model from suggesting chords on higher frets, which are uncommon in rhythm guitar. Using this augmentation strategy makes the training sets more than three times larger. We also confirm with Figure 7.11 that the augmentation strategy allows to reduce pitch class imbalance, even though some notes are still encountered more often than others.

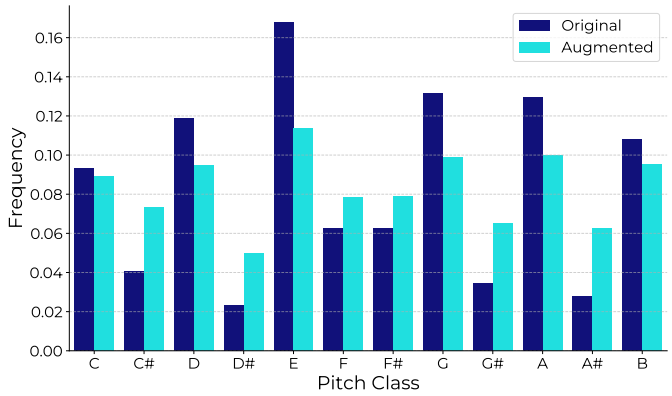


Figure 7.11: Pitch class histograms of the DadaGP training set chords before and after data augmentation.

7.4 Experiments

As we illustrated previously for the G major chord, there can be multiple possible diagrams with only one note changing. If the model were to suggest `3.2.0.0.3.3` instead of `3.2.0.0.0.3` for a chord pair in the dataset, the difference is minimal and should not be considered an error *per se*. To account for those possible situations, we introduce several new metrics to evaluate the quality of the diagrams suggested, and present the results of our model according to those metrics.

7.4.1 Evaluation Metrics

Like most tasks on probability estimation, we can evaluate the performance of the model through its Precision (P), Recall (R) and F1-score. However, in the case of diagram suggestion, it could be argued that the so-called ground-truth is not the only acceptable diagram. For this reason, we propose hereafter several automated metrics to account for the possible differences that can be encountered between the suggested diagrams and the reference ones.

Pitch Metrics. Similarly to Wiggins et al. (2019), we want to measure to which extent the suggested diagrams include all pitch-classes of the target chord labels. For this purpose, we compute for a chord diagram d the set of pitch classes $\mathcal{S}_P(d)$ it contains and compare it with the pitch classes associated with the chord label ℓ: $\mathcal{S}_P(\ell)$.
From these sets, we can compute Pitch Precision (P_P), Pitch Recall (R_P) and a Pitch F1-score ($F1_P$). For each pitch class p_i from the expected ones $\mathcal{S}_P(\ell)$, a True Positive means that $p_i \in \mathcal{S}_P(d)$, a False Negative is when $p_i \notin \mathcal{S}_P(d)$ and a False Positive is when there is a pitch class p_j in the diagram $\mathcal{S}_P(d)$ with $p_j \notin \mathcal{S}_P(\ell)$. True Negatives are undefined with our definition because we base our values on pitch classes that are either in $\mathcal{S}_P(d)$ or $\mathcal{S}_P(\ell)$. We thus set the amount of True Negatives to 0 in all our formulae.

Fretboard Metrics. Drawing insights from Wiggins et al. (2019), we also define metrics to measure how similar the model suggestions are to the reference diagrams, when

comparing them on the fretboard. Similarly to the pitch metrics, we compute the set of string-fret (SF) pairs on the predicted diagram and compare them with those of the expected one. We can use those sets to compute String-Fret Precision (P_{SF}), String-Fret Recall (R_{SF}) and the corresponding F1-score ($F1_{SF}$). The definitions are equivalent to those of pitch metrics.

Detect Unplayable Diagrams. To evaluate how often the model can return an unplayable diagram, we implement a playability metric inspired by Wortman et al. (2021). In this paper, the authors define an *anatomical score* to evaluate the ease of playing a chord. They first define a score for pairs of fingers to determine if the distance between them is comfortable:

$$SF(x, a, b) = \begin{cases} 1 + (x - 0.99a)^3 & x < a, \\ 1 - \left(\frac{x - 0.99a}{1.01b - 0.99a}\right)^2 & x \geq a. \end{cases} \tag{7.1}$$

with x the distance between two fingers, and a and b the minimum and maximum comfortable distance values, respectively. Those comfortable values were obtained from a "model of the player's hand" and for instance state that the comfortable range for the distance between the index finger and middle finger is between 5 mm and 80 mm. In the original paper, users can adjust the comfortable ranges to better model their own hand, but we use the default values of the paper for our metric. From the finger-pairs scores, the final anatomical score of a diagram d can be measured:

$$AS(d) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} SF(x_{ij}, a_{ij}, b_{ij})}{\max(n^2 - n, 1)} \tag{7.2}$$

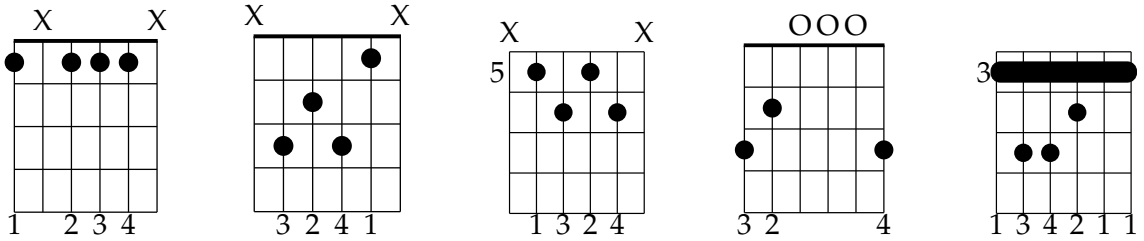
with n the number of fingers used for playing the chord, and x_{ij}, a_{ij}, b_{ij} the observed distance and the min and max of the comfortable distance range between fingers i and j. This metric allows us to detect unplayable diagrams by their low anatomical score. We settled for an anatomical score threshold of $t = 0.2$ after manual analysis of anatomical scores and playability of the model's suggestions.

Ease of Transition. While it is necessary that the suggested diagrams are playable, playability of chord sequences also depends on the chord transitions involved. To assess the ease of the transitions suggested, we also implement the chord change metric proposed in Yazawa et al. (2014). This metric analyses the transition between two diagrams through two movements: the *wrist movement* m_w, which is obtained as the absolute difference of the index finger fret in the two chords; and the *fingers movement* m_f, defined as the Manhattan distance between the positions of each finger. We define the final value for the ease of transition as:

$$\tau = \frac{1}{1 + m_w + m_f} \tag{7.3}$$

τ is highest (1) when the chord change is easy and close to 0 when the transition is hard.

The computation of the last two metrics requires to know which fingers of the fretting hand are used. This information was derived using a naive heuristic based on observations of the most common chords in the datasets. For these chords, the index finger



(a) Fm7 (b) C7M (c) Dm7♭5 (d) G (e) G

Figure 7.12: Guitar Chord Diagrams with possible fingerings below. Fingers are numbered as such: 1 = index, 2 = middle, 3 = ring, 4 = little.

is always on the lowest fret (closest to the guitar head) and the lowest string played on that fret. Then, the middle, ring and little finger follow, ordered from lowest to highest fret and lowest to highest string. A few examples to illustrate this principle are provided Figure 7.12. Those rules work well with chords using four fingers, but might be wrong when only three fingers or less are used. The G open chord illustrates this limitation (Figure 7.12d) because the most common shape usually does not use the index finger. Our heuristic also does not account for chords with a *barré* with another finger than the index. Future work could benefit from implementing existing work on chord fingering prediction to better estimate the fingers used (A. M. Barbancho et al. 2012; Hori 2021).

7.4.2 Results

Experiments were performed by applying a 60-20-20 train-validation-test split to both datasets and average the results over four different splits. Final results are presented Table 7.1. For fair comparison with the baseline, that does not take the previous diagram d_{t-1} into account, chord pairs which have identical ℓ_t and d_t but different d_{t-1} are considered duplicates and skipped during testing. The data augmentation strategy presented earlier is used for the full model but not for the baseline as it decreased its performance. This observation suggests that the results of the baseline should be taken with care, as prediction quality might decrease significantly on key signatures less represented in the training set. The first observation from Table 7.1 is that the proposed model surpasses the baseline on standard and string-fret F1-scores. However, both implementations perform well on pitch metrics, showing that the proposed diagrams contain overall the expected pitch content. There is also a significant improvement over the baseline on string-fret metrics, suggesting that information from the previous diagram helps the model choose the correct fretboard area.

Unplayability and transition ease of the ground-truth and the models' prediction are shown in Table 7.2. It can be observed that the proposed model suggests unplayable diagrams 15% of the time, which is moderately more than the baseline, with a clearer gap on DadaGP. As a reference, 3-4% of the ground-truth diagrams are deemed unplayable, most of them because the metric does not recognise the *barré* technique with other fingers than

Table 7.1: Results of the baseline and the full model on MSB (top) and DadaGP (bottom). Precision and Recall measures were omitted for clarity.

		F1	$F1_P$	$F1_{SF}$
MSB	Baseline	0.40±.01	0.87±.01	0.46±.04
	Full Model	0.72±.02	0.90±.01	0.67±.02
DGP	Baseline	0.38±.01	0.88±.00	0.45±.02
	Full Model	0.63±.01	0.88±.01	0.60±.02

Table 7.2: Ratio of unplayable diagrams and average ease of transition τ measure on MSB (top) and DadaGP (bottom).

		Unplayable diagrams	Ease of Transition
MSB	Baseline	0.12±.02	0.15±.02
	Full Model	0.15±.03	0.21±.03
	Test Set	0.03±.02	0.31±.02
DadaGP	Baseline	0.09±.01	0.13±.01
	Full Model	0.15±.01	0.19±.01
	Test Set	0.04±.01	0.29±.01

the index (Figure 7.13). However, the proposed diagrams permit slightly easier transitions than the baseline, which is probably again due to the context information that keeps the suggested diagram in the same fretboard area, limiting wrist and fingers movements. Nonetheless, even our model with context information has a 0.1 difference with the transition ease measured in the datasets, showing that the transitions are still too complicated. Examples of simple and complex transitions are shown Figure 7.14. Nonetheless, the evaluations of the transitions should be taken with care since the heuristic that predicts fingerings for chords does not take context into account. While a guitar player would usually look ahead and adapt its current hand shape to incoming chords, our heuristic does not, and can cause the estimated transitions to be more complex than they should be.

Overall, one can note that performance is similar on MSB and DadaGP. We still decided to share results on both datasets because we deemed relevant the fact that the proposed approach improves diagram suggestion even when training on data where chords are not always labelled correctly. We also observe that a larger amount of chord pairs with an unbalanced distribution does not significantly increase the bias towards common diagrams.

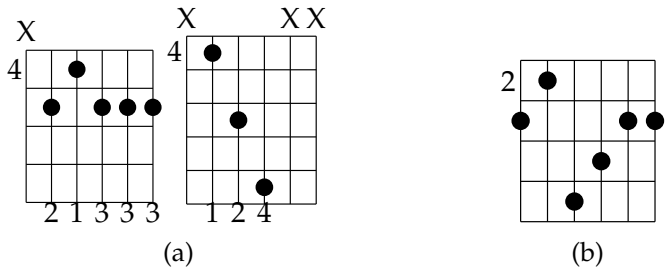


Figure 7.13: Two diagrams from the dataset (left) and one generated by the full model (right) detected as unplayable with our metric. The two from the datasets are actually playable with the provided fingerings, but are not supported by our heuristic.

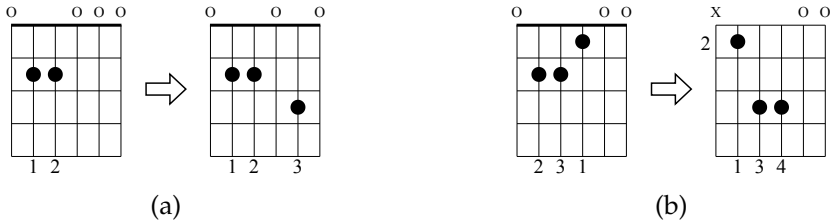


Figure 7.14: Easy (left) and hard (right) chord transitions.

Table 7.3: Texture metrics for chord suggestions of the proposed model (Full), the baseline (BL) and the corresponding test set (data). $\delta(.)$ denotes the absolute difference of the previous metric between the two chords of a transition. Bold values are the ones significantly closer to the reference dataset.

	DadaGP			MSB		
	BL	Full	Data	BL	Full	Data
Muted notes	0.15 ± .01	**0.27** ± .01	0.28 ± .01	0.32 ± .02	0.32 ± .01	0.33 ± .01
$\delta(.)$	0.20 ± .01	**0.08** ± .00	0.10 ± .01	0.18 ± .02	**0.08** ± .01	0.08 ± .01
Open Strings	0.25 ± .01	**0.19** ± .02	0.20 ± .01	**0.20** ± .01	0.17 ± .02	0.20 ± .02
$\delta(.)$	0.22 ± .01	**0.15** ± .01	0.17 ± .00	0.19 ± .01	**0.15** ± .01	0.15 ± .01
String Centroid	0.53 ± .00	0.54 ± .01	0.55 ± .00	0.60 ± .01	0.59 ± .01	0.59 ± .01
$\delta(.)$	0.10 ± .00	**0.05** ± .00	0.06 ± .00	0.11 ± .01	**0.05** ± .01	0.06 ± .01
Unique Notes	0.61 ± .00	**0.71** ± .00	0.80 ± .01	0.68 ± .01	0.72 ± .01	0.80 ± .01
$\delta(.)$	0.21 ± .01	**0.13** ± .00	0.12 ± .00	0.20 ± .02	**0.13** ± .01	0.11 ± .01

Diagram Texture Consistency Finally, we propose to define and measure chord *texture* to determine if the suggested chord share common timbral characteristics. While texture is usually defined from audio (Lu et al. 2004), it can also be defined from symbolic data (Couturier 2024; Giraud et al. 2014). Guitar chords have different texture depending on whether they are played as open chords or barre chords higher on the fretboard, because strings are going to resonate differently. Likewise, chord voicings, depending on which note they repeat and where there are played on the fretboard, can sound different. We therefore want to evaluate if the *texture* change observed when using a suggested diagram is similar to the one from the reference. To measure this texture, we implement some of the sound quality measures of Wortman et al. (2021) and variations of them. More

precisely, we extract for each diagram: the ratio of open strings; the ratio of muted strings; the string centroid; and the ratio of unique notes *i.e.* counting only once notes that are repeated on several octaves. We also compute the difference of these metrics from one chord to the next to assess how consistent they are through a transition. The results are reported in Table 7.3, all values go from 0 to 1. The first observation we can make is that the full model is closer to the texture of the reference data than the baseline on all metrics except the amount of open strings compared to the MSB dataset. We also notice that the performances of the full models trained and tested on DadaGP and MSB are similar. However, this experiment does show slight differences in the datasets, on the amount of muted notes and the string centroids in particular. Those differences might be due to discrepancies in the musical style distribution of each dataset. The biggest difference in the results is the improvement of the baseline's performance on individual chord metrics for MSB, which could be due to the fact that the dataset is smaller and less varied. From the δ values for both datasets, it appears that the metrics are rather consistent from one chord to the next. We can also observe that the baseline trained on DadaGP tends to play too many notes on each chord (lower ratio of muted notes) probably by repeating them on different octaves (lower ratio of unique notes). Finally, an encouraging result is that the model using context suggests diagrams with a texture similar to the ground-truth (lower δ values). It should however be noted that it also tends to repeat more notes than necessary, while still missing some.

7.5 Discussion

In this chapter, we confirmed that using information from the past chord allows to suggest diagrams that are closer to the expected reference ones, both in terms of texture and location on the fretboard. However, a few limitations still exist. First of all, the MLP we use for diagram suggestion does not ensure that the output diagram is playable. In fact, we observed that the model would sometimes merge two different diagrams of a same chord, resulting in an unplayable diagram that could span over a large part of the fretboard and unrealistic fingerspans for most players. Future work should consider potential solutions to address this limitation, for instance by training the model with a loss that explicitly penalises unplayable diagrams. In a more practical sense, one could also simply imagine conducting a post-processing step, and forcing the model to suggest a new diagram if the first one is unplayable. Finally, it might also be worth considering a multi-criteria optimisation approach rather than a neural network.[3] It could indeed be more efficient to use a diagram finder system like Wortman et al. (2021) that can only suggest known chord positions, and design an optimisation model that can choose between possible diagrams to guarantee texture continuity, playability, stylistic consistency,

[3] I am indebted to Sean Luke I met at SMC 2024 for the idea.

etc. Such an approach would also have the advantage to be applicable to full chord progressions instead of chord pairs. Indeed, while the model we presented in this chapter can technically be used repeatedly to suggest diagrams for as many chords as necessary, only one previous chord is taken into account when making suggestions. Because of this limited past context, there is a risk of the model "drifting" and that suggestions after 4 or 5 iterations lack consistency with the first chord. This can be an issue because of the repetitive nature of WPM, where chord progressions are usually evolving in cycles and not linearly in time.

Finally, we observed when analysing the datasets that chords are not always labelled correctly (Figure 7.10), or that diagrams can sometimes be missing from the tablatures. As we discussed, developing methods to automatically annotate tablatures with diagram information, or correct chord labelling, could be beneficial to a diagram suggestion system. However, simply correcting chord labels without further analysis could be a mistake. Indeed, we mentioned that metal songs could sometimes have power chords labelled as major chords, or that inverted chords were not always labelled consistently in the datasets (and DadaGP in particular). If those "wrong" labels are what guitar players use because it makes sense in context, maybe a diagram suggestion model should occasionally suggest inverted chords or slightly different chord natures than the one originally asked for. In other words, while correcting chord labels as music theory dictates might be tempting, it might not be necessary and maybe even detrimental for a system that aims at suggesting diagrams to WPM guitarists, who develop their own idioms through their own practice-based culture that may differ from modern Western music theory (L. Green 2002).

7.6 Conclusion

In this chapter, we have shown that chords used in the guitar WPM repertoire are varied but also highly unbalanced, with some common chords and diagrams being used much more frequently than others. From this observation, we proposed a new approach to suggest guitar chord diagrams in the context of WPM. We showed with several newly introduced metrics that adding context information through the previous diagram improves the quality of the suggestions while also maintaining a better consistency of texture between chords. All experiments were conducted on two datasets, one proprietary and one public, to further guarantee the validity of the conclusions. The Python implementation is openly available under a GPL-3.0 license, along with a demonstration website, at algomus.fr/code. All diagrams observed for each chord label are also released openly under an ODbL license, along with the extracted chord pairs. The diagram suggestion model we introduced could be used as a first step towards rhythm guitar tablature continuation (D'Hooge et al. 2023b). In the next chapter, we present an approach to generate picking patterns that, combined with diagram information, can be used to conduct complete rhythm guitar tablature continuation.

8

Picking Pattern Generation for Rhythm Guitar Tablature Continuation

"Just pick up a guitar that sounds so good and feels so good and then play chords that sound perfect, I could do that all day, it just makes me happy."

Ian (2018)

Contents

Suggesting chord diagrams like presented in the previous chapter is a way to assist guitarists who play accompaniment parts. However, while this might suggest chord positions for comping (A. Green 2017), it is not sufficient for guitarists who practise or compose with tablatures. In this chapter, we study the task of *picking pattern* generation that, combined with chord diagrams information, can be used to propose possible continuations to an existing rhythm guitar prompt measure. To do so, we present a rule-based and a GPT-based model, and compare their performance through quantitative metrics and a subjective user study.

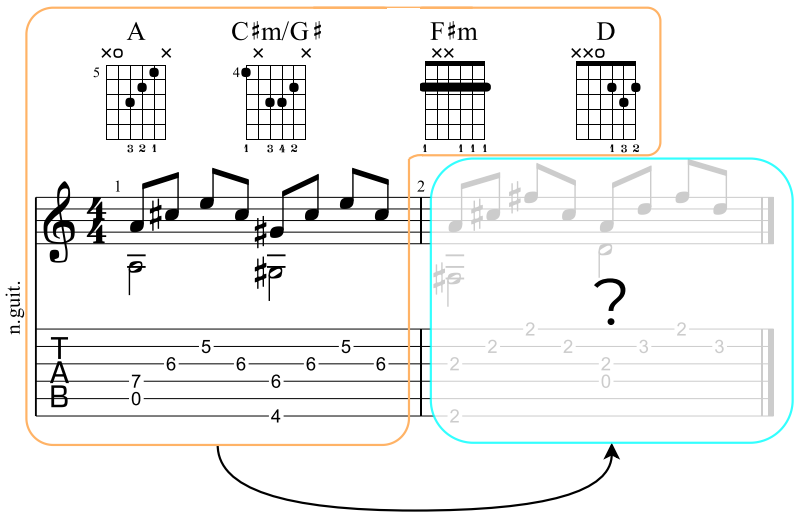


Figure 8.1: Excerpt of a rhythm guitar tablature and summary of the task studied. Chord positions are specified with chord diagrams above the score.

8.1 Picking Pattern Generation

In this section, we introduce the task of picking pattern generation by first presenting context and motivation for this task as well as defining picking patterns. The objective of this picking pattern generation is to conduct rhythm guitar tablature continuation given a prompt measure and a set of conditioning controls. This section also introduces the conditioning controls and texture metrics used as input. We conclude by presenting the two architectures used in this chapter: a transformer-based model as well as a rule-based model.

8.1.1 Definitions

Task Presentation This chapter proposes a new approach to generate rhythm guitar tablature excerpts from a chord progression (Figure 8.1). We suggest splitting this generation into two steps: i) predicting *chord positions* for choosing an exact way to play the chord progression, ii) generating a *picking pattern* that contains information on what strings should be played at each time. In this chapter, we focus on picking pattern generation and assume that chord diagrams are available, either provided by the composer or generated with the approach presented in chapter 7. Our approach is based on the observation that the accompaniment tracks in guitar popular music, are often repetition-based (Adkins et al. 2023; Margulis 2014). In particular, when using a tablature notation software to write accompaniment parts, guitar players typically use copy-pasting tools and then apply minor edits to the pasted sections to add variations (Bacot 2023). The approach presented in this chapter aims at suggesting alternatives to this copy-pasting practice that might lead composers to introduce more variation into their work.

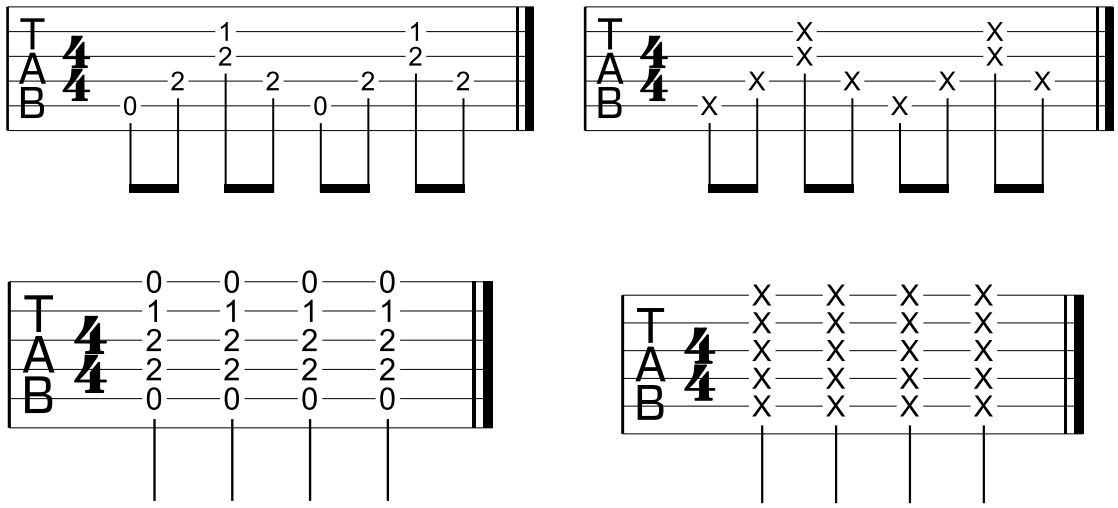


Figure 8.2: Tablature excerpts (left) and their corresponding *picking patterns* (right). The `x` symbol signifies that no specific fret (and thus no pitch) is defined for a note in a picking pattern. Pitched notes are not distinguished from *dead notes* in our approach.

Picking Patterns As chord diagrams provide information regarding the *fretting hand*, picking patterns inform on the role of the *picking hand*. This hand is not completely independent from the first one because guitar players will usually only play strings that are pressed down by the fretting hand (unless they need to play *dead notes* or open strings). Nonetheless, there are still several degrees of freedom to choose which strings to strum and with which rhythm. For instance, the same A minor chord (`x.0.2.2.1.0`) could be strummed fully on quarter notes (Figure 8.2, bottom) or arpeggiated on 8th notes (Figure 8.2, top), depending on context.

8.1.2 Texture and Conditioning Controls

We aim at generating a possible continuation of a single measure provided as a prompt, using picking patterns. A summary of the full pipeline is represented Figure 8.3, and the next paragraphs present the different elements of this figure.

The prompting measure is already a way in itself for guitar players to have some control over the generated strumming pattern. Indeed, as with any architecture that relies on *priming*, the generated content should be consistent with the priming sequence, keeping some of its textural characteristics intact. This is however limited in its control possibilities, which is why we present below our approach to guide the generation through texture control values. The main difference with simple priming is that the provided texture can drive the model towards generating measures that are unexpected or different from the prompt by driving the model to change rhythmic or note densities for instance (S.-L. Wu et al. 2023). These texture variations can generally be observed in WPM (Bimbot et al. 2016) and might be more desirable than being over-consistent with the priming measure.

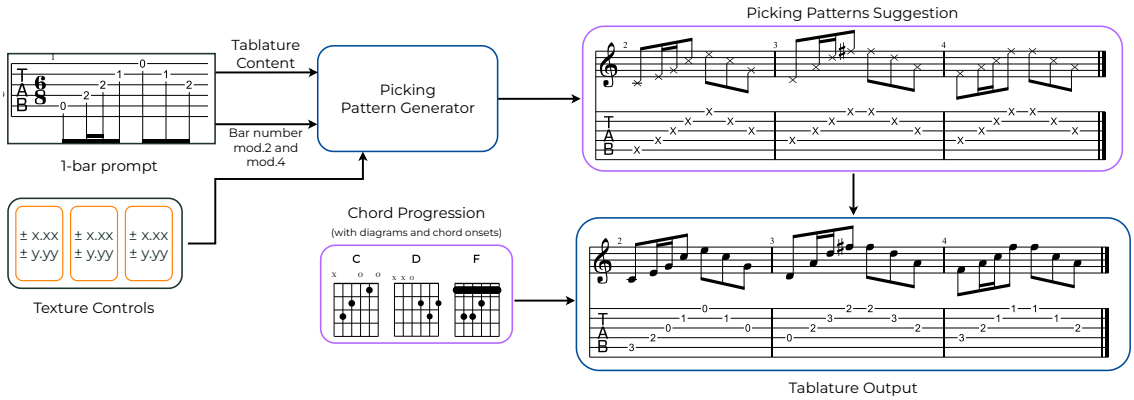


Figure 8.3: Overview of the picking pattern generation pipeline. The final tablature is obtained by combining the chord diagrams with the generated picking patterns. The texture controls are optional because the model can generate patterns even if all controls are equal to zero, it simply means that the texture should be constant across the generation.

Tablature Texture metrics and distances WPM is heavily based on repetition and this can be observed within tablatures (Adkins et al. 2023), but there can be variations of musical characteristics within looping excerpts nonetheless. Building upon the definitions of texture for guitar chord diagrams presented in section 7.4.2, we extend the definition of texture to tablatures. We also adapt metrics from Couturier et al. (2023), where the authors introduce *vertical* and *horizontal* texture measures for piano music. Those metrics capture information about the pitches used (vertical axis in standard notation) or the rhythmic density (horizontal axis in standard notation), respectively. We use the horizontal texture metrics without any change, computing the *average number of onsets per beat* and the *standard deviation of the duration between consecutive onsets*. Regarding vertical texture, three metrics are derived from tablature data:

1. *average thickness*: the number of notes averaged by onsets, weighting the notes by duration where a quarter note has a weight of 1;

2. *number of strings standard deviation*, from all played strings at any non-rest onset;

3. *average string centroid*, obtained from the string centroids (mean of the played strings numbers) of each non-rest onset.

For instance, given the top-left bar in Figure 8.2, each eighth note has a weight of 0.5 and we obtain: a thickness of $(10 * 0.5)/8 = 0.625$; a number of strings standard deviation of ~0.433; and an average string centroid of 3.875 (we then divide it by the number of strings for normalisation). These first two vertical metrics are taken from Couturier et al. (2023), after reformulating them to acknowledge the fact that the vertical axis in tablature notation represent strings. The last metric is adapted from section 7.4.2, to also include a metric related to the hand position on the fretboard.

Finally, rather than using those values directly, we condition the generation on texture variations to convey relative differences rather than trying to reach exact values. To do so, we simply sum differences of texture values to obtain a *horizontal texture variation* and a *vertical texture variation*, from the horizontal and vertical metrics respectively. While we would expect the prompt to provide the model with texture information, that sole information cannot help the model decide when and how to diverge from the original texture. We compute these texture variation values to allow better control on the generated output by letting the user choose how the texture should evolve. Because the texture controls are ultimately designed to be used as manual sliders in notation software, we also round the values to 2 decimal places, as a higher precision would render control of the model needlessly complex. Besides, if the model is trained with high-precision inputs, users entering numbers with lower precision might decrease performance.

Additional Conditioning and Pipeline Summary In addition to the texture variation information, we condition the picking pattern generation with two additional musical signals: chords diagrams for the sequence, and an indication of the structural position of the priming measure. The chord diagrams are supposed to be available, either directly provided by the user or suggested from chord labels as presented in chapter 7. Those diagrams inform the model on what strings are played for each chord, which should guide the proposed continuations to not use unwanted strings. For example, with a C major chord, if it is played as a barre chord on the 8th fret, it uses all 6 strings (`8.6.6.7.8.8`) while if it is played as an open chord, the 6th string should not be played (`x.3.2.0.1.0`).

The second conditioning signal, related to the position of the prompting measure in the structure of the song, aims at helping the model appropriately suggest variations and differences from the prompt measure. Indeed, even though WPM is repetitive, 4-bar cycles will often end up with a contrasting measure, as studied in Bimbot et al. (2016). For this reason, we take the number of the measure that is used as a prompt and provide the model with this value *modulo-2* and *modulo-4*. By informing the model with the relative position of the first bar in such cycles, we try to enforce that the generated measures better fit with the overall song. All conditioning signals are shown Figure 8.3.

8.1.3 Deep Learning Model

Inspired by Dalmazzo et al. (2024b) that uses a decoder-only implementation of a transformer network to generate chord progressions from small prompts, we base our code on *minGPT*,[1] an open-source implementation of the GPT-2 model (Radford et al. 2019). We supplement the existing architecture with fully-connected linear networks to provide conditioning information using the *pre-attention* mechanism proposed in (S.-L. Wu et al. 2023), visible in Figure 8.5. It consists in concatenating the conditioning vectors to the input tokenised data. Finally, because preliminary experiments with larger networks showed that

[1] https://github.com/karpathy/minGPT, accessed in July 2025.

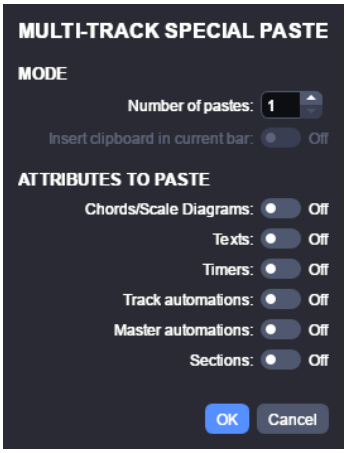


Figure 8.4: Screenshot of the "special paste" window from Guitar Pro.

the task at hand can be tackled by a network of limited size, we base our architecture on the `gpt-nano` template of minGPT, using 3 layers of attention blocks, all featuring 3 attention heads, while setting positional and vocabulary embeddings to a size of 48. A graphical representation of the full architecture is provided in Figure 8.5.

Loss Functions For training, the model uses a standard *Cross Entropy* loss. Nevertheless, to better enforce the chord diagrams constraints and limit the amount of notes on OoD strings, we devise a guitar chord diagram loss and also use it for training. We combine the losses by simply summing them, after scaling the diagram loss by 0.01 to make its values closer to those of the cross entropy loss. To obtain the diagram loss value in a differentiable fashion, we use a *softargmax* function to determine which notes the model predicted at each time, and apply a mask of the authorised strings according to the chord diagrams. The resulting non-zero values are simply summed to obtain the total loss of processing the studied sample (or batch).

8.1.4 Rule-Based Model

As an alternative to the deep learning approach presented previously, we devise a deterministic rule-based approach which aims at reproducing the copy-pasting process used by guitarist composers (Bacot et al. 2024). In particular, after copying exactly the rhythm used in a bar to another, the composer will typically need to correct the tablature for the frets and strings to match the new chord, like in Figure 8.6. We propose a set of rules that apply minimal transformations to a tablature region to adapt it to a new chord diagram. This is, to the best of our knowledge, the first time that such a "copy-pasting" approach is proposed for tablatures or even symbolic music, since copying is usually something that is avoided rather than sought (Batlle-Roca et al. 2024; Yin et al. 2022). It is also worth mentioning that the *special paste* (*Guitar Pro8 User Guide* 2025, p. 89) feature of the Guitar Pro software does not permit to automatically modify the pasted content but rather allows to copy parts of the musical notation independently (Figure 8.4).

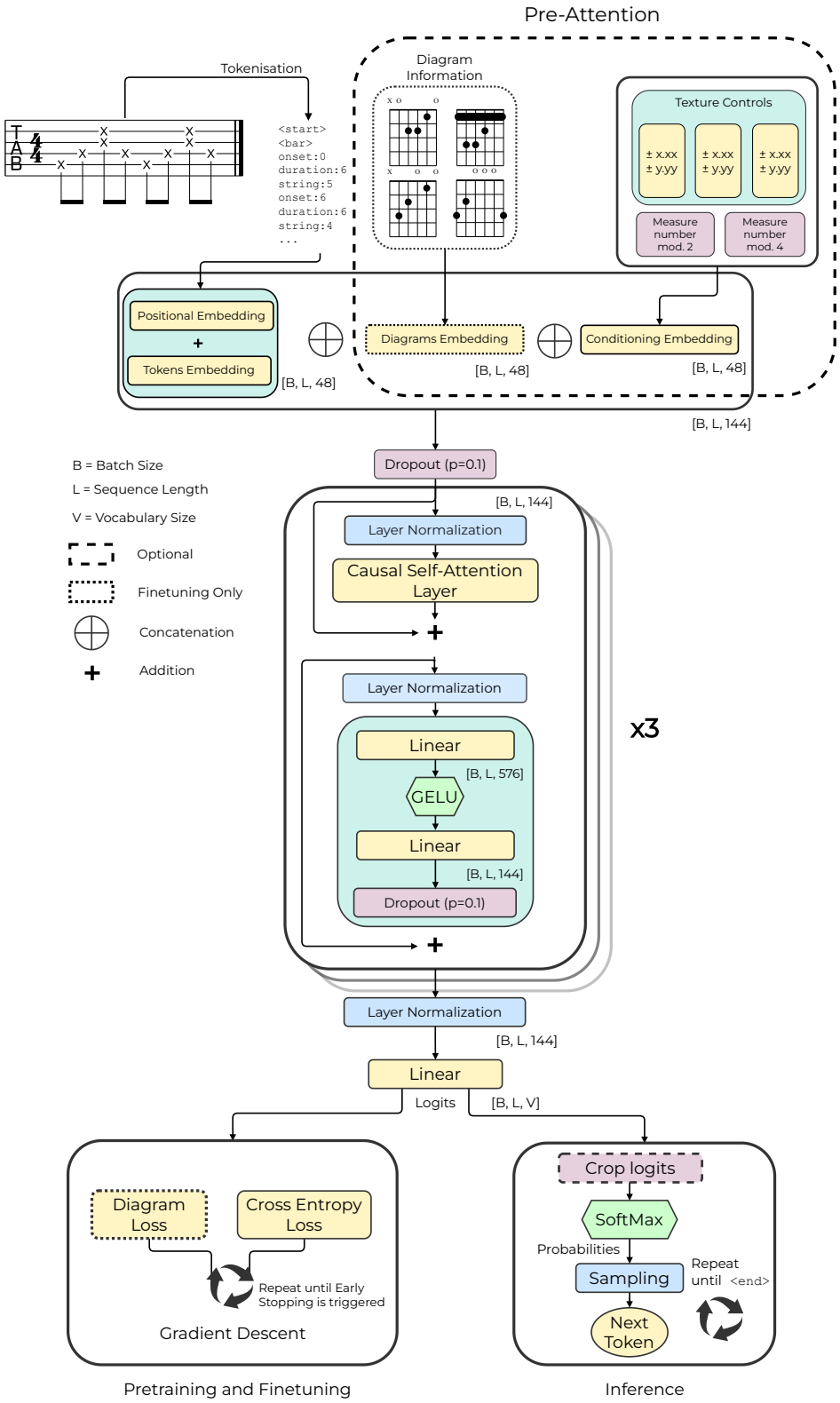


Figure 8.5: Summary diagram of the implemented model. The main architecture is A. Karpathy's minGPT model, with slight modifications to allow for additionnal conditioning. Numbers in brackets denote the dimension of the matrices at each step. The main block is repeated three times, by connecting the output of one block to the input of the next.

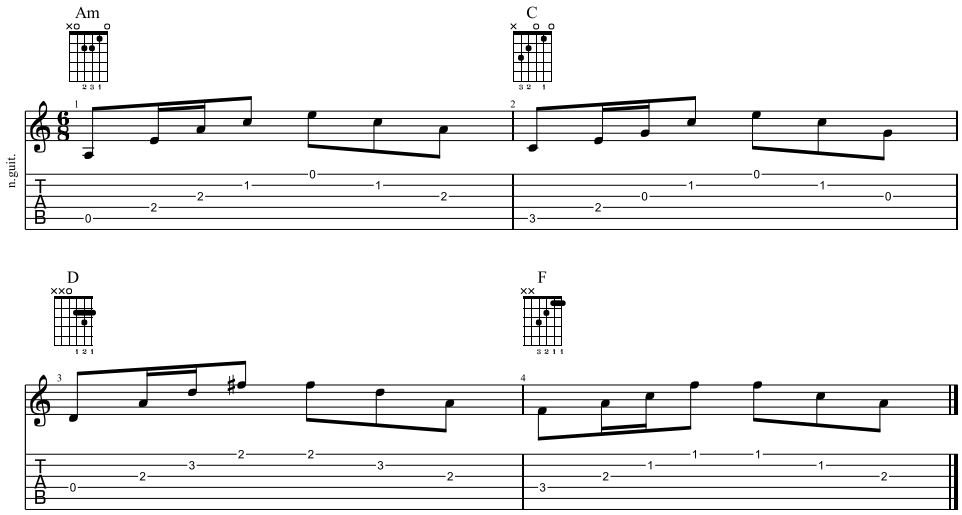


Figure 8.6: Example of a rhythm guitar part tablature including four chord-regions with similar/uniform texture, excerpt from *The House of the Rising Sun* by The Animals. Improved copy-pasting could be used to generate the three following bars given the first bar and the chord diagrams.

The pasting tool we propose is designed to, given a certain musical excerpt as input, return a musical excerpt of the same duration, with the same rhythm, but where the strings values (1 to 6) are updated to match a new chord diagram.

Let $\mathcal{S}_P$ and $\mathcal{S}_D$ be the lists of the strings played in the prompt tablature excerpt and in the new chord diagram respectively. $|\mathcal{S}|$ denotes the length of the list $\mathcal{S}$ and s_D^i denotes the number of the i-th string in $\mathcal{S}_D$, as it would be numbered by guitar players, i.e. 1 to 6 from thinnest (highest in pitch) to thickest (lowest). For example, if we consider a chord that uses the 3 lower strings like a G power chord, or G5 (`3.5.5.x.x.x`):

$$\begin{aligned}\mathcal{S}_{D(\text{G5})} &= [6, 5, 4],\\ s^0_{D(\text{G5})} &= 6,\\ s^1_{D(\text{G5})} &= 5,\\ s^2_{D(\text{G5})} &= 4.\end{aligned}$$

The first rule of our model is fairly simple: no matter how many strings differ between $\mathcal{S}_P$ and $\mathcal{S}_D$, whenever all strings are strummed in the original bar, all strings are strummed in the copy. However, in many cases, strings of a chord may be strummed independently (arpeggios, bass played separately...). In those situations, we map the strings from $\mathcal{S}_P$ to $\mathcal{S}_D$ according to the following rules:

- If $|\mathcal{S}_P| = |\mathcal{S}_D|$, for instance when changing from C Major (`x.3.2.0.1.x`) to D Major (`x.x.0.2.3.2`) or G7 (`3.x.3.4.3.x`). In that case, we simply proceed with the mapping $s_P^i \rightarrow s_D^i$, $i \in [\![0, |\mathcal{S}_P|]\!]$. If $\mathcal{S}_P = \mathcal{S}_D$, the mapping is the identity function;

- if $|\mathcal{S}_P| < |\mathcal{S}_D|$, the lowest strings are mapped to one another $s_P^0 \rightarrow s_D^0$ and all other strings are chosen based on their distance to the highest string: $s_P^i \rightarrow s_D^{\text{high}_D - (\text{high}_P - i)}$.
- if $|\mathcal{S}_1| > |\mathcal{S}_2|$, we map the lowest strings together: $s_1^0 \rightarrow s_2^0$, and the other strings are mapped according to the following equation:

$$s_1^i \longrightarrow \begin{cases} s_2^{\text{high}_2 - (\text{high}_1 - i)} & , \forall i \text{ where } s_1^i < \frac{s_1^{\text{high}} + s_1^0}{2} \\ s_2^{\text{i}} & , \text{otherwise.} \end{cases}$$

  If any duplicate string appears from that mapping on a given onset, the note is simply discarded.

Once the picking pattern has been copied, frets are added on all notes based on the chord diagrams used at the corresponding time.

8.2 Data Preparation

This study is based on the DadaGP dataset, presented in section 3.1.3. In this section, we detail the pre-processing steps required to use the dataset, which consist in identifying rhythm guitar tracks, and extract 4-bar patterns for training and evaluating our proposed models.

Rhythm Guitar Identification While we aim at generating rhythm guitar tablature only, the DadaGP dataset contains all sorts of guitar tracks, some being melodic or solo sections that are out of scope for this project. To detect what tracks are rhythm guitar, we implement the approach from Régnier et al. (2021) like in previous chapters. On the whole dataset that contains over 2.9M bars of guitar, we detect that 2.3M bars are labelled as rhythm guitar with at least a 50% chance. This means that almost 80% of the data is rhythm guitar, which is quite close to the ratio of rhythm guitar bars manually annotated in Régnier et al. (2021) (82%) on a subset of the MSB dataset.

4-bar Sequences extraction In Bimbot et al. (2016), the authors note that most *contrastive* measures, i.e. measures that breaks the continuity of a sequence's texture, will happen within a 4-bar cycle. Preliminary analyses through self-similarity matrices showed that this also applied to the DadaGP dataset, which is why we settled on 4-bar sequences for our models, leaving the extension to any duration as future work.

Some additional constraints are applied to ensure that the patterns are varied and reflect different musical contexts. Firstly, the four-bar patterns are obtained in each file with a sliding window using a 1-bar step. While this means that most bars are added 4 times in the dataset, it ensures that the model can generalise to any part of the 4-bar cycles we observed. More precisely, this ensures that the contrasting measure is not always the last

Table 8.1: Tokens in the vocabulary divided in their respective categories. `token:n-m` means that the value for `token` can be any integer between `n` and `m` inclusive.

Type	Tokens	Voc. Size
Metadata	`<start>`, `<end>`, `<bar>`, `<PAD>`	4
Note	`string:1-6`, `rest`	7
Onset	`onset:0-47`	48
Duration	`duration:1-96`	96
Total		155

in a sequence (Bimbot et al. 2016) so that the DL model can be used at any point in a tablature. Secondly, we drop any pattern that contains over 75% of rest time, or whenever the prompt bar is entirely silent. While this threshold might seem high, it is required to include patterns that could contain *staccato* notes that occupy a small part of each bar. Finally, to simplify the implementation of the diagram loss and because it still encompasses almost 90% of the data, we only consider bars in $\frac{4}{4}$ time signature. With those constraints, we obtain 2 679 628 4-bar sequences, out of which 1 018 745 are unique.

Lastly, we isolate patterns that contain 2 to 8 chord diagrams. We separate the data in such a fashion to allow for an approach akin to pretraining and finetuning. In doing so, we can pretrain the model on as much data as possible, without any chord information, to generate consistent continuations of a first measure in tablature format. Once the model is pretrained, we conduct finetuning on 4-bar sequences with chord diagram information, to enforce the generated tablatures to match the strings played in each chord (see Figure 8.5). 31 596 patterns match our constraints and will be used to finetune the final model, which leaves 987 149 patterns (~97%) for pretraining. Most patterns of the latter category do not explicitly mention chord diagrams, even though they usually have chord labels defined. Those observations suggest that future work could strongly benefit from some kind of model or heuristic that reliably recognise chord diagrams from tablature excerpts to increase the size of the finetuning dataset.

Tokenisation Like standard sheet music, tokenising a tablature requires encoding the duration of each note along with its pitch, except that the pitch can be described indirectly with a *string-fret* pair. However, in our work, fret information is held within the chord diagrams and picking patterns only gather string information with their musical timing. For this reason, we use a tokenisation scheme inspired by Y.-H. Chen et al. (2020): each new note is described by three tokens, its `onset`, its `duration`, and the `string` it is played on. However, the original tokenisation scheme was restricted to a sixteenth note resolution while we wanted to be able to also account for triplets that can be encountered in the dataset. We settle for a resolution of 12 ticks per quarter note i.e. 48 ticks per bar in

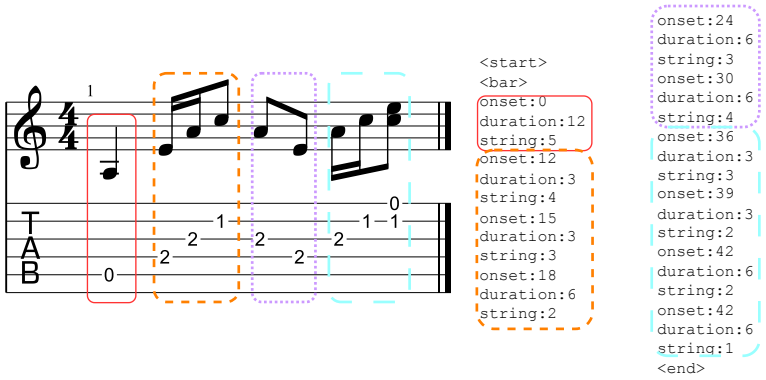


Figure 8.7: Example of the tokens obtained from a single bar tablature excerpt.

$\frac{4}{4}$. This resolution supports 16th note, as well as triplets and sextuplets. A summary of the vocabulary is provided Table 8.1, along with an example of a tokenised excerpt Figure 8.7.

8.3 Transformer Model Training and Inference Details

This section presents the training details for the DL model introduced previously. The rule-based model does not require any parameter optimisation and can be used directly for inference.

Pretraining and Finetuning As mentioned in section 8.2, the DL model is trained in two phases. During *pretraining*, it has to generate a 3-bars continuation of the first bar used as input, guided by texture controls and the structural position of the first bar, but without any information about the chord progression. Afterwards, it is *finetuned* on the subset of data with chord diagrams information, so that it also exploits the chord conditioning in its generation.

Both pretraining and finetuning steps use an *AdamW* ($\beta_0 = 0.9$, $\beta_1 = 0.95$) optimiser with weight decay ($\delta = 0.1$) on linear layers.

Pretraining uses a batch size of 256 samples, a learning rate $\lambda = 5 \times 10^{-4}$ and an early-stopping strategy with a patience of 3 epochs, any improvement over the validation loss resetting the patience count. The pretrain dataset is split into a train and a validation set with a 90/10 ratio. Training stopped after 27 epochs, which took approx 20 hours on a Nvidia L40S Graphical Processing Unit (GPU). A discussion on the energy and computational cost of the DL model, compared to the rule-based one, is provided appendix B.

Finetuning used a batch size of 64, a learning rate of $\lambda = 5 \times 10^{-5}$ and the similar early-stopping strategy but with a patience of 20 epochs. The finetuning dataset is split in train, validation and test sets with 70/15/15 ratio. Note that no weights were frozen in this step, only the learning rate was decreased. Training stopped after 37 epochs which took 4 hours on a RTX2080 Ti GPU.

Inference At inference, tablatures are generated by the DL model using *top-k* sampling with $k = 5$. The generation is automatically stopped as soon as an `<end>` token is produced. Although it depends on the amount of notes in each of the 3 bars, a full sample is obtained in less than 5 seconds on a standard laptop CPU (Intel Core i5). It might be worth looking into ways of optimising the model to reduce this computational time and ensure it would easily fit in a composer's pipeline. However, we want to highlight that the fact that the model does not need a GPU to generate content in a relatively small amount of time is an important quality for enabling its use on standard computers for all users.

8.4 Quantitative Results

After training the transformer model, we can test it and compare it with the rule-based model. This section introduces the quantitative evaluation metrics used, and discuss the comparative performance of both models, using the original 4-bar sequences as a reference.

8.4.1 Evaluation Metrics

To conduct a first automatic evaluation of our models, we implemented several metrics to assess how the strumming pattern we generate compare to the reference tablatures. The use of these metrics assumes that the reference is the best possible answer, which is debatable from the perspective of an open composition situation, but a convenient assumption for automated quantitative evaluation. More specifically, these metrics do not aim at quantifying the quality of the generated continuation, but rather at measuring its proximity to a continuation choosen by a human composer, which can therefore be considered as one reasonable solution, possibly among many.

We first use a *Levenshtein edit distance* to assess how close predictions are to the reference. The edit distance being dependent on the size of the sequences, we compute a normalised version, using the reference as the target length. Note that this normalised version is not necessarily smaller than 1 as, for instance, an entirely wrong prediction twice the size of the reference would yield a value of 2.

Second, to evaluate the impact of the conditioning on the predictions, we compute the ratio of *Out-of-Diagram (OoD) notes*, i.e. notes that are played on strings that are excluded in the current chord diagram, for example the lowest string in an open A minor: `x.0.2.2.1.0`. An example of an OoD note in a tablature is the circled note in Figure 8.8. We also compare the *symbolic texture* of each generated bar compared to the expected texture that was provided as conditioning, based on the texture definition of section 8.1.2. More precisely, since the conditioning actually consists in textural distances of each bar to the prompt measure, we compute the Manhattan distance between the expected and observed texture distances. Through this measure, we aim at determining if the model takes the texture conditioning into account during generation and, if so, how close to the

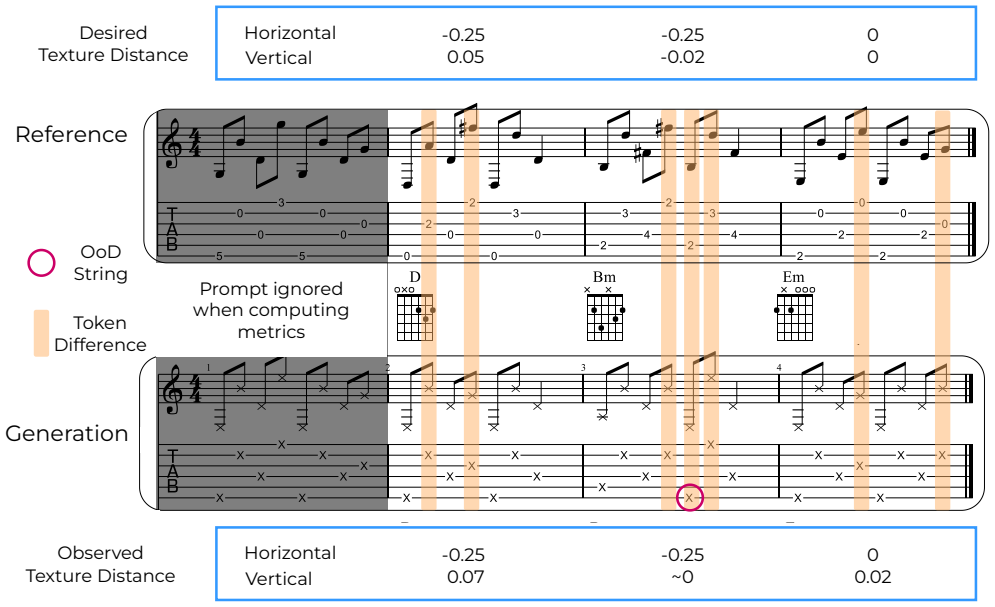


Figure 8.8: Generated picking pattern (bottom) and the expected reference tablature (top). There are 7 notes that are placed on different strings, so the edit distance is 7, which is 0.1 after normalisation by the number of tokens. Since there is a single note Out-of-diagram over the 22 notes in the last 3 bars, the ratio of OoD notes is ~0.05. Finally, the texture distance is obtained by summing the absolute difference between texture values (Manhattan distance), and yields 0.06 for that example.

expected texture the generation can be. An example of a reference tablature, along with a generated one and details on the metrics values for those is shown Figure 8.8. In this example, because the generation has the same rhythm as the reference, no difference in horizontal texture is measured. However, because the strings are played slightly differently, the string centroids are not equal and a 0.02 vertical texture difference is observed on each bar.

During evaluation, we compute those metrics by generating predictions on the test set with the rule-based model, and variants of the DL model. Those variants are a model pretrained but not finetuned, and two finetuned models, with or without diagram loss (see Table 8.2). We compare deep learning-based models to the rule-based one to assess the validity of the rules and determine whether training on large amounts of data significantly affects the performance. Besides, we compare pretrained and finetuned models to analyse the importance of providing diagram information for limiting OoD notes. We compare models with or without diagram loss for the same reason.

Final results for all those metrics are presented Table 8.2.

8.4.2 Discussion on Performance

Out-of-Diagram Notes. The first observation we want to make on the results presented in Table 8.2 is that 4.1 % of the notes in the dataset are OoD, meaning that the tablatures include strings that should not be played according to the provided chord diagrams. The

Table 8.2: Results obtained on our evaluation metrics. Most values are N/A (Not Applicable) for the test set because there is not point in rating the fidelity of the reference to itself. For all metrics, a value closer to 0 is better. All values' differences are statistically significant (p-value < 0.001 in paired t-tests), except the ones that share a ⁕ symbol. Values with a ★ denote data series that could not be used in statistical tests because their value is constant.

	OoD Notes Ratio	Edit Distance	Texture Distance
Test Set	0.041	N/A	N/A
Rule-Based Copy	$0^{\star}$	0.605	2.063
Pretrained	0.109	0.456	0.362
Finetuned	0.091	0.429^{*}	0.268^{*}
Finetuned w/ Diag. Loss	0.077	0.424^{*}	0.273^{*}

main possible reasons for such OoD notes are: i) imprecise notation: some transcribers will not always ensure that the diagram exactly matches the excerpt its referring to, it can also happen that a chord diagram is not written at all, the previous diagram then holds for a longer duration than it should; ii) use of muted strings to play *dead notes*, either voluntarily to add unpitched notes to the part, or out of convenience by simply muting unwanted strings and then strumming all strings regardless; iii) to play melodic ornaments or additional notes that technically do not belong to the current chord annotation but make sense in context. Those observations remind us that diagrams often have the practical aim to provide the information of a known *hand position* to the performer, rather than indicating the precise set of played strings. Even though perfectly reproducing the reference is not necessarily desirable, we can use the value observed in the dataset as a higher-bound for the ratio of notes that are played on diagrams' strings, and use it to assess the performance of the other models.

The *Rule-Based model* has a perfect score on OoD notes because it guarantees that the diagrams are respected by construction. In contrast, the *pretrained model*, which lacks diagram conditioning, achieves a reasonably high 89.1 % in-diagram note rate. This value is actually expected not to be low since the average number of strings played in each diagram in the test set is 4.7 which means that, if strings were played randomly, ~ 78 % of notes would be in-diagrams for a 6-strings guitar. Nonetheless, finetuning with diagram conditioning and an additional loss reduces the amount of OoD notes further, although it still does not fully reach the dataset's 4.1 % out-of-diagram note rate. Note that we also experimented with zeroing-out probabilities of unwanted tokens based on the current chord diagram to eliminate all out-of-diagrams notes and match the performance of the baseline on that aspect. However, these constraints came at the expense of other performance and worsened all metrics, in particular texture distance, while the sequences also tended to be ill-formed, `onset`, `duration` and `string` no longer coming in triplets. We thus deemed this approach non-viable and do not report its results in this chapter.

Edit Distance. The edit distance metric reflects how close generated sequences are

to the reference. The *Rule-Based Copy* model achieves reasonable edit distances, likely because it faithfully replicates the prompt structure. This supports the hypothesis that copying one bar to the next can actually be a valid approach for generating rhythm guitar tablatures. The *pretrained model* improves on this metric which suggests that the training procedure was fruitful, and the results are further improved in the *finetuned model*, even though the addition of the diagram loss has no significant impact on the edit distance. This last observation suggests that the "errors" that increase the edit distance are not necessarily OoD notes that are effectively reduced by the diagram loss.

Texture Distance. The texture distance of the DL model is much lower than the rule-based one, showing an effective use of texture conditioning. Finetuning maintains low texture distance despite added constraints, indicating robustness. The worse performance of the rule-based copy on these metrics is actually expected, since the rules implemented do not impose any texture constraints. This however shows to what extent texture variation in WPM still limits the adequacy of mere copying in notation and composition.

8.5 Subjective Evaluation

While the metrics we introduced allow to assess the overall performance of the proposed models, their link to the actual musicality of the results is debatable and hard to establish. For this reason, we devise an online survey to gather subjective feedback on the proposed approach. This study got approved by the ethics committee of *Université de Lille* (project reference: 2024-823-S132).

8.5.1 User Study Details

The main goal of the survey we designed is to assess what generation model is preferred by participants on various musical aspects, and why. To keep the study to a reasonable duration, we decided to ask participants to rate 5 different samples from 3 different configurations: the ground-truth, the rule-based copier, and the finetuned model that uses the diagram loss. The samples used are shown in Appendix A. All samples are rated on a 1-7 Likert scale on 4 criteria:

1. **Playability**: *is the tablature shown playable on guitar?* This questions aims at measuring the perceived playability, likely correlated with difficulty, of the samples;
2. **Consistency**: *are the strumming and rhythm consistent throughout the excerpt?* While some variation might be desirable, some coherence is expected between bars;
3. **Interest**: *is the musical content shown interesting?* This question aims at gathering subjective feedback on the tablatures shown;
4. **Usability**: *would the proposed tablature be usable in a performance or composition context without editing it?* Since the ultimate goal of our system is to assist writing tablatures, we collect direct feedback on the possible usage of the tablatures.

Snippet 8.1 Questions related to musical practice.

- *What guitar type do you play the most?* One of [Classical Guitar, Electric Guitar, Folk Guitar];
- *Do you know how to:*
  - *Read rhythm notation?* Yes/No
  - *Read notes on a score?* Yes/No
  - *Read chord charts?* Yes/No
- *On the guitar type you play the most, how would you rate your own level?* 5 point choice, from 1 (Beginner) to 5 (Expert);
- *What musical style would you say you practise the most?* One of [Classical Music, Jazz, Folk/Country, Pop, Rock, Metal, Ska/Reggae, Blues/Soul, Funk, Other];
- *Select all guitar types you play:* Multiple choice from [Classical Guitar, Electric Guitar, Folk Guitar].

Snippet 8.2 Questions subset from the Goldsmith-MSI test.

- *I engaged in regular, daily practice of a musical instrument (including voice) for X years.* One of [0, 1, 2, 3, 4-5, 6-9, 10+];
- *At the peak of my interest, I practised my primary instrument for X hours per day.* One of [0, 0.5, 1, 1.5, 2, 3-4, 5 or more];
- *I have had formal training in music theory for X years.* One of [0, 0.5, 1, 2, 3, 4-6, 7+];
- *I have had X years of formal training on a musical instrument (including voice) during my lifetime.* One of [0, 0.5, 1, 2, 3-5, 6-9, 10+];
- *I can play X musical instruments (including voice).* One of [0, 1, 2, 3, 4, 5, 6+].

Personal and Music-related Questions. We ask several other questions to assess the representativeness of our population study and the musical expertise of the participants. Regarding personal characteristics, we ask participants in which *country* they learnt music, their *age*, and their *gender*. When it comes to their musical practice, the survey included two different sets of questions: one was related to their guitar and musical practice, the other one was a subset of the Goldsmith-MSI test to assess their "musical expertise" (Müllensiefen et al. 2014). The questions asked are provided in Snippets 8.1 and 8.2. We do not consider that the answers to those questions will reflect the actual musical level – as a practitioner – of the participants. Rather, the questions aim at identifying possible clusters of respondents that might be correlated to ratings of the samples.

The study was deployed on an institutional instance of LimeSurvey and advertised on public guitar and music forums and to acquaintances of the authors. All answers were gathered anonymously and will be shared publicly upon publication of this work.[2]

[2]Code and data will be available at https://algomus.fr/code.

Table 8.3: Number of participants per geographical regions.

Region	Number of Participants
Western Europe	33
Southern Europe	7
Northern Europe	2
Eastern Europe	3
Oceania	1
South America	2
Northern America	6

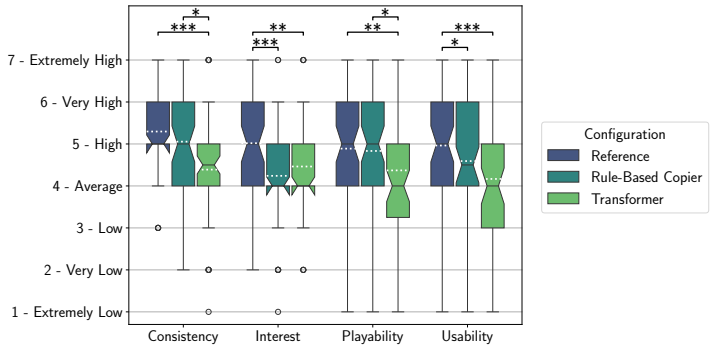
8.5.2 Questions Results

The goal of our user study is to to assess if one configuration has higher ratings than another (ground-truth vs rule-based vs attention-based). We conducted an *a priori* statistical analysis to determine the required sample size using the G*Power software (Faul et al. 2007). We based our estimation on paired t-tests (since all models are evaluated on the same test data), aiming for a significance level of $\alpha = 0.05$, a power $(1 - \beta) = 0.95$ and a default effect size $dz = 0.5$ considering a two-tails distribution. This analysis identified a minimum sample size of *54 participants*, and we closed the study when this amount was reached.

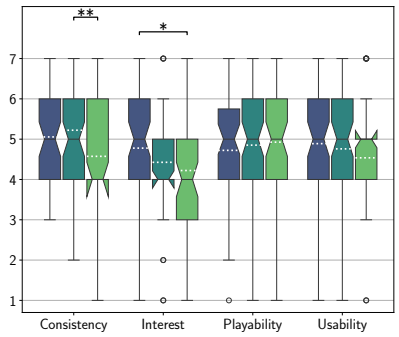
The majority of the participants were male ($n = 41$), with only 3 female participants and 10 who did not answer this question. Participants were 46 years old on average (the median is similar), with the following repartition: 18-29: 13; 30-39: 8; 40-49: 9; 50-64: 16; 65+: 8. The countries where each participant learned music are grouped according to the UN Geographic Regions taxonomy[3] and shown in Table 8.3. Unfortunately, our panel of participants is biased towards male and Western European guitar players. Conducting another study with more diverse respondents might change the conclusions of this subjective evaluation in some aspects, but it could not be studied in this work.

Boxplots comparing the statistical distribution of the answers are provided Figure 8.9. Those graphs show firstly that even though the distributions of answers vary depending on samples, they are above average in most cases. It is also worth noting that differences between the answers distribution are not always significant, depending on the sample considered. Interestingly, the ratings of the reference are not much better than the rule-based copier, which is even better rated at times, on *Playability* for the third sample for instance. The median ratings of the transformer model we proposed are never more than one point away from the other configurations, but still significantly lower than the reference on almost all questions and samples. However, the transformer model's ratings are also rarely significantly different from the rule-based one, which suggests that the participants did not deem one better than the other on the queried criteria.

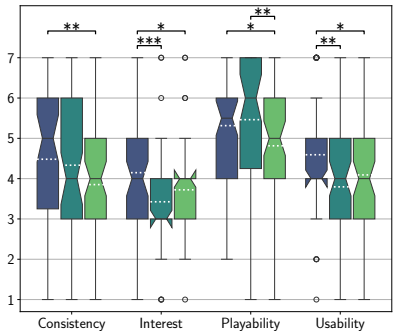
[3] https://unstats.un.org/unsd/methodology/m49/#ftn13, accessed in May 2025.



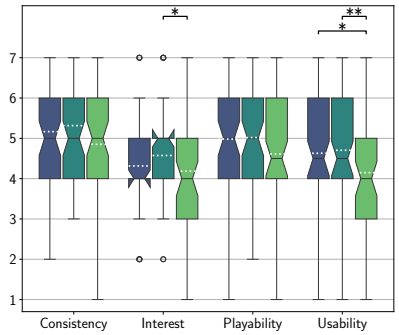
(a) First Sample.



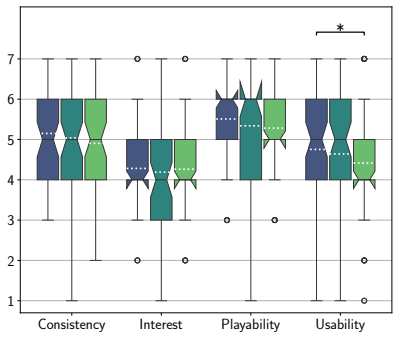
(b) Second Sample.



(c) Third Sample.



(d) Fourth Sample.



(e) Fifth Sample.

Figure 8.9: Boxplots of the participants' answers for each question and each sample. The boxes indicates the distributions' quartiles, while the whiskers describe all others samples that are not considered outliers (denoted by empty circles). Notches around the median shows the confidence interval, and the dotted lines represent the average answer values. Brackets indicate statistical significance in pairwise Wilcoxon signed-rank tests (non-normality of the data was checked beforehand) using a Bonferroni-adjusted α_B level of $\frac{.05}{3}$: $*\rightarrow \alpha_B < 0.05$, $**\rightarrow \alpha_B < 0.01$,$***\rightarrow \alpha_B < 0.001$.

Those observations are confirmed when looking at the cumulated answers on all samples for the different configurations (Figure 8.10), where the differences between the models are also more significant. More precisely, analysing the cumulated answers from Figure 8.10 shows that the rule-based model is considered significantly better than the DL model on Consistency, Playability, and Usability. The better performance on consistency

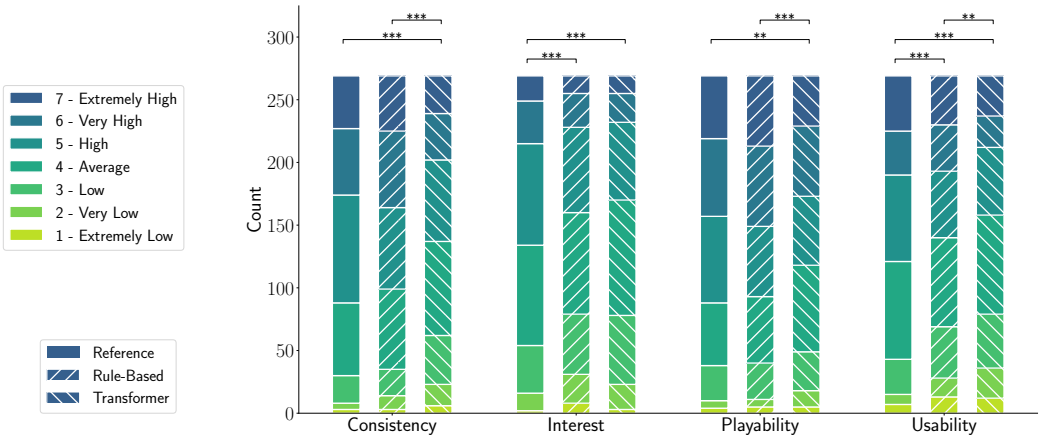


Figure 8.10: Cumulated answers on all 5 questions for each configuration. Brackets indicate statistical significance in pairwise Wilcoxon signed-rank tests (non-normality of the data was checked beforehand) using a Bonferroni-adjusted α_B level of $\frac{.05}{3}$: $\star \rightarrow \alpha_B < 0.05$, $\star\star \rightarrow \alpha_B < 0.01$,$\star\star\star \rightarrow \alpha_B < 0.001$.

was expected, because the copy-pasting approach enforces almost identical reproduction of the prompt bar's content. The higher playability rating of the rule-based model might also be due to this copy mechanism, because playing the 4 bars require less effort if the first bar's rhythm and picking pattern is repeated, while the transformer model might generate more varied content that is comparatively harder to play. Finally, analyses of text answers from the participants show that some generations of the transformer model were deemed unusable because of too many "mistakes" or "errors" in the final tablature. We discuss these mistakes in more details in the following section, but the ratings suggest that guitarists consider a robust system like the rule-based copier more usable because it is less likely to generate tablatures with errors, as long as the prompt does not have any.

To further analyse what influence the answers of participants, we implement a linear mixed effects model (Pinheiro et al. 2000). Results show that the measured personal characteristics (age, gender, country where they learned music, music proficiency level) of the participants do not significantly impact the ratings given. The results for a subset of the most significant variables are reported Table 8.4. The "scale" of the test, *i.e.* the residual variance is 1.6 (standard deviation of ~ 1.265) which means that a strong part of the variability was not captured by our study. It is unclear what might be explain those residual errors since the variance due to participants is 0.503. Likewise, the model type contribute to a change in ratings around 0.5. Besides, only sample 3 and 5 are rated significantly differently than sample 1 (taken as the reference). It is possible that our survey failed to capture some aspects of musical practice related to tablatures, and it might be worth studying this issue further in future studies.

Table 8.4: Results of the linear mixed effects model analysis. The intercept is the base answer value observed, while the models and samples are compared to the reference or the first sample, respectively.

Variable	Coeff.	Std. Err.	z	p-value
Intercept	5.692	0.382	14.917	< 0.001
Model: Rule-Based	-0.156	0.056	-2.814	0.005
Model: Transformer	-0.402	0.056	-7.225	< 0.001
Sample 2	0.032	0.072	0.448	0.654
Sample 3	-0.361	0.072	-5.035	< 0.001
Sample 4	0.026	0.072	0.358	0.720
Sample 5	0.143	0.072	1.978	0.048
Participant Variance	0.503	0.094		

8.5.3 Thematic Analysis

Finally, the survey contained free-text questions to allow for participants to explain their ratings if they wanted to. The free-text questions asked the participants to discuss what guided their ratings in the previous Likert questions. There was one question for each sample, where they could discuss any of the three configurations evaluated. 31 participants wrote at least one text answer when rating samples. We analysed the answers by conducting a *thematic analysis*, as defined in Braun et al. (2006), to gain better understanding of what guided participants' ratings. After going through the answers multiple times to define *codes*, we assembled them in *themes* represented Figure 8.11. The rest of this section provide example quotes that relate to each theme, and analyse what it implies for the models used and the survey we designed.

Ratings based on musical characteristics. As expected, the participants explained what drove their ratings. Rhythm was mentioned by 10 participants (P 6, 7, 8, 10, 11, 12, 17, 22, 23, 28), a participant for instance explained looking for *"rhythm that gets more complicated"* (P 10). Consistency with the first bar was also evaluated as expected (P 8, 13, 20, 23, 26), and was based on specific musical aspects:

> *"[The] consistency rating I based on how similar the continuation was to the first measure in rhythmic pattern or melodic motion. However more consistent examples tended to score lower on musical interest."*
>
> — Participant 23

This answer also supports the hypothesis that the rule-based model, even though it is consistent and efficient in its generation, might be less interesting than the transformer

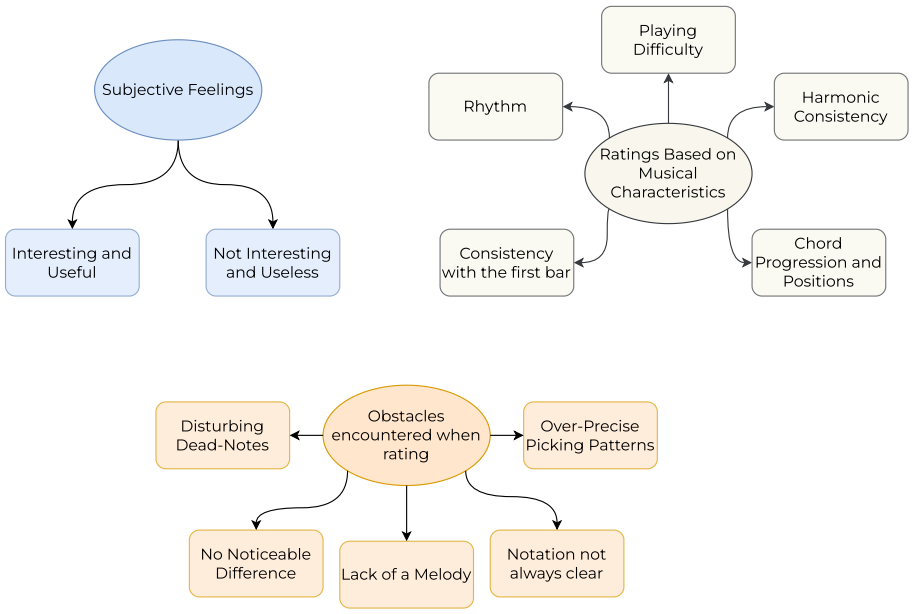


Figure 8.11: Final themes and sub-themes identified from the text answers.

model. While further human studies are required to understand which parts of the reference and transformer model generations were interesting, it suggests that being overly-consistent is not interesting from a musical point of view. The non-deterministic nature of the transformer's generation intrinsically allows for more varied samples.

13 participants explained what drove their playability ratings (P 0, 5, 8, 10, 11, 12, 15, 18, 20, 22, 24, 28, 29). Some excerpts were for instance judged too fast:

> *"[T]he harmonics at that speed are not playable. The change from G to G/F# isn't easy at that speed - it requires moving two fingers on strings 6 and 5"*
>
> — Participant 5

But other excerpts were found easy to play: *"the tablatures (identical, only the key signature changes) are super simple and very easy to play"* (P 28).

6 participants (P 0, 12, 13, 14, 17, 24) were also influenced by their perception of how the excerpts should be strummed or picked:

> *"[I]f we play with fingers and not a pick we need more fingers, and if we use a pick the dead notes are not located in the right place, sometimes some are missing, and others are useless."*
>
> — Participant 12

Such comments support a need to better indicate the way accompaniment is played on guitar, and add playability constraints in the transformer model so that the tablatures are always realistic. The MIR community could also study the prediction of strumming patterns for rhythm guitar, possibly through the adaptation of existing research on strumming pattern transcription (I. Barbancho et al. 2014; Murgul et al. 2025).

Unfortunately, feedback from 6 participants suggests that the survey explanations were misunderstood as those participants justified their rating from the chord progression, even though we explained that it was chosen beforehand.

> *"[T]he harmonic progression is weak from the start, the arrival on F#Major is weirdly placed. The progression of this cycle is too random and doesn't take the repetitiveness of music and the location of chords in cycles of tension and release into account, which leads to an usable chord progression that doesn't make sense".*
>
> — Participant 8

This feedback and others of the same type suggest that the task could have probably been explained better, so that the participants do not rate the chord progression itself but the way it is played (it was explained in the task description but it might have been misunderstood). It is also possible that asking participants to rate rhythm guitar guitar excerpts while asking them to ignore chord progression details was unrealistic.

Finally, while chord diagrams were also determined beforehand, they were at times judged inappropriate or too difficult by 7 participants (P 2, 5, 8, 11, 13, 20, 22):

> *"Most guitarists would use movable chord shapes with typical fingerings instead. These tabs are misleading for novice players."*
>
> — Participant 22

Again, these aspects should technically not have been rated by the participants, but their mention by 7 participants highlight the importance of chord positions in the perceived playing difficulty of rhythm guitar tablatures.

Subjective Feelings As there were questions on the *Interest* and *Usability* of tablature excerpts shown, participants expressed both positive and negative feedback. The justifications were often based on subjective feelings like the *"naturalness"* (P 13) or the *"musicality"* (P 12, 0, 7) of the excerpt. For instance, one participant comments:

> *"what I found interesting here are the suspended chords and the corresponding rhythmic pattern"*
>
> — Participant 28

while another mentioned the *"awful sound and uninteresting and boring melody"* (P 6). Others explicitly discussed the usability of the examples, mostly negatively: *"the rhythm is weak so no possibility or reusing it identically or even partially"* (P 8); *"The usability is questionable for me. In general, I choose the chord positions, voicings, and rhythmic accents myself."* (P 28)

Overall, the answers show limited interest in the suggestions provided, and they were rarely considered usable directly. One participant in particular (last quote) explained that

the tablature made little sense to them in a context of rhythm guitar, since they would usually play a rhythm of their choosing on the specified chords. However, while there is an observable negativity bias in the answers (Norris 2021), many answers are also neutral as participants simply explain what guided their ratings. From those answers, we hypothesise that a limitation of the survey is that it only allowed participants to rate a static set of sample, where all conditioning values were fixed. Conducting interviews with guitar players where they can use the transformer model in real-time by controlling the different conditioning parameters might change their perception of the interest of having such a tool. Real-time interactions with the models could also allow them to further explore whether consistency is opposed to interest or not.

Obstacles encountered when rating Finally, participants also mentioned what bothered them when rating the excerpts. Some (P 8, 12, 16) found the notation unclear *"interesting rhythm pattern, but not well-written in the score"* (P 8). Participants 11 and 13 sometimes found the tablature notation overly precise:

> *"to request strumming sometimes 2 strings, sometimes 3 and sometimes 4 strings feels like a bit too difficult to play exactly live."*
>
> — Participant 11

This overly precise string notation by the transcribers might be motivated by the goal of an optimal audio playback rendering within the tablature notation software (Bacot et al. 2024). Indicating precisely what strings are strummed allow the playback sound to resemble the original recording closely, but this level of precision is detrimental when a guitarist tries to learn the tablature. This observation suggest that tablatures with different levels of details might be necessary depending on the use-case envisioned for them, like precise playback as a background loop for composing, or clear picking patterns for learning. Another crucial issue with the notation, mentioned by 10 participants, is the use of *dead-notes* (muted noted, denoted by x) in the tablatures. Some were in the original excerpts, but many were added due to the transformer model predicting OoD notes. Using dead notes seemed the natural way of notating such OoD notes to us, as we cannot predict an appropriate fret for those notes automatically. Keeping those suggested OoD notes but making them muted seemed like a fair way to evaluate the transformer model, but it bothered participants:

> *"In [the first example] the 'X' in the score is saying 'play the strings, but muting them'. (which is a different meaning than the 'X' had in [previous] examples, where X was 'avoid plucking'. […] in [the second example] the 'X' feels like meaning 'avoid plucking this'. I do not like it. I prefer how it is notated in [the third sample], there are no X and it feels [like] the best notation."*
>
> — Participant 11

The importance given to those dead-notes by participants suggest that reducing the amount of OoD notes should be a priority in improving the transformer model. They could also be removed through a (possible manual) post-processing step, without reducing the model's performance.

8.6 Conclusion

In this chapter, we presented the task of picking pattern generation that can, when combined with chord diagram suggestion, be used for rhythm guitar tablature continuation. We proposed two different models for picking pattern generation, one rule-based and one transformer-based. The transformer based model can use multiple conditioning signals to control the output, while the rule-based model implements an adaptive copy that is simpler and faster. We devised multiple metrics to automatically evaluate the performance of the models, and conducted an online survey to get subjective feedback on generated samples. While digital metrics suggest that the transformer model performs best, subjective feedback rated the rule-based copier closer to the reference data. Detailed analyses of the online study also gave interesting feedback on what guitar players look at most when considering rhythm guitar tablatures. They however also highlighted the limitations of this online study, and we hypothesise that the transformer-based model would be better evaluated through interactive use and semi-guided interviews. Considered approaches for improving the transformer model include adding dead notes in the tokenisation process to explicitly model them, and developing a second post-processing model that could add meaningful out-of-diagram notes like for transitioning between chords. All code for reproducing this chapter's experiments and results will be shared publicly after publication of the results.

9

Conditional Bass Tablature Generation

"Some guitarists had to play bass because there weren't enough to go around. You can always tell when a bass player is actually a guitarist, they look like they've been demoted. But I don't see it like that, I really like being the bass player. "

Baldwin (2022)

Contents

Many guitarists can be tempted to play bass guitar because of the instruments' apparent similarity. However, the role bass plays in WPM is fundamentally different from that of rhythm guitar (Goyal 2008). For this reason, composers who play guitar do not necessarily have the knowledge to write realistic and idiomatic bass parts in their tablatures. In this chapter, we present a model for bass tablature generation, given an existing rhythm guitar tablature. This task was one of the first identified in the TABASCO project and expressed by guitarist composers as of interest to them, along with drum track generation (Bacot et al. 2024). Olivier Anoufa programmed all code for this task during his second year of a Master's program in *Data Science*, as a research project supervised by Ken Déguernel and me. This chapter is partly based on his report, which was updated and completed by Ken Déguernel and me for the following publication:

"Conditional Generation of Bass Guitar Tablature for Guitar Accompaniment in Western Popular Music"

Olivier Anoufa, Alexandre D'Hooge, Ken Déguernel

Proceedings of the AI Music Creativity Conference (AIMC), 2025. (Anoufa et al. 2025)

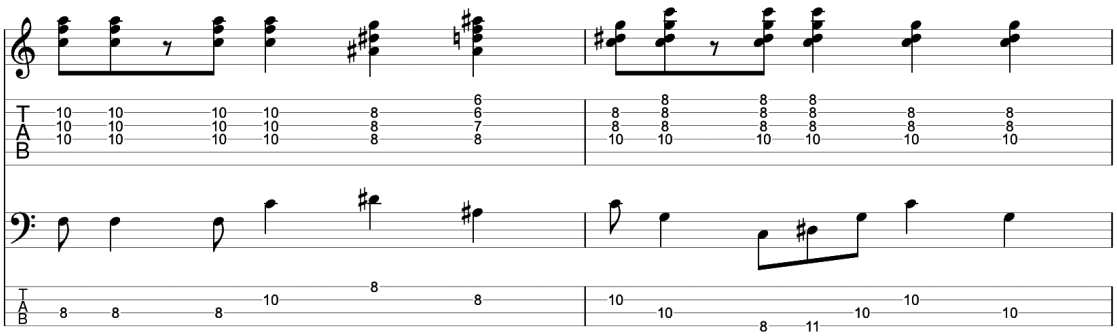


Figure 9.1: Tablatures for rhythm guitar (top) and bass (bottom). Excerpt from *Living in the Past* by Jethro Tull. One can see that, overall, the bass and rhythm guitar are consistent harmonically and rhythmically, but the bass part does not copy the rhythm guitar.

9.1 Introduction

In WPM, the bass guitar provides the harmonic foundation of a tune whilst driving the rhythm section (Goyal 2008; Hove et al. 2014). Composing engaging bass lines is therefore an important aspect of song writing. A user study by Bacot et al. revealed a significant demand among guitarists for accompaniment generation tools (Bacot et al. 2024). More precisely, several guitarists expressed during semi-structured interviews the desire for a tool which, given a guitar part they composed themselves, would generate bass lines and drum parts. Bacot et al. (2024) explain that, during the composition process, whilst preparing their demo, guitarists often resort to writing basic bass and drum lines to accompany their compositions, due to a limited familiarity with these instruments. In this chapter, we consider the task of composing a bass line from a rhythm guitar tablature alone, which can be typical for a singer-songwriter guitar player. More specifically, we focus on conditional symbolic music generation for bass guitar tablatures (Figure 9.1) within the context of WPM. Such a system might allow guitarist composers to assess what their compositions would sound like with a bass part, before actually working with a bass player.

While the role bass plays in WPM is usually more than a mere accompaniment, it is rarely the lead instrument of a song, and generating plausible bass tablatures is thus akin to the MIR task of accompaniment generation. Suggesting bass tracks is already studied in the audio domain for WPM in Grachten et al. (2020) and Pasini et al. (2024), an approach generalised to any accompaniment tracks of a lead melody in Nistal et al. (2024). However, to the best of our knowledge, there is no published work that studies the generation of bass guitar tablatures. Preliminary experiments with the GTR-CTRL model (Sarmento et al. 2023a) showed that control tokens in a prompt are not enough to drive the model towards generating bass tracks, as other instruments eventually appear in the generated score. That approach is also restricted to free generation so the bass tracks can never really match an existing guitar tablature.

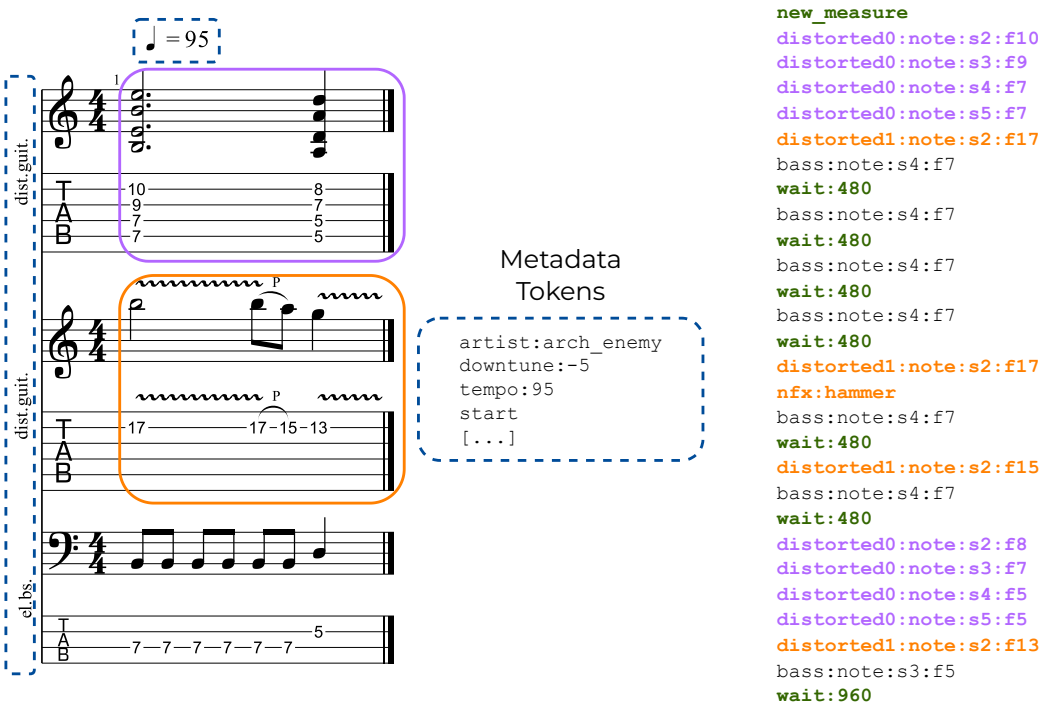


Figure 9.2: Example measure of a tablature with two distorted guitars and a bass, and the corresponding tokens (right) from the DadaGP dataset. Excerpt from *Pilgrim* by Arch Enemy.

Those experiments confirmed the need for a custom model and method for generating bass guitar tablatures for accompaniment. As a first step towards this goal, the contributions of this paper are as follows: i) Isolating bass guitar and rhythm guitar tracks from the DadaGP dataset (section 9.2); ii) Adapting an existing conditional accompaniment generation architecture to bass guitar tablature (section 9.3); iii) A qualitative analysis of the generated content, with future steps to improve bass tablature generation (section 9.4).

9.2 Data Preparation

The DadaGP datasets contain multiple songs with bass tablatures and is therefore the dataset of choice for this work. Unlike other chapters where the Guitar Pro files were used, this chapter use the preprocessed token files directly. The data preparation pipeline presented in this section consists of three stages: (1) parsing and preprocessing the DadaGP token files to extract bass guitar tablature information, (2) identifying rhythm guitar tracks to serve as conditioning input, and (3) segmenting the data into manageable sequences for model training.

9.2.1 Tokenisation and Preprocessing

The DadaGP tokenisation format adopts an event-based approach (Le et al. 2024), similar to other music generation models, by representing musical events as discrete tokens. A measure and its corresponding tokens is shown Figure 9.2 for reference. All token files start with *metadata* tokens that state the artist (if known), the starting tempo of the song and an optional *downtuning*. This last element refers to the distance in semitones between the current string pitches and the *standard tuning*: (E_1, A_1, D_2, G_2) for a 4-string bass. The musical content is then encoded with `new_measure` tokens and notes are written in the form of `<instrument>:note:<string>:<fret>` (e.g. `bass:note:s4:f7`) as a way to directly encode tablature notation. Each note token can be followed by note effects token (`nfx`) that represent playing techniques that are also denoted in the tablatures (Josel et al. 2014). Finally, unlike REMI encoding where the note duration is provided right after the note token (Y.-S. Huang et al. 2020), the DadaGP tokenisation uses `wait` tokens that indicate the number of ticks to wait before playing the next note (960 ticks per quarter note). This approach allows for shorter token sequences since a single `wait` token may apply to multiple notes, while also being more robust than MIDI-like encodings since every new note mutes the previous one (on the same instrument), suppressing the risk of missing `NOTE_OFF` tokens.

To prepare the data for training, we follow a structured preprocessing pipeline that ensures clean and well-formatted input sequences. The first step involves retrieving and cleaning the DadaGP dataset, extracting tokens specific to selected instruments while preserving relevant note effect tokens and metadata. As we want to generate bass guitar tablatures conditioned on rhythm guitar, we developed a function to extract tokens specific to those instruments. Preliminary experiments showed that reprocessing all GuitarPro files one track at a time would have required unnecessarily high computational power and time. To perform the extraction of instrument-specific tokens, we choose to filter tokens from the already existing files. We leverage the fact that tokens of a given instrument start with the instrument name followed by a colon, e.g. "`bass:`" for the bass guitar, to automatically detect such tokens. However, we must be careful not to miss the potential note effect tokens that are not instrument-specific, but come right after the token they are related to. After extracting those tokens and the general tokens (metadata tokens and wait tokens), we sum any consecutive wait tokens that were previously separated by notes from instruments that are no longer present (Figure 9.3). We end up with 14 480 bass track token files. We also remove `bfx` (bar effects, for tempo changes for instance) tokens to reduce the sequences' complexity. Those `bfx` tokens are unrelated to the meter and have no impact on the final rhythm conversion of the detokeniser algorithm. We consider that removing `bfx` tokens is a minor simplification, as modifications like tempo changes can always be added in a post-processing step if the guitarist needs them.

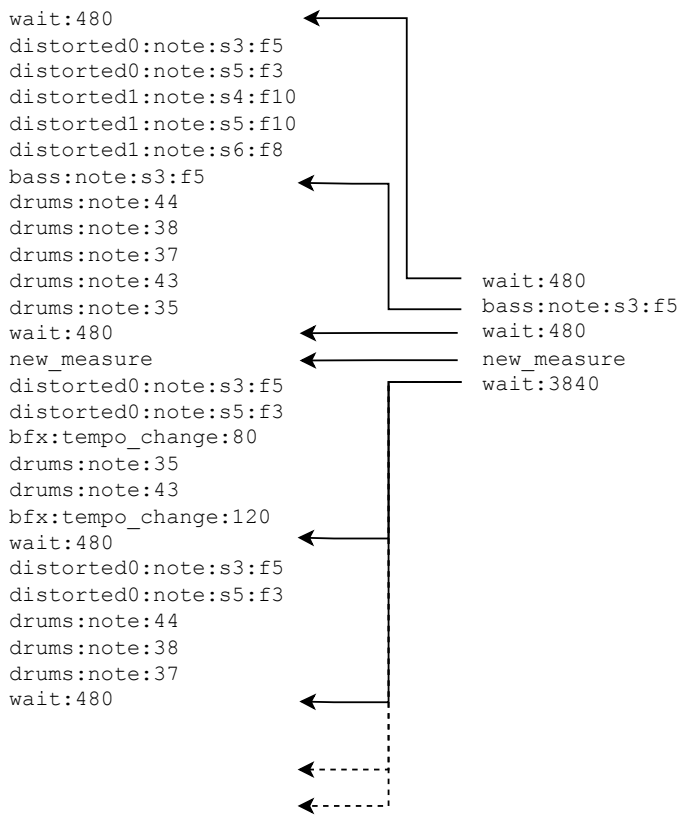


Figure 9.3: Example of an extraction of bass guitar tokens in *Last Nite* by The Strokes.

Rhythm guitar identification The method presented above allows us to extract tokens for any instrument in the DadaGP dataset. While the bass guitar in WPM often establishes the low-end foundation and root notes of the chord progression, the rhythm guitar complements this by providing the full harmonic voicings and rhythmic texture in a higher frequency range. These characteristics makes rhythm guitar therefore particularly relevant for conditioning bass generation tasks and are thus used as the input of our system. Drum tracks could also provide the bass generation model with a strong rhythmic foundation but, because our objective is to assist guitarist composers who do not play the drums or bass, assuming a pre-existing drum track seems unrealistic. We therefore decide to condition the generation of bass tablatures on rhythm guitar tablatures only. Rhythm guitar tracks are identified using the methods from Régnier et al. (2021) like in previous chapters.

9.2.2 Sequence Extraction and Filtering

To avoid sequences that are too long and that could not be processed by the model used in this chapter (BiLSTM), we chose to extract sequences of 16 measures from the dataset's songs. We extract sequences with a sliding window of 8 measures, which allows for some overlap between the sequences. This will help the model learn transitions between measure groups, and observe the various possible contexts around a given sequence.

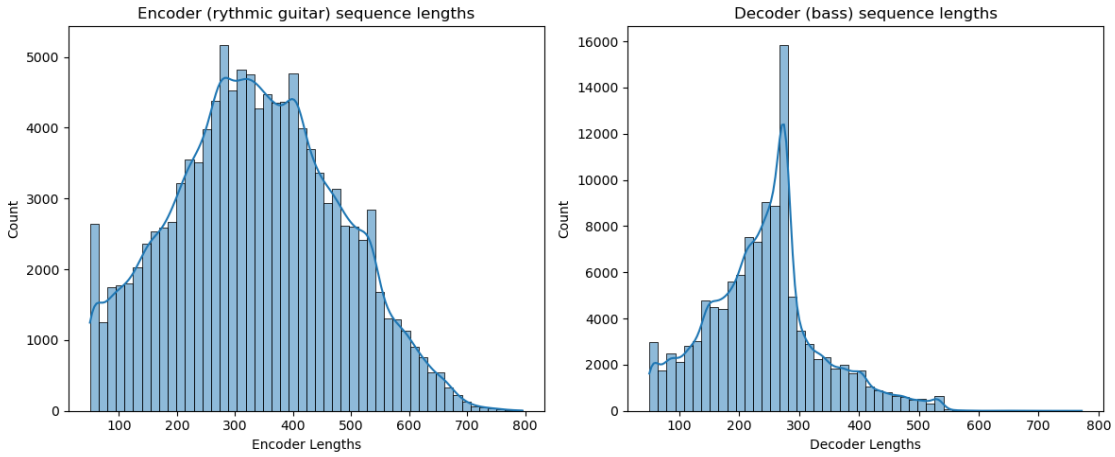


Figure 9.4: Distribution of sequence lengths in the training dataset

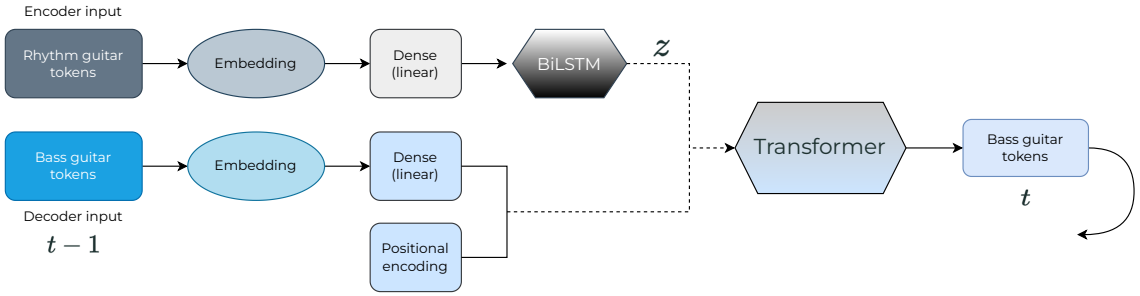


Figure 9.5: Adapted implementation of the model from Makris et al. (2022).

A maximum threshold of 800 tokens, and a minimum threshold of 50 tokens were also set to avoid training on sequences that are either too long or that almost do not contain rhythm guitar or bass. Manual verification showed that no artists were discarded when applying those thresholds. Figure 9.4 shows the distribution of sequence lengths in the training dataset. The majority of the sequences are between 200 and 400 tokens long for the rhythm guitar and between 100 and 300 tokens long for the bass. The final dictionary of sequences is finally split into training, validation and test sets with proportions 0.8, 0.1 and 0.1 respectively. We ensure that all sequences from a same file are in the same subset, to limit data-leakage risks. We extracted 118 167 sequences in total, which leads to 94 533 sequences in the training set and 11 817 in both the validation and test sets.

9.3 Method

We base our model on the conditional drum generation approach of Makris et al. (2022). It is based on a bidirectional Long Short-Term Memory (BiLSTM) that first encodes the input track (rhythm guitar in our case) before sending it to a Transformer Decoder with Relative Global Attention (Y.-S. Huang et al. 2020) that generates the output tokens (the bass guitar track).

Figure 9.5 shows the model adapted to our task. Unlike the original implementation that uses a Compound-Word architecture, we merge all encoders and process the input tokens sequentially with a single encoder network to use the DadaGP tokenisation format directly. The decoder network also generates tokens in a single sequence.

The Encoder processes rhythm guitar tokens using a BiLSTM network, allowing the model to capture dependencies both forward and backward in the sequence. The BiLSTM generates a latent variable z, which encodes both rhythmic and structural information from the input sequence. This latent variable serves as a compressed representation of the rhythm guitar part. The Decoder follows an autoregressive approach, predicting each token sequentially. At each time step $t-1$, the Decoder receives the previously generated token, embedded through a dense layer, along with the latent variable z from the Encoder, which provides contextual information. The Decoder is based on a Transformer architecture with self-attention layers, allowing it to model long-range dependencies within the generated sequence. Unlike standard Transformers that rely solely on absolute positional encodings, it incorporates Relative Global Attention to enhance the model's ability to capture hierarchical musical structures. This mechanism enables the model to account for both global position information and relative relationships between musical events, which is crucial for generating coherent bass lines that align with the rhythm guitar.

We set the relative attention window size to half of the total sequence length. This balances local context within measures and global context across phrases, improving the overall consistency of the generated bass lines. The final output of the Transformer Decoder consists of relation-aware self-attention activations h_t, which are passed through a Dense layer followed by a softmax activation. Our adapted model consists of approximately 21 million parameters, comparable to other large models such as the Pop Music Transformer, which has around 41 million parameters (Y.-S. Huang et al. 2020). To train our model, we use a loss function based on categorical cross-entropy and an accuracy metric designed for sequence modelling. The loss function measures how well the predicted probability distribution aligns with the true tokens and is defined as:

$$\mathcal{L}(\hat{y}, y) = -\sum_{t=1}^{T}\sum_{k=1}^{K} y_{t,k} \log \hat{y}_{t,k} \tag{9.1}$$

where T is the sequence length, K is the vocabulary size, $y_{t,k}$ is the ground-truth token at time step t, and $\hat{y}_{t,k}$ is the predicted probability for token k. The ground-truth token is the token that effectively appears in the original bass guitar sequence at time step t. Our accuracy metric computes the proportion of correctly predicted tokens in a sequence. Both functions incorporate a masking mechanism to handle padding tokens and remove their contribution to the calculations. While the reference bass sequence is by no means the only valid answer in a musical sense, it is used during training as the expected prediction of the model.

Implementation details The architecture is the same as Makris et al. (2022), with only the sizes of the encoding and input/output layers adapted to our problem. The resulting model contains 4 attention layers with 8 attention heads, with encoder and decoder embeddings being of size 240 and 192, respectively. The BiLSTM network has 1024 units in total, 512 for each direction.

Training is done on GPU with an early-stopping criterion of 5 epochs without improvement on the validation loss. We use the Adam optimizer with the following hyperparameters: $\alpha = 5 \times 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 10^{-9}$, and a batch size of 32.

9.4 Qualitative Analysis of the Generation

As this bass line generation tool is currently a proof-of-concept, this section focuses on a qualitative analysis of generated bass guitar tablatures. While quantitative evaluation methods are valuable for measuring model accuracy in well-defined tasks, they are less suitable for assessing computational creativity tasks where multiple valid outputs may exist (Jordanous 2012). For instance, edit distance or perplexity metrics only assess if the generation is close to the reference, which does not measure the musical quality of the output. Other high-level metrics might seem attractive, to measure harmonic and rhythmic consistency for instance, but a bass playing "dissonant" notes or diverging from the guitar's rhythm are not necessarily wrong musical decisions. We thus choose to not implement any automatic quantitative metrics, as we believe they would bring little value to the current evaluation.

The model presented in this paper is designed to help guitarist composers generate idiomatic and musically coherent bass lines given their composed rhythm guitar tablatures. In this context, qualitative analysis provides more informative insights into the musical relevance, performability, as well as the limitations of the generated bass lines (Ritchie 2019). Analyses were conducted by defining codes and assigning examples to each code. The generations were explored randomly thanks to a demonstration website, and the exploration was stopped once no new codes seemed necessary to reflect the content of the generated tablatures. The final categories were cross-checked by the other authors for further grouping of similar codes. We are aware of the limitations and possible biases that can occur from such subjective analyses with limited raters. Nevertheless, the evaluation was conducted in an inductive way, meaning that codes emerged from the samples, and were later linked with musical knowledge to reflect upon the model's performance. By making an exploration website publicly available,[1] we leave to readers the opportunity to check the generated samples freely, allowing them to verify our claims.

[1] https://github.com/adhooge/BassTablatureGeneration/, accessed in June 2025.

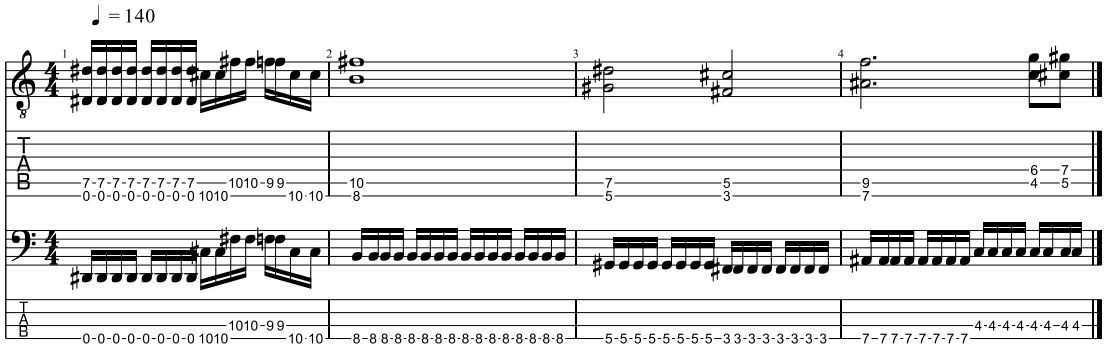


Figure 9.6: Generated example 518. Here, the bass plays full 16th notes even though the guitar part is more sparse.

We detail the codes identified hereafter. First, we analyse the harmonic and rhythmic consistency of the generated bass lines with respect to the conditioning guitar part. Second, we assess the naturalness of the resulting tablatures in term of bass guitar technique, focusing on physical playability and idiomatic movement across the instrument. Finally, we examine the system's ability to stay stylistically relevant to the guitar part. Each code is illustrated by tablature excerpts and we also point to the identifiers of additional examples from the demonstration website with hyperlinked numbers in parentheses.[2]

9.4.1 Harmonic and Rhythmic Features

We first observe that the generated bass tablatures can *highlight chord tones* even when they are not explicitly played in the guitar part. The bass also tends to *fill voids and rests* with relevant rhythms and notes (ids 518, 1029, 1215, Figure 9.6). The generated bass parts can also *anticipate* the upcoming guitar chords and play notes that match the upcoming key (899). Notes that are unexpected in the current harmonic context can also be encountered, even without being completely dissonant (1133, 1306). Rhythmically, the model has a *tendency to generate eighth notes*, or sixteenth notes in a filling non-stop fashion (1151). *Less common rhythms* can however be encountered, either following the guitar (873, 26) or diverging from it (172). Nonetheless, it happens that the bass is rhythmically consistent with itself without really matching with the rhythm of the guitar (877). Another limitation is that the bass part can be harmonically and rhythmically consistent but *simplistic* (952). The model might benefit from more varied data where the bass diverges from the guitar in an interesting way.

[2]You can also access each example by changing the website URL, e.g. for id 222: https://adhooge.github.io/BassTablatureGeneration/?id=222, accessed in July 2025.

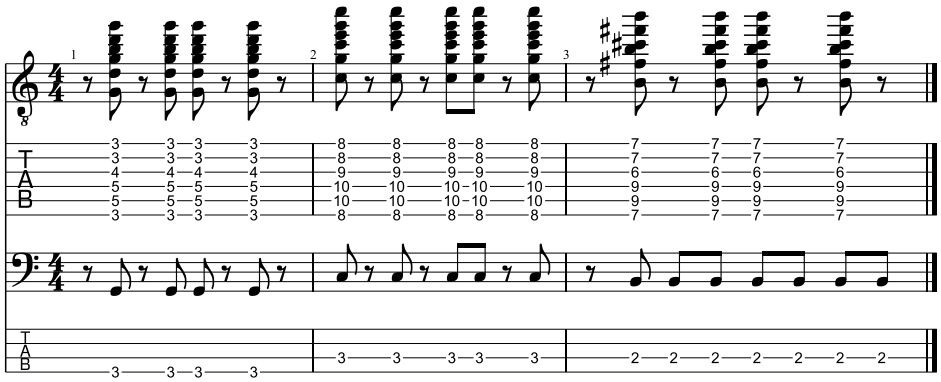


Figure 9.7: Generated example 209. While the guitar goes up the fretboard, the bass moves perpendicularly.

9.4.2 Gesture and Bass Guitar Idiomaticity

A satisfying observation is that the generated tablatures usually seem to replicate idiomatic bass guitar playing. For example, while the bass tends to replicate what the guitar plays, it plays *single notes rather than chords*, that are harder and less common in bass guitar (121, 126, 949). Playing techniques like dead notes can also be generated (22), as can be slides or hammer-on and pull-off. Another important observation is that the tablatures tend to contain moves perpendicular to the fretboard (209, 752, 1209, Figure 9.7) which is more common among bass players, notes resonating more fully closer to the beginning of the fretboard and limiting hand movements being an important consideration. Octave-jumps are also typical bass-guitar playing and can be encountered (534, Figure 9.8) as well as typical string-jumps (1275). It is also worth noting that the bass is not always playing accompaniment and melodic parts are sometimes generated (917, 961, 1351). Similarly, "transition" notes (1354) or chromatic notes (965), typically encountered in bass guitar parts, are generated.

But the generated content can also be very repetitive (1119, 83, 474, 1067) and while this can happen in actual WPM bass parts, it would be more interesting to generate parts with more variation to suggest varied parts to guitar players. Finally, the bass can sometimes stays too close to the rhythmic guitar input, resulting in difficultly playable tablatures with bends for instance (1167).

9.4.3 Stylistic Consistency

A good example of how the model adapts to different musical styles is on metal subgenres, largely represented in the DadaGP dataset. We observe that the generated bass matches the required style by closely following the rhythm guitar (268, 288, 718, 719, Figure 9.9). Likewise, rock music (or any subgenre) is usually well reproduced, closely following the guitar part but with additional minor variations (844). Conversely, styles of guitar playing that are less represented in DadaGP leads to less appropriate bass parts (1355), like when more complex chords akin to jazz music are used by the guitar.

When the guitar is more melodic (which can happen even if the track is rhythm guitar overall), the model still manages to suggest a bass track that matches the harmony and the style of the melody (795, 499, 166, 588, Figure 9.10). However, the bass part can sometimes repeat itself too much and miss mood changes (68), which looks like a mistake even though it is up to musicians to decide if this is unacceptable. Another limitation to the current system is that it sometimes seems to be late (725) or early (84) on following the changes in the way the guitar plays.

9.5 Conclusion

In this chapter, we explored the conditional generation of bass guitar tablatures using a transformer-based model trained on the DadaGP dataset. We implemented a detailed data preprocessing pipeline, including instrument-specific token extraction, rhythm guitar identification, and structured sequence formatting. Our approach enables the generation of bass lines that are harmonically and rhythmically coherent with a given rhythm guitar part. Our results demonstrate that the model effectively captures the structural role of the bass guitar in WPM, successfully aligning with the rhythmic and harmonic elements of the accompaniment. However, we also identified certain limitations, such as the model's tendency to replicate rhythmic patterns excessively and being less idiomatic in styles that are less represented in the dataset.

While our study establishes a strong foundation for conditioned bass tablature generation, further refinements in model design, data representation, and evaluation methods could enhance its practical applicability. One potential enhancement is simplifying the model by removing one of the dense layers in the embedding process, which could reduce complexity without significantly impacting performance. Additionally, experimenting with compound word tokenisation as an alternative to the current DadaGP method might yield more meaningful representations and help the model during the training phase. To further improve the model's flexibility and idiomaticity across diverse musical styles, we could fine-tune dedicated sub-models for specific styles or incorporate style descriptor tokens during training to condition the generation on a target style.

Looking ahead, such a bass tablature generation tool should be evaluated by bass players to assess the quality of the tablatures, and by guitar players who could rate its usefulness. Because the validity of the tablatures is a subjective notion, an online survey with pre-defined rhythm guitar tablatures might not allow participants to correctly assess the capabilities of the model. Semi-guided interviews where guitarist composers have access to the system and can use it freely and on specific tasks would probably yield more relevant feedback.

All code and data to reproduce this work are shared publicly[3] under GPL-3.0 (for the code) and ODbL (for the data) licenses.

[3] https://github.com/adhooge/BassTablatureGeneration/, accessed in July 2025.

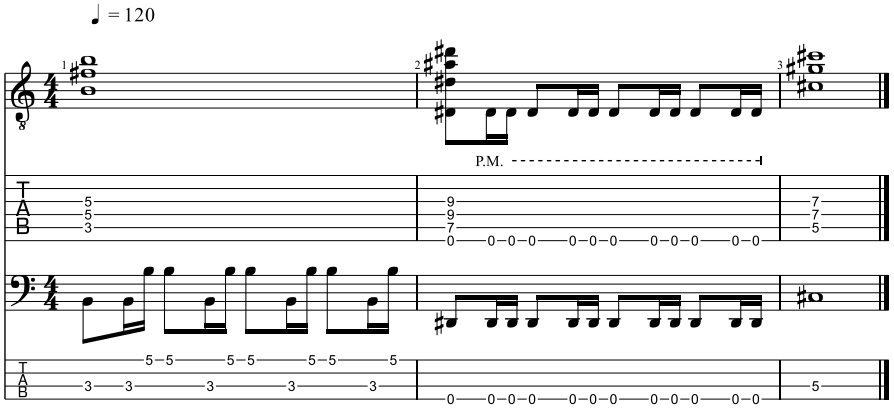


Figure 9.8: Generated example 534. Example of idiomatic octave jumps in the bass part.

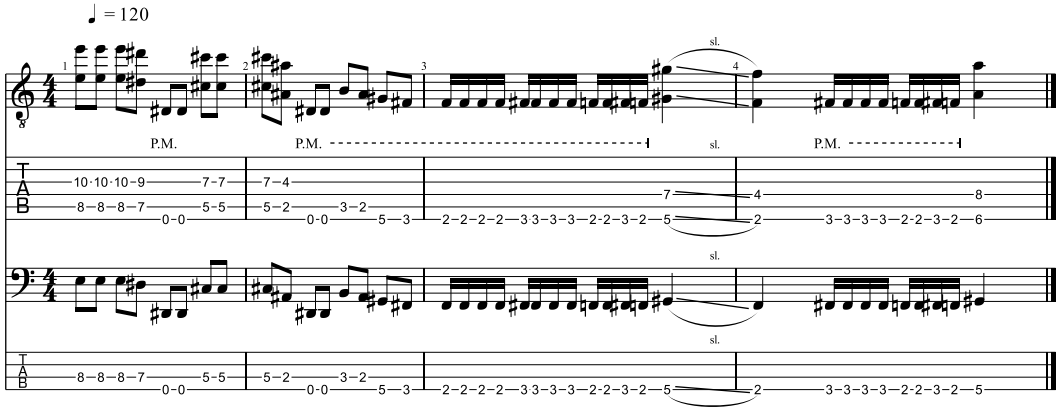


Figure 9.9: Generated example 719. This is a fast metal excerpt, the bass follows the guitar closely but reduces chords to single notes.

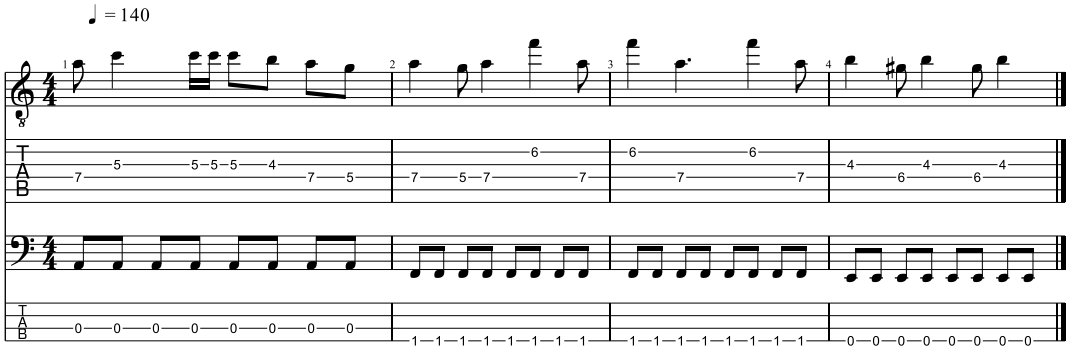


Figure 9.10: Generated example 588. Even though the guitar plays a melodic part, the bass manages to make the underlying chords heard.

Part IV

Conclusion and Perspectives

Summary of Contributions

The contributions of this thesis focus on two core aspects of guitar practice in WPM: learning and composing.

Regarding *guitar learning*, this thesis introduced two approaches for assisting WPM guitar players in their learning of the instrument by helping them navigate the ever-increasing amount of learning materials available. In chapter 4, we built upon difficulty ratings of songs to devise a recommendation system for guitar learners, based on an estimation of their current level. This system was designed and evaluated in collaboration with a guitar teacher, and is deployed in a commercial application that will allow to garner user feedback for future improvements. Chapter 5 extended existing work on song difficulty analysis to guitar and bass tablatures, based on a newly gathered dataset and detailed analyses of possible difficulty-related features. Using a Naive-Bayes model yielded satisfactory results for difficulty prediction but questions persist on which visualisations and features would be most helpful to learners.

Regarding *composition*, this thesis presented various methods destined to help guitarists compose with tablatures, with the objective to change the current usage of tablature notation as a means of learning and communication. In chapter 1, we presented the specificities of tablature notation and how it is used by WPM guitarists. In particular, we argued that guitar players rarely use tablature notation while composing, motivating our research on AI methods that could make composition within tablature notation software more appealing. After introducing the ML techniques used throughout this thesis (chapter 2), we showed in chapter 3 that while research on guitar tablatures for automatic transcription or generation has been studied, little importance was given to composers and how they might benefit from this research. In an effort to assist composers, we proposed new models that can help writing both lead guitar and rhythm guitar parts. For lead guitar, chapter 6 presented a method to suggest where to add guitar playing techniques in a tablature, with the objective to make the resulting song more idiomatic. Improving the parsing of bends also led to a contribution to the `music21` Python library.[1] Focusing on bends, we showed that they can be modelled and suggested efficiently, with a model that can ultimately be integrated in tablature notation software with gradual controls. For rhythm guitar, chapter 7 analysed how chords are played on guitar to design a model that can suggest appropriate chord positions from minimal context. Chapter 8 was built upon this previous work and presented two models for suggesting picking patterns that, combined with a chord progression and the corresponding positions, can propose possible continuations to a rhythm guitar prompt in tablature format. Finally, chapter 9 studied a transformer model to provide guitar composers with a possible bass accompaniment, in tablature format, given a rhythm guitar tablature they composed. Overall, the models presented were evaluated quantitatively when possible, i.e. when clear metrics could be

[1] https://github.com/cuthbertLab/music21/pull/1580, accessed in July 2025.

used to assess performance. Some metrics were newly introduced in this thesis, for instance to describe the texture of guitar tablatures. However, the musical tasks studied also often called for qualitative analyses of the results. Most were conducted by the authors, based on their own musical expertise except for Chapter 8 that was evaluated by external participants through an online questionnaire.

We conclude this thesis with perspectives on new research that could be conducted to further the contributions of this thesis.

Perspectives

The research presented in this thesis tackled some of the tasks identified to assist guitarists when learning or composing, but also opened perspectives on new tasks that could be studied. This section presents those perspectives, starting from improvements to the thesis' contributions and generalising to new tasks.

Improving Usability The tools presented in this thesis would need some refining before being deployed for guitarists to use them. The key limitation of our current systems is that most need programming knowledge to be used, which would exclude many guitarists. Next steps therefore include making the systems more user-friendly by directly including them in tablature notation software. In that regard, Léo Dupouey worked as a research engineer to integrate some of the suggestion tools to a tablature viewer based on *alphaTab*[2] (Figure 6.19) and a second research engineer might continue his work to further deploy the models, in the scope of the ANR TABASCO project. Ultimately, the AI models presented in this thesis might be of interest to the open-source developer community, to be integrated in Musescore or TuxGuitar for instance. Proprietary websites and software could also be interested in deploying such tools, be it for difficulty evaluation (Songsterr, UltimateGuitar) or assisted composition (Guitar Pro). Integration of the models presented in this thesis to existing software and websites would be an ideal scenario, as it would make them available to a wide range of guitarists without requiring them to add new tools in their workflow.

Improving Computational Efficiency The models' efficiency should also be improved as much as possible to reduce the computational load and time required to generate results, to permit their usage on a wide range of users' devices. Indeed, some of the models used in this thesis like the bass tablature generation one (chapter 9) need to be ran on GPUs and require several dozens seconds to produce a result, limiting their usability by musicians. Implementing methods like pruning, quantisation or model distillation are possibilities to increase the computational efficiency of the models. It would also be worth for future work to consider new performance metrics in the development phase of AI models, to evaluate models both in terms of evaluation metrics and energy requirements. This is important because the energy consumption of AI models is far from negligible. For instance, the contributions presented in this thesis were estimated to have required 1 MWh of energy in total, from the early experimentations to the final inferences on test sets. More explanations that led to this final figure as well as more performance metrics can be found in Appendix B.

[2] https://alphatab.net/, accessed in June 2025.

Subjective Evaluations Once the models are deployed in tablature notation software, they should be evaluated subjectively. Online studies could be deployed, but in-depth ethnographic interviews with guitar composers would be ideal, replicating the preliminary studies conducted by Baptiste Bacot (Bacot 2023; Bacot et al. 2024) by conducting semi-guided interviews and other qualitative methods where participants can use the proposed tools for a few hours. Such prolonged use would clearly expose the strengths and weaknesses of the models designed, and provide additional guidance for future improvements.

Studying New Tasks Many new ways of assisting the composition of guitar tablatures could also be studied, like *inpainting* (Hadjeres et al. 2021) or *overpainting* (Row et al. 2023) to suggest variations on an existing tablature. One could also imagine a model for suggesting riffs or solo licks, to help guitarists consider playing styles they are less familiar with, with uncommon rhythm or chords for the former, or new scales and modes for the latter. Another promising task to study would be to combine our tablature difficulty analysis model with generation systems to allow for arranging a song into different difficulty levels, similarly to what has already been studied for piano scores (Gover et al. 2022). It might also be interesting to analyse the playing difficulty of tablatures to provide the players with exercises focusing on those difficult aspects, either drawing them from a preassembled database, or generating them on-demand by deriving them from the tablature studied.

Computational Musicology Studies of Guitar Tablatures Considering a different approach, many computational musicology studies could be conducted on the tablature data available to improve our understanding of the guitar WPM repertoire. As done by Sarmento et al. 2023b, one could analyse tablatures to determine whether famous guitarists styles are detectable directly from transcriptions, or if some artists are better identified by performance aspects currently not captured by tablatures such as dynamics, effect pedals, or special performing gestures. While some analyses on the datasets were conducted in this thesis, the thousand of tablatures available could provide more insights on guitar WPM. It would also be interesting to compare multiple tablature transcriptions of the same song, to determine why multiple versions can exist as it might reflect different perspectives on the transcription of WPM. Furthermore, one could imagine studying specific subsets of the DadaGP dataset, for instance only bands with female players, to see if specific trends emerge from those songs compared to the majority of bands with only male players.

Towards Multimodal Tablature Research Future work might benefit from adopting a multimodal approach, i.e. combining symbolic tablature data with audio or video recordings. By doing so, one might combine Automatic Music Transcription models with tablature-based models to directly suggest modifications or additions to what a guitarist

just played. Incorporating audio data could also allow tablature models to benefit from performance data currently absent from (most) tablature notation, like dynamics or effect pedals. Video recordings could also be used to enhance models for fingering prediction (what finger plays what note on the fretboard) or strumming directions for chords. With advanced multimodal models, one could even imagine interactive digital agents that generate tablatures and play them through digital instruments, for a guitarist to interact with while playing music. Considering a model that could dynamically react to what the guitarist plays, it could provide suggestions, alternatives or co-create in tablature or audio format in a way that does not require popular music guitar players change the way they usually compose songs, which is by jamming individually or in a band (Bacot et al. 2024; L. Green 2002).

Supporting Artists and Limiting Plagiarism All those perspectives would greatly benefit from a new multimodal dataset that contains tablatures, video and audio recordings from different contexts. It would be particularly beneficial to the community that such a dataset is assembled with the artists' consent, to guarantee that its usage is legal for all (or most) research applications, and to participate in strengthening the link between artists (and the general public) and the AI research community in contexts where distrust can increase. Indeed, it is unclear under current EU law if generative AI models can be trained on copyrighted data without the authors' consent. Possible risks of plagiarism also exist and should be considered, as generative AI models can sometimes memorise and reproduce part of their training set identically. More details on EU law regarding AI and possible risks of plagiarism of the contributions of this thesis are provided in Appendix C.

Reflect upon the Interactions between AI and Guitarists Finally, while all models presented in this thesis are designed to help guitar learners and composers, AI tools can be a source of distrust between scientists and artists. Even though we consider that the contributions presented are unlikely to impact artists or music teachers negatively because of the specific tasks they focus on, collecting subjective feedback would strengthen those claims. Ultimately, AI music models should be developed iteratively with musicians, to ensure that the models actually support the creative process, rather than models that create artificial needs. AI distrust and the possible risks of the contributions of this thesis replacing guitarists are discussed further in Appendix D. Overall, we believe that well-designed AI methods have the potential to contribute significantly to co-creative practices. For musicians to notice their potential and overcome the reluctance of using AI in their work, a focus should be put on developing transparent and understandable models that might even lead more people to start learning, performing and composing music.

Bibliography

ABRSM (2021). *Music Performance Grades: Guitar* (cit. on pp. 90, 92).

AC/DC, ed. (2008). *Black Ice*. Wise Publications (cit. on p. 21).

Adkins, S. et al. (2023). "LooperGP: A Loopable Sequence Model for Live Coding Performance Using GuitarPro Tablature". In: *Proc. of the Int. Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)* (cit. on pp. 50, 61, 144, 146).

Agostinelli, A. et al. (2023). *MusicLM: Generating Music From Text*. Pre-published (cit. on p. 57).

Alammar, J. (2018). *The Illustrated Transformer*. URL: https://jalammar.github.io/illustrated-transformer/ (visited on 2025-04) (cit. on p. 41).

Alonso-Jiménez, P. et al. (2023). "Efficient Supervised Training of Audio Transformers for Music Representation Learning". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 38, 88).

Anoufa, O. et al. (2025). "Conditional Generation of Bass Guitar Tablature for Guitar Accompaniment in Western Popular Music". In: *Proc. of the AI Music Creativity Conf. (AIMC)* (cit. on pp. 8, 167).

Apel, W. (1942). *The Notation of Polyphonic Music, 900-1600*. Cambridge, Mass., Mediaeval Academy of America (cit. on pp. 12, 14, 19).

Ariga, S. et al. (2017a). "Song2Guitar: A Difficulty Aware Arrangement System for Generating Guitar Solo Covers from Polyphonic Audio of Popular Music". In: *Proc. of the 18th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 4, 53, 61, 63).

Ariga, S. et al. (2017b). "Strummer: An Interactive Guitar Chord Practice System". In: *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)* (cit. on pp. 4, 53, 64).

Arobas-Music (2025). *Guitar Pro - Logiciel d'édition de Tablatures Pour Guitare, Basse, Piano, Batterie et Plus...* URL: https://www.guitar-pro.com/fr/ (visited on 2025-02) (cit. on p. 15).

Arsenault, D. (2008). "Guitar Hero:" Not like Playing Guitar at All"?" In: *Loading...* 2.2 (cit. on pp. 65, 66).

Artist Rights Alliance (2024). *200+ Artists Urge Tech Platforms: Stop Devaluing Music*. Medium. URL: https://artistrightsnow.medium.com/200-artists-urge-tech-platforms-stop-devaluing-music-559fb109bbac (visited on 2025-05) (cit. on p. 221).

Assayag, G. et al. (2006). "OMax Brothers: A Dynamic Topology of Agents for Improvization Learning". In: *Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia* (cit. on p. 58).

Bacot, B. (2023). *Composing with Tablature Software: An Inquiry*. URL: https://hal.science/hal-04432306 (visited on 2025-01). Pre-published (cit. on pp. 144, 184).

Bacot, B. et al. (2024). "Enjeux Du Logiciel de Tablatures Dans l'acte de Création En Musiques Actuelles : Méthode d'entretien et Analyse d'une Pratique". In: *Actes Des Journées d'Informatique Musicale (JIM)* (cit. on pp. 3, 6, 20, 22, 123, 148, 165, 167, 168, 184, 185).

Baldwin, B. (2022). *The Gearbox Interview: Becky Baldwin – Bass Guitarist*. URL: https://devolutionmagazine.co.uk/2022/06/03/gearbox-interview-becky-baldwin-bass-guitarist (cit. on p. 167).

Balman, F. (2020). *Tablature Disdain: Music Pedagogues' Preference for Staff Notation and Its Impact on the Average Guitarist*. Pre-published (cit. on p. 11).

Barbancho, A. M. et al. (2012). "Automatic Transcription of Guitar Chords and Fingering From Audio". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.3 (cit. on p. 138).

Barbancho, I. et al. (2014). "Estimation of the Direction of Strokes and Arpeggios". In: *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 163).

Barenboim, G. et al. (2024). "Exploring How a Generative AI Interprets Music". In: *Neural Computing and Applications* 36.27 (cit. on p. 57).

Barthet, M. et al. (2011). "Music Recommendation for Music Learning: Hottabs, a Multimedia Guitar Tutor". In: *Workshop on Music Recommendation and Discovery (WOMRAD)* (cit. on pp. 3, 15, 64).

Bas, C. L. (2020). "Frugal Innovation as Environmental Innovation". In: *International Journal of Technology Management* 83.1–3 (cit. on p. 215).

Batlle-Roca, R. et al. (2024). "Towards Assessing Data Replication in Music Generation with Music Similarity Metrics on Raw Audio". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 148, 219, 220).

Beato, R., director (2021). *The Modern Guitar Discussion w/ Tosin Abasi, Tim Henson & Misha Mansoor*. URL: https://www.youtube.com/watch?v=6wb4AcfXSyo (visited on 2025-05) (cit. on p. 105).

Bertrand, F. (1998). "La « Composition » Pour Guitare Dans Le Rock'n'roll : Problèmes d'Analyse". In: *Musurgia* 5.2 (cit. on p. 20).

Bimbot, F. et al. (2016). "System & Contrast". In: *Music Perception* 33.5 (cit. on pp. 18, 145, 147, 151, 152).

Blalock, D. et al. (2020). "What Is the State of Neural Network Pruning?" In: *Proc. of the 3rd MLSys Conf.* (Cit. on p. 215).

Bontempi, P. (2025). "An Innovative Computer-Based Model for the Generation of Expressive Lead Guitar Performances". PhD thesis. Università degli Studi di Padova (cit. on pp. 106, 107).

Bontempi, P. et al. (2024). "From MIDI to Rich Tablatures: An Automatic Generative System Incorporating Lead Guitarists' Fingering and Stylistic Choices". In: *Proc. of the 21st Sound and Music Computing Conf. (SMC)* (cit. on pp. 7, 106).

Born, G. et al. (2021). *Artificial Intelligence, Music Recommendation, and the Curation of Culture*. White Paper. URL: http://hdl.handle.net/1807/129105 (visited on 2025-07) (cit. on p. 3).

Bosch, J. J. et al. (2012). "A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals". In: *Proc. of the 13th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 46).

Bosseur, J.-Y. (2005). *Du son au signe : histoire de la notation musicale*. Editions Alternatives (cit. on p. 19).

Braun, V. et al. (2006). "Using Thematic Analysis in Psychology". In: *Qualitative Research in Psychology* 3.2 (cit. on p. 162).

Breiman, L. (2017). *Classification and Regression Trees*. Routledge (cit. on p. 117).

Breiman, L. et al. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC (cit. on p. 32).

Briot, J.-P. et al. (2019). *Deep Learning Techniques for Music Generation – A Survey*. Prepublished (cit. on p. 58).

Briot, J.-P. et al. (2020). *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer Nature Switzerland (cit. on pp. 29, 34).

Burgoyne, J. A. et al. (2011). "An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis". In: *Proc. of the 12th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 53, 54, 70).

Burgoyne, J. A. et al. (2015). "Music Information Retrieval". In: *A New Companion to Digital Humanities*. John Wiley & Sons, Ltd (cit. on p. 1).

Cabral, G. et al. (2005). "Automatic X Traditional Descriptor Extraction: The Case of Chord Recognition". In: *Proc. of the 6th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 46).

Cádiz, R. F. et al. (2021). "Creativity in Generative Musical Networks: Evidence From Two Case Studies". In: *Frontiers in Robotics and AI* 8 (cit. on p. 57).

Cavanagh, T. (2019). *What Inspired Dicey Dungeons?* distractionware. URL: https://distractionware.com/blog/2019/08/4-what-inspired-dicey-dungeons/ (visited on 2025-02) (cit. on p. 45).

Challis, B. (2009). *The Song Remains the Same: A Review of the Legalities of Music Sampling*. WIPO. URL: https://www.wipo.int/web/wipo-magazine/article-details/?assetRef=37091&title=the-song-remains-the-same-a-review-of-the-legalities-of-music-sampling (visited on 2025-05) (cit. on p. 219).

Chan, T. F. et al. (1982). "Updating Formulae and a Pairwise Algorithm for Computing Sample Variances". In: *COMPSTAT 1982 5th Symposium Held at Toulouse*. Ed. by H. Caussinus et al. Physica-Verlag HD (cit. on p. 98).

Chatti, M. A. et al. (2024). "Visualization for Recommendation Explainability: A Survey and New Perspectives". In: *ACM Trans. Interact. Intell. Syst.* 14.3 (cit. on p. 99).

Chawla, N. V. et al. (2002). "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (cit. on p. 116).

Chemla–Romeu-Santos, A. et al. (2022). "Challenges in Creative Generative Models for Music: A Divergence Maximization Perspective". In: *Proc. of the AI Music Creativity Conf. (AIMC)* (cit. on p. 57).

Chen, Y.-H. et al. (2020). "Automatic Composition of Guitar Tabs by Transformers and Groove Modeling". In: *Proc. of the 21st Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 5, 51, 54, 60, 152).

Chen, Y.-H. et al. (2022). "Towards Automatic Transcription of Polyphonic Electric Guitar Music: A New Dataset and a Multi-Loss Transformer Model". In: *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (cit. on pp. 46, 55).

Chen, K. et al. (2020). "Music Sketchnet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm". In: *Proc. of the 21st Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 58).

Chen, T.-P. et al. (2020). "Chord Jazzification: Learning Jazz Interpretations of Chord Symbols". In: *Proc. of the 21st Int. Society for Music Information Retrieval Conf.* (Cit. on p. 129).

Chen, W. et al. (2019). "Data Usage in MIR: History & Future Recommendations". In: *Proc. of the 20th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 219).

Cheng, J. et al. (2008). "A Neural Network Approach to Ordinal Regression". In: *Proc. of the IEEE Int. Joint Conf. on Neural Networks (IJCNN)* (cit. on p. 97).

Chiu, S.-C. et al. (2012). "A Study on Difficulty Level Recognition of Piano Sheet Music". In: *IEEE International Symposium on Multimedia* (cit. on pp. 63, 90, 92).

Cho, K. et al. (2014). "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)* (cit. on p. 40).

Choudhury, M. (2023). "Generative AI Has a Language Problem". In: *Nature Human Behaviour* 7.11 (cit. on p. 57).

Coffman, T. (2023). *James Hetfield Is the Greatest Rhythm Guitarist of All Time*. Far Out. URL: https://faroutmagazine.co.uk/hear-me-out-james-hetfield-is-the-greatest-rhythm-guitarist-of-all-time/ (visited on 2025-02) (cit. on p. 23).

Colton, S. et al. (2012). "Computational Creativity: The Final Frontier?" In: *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI'12)* (cit. on p. 57).

Cooper, A. F. et al. (2023). *Report of the 1st Workshop on Generative AI and Law* (cit. on pp. 26, 28).

Cooper, A. F. et al. (2024). *The Files Are in the Computer: Copyright, Memorization, and Generative AI*. Pre-published (cit. on pp. 217, 219).

Corder, G. W. et al. (2009). *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. John Wiley & Sons (cit. on p. 89).

Cournut, J. et al. (2020). "Encodages de Tablatures Pour l'analyse de Musique Pour Guitare". In: *Actes Des Journées d'Informatique Musicale (JIM)* (cit. on pp. 2, 17, 48, 50).

Cournut, J. et al. (2021). "What Are the Most Used Guitar Positions?" In: *8th Int. Conf. on Digital Libraries for Musicology (DLFM'21)* (cit. on pp. 2, 5, 48, 52).

Couturier, L. (2024). "Modélisation Informatique De La Texture Pour L'analyse De Musique Symbolique Et L'aide À La Composition." PhD thesis. Université de Picardie Jules Verne. URL: https://theses.fr/s301124 (visited on 2025-06) (cit. on p. 140).

Couturier, L. et al. (2023). "Comparing Texture in Piano Scores". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 146).

Craig, C. (2022). "The AI-Copyright Challenge: Tech-Neutrality, Authorship, and the Public Interest". In: *All Papers* 360 (cit. on p. 219).

Craig, C. J. (2024). "The AI-Copyright Trap". In: *Osgoode Legal Studies Research Paper* (cit. on p. 218).

Craig, C. J. et al. (2021). "The Death of the AI Author". In: *Ottawa Law Review* 52.1 (cit. on p. 220).

Cui, W. et al. (2024). "MoodLoopGP: Generating Emotion-Conditioned Loop Tablature Music with Multi-Granular Features". In: *Proc. of the Int. Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)* (cit. on pp. 50, 61).

Cunha, N. et al. (2018). "Generating Guitar Solos by Integer Programming". In: *Journal of the Operational Research Society* 69.6 (cit. on pp. 56, 60).

Cuthbert, M. S. et al. (2010). "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data". In: *Proc. of the 11th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 17, 48, 113).

Cwitkowitz, F. et al. (2022). "A Data-Driven Methodology for Considering Feasibility and Pairwise Likelihood in Deep Learning Based Guitar Tablature Transcription Systems". In: *Proc. of the 19th Sound and Music Computing Conf. (SMC)* (cit. on p. 50).

D'Hooge, A. et al. (2023a). "Modeling Bends in Popular Music Guitar Tablatures". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 7, 105).

— (2023b). "Rhythm Guitar Tablature Continuation from Chord Progression and Tablature Prompt". In: *Digital Music Research Network One-day Workshop (DMRN+18)* (cit. on p. 142).

D'Hooge, A. et al. (2024a). "Guitar Chord Diagram Suggestion for Western Popular Music". In: *Proc. of the 21st Sound and Music Computing Conf. (SMC)* (cit. on pp. 7, 127).

D'Hooge, A. et al. (2024b). "Suggestions Pédagogiques Personnalisées Pour La Guitare". In: *Actes Des Journées d'Informatique Musicale (JIM)* (cit. on pp. 8, 69, 86).

Dahia, M. et al. (2004). "Using Patterns to Generate Rhythmic Accompaniment for Guitar". In: *Actes Des Journées d'Informatique Musicale (JIM)* (cit. on pp. 53, 59).

Dai, Z. et al. (2019). "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics* (cit. on p. 60).

Dalitz, C. et al. (2013). "From Facsimile to Content Based Retrieval: The Electronic Corpus of Lute Music". In: *Phoibos - Zeitschrift für Zupfmusik* 2 (cit. on p. 13).

Dalmazzo, D. et al. (2024a). "ChromaFlow: Modeling and Generating Harmonic Progressions with a Transformer and Voicing Encoding". In: *MML 2024: 15th Int. Workshop on Machine Learning and Music*. URL: https://hal.science/hal-04710950 (cit. on p. 129).

— (2024b). "The Chordinator: Modeling Music Harmony By Implementing Transformer Networks and Token Strategies". In: *Proc. of the Int. Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)* (cit. on pp. 129, 147).

Dart, T. et al. (2001). *Tablature*. Vol. 1. Oxford University Press (cit. on pp. 12, 19).

Das, O. et al. (2018). "Analyzing and Classifying Guitarists from Rock Guitar Solo Tablature". In: *Proc. of the 15th Sound and Music Computing Conf. (SMC)* (cit. on pp. 51, 54).

Déguernel, K. et al. (2022). "Personalizing AI for Co-Creative Music Composition from Melody to Structure". In: *Proc. of the 19th Sound and Music Computing Conf. (SMC)* (cit. on p. 58).

Déguernel, K. et al. (2023). "Emotion, Motion, and Abstract Notions: Insights in the Role of Imagination in Professional Musicians Practices from Semi-Guided Interviews". In: *Proc. of the Int. Conf. on Music Perception and Cognition (ICMPC)* (cit. on p. 62).

De Prisco, R. et al. (2016). "Visualization of Music Plagiarism: Analysis and Evaluation". In: *Proc. of the 20th Int. Conf. Information Visualisation (IV)* (cit. on p. 220).

De Sa, V. (1993). "Learning Classification with Unlabeled Data". In: *Advances in Neural Information Processing Systems* (cit. on p. 29).

Devlin, J. et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. arXiv (cit. on p. 42).

Dhariwal, P. et al. (2020). *Jukebox: A Generative Model for Music*. Pre-published (cit. on p. 57).

Dib, L. (2024). "Formal Definition of Interpretability and Explainability in XAI". In: *Intelligent Systems and Applications (IntelliSys)*. Springer, Cham (cit. on p. 31).

Dickinson, S. (2019). "Building From Within: A Harmonic Self-Help Guide For Guitar Chord Voicings, Their Densities, and Applications". Doctoral Essay. University of Miami (cit. on p. 221).

Dornis, T. W. et al. (2025). *Generative AI Training and Copyright Law*. Pre-published (cit. on pp. 218, 219).

Douwes, C. et al. (2021). *Energy Consumption of Deep Generative Audio Models*. Pre-published (cit. on p. 213).

Downie, J. S. (2003). "Music Information Retrieval". In: *Annual Review of Information Science and Technology* 37 (cit. on p. 1).

Edwards, D. et al. (2024). "MIDI-to-Tab: Guitar Tablature Inference via Masked Language Modeling". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 5, 50, 61, 109, 122).

Eiben, A. E. et al. (2010). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer (cit. on p. 61).

Eichner, M. et al. (2006). "Instrument Classification Using Hidden Markov Models". In: *Proc. of the 7th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 46).

Eldby, T. (2021). "Investigating Representation of Tablature Data for NLP Music Prediction". MA thesis. UiT Norges arktiske universitet (cit. on pp. 50, 54).

Elowsson, A. et al. (2013). "Modelling Perception of Speed in Music Audio". In: *Proc. of the 10th Sound and Music Computing Conf. (SMC)* (cit. on p. 70).

Ens, J. et al. (2020). *MMM : Exploring Conditional Multi-Track Music Generation with the Transformer*. Pre-published (cit. on p. 58).

Eremenko, V. et al. (2020). "Performance Assessment Technologies for the Support of Musical Instrument Learning". In: *Proc. of the 12th Int. Conf. on Computer Supported Education (CSEDU)* (cit. on pp. 71, 82).

Faul, F. et al. (2007). "G*Power 3:A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences". In: *Behavior Research Methods* 39.2 (cit. on p. 159).

Ferretti, S. (2016). "Guitar Solos as Networks". In: *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)* (cit. on pp. 51, 53).

Fitch, W. T. et al. (2007). "Perception and Production of Syncopated Rhythms". In: *Music Perception* 25.1 (cit. on p. 91).

Furcy, D. et al. (2005). "Limited Discrepancy Beam Search". In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)* (cit. on p. 84).

Garcia, E. et al. (2007). "The Evolution of Robotics Research". In: *IEEE Robotics & Automation Magazine* 14.1 (cit. on p. 26).

Gelling, P. (2003). *Learn to Play Lead Guitar Manual: Progressive Complete*. LearnToPlayMusic.com Pty Limited (cit. on p. 20).

Gholami, A. et al. (2021). "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *Low-Power Computer Vision*. Chapman and Hall/CRC (cit. on p. 216).

Giraud, M. et al. (2014). "Towards Modeling Texture in Symbolic Data". In: *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*. URL: https://hal.science/hal-01057017 (visited on 2023-08) (cit. on pp. 31, 140).

Goertzel, B. (2014). "Artificial General Intelligence: Concept, State of the Art, and Future Prospects". In: *Journal of Artificial General Intelligence* 5.1 (cit. on p. 26).

Gomez, P. J. (2016). "Modern Guitar Techniques; a View of History, Convergence of Musical Traditions and Contemporary Works (A Guide for Composers and Guitarists)". PhD thesis. UC San Diego (cit. on pp. 20, 108).

Goodfellow, I. et al. (2016). *Deep Learning*. Adaptive Computation and Machine Learning. The MIT Press (cit. on pp. 28–30, 34).

Gotham, M. et al. (2023). "When in Rome: A Meta-corpus of Functional Harmony". In: *Transactions of the Int. Society for Music Information Retrieval* 6.1 (cit. on p. 31).

Gover, M. et al. (2022). "Music Translation: Generating Piano Arrangements in Different Playing Levels". In: *Proc. of the 23rd Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 58, 184).

Goyal, N. (2008). "Bass Guitar in Rock Music: Current State, Potential and Possibilities". Post-Graduate Dissertation. Mudra Institute of Communications, Ahmedabad (cit. on pp. 167, 168).

Grachten, M. et al. (2020). "BassNet: A Variational Gated Autoencoder for Conditional Generation of Bass Guitar Tracks with Learned Interactive Control". In: *Applied Sciences* 10.18 (cit. on pp. 6, 168).

Green, A. (2017). *Jazz Guitar Comping: Raising Your Chord Awareness*. Mel Bay Publications (cit. on p. 143).

Green, L. (2002). *How Popular Musicians Learn: A Way Ahead for Music Education*. Ashgate Publishing, Ltd. (cit. on pp. 1, 14, 18, 20, 58, 142, 185, 222).

Griffiths, J. (2021). "Turning the Tables: Reassessing Tablature". In: *MSA Annual Conf.* (cit. on pp. 11, 12, 14, 19).

Grimes, D. R. (2014). "String Theory - The Physics of String-Bending and Other Electric Guitar Techniques". In: *PLoS ONE* 9.7 (cit. on p. 108).

Guilluy, Q. et al. (2025). "Vers Une Taxonomie et Une Analyse Des Gestes Guitaristiques Dans Le Brutal Death Metal". In: *Actes Des Journées d'Informatique Musicale (JIM)* (cit. on pp. 7, 8).

Guitar Pro8 User Guide (2025). URL: https://static.guitar-pro.com/gp8/manual/Guitar-Pro-8-user-guide.pdf (visited on 2025-02) (cit. on pp. 20, 148).

Guo, C. et al. (2017). "On Calibration of Modern Neural Networks". In: *Proc. of the 34th Int. Conf. on Machine Learning (ICML)* (cit. on p. 120).

Hadjeres, G. et al. (2021). *The Piano Inpainting Application*. Pre-published (cit. on pp. 58, 184).

Hafsa, M. et al. (2022). "A Multi-Objective e-Learning Recommender System at Mandarine Academy." In: *Proceedings of the 2nd Workshop on Multi-Objective Recommender Systems Co-Located with 16th ACM Conference on Recommender Systems (RecSys 2022)*. URL: https://hal.science/hal-03956402 (cit. on p. 64).

Hamel, P. et al. (2009). "Automatic Identification of Instrument Classes in Polyphonic and Poly-Instrument Audio". In: *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 46).

Hart, P. E. et al. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (cit. on p. 83).

Hassein-Bey, Z. et al. (2025). "What Song Now? Personalized Rhythm Guitar Learning in Western Popular Music". In: *Proc. of the 26th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 8, 69, 86).

Havre, S. J. et al. (2019). "Playing to Learn or Learning to Play? Playing *Rocksmith* to Learn Electric Guitar and Bass in Nordic Music Teacher Education". In: *British Journal of Music Education* 36.1 (cit. on p. 65).

Herbst, J.-P. et al. (2024). *Rock Guitar Virtuosos: Advances in Electric Guitar Playing, Technology, and Culture*. 1st ed. Cambridge University Press (cit. on p. 221).

Herbst, J.-P. et al. (2025). "The Benefits of Collaborative Popular Music Songwriting: A Spectrum of Artist-Songwriter Involvement". In: *Popular Music and Society* 48.1 (cit. on p. 222).

Herremans, D. et al. (2017). "A Functional Taxonomy of Music Generation Systems". In: *ACM Computing Surveys* 50.5 (cit. on p. 58).

Hess, J. (2020). "Finding the "Both/and": Balancing Informal and Formal Music Learning". In: *Int. Journal of Music Education* 38.3 (cit. on p. 222).

Hiller, L. et al. (1979). *Experimental Music; Composition with an Electronic Computer*. Greenwood Publishing Group Inc. (cit. on p. 27).

Hochreiter, S. et al. (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8 (cit. on p. 39).

Holzapfel, A. et al. (2018). "Ethical Dimensions of Music Information Retrieval Technology". In: *Transactions of the Int. Society for Music Information Retrieval* 1.1 (cit. on pp. 219, 220).

Holzapfel, A. et al. (2024). "Green MIR? Investigating Computational Cost of Recent Music-Ai Research in Ismir". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 213, 214).

Hori, G. (2021). "Three-Level Model for Fingering Decision of String Instruments". In: *Proc. of the 15th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)* (cit. on pp. 123, 138).

Hori, G. et al. (2013). "Input-Output HMM Applied to Automatic Arrangement for Guitars". In: *Journal of Information Processing* 21.2 (cit. on p. 92).

Hove, M. J. et al. (2014). "Superior Time Perception for Lower Musical Pitch Explains Why Bass-Ranged Instruments Lay down Musical Rhythms". In: *Proc. of the National Academy of Sciences* 111.28 (cit. on p. 168).

Huang, C.-Z. A. et al. (2019). "Music Transformer: Generating Music with Long-Term Structure". In: *Proc. of the 7th Int. Conf. on Learning Representations (ICLR)* (cit. on p. 58).

Huang, C.-Z. A. et al. (2020). "AI Song Contest: Human-AI Co-Creation in Songwriting". In: *Proc. of the 21st Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 58).

Huang, Y.-S. et al. (2020). "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions". In: *MM '20: Proc. of the 28th ACM Int. Conf. on Multimedia* (cit. on pp. 2, 59, 60, 170, 172, 173).

Humphrey, E. J. et al. (2014). "JAMS: A JSON Annotated Music Specification for Reproducible MIR Research". In: *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 55).

Ian, S. (2018). *'The Magic of Malcolm Young'*. URL: https://www.youtube.com/watch?v=w4IFImDwnHk (visited on 2025-06) (cit. on p. 143).

Ji, S. et al. (2023). "A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges". In: *ACM Computing Surveys* 56.1 (cit. on pp. 58, 59).

Jordanous, A. (2012). "A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What It Is to Be Creative". In: *Cognitive Computation* 4.3 (3) (cit. on pp. 30, 174).

— (2016). "Four PPPPerspectives on Computational Creativity in Theory and in Practice". In: *Connection Science* 28.2 (cit. on p. 61).

Josel, S. et al., eds. (2014). *The Techniques of Guitar Playing*. Bärenreiter (cit. on p. 170).

Kaliakatsos-Papakostas, N. et al. (2022). "A Machine Learning Approach for MIDI to Guitar Tablature Conversion". In: *Proc. of the 19th Sound and Music Computing Conf. (SMC)* (cit. on p. 61).

Keating, M. et al. (2024). "Jazz Guitar Voice-Leading Chord Fingerings With Long Short-Term Memory". In: *Proc. of the AI Music Creativity Conf. (AIMC)* (cit. on pp. 56, 65).

Kehling, C. et al. (2014). "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-related Parameters". In: *Proc. of the 17th Int. Conf. on Digital Audio Effects (DAFx-14)* (cit. on p. 46).

Kelly, J. (2007). "Pop Music, Multimedia and Live Performance". In: *Music, Sound and Multimedia: From the Live to the Virtual*. Jamie Sexton. Edinburgh University Press (cit. on p. 222).

Kendall, M. G. (1945). "The Treatment of Ties in Ranking Problems". In: *Biometrika* 33.3 (cit. on p. 93).

Kim, S. (2025). "Édition, Distribution, Utilisation". In: *Journées d'accélération de l'ICCARE-LAB : Partition et Numérique* (cit. on p. 15).

Kingma, D. P. et al. (2015). "Adam: A Method for Stochastic Optimization". In: *3rd Int. Conf. for Learning Representations (ICLR)* (cit. on p. 38).

Kolb, T. (2020). *String-Bending Masterclass: How to Make Your Guitar Wail and Sing Like the Pros*. Guitar Player. URL: https://www.guitarplayer.com/lessons/string-bending-masterclass-how-to-make-your-guitar-wail-and-sing-like-the-pros (visited on 2023-01) (cit. on p. 108).

Koozin, T. (2011). "Guitar Voicing in Pop-Rock Music: A Performance-Based Analytical Approach". In: *Music Theory Online* 17.3 (cit. on pp. 51, 221).

Krizhevsky, A. et al. (2017). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (cit. on p. 26).

Laato, S. et al. (2022). "How to Explain AI Systems to End Users: A Systematic Literature Review and Research Agenda". In: *Internet Research* 32.7 (cit. on p. 99).

Laliberté, M. (2010). "Notations pour la guitare électrique". In: *Appareil* 5 (cit. on pp. 14, 20).

Larson, T. (2018). *Sarah Longfield, Queen of the 8-String*. YouTube - Music is Win. URL: https://youtu.be/LkIFETF17qc?si=znZ6GiRi_Pn6Isnp&t=187 (cit. on p. 69).

Le, D.-V.-T. et al. (2024). "Natural Language Processing Methods for Symbolic Music Generation and Information Retrieval: A Survey". In: *ACM Computing Surveys* 57.7 (cit. on pp. 1, 2, 39, 46, 58, 59, 170).

Le Cun, Y. (2019). *Quand La Machine Apprend : La Révolution Des Neurones Artificiels et de l'apprentissage Profond*. Odile Jacob (cit. on p. 34).

Leech-Wilkinson, D. (2012). "Compositions, Scores, Performances, Meanings". In: *Music Theory Online* 18.1 (cit. on p. 58).

Longuet-Higgins, H. C. et al. (1984). "The Rhythmic Interpretation of Monophonic Music". In: *Music Perception* 1.4 (cit. on p. 91).

Loth, J. et al. (2023). "ProgGP: From GuitarPro Tablature Neural Generation To Progressive Metal Production". In: *Proc. of the 16th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)*. Zenodo (cit. on pp. 50, 60, 62).

Lourenço, J. M. (2021). *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. iii).

Lu, L. et al. (2004). "Audio Textures: Theory and Applications". In: *IEEE Transactions on Speech and Audio Processing* 12.2 (cit. on p. 140).

Lundberg, S. M. et al. (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. (cit. on pp. 32, 100).

Ma, H. et al. (2023). "Classification of Guitar Tab and Numbered Musical Notation Using ResNet50 Network". In: *Proc. of the 3rd Int. Conf. on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC)* (cit. on pp. 52, 54).

Maccarini, F. (2024). "Modeling Orchestration for Computer Assisted Analysis and Human-AI Co-Creativity". PhD thesis. Université de Lille (cit. on p. 58).

Macrae, R. et al. (2011). "Guitar Tab Mining, Analysis and Ranking". In: *Proc. of the 12th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 64).

Makris, D. et al. (2022). "Conditional Drums Generation Using Compound Word Representations". In: *Proc. of the Int. Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)*. Ed. by T. Martins et al. Springer International Publishing (cit. on pp. 172, 174).

Margoudi, M. et al. (2016). "Game-Based Learning of Musical Instruments: A Review and Recommendations". In: *10th European Conf. on Games Based Learning (ECGBL)* (cit. on p. 82).

Margulis, E. H. (2014). *On Repeat : How Music Plays the Mind*. New York, NY : Oxford University Press (cit. on pp. 91, 144).

McCulloch, W. S. et al. (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (4) (cit. on pp. 34, 35).

McKinna, D. R. (2014). "The Touring Musician: Repetition and Authenticity in Performance". In: *IASPM Journal* 4.1 (1) (cit. on pp. 19, 222).

McQueen, H. et al. (2018). "Teachers' and Students' Music Preferences for Secondary School Music Lessons: Reasons and Implications". In: *Music Education Research* 20.1 (cit. on p. 62).

McVicar, M. et al. (2014a). "AutoLeadGuitar: Automatic Generation of Guitar Solo Phrases in the Tablature Space". In: *12th Int. Conf. on Signal Processing (ICSP)*. IEEE (cit. on pp. 53, 60).

— (2014b). "AutoRhythmGuitar: Computer-aided Composition for Rhythm Guitar in the Tab Space". In: *Proc. of the 40th Int. Computer Music Conf. Joint with the 11th Sound and Music Computing Conf. (ICMC | SMC)* (cit. on pp. 53, 60).

— (2015). "AutoGuitarTab: Computer-Aided Composition of Rhythm and Lead Guitar Parts in the Tablature Space". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. Vol. 23 (cit. on pp. 5, 53, 60, 135).

Merchi, G. (1700–1799). *Le Guide Des Écoliers de Guitarre* (cit. on p. 19).

Mesbur, E. E. (2006). "Choosing to Play: Adolescent Girls and Informal Music Learning". Master of Arts. University of Toronto (cit. on p. 222).

Middleton, R. et al. (2001). "Popular Music". In: *Oxford Music Online*. Oxford University Press (cit. on p. 17).

Minsky, M. et al. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press (cit. on p. 35).

Mistler, E. (2017). "Generating Guitar Tablatures with Neural Networks". Master of Science Thesis in Data Science. University of Edinburgh (cit. on pp. 54, 61).

Mitchell, T. M. (1997). *Machine Learning*. Nachdr. McGraw-Hill Series in Computer Science. McGraw-Hill (cit. on pp. 25, 28).

Morreale, F. et al. (2023). "Data Collection in Music Generation Training Sets: A Critical Analysis". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 219).

Müllensiefen, D. et al. (2014). "The Musicality of Non-Musicians: An Index for Assessing Musical Sophistication in the General Population". In: *PLoS ONE* 9.2 (cit. on p. 158).

Müllerschön, M. et al. (2025). "Playability Prediction in Digital Guitar Learning Using Interpretable Student and Song Representations". In: *Proc. of the 26th Int. Society for Music Information Retrieval Conf. (ISMIR)*. URL: https://ismir2025program.ismir.net/poster_238.html (visited on 2025-11) (cit. on p. 64).

Murgul, S. et al. (2025). "Joint Transcription of Acoustic Guitar Strumming Directions and Chords". In: *Proc. of the 26th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 163).

Narici, I. (2025). "Édition, Distribution, Utilisation". In: *Journées d'accélération de l'ICCARE-LAB : Partition et Numérique* (cit. on p. 15).

Navarret, B. (2013). "Caractériser la guitare électrique : définitions, organologie et analyse de données verbales". Thèse de Doctorat. Université Paris 8 (cit. on pp. 14, 15, 19, 20).

Nemeroff, B. (2024). *Lead vs. Rhythm Guitar: What's the Difference?* Fender. URL: https://www.fender.com/articles/instruments/lead-vs-rhythm-guitar (visited on 2025-01) (cit. on p. 22).

Nielsen, S. G. et al. (2023). "Selecting Repertoire for Music Teaching: Findings from Norwegian Schools of Music and Arts". In: *Research Studies in Music Education* 45.1 (cit. on p. 62).

Nierhaus, G. (2009). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Science & Business Media (cit. on p. 57).

Nika, J. et al. (2017). "DYCI2 Agents: Merging the "free", "reactive", and "scenario-Based" Music Generation Paradigms". In: *Proc. of the Int. Computer Music Conf. (ICMC)* (cit. on p. 58).

Nistal, J. et al. (2024). "Diff-a-Riff: Musical Accompaniment Co-Creation Via Latent Diffusion Models". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 6, 168).

Norris, C. J. (2021). "The Negativity Bias, Revisited: Evidence from Neuroscience Measures and an Individual Differences Approach". In: *Social Neuroscience* 16.1 (cit. on p. 165).

Norton, J. C. (2008). "Motion Capture to Build a Foundation for a Computer-Controlled Instrument by Study of Classical Guitar Performance". PhD thesis. Stanford University (cit. on p. 20).

Novack, Z. et al. (2025). "Presto! Distilling Steps and Layers for Accelerating Music Generation". In: *Proc. of the 13th Int. Conf. on Learning Representations (ICLR)* (cit. on p. 216).

OpenAI (2025). *ChatGPT — Release Notes*. URL: https://help.openai.com/en/articles/6825453-chatgpt-release-notes (visited on 2025-05) (cit. on p. 26).

Owens, J. T. (2017). "Power Chords, Blast Beats, and Accordions: Understanding Informal Music Learning in the Lives of Community College Musicians". PhD thesis. Kent State University (cit. on p. 222).

Palmer, C. (2006). "The Nature of Memory for Music Performance Skills". In: *Music, Motor Control and the Brain*. Oxford University Press (cit. on p. 90).

Pasini, M. et al. (2024). "Bass Accompaniment Generation via Latent Diffusion". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (cit. on p. 168).

Paul, D. et al. (2020). "A Survey of Music Recommendation Systems with a Proposed Music Recommendation System". In: *Emerging Technology in Modelling and Graphics*. Springer (cit. on p. 64).

Pease, A. et al. (2011). "On Impact and Evaluation in Computational Creativity: A Discussion of the Turing Test and an Alternative Proposal". In: *Proc. of the AISB Symposium on Computing and Philosophy* (cit. on p. 27).

Pedregosa, F. et al. (2011). "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (cit. on p. 32).

Pedroza, H. et al. (2024). "Leveraging Real Electric Guitar Tones and Effects to Improve Robustness in Guitar Tablature Transcription Modeling". In: *Proc. of the 27th Int. Conf. on Digital Audio Effects (DAFx24)* (cit. on p. 50).

Peeters, G. (2004). *A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project*. Technical Report 1 (cit. on p. 31).

Peterson, J. (2009). "Computer Notation-based Music Composition and the Delayed Introduction of Musical Expression Markings". In: *Journal of Education, Informatics and Cybernetics* 1 (cit. on p. 22).

Petrucci, J. (1995). *Rock Discipline*. Youtube - Guitar Channel. URL: https://youtu.be/3Ababg5Y8kA?si=5LTOkCJz2UU_WtYm&t=2395 (visited on 2025-05) (cit. on p. 85).

Pinheiro, J. et al. (2000). "Linear Mixed-Effects Models: Basic Concepts and Examples". In: *Mixed-Effects Models in S and S-PLUS*. Springer, New York, NY (cit. on p. 161).

Polin, A. (2022). *Guitar Chords For Dummies*. John Wiley & Sons (cit. on p. 128).

Radford, A. et al. (2018). *Improving Language Understanding by Generative Pre-Training*. Pre-published (cit. on p. 42).

Radford, A. et al. (2019). *Language Models Are Unsupervised Multitask Learners*. Pre-published (cit. on p. 147).

Raffel, C. et al. (2014). "'mir_eval': A Transparent Implementation of Common MIR Metrics". In: *Proc. of the 15th Int.Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 30).

Ramoneda, P. et al. (2022). "Score Difficulty Analysis for Piano Performance Meducation Based on Fingering". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Institute of Electrical and Electronics Engineers (IEEE) (cit. on p. 63).

Ramoneda, P. et al. (2023). "Predicting Performance Difficulty from Piano Sheet Music Images". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR 2023)* (cit. on pp. 63, 86).

Ramoneda, P. et al. (2024a). "Combining Piano Performance Dimensions for Score Difficulty Classification". In: *Expert Systems with Applications* 238 (cit. on pp. 63, 86, 98).

Ramoneda, P. et al. (2024b). "Towards Explainable and Interpretable Musical Difficulty Estimation: A Parameter-efficient Approach". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 7, 85, 90, 95, 97, 98, 100).

Randel, D. M. (2003). *The Harvard Dictionary of Music*. Harvard University Press (cit. on p. 58).

Régnier, D. et al. (2021). "Identification of Rhythm Guitar Sections in Symbolic Tablatures". In: *Proc. of the 22nd Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 2, 23, 31, 48, 106, 114, 151, 171).

Rhodes, M. (1961). "An Analysis of Creativity". In: *The Phi Delta Kappan* 42.7 (cit. on p. 221).

RIAA (2024). *Record Companies Bring Landmark Cases for Responsible AI Against Suno and Udio in Boston and New York Federal Courts, Respectively*. RIAA. URL: https://www.riaa.com/record-companies-bring-landmark-cases-for-responsible-ai-againstsuno-and-udio-in-boston-and-new-york-federal-courts-respectively/ (visited on 2025-05) (cit. on p. 217).

Ribeiro, M. T. et al. (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *KDD '16: Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (cit. on p. 32).

Riley, X. et al. (2024a). "GAPS: A Large and Diverse Classical Guitar Dataset and Benchmark Transcription Model". In: *Proc. of the 25th Int. Society for Music Information Retrieval Conf.* (Cit. on pp. 46, 55).

Riley, X. et al. (2024b). "High Resolution Guitar Transcription Via Domain Adaptation". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (cit. on p. 54).

Ritchie, G. (2019). "The Evaluation of Creative Systems". In: *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*. Ed. by T. Veale et al. Computational Synthesis and Creative Systems (CSACS). Springer International Publishing (cit. on p. 174).

Roberts, A. et al. (2019). "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *Proc. of the 35th Int. Conf. on Machine Learning PMLR* (cit. on p. 57).

Rodriguez, R. et al. (2020). "Learning beyond the Game: A Multimodal Analysis of Rocksmith Users' Interactions". In: *Acta Ludologica* 3.2 (cit. on p. 65).

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological Review* 65.6 (cit. on pp. 35, 36).

Row, E. et al. (2023). "JAZZVAR: A Dataset of Variations Found within Solo Piano Performances of Jazz Standards for Music Overpainting". In: *Proc. of the 16th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)* (cit. on pp. 59, 184).

RSL Awards (2023). *Electric Guitar Syllabus* (cit. on pp. 90, 92).

Ruismäki, H. et al. (2012). "The Internet as a Learning Environment in Guitar Playing: Rane's Search for Information and Expertise". In: *Procedia - Social and Behavioral Sciences* 45 (cit. on p. 222).

Rumelhart, D. E. et al. (1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323 (cit. on p. 38).

Sakai, S. et al. (2024). "Tablature Generation from Lead Sheets for Finger-Style Solo Guitar". In: *Proc. of the 21st Sound and Music Computing Conf. (SMC)* (cit. on p. 61).

Samuel, A. L. (1959). "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* (cit. on p. 27).

Sanh, V. et al. (2020). "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter". In: *EMC^2: 5th Edition Co-located with NeurIPS'19*. arXiv (cit. on p. 216).

Sarmento, P. (2024). "Guitar Tablature Generation with Deep Learning". PhD thesis. Queen Mary University of London (cit. on pp. 48, 219).

Sarmento, P. et al. (2021). "DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models". In: *Proc. of the 22nd Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 5, 45, 48–51, 55, 60).

Sarmento, P. et al. (2023a). "GTR-CTRL: Instrument and Genre Conditioning for Guitar-Focused Music Generation with Transformers". In: *Proc. of the Int. Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)* (cit. on pp. 6, 50, 60, 168).

Sarmento, P. et al. (2023b). "ShredGP: Guitarist Style-Conditioned Tablature Generation with Transformers". In: *Proc. of the 16th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)*. Zenodo (cit. on pp. 50, 51, 60, 123, 184).

Satvaty, A. et al. (2025). *Undesirable Memorization in Large Language Models: A Survey*. Prepublished (cit. on p. 219).

Savage, P. E. et al. (2018). "Quantitative Evaluation of Music Copyright Infringement". In: *Proc. of the 8th Int. Workshop on Folk Music Analysis* (cit. on p. 220).

Sawayama, K. et al. (2006). "A System Yielding the Optimal Chord-Form Sequence on the Guitar". In: *Proc. of the Int. Conf. on Music Perception and Cognition (ICMPC)*. Citeseer (cit. on p. 129).

Sayegh, S. I. (1989). "Fingering for String Instruments with the Optimum Path Paradigm". In: *Computer Music Journal* 13.3 (cit. on p. 46).

Sayood, K. (2018). "Information Theory and Cognition: A Review". In: *Entropy* 20.9 (9) (cit. on p. 90).

Schubert, E. (2013). "Emotion Felt by the Listener and Expressed by the Music: Literature Review and Theoretical Perspectives". In: *Frontiers in Psychology* 4 (cit. on p. 62).

Schwartz, R. et al. (2020). "Green AI". In: *Commun. ACM* 63.12 (cit. on pp. 213–215).

Sébastien, V. et al. (2012). "Score Analyzer: Automatically Determining Scores Difficulty Level for Instrumental e-Learning". In: *Proc. of the 13th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 62).

Shiga, J. (2016). "Copying Machines: Unconscious Musical Plagiarism and the Mediatisation of Listening and Memory". In: *Transposition. Musique et Sciences Sociales* 6 (6) (cit. on p. 219).

Shmulevich, I. et al. (2000). "Measures of Temporal Pattern Complexity". In: *Journal of New Music Research* 29.1 (cit. on p. 91).

Sioros, G. (2014). "Syncopation as Transformation". PhD thesis. Universidade do Porto (cit. on p. 92).

Skreinig, L. R. et al. (2022). "AR Hero: Generating Interactive Augmented Reality Guitar Tutorials". In: *Proc. of the IEEE Conf. on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (cit. on p. 65).

Skreinig, L. R. et al. (2023). "guitARhero: Interactive Augmented Reality Guitar Tutorials". In: *IEEE Transactions on Visualization and Computer Graphics* 29.11 (cit. on p. 65).

Snapes, L. (2024). "Miley Cyrus Sued over Allegedly Copying Bruno Mars Song on Flowers". In: *The Guardian. Music*. URL: https://www.theguardian.com/music/2024/sep/17/miley-cyrus-sued-over-allegedly-copying-bruno-mars-song-on-flowers (visited on 2025-05) (cit. on p. 219).

Somdahl-Sands, K. et al. (2015). "Media, Performance, and Pastpresents: Authenticity in the Digital Age". In: *GeoJournal* 80.6 (6) (cit. on p. 19).

Stein, M. et al. (2010). "Automatic Detection of Audio Effects in Guitar and Bass Recordings". In: *Journal of the Audio Engineering Society* 8013 (cit. on p. 46).

Stokes, S. (2019). *Digital Copyright: Law and Practice*. 1st ed. Bloomsbury Publishing Plc (cit. on p. 218).

Sturm, B. L. (2012). "Two Systems for Automatic Music Genre Recognition: What Are They Really Recognizing?" In: *Proc. of the Second Int. ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*. MIRUM '12. Association for Computing Machinery (cit. on p. 29).

— (2017). "The "Horse" inside: Seeking Causes behind the Behaviors of Music Content Analysis Systems". In: *Computers in Entertainment (CIE)* 14.2 (cit. on p. 32).

Temperley, D. (2018). *The Musical Language of Rock*. Oxford University Press (cit. on p. 113).

Thorpe, V. (2007). "We Made This Song. The Group Song Writing Processes of Three Adolescent Rock Bands". Master of Music. New Zealand School of Music (cit. on p. 222).

Tobias, E. S. (2013). "Composing, Songwriting, and Producing: Informing Popular Music Pedagogy". In: *Research Studies in Music Education* 35.2 (cit. on pp. 20, 22).

Tostig, A. (2025). *The Amateur Guitar Player's Chord-Strummin' Songbook*. Songbook. URL: https://songbook19.my-free.website/ (visited on 2025-06) (cit. on p. 128).

Trinity College London (2017). *Guitar Syllabus: Rock & Pop* (cit. on pp. 90, 92).

Tuck, M. (2025). *BFMV x Trivium – Guitar Talk & Playthroughs*. Youtube - Bullet For My Valentine. URL: https://youtu.be/TdWvrBp-5SQ?si=jLfaN3y8pw66dpz4&t=281 (visited on 2025-06) (cit. on p. 127).

Tuohy, D. R. et al. (2005). "A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature". In: *Proc. of the Int. Computer Music Conf. (ICMC)* (cit. on p. 61).

— (2006). "GA-based Music Arranging for Guitar". In: *Proc. of the IEEE Int. Conf. on Evolutionary Computation* (cit. on pp. 53, 61).

Turing, A. M. (1950). "Computing Machinery and Intelligence". In: *Mind* 49 (cit. on p. 27).

Ultimate Guitar Tablature Guide (2025). URL: https://www.ultimate-guitar.com/contribution/help/rubric#ii5 (visited on 2025-02) (cit. on p. 20).

Vaswani, A. et al. (2017). "Attention Is All You Need". In: *Proc. of the 31st Conf. on Neural Information Processing Systems (NIPS)*. URL: http://arxiv.org/abs/1706.03762 (visited on 2022-09) (cit. on pp. 32, 41).

Vazirani, V. V. (2003). *Approximation Algorithms*. Springer (cit. on p. 83).

Vélez Vásquez, M. A. et al. (2023). "Quantifying the Ease of Playing Song Chords on the Guitar". In: *Proc. of the 24th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 3, 4, 54, 63, 70, 71, 86, 95, 129).

Wang, B. et al. (2021). "Soloist: Generating Mixed-Initiative Tutorials from Existing Guitar Instructional Videos Through Audio Processing". In: *Proc. of the CHI Conf. on Human Factors in Computing Systems (CHI '21)* (cit. on pp. 4, 64).

Wang, X. et al. (2024). "Deep Reinforcement Learning: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.4 (cit. on p. 82).

Wang, Z. et al. (2021). "MuseBERT: Pre-Training of Music Representation for Music Understanding and Controllable Generation". In: *Proc. of the 22nd Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 2).

Wiggins, A. et al. (2019). "Guitar Tablature Estimation with a Convolutional Neural Network". In: *Proc. of the 20th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 136).

Windsor, M. (2021). "Using Machines to Define Musical Creativity". MUSSS3140: Dissertation. University of Leeds (cit. on p. 57).

WIPO (1979). *Berne Convention for the Protection of Literary and Artistic Works, Article 6bis*. World Intellectual Property Organization. URL: https://www.wipo.int/treaties/en/ip/berne/ (cit. on p. 218).

Wortman, K. A. et al. (2021). "CombinoChord: A Guitar Chord Generator App". In: *IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (cit. on pp. 5, 65, 93, 129, 137, 140, 141).

Wu, C.-J. et al. (2022). "Sustainable AI: Environmental Implications, Challenges and Opportunities". In: *Proc. of Machine Learning and Systems* (cit. on pp. 213–215).

Wu, S.-L. et al. (2023). "MuseMorphose: Full-Song and Fine-Grained Piano Music Style Transfer With One Transformer VAE". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 31 (cit. on pp. 58, 145, 147).

Xi, Q. et al. (2018). "Guitarset: A Dataset for Guitar Transcription". In: *Proc. of the 19th International Society for Music Information Retrieval Conf. (ISMIR)* (cit. on pp. 46, 55).

Yang, R. et al. (2025). "Music with Numbers: Jianpu Number-Based Notation in Cultural Heritage and Digital Humanities". In: *Proc. of the Music Encoding Conf. (MEC)* (cit. on p. 52).

Yazawa, K. et al. (2014). "Automatic Transcription of Guitar Tablature from Audio Signals in Accordance with Player's Proficiency". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (cit. on p. 137).

Yim, G. (2011). "Affordant Chord Transitions in Selected Guitar-Driven Popular Music". MA thesis. The Ohio State University (cit. on p. 51).

Yin, Z. et al. (2022). "Measuring When a Music Generation Algorithm Copies Too Much: The Originality Report, Cardinality Score, and Symbolic Fingerprinting by Geometric Hashing". In: *SN Computer Science* 3.5 (5) (cit. on p. 148).

Yuan, Y. et al. (2023). "Perceptual and Automated Estimates of Infringement in 40 Music Copyright Cases". In: *Transactions of the International Society for Music Information Retrieval* 6.1 (cit. on p. 219).

Yue, Y. et al. (2025). "Artificial Intelligence in Music Education: Exploring Applications, Benefits, and Challenges". In: *Proc. of the 14th Int. Conf. on Educational and Information Technology (ICEIT)* (cit. on p. 64).

Zang, Y. et al. (2024). "SynthTab: Leveraging Synthesized Data for Guitar Tablature Transcription". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (cit. on p. 50).

Zappa, F. (2017). *The Frank Zappa Guitar Book*. Hal Leonard Corporation (cit. on p. 111).

Zeng, Z. et al. (2024). "A Survey of Music Recommendation Systems". In: *Proc. of the 5th Int. Conf. on Computer Information and Big Data Applications (CIBDA '24)* (cit. on pp. 3, 64, 82).

Zhang, X. et al. (2008). "Chord Recognition Using Instrument Voicing Constraints". In: *Proc. of the 9th Int. Society for Music Information Retrieval Conf. (ISMIR)* (cit. on p. 46).

Zhou, Y. et al. (2022). *AnimeTAB: A New Guitar Tablature Dataset of Anime and Game Music*. Pre-published (cit. on p. 55).

Zhu, J.-Y. et al. (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *Proc. of the Int. Conf. on Computer Vision (ICCV)* (cit. on p. 61).

Zhuang, X. (2023). "Symbolic Guitar Music Style Transfer with Playable Guitar Tablatures". Master of Science Thesis in Embedded Systems. Delft University of Technology (cit. on p. 61).

Ziv, J. et al. (1978). "Compression of Individual Sequences via Variable-Rate Coding". In: *IEEE Transactions on Information Theory* 24.5 (cit. on p. 91).

A

Evaluation Samples of the Rhythm Guitar User Study

Below are shown the 5 samples that participants were asked to rate for the subjective evaluation of the models presented in chapter 8. Each sample had 3 versions: the reference, the transformer generation, and the rule-based generation. Please note that the original study featured videos with sound for each example to help the participants rate the excerpts.

A.1 Sample 1

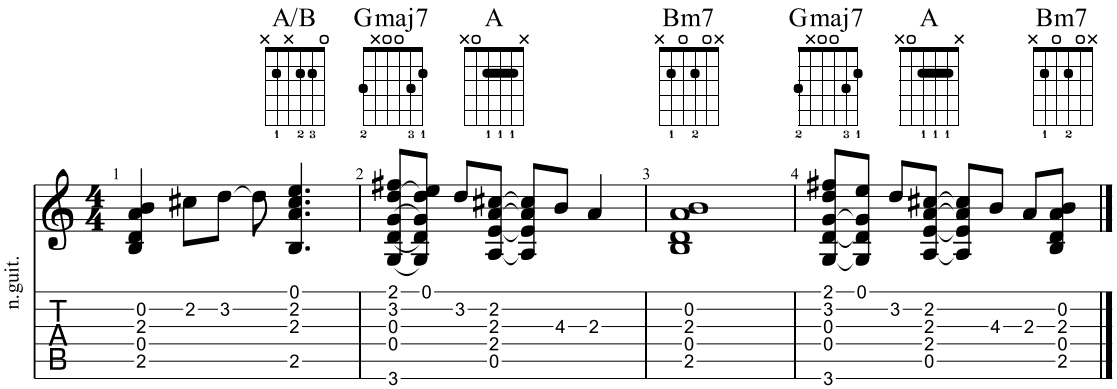


Figure A.1: Reference tablature of the first sample.

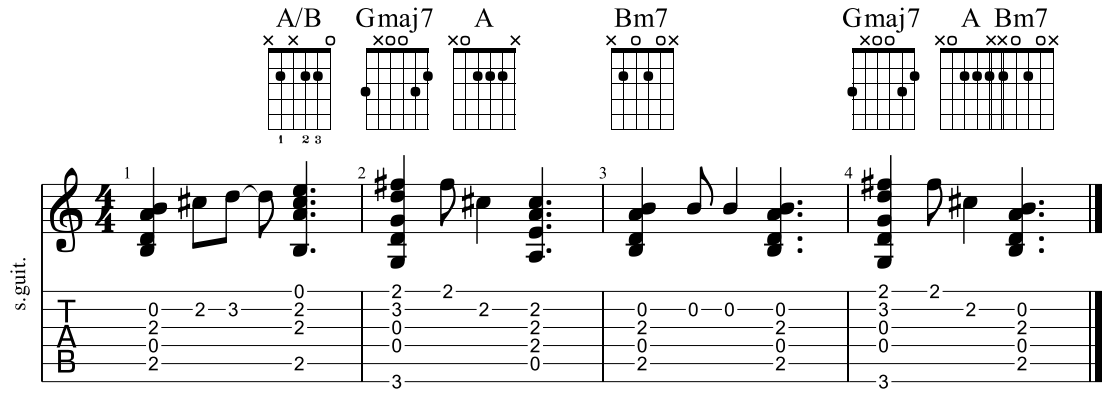


Figure A.2: Rule-based generated tablature of the first sample.

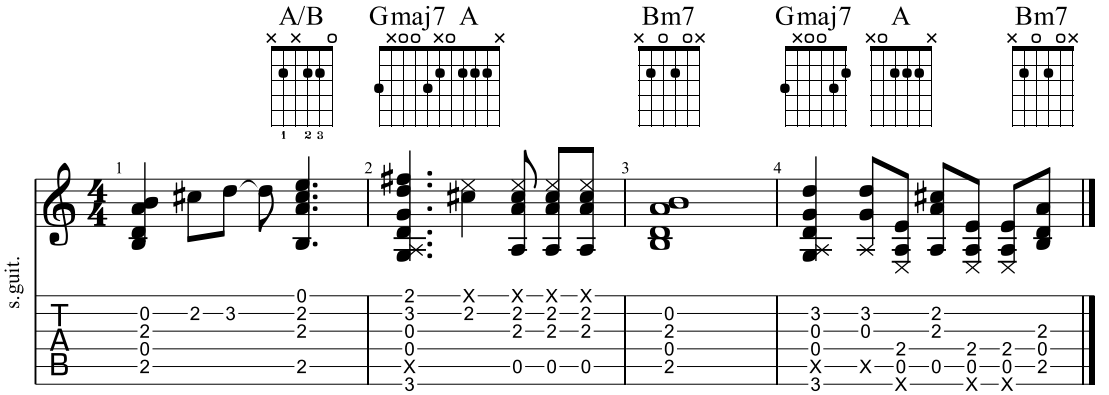


Figure A.3: Transformer generated tablature of the first sample.

A.2 Sample 2

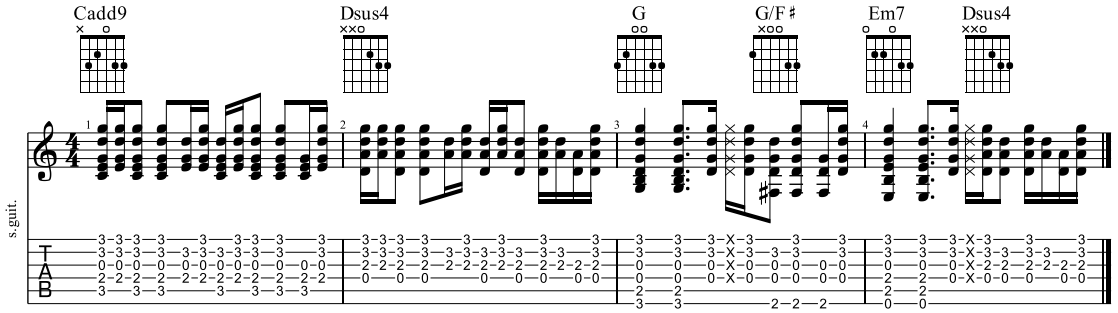


Figure A.4: Reference tablature of the second sample.

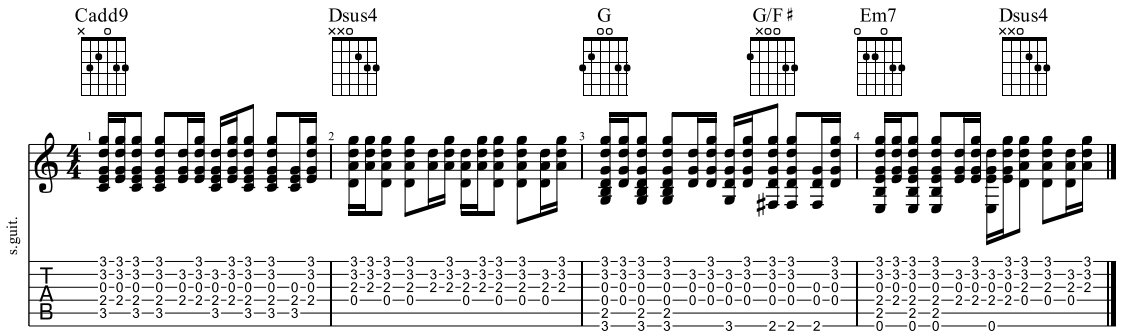


Figure A.5: Rule-based generated tablature of the second sample.

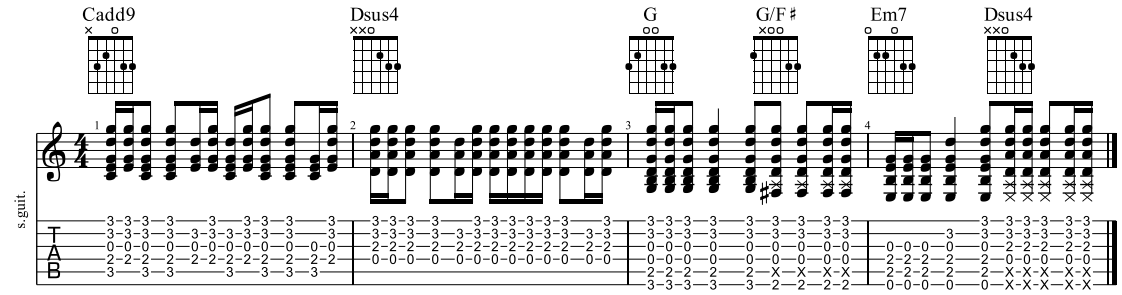


Figure A.6: Transformer generated tablature of the second sample.

A.3 Sample 3

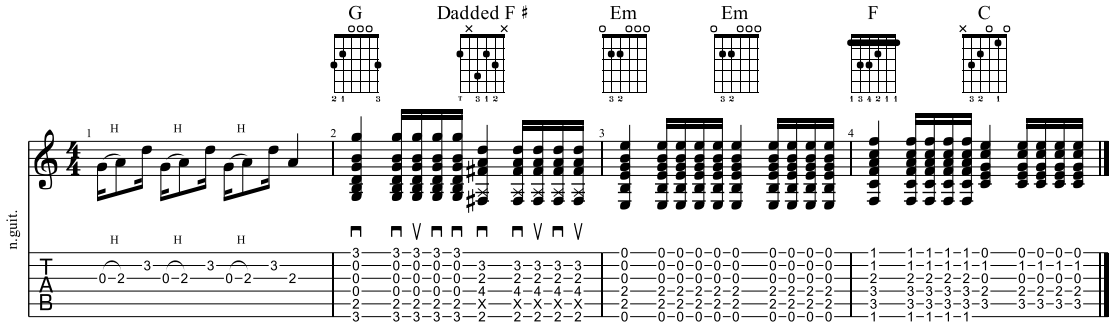


Figure A.7: Reference tablature of the third sample.

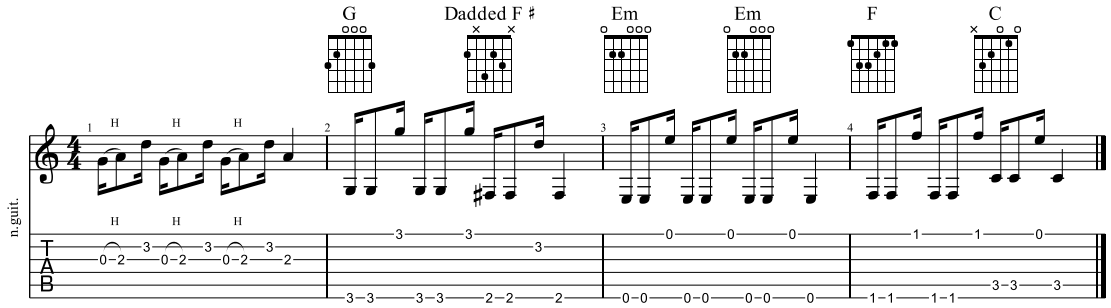


Figure A.8: Rule-based generated tablature of the third sample.

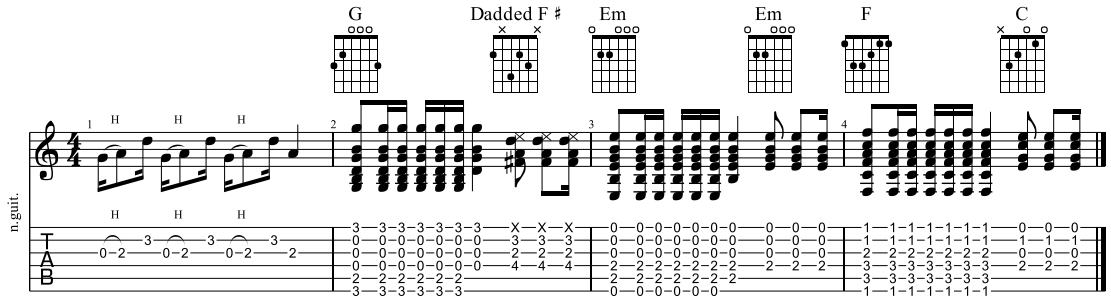


Figure A.9: Transformer generated tablature of the third sample.

A.4 Sample 4

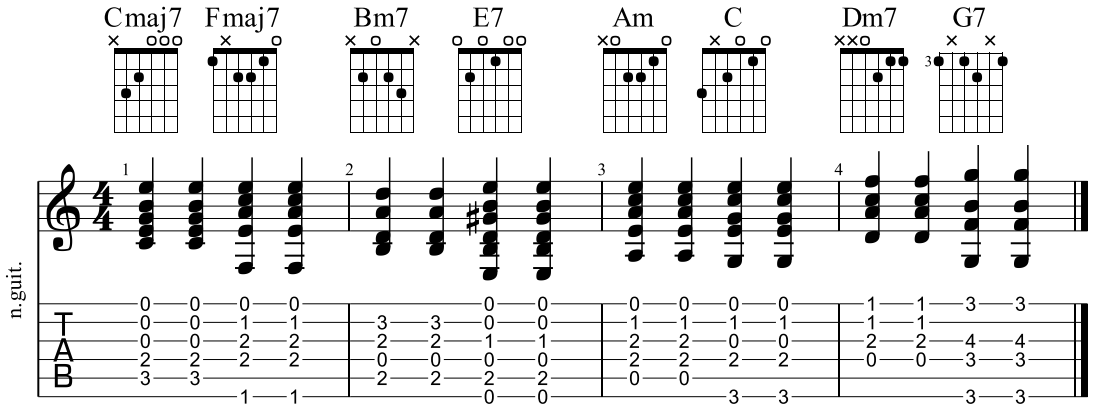


Figure A.10: Reference tablature of the fourth sample.

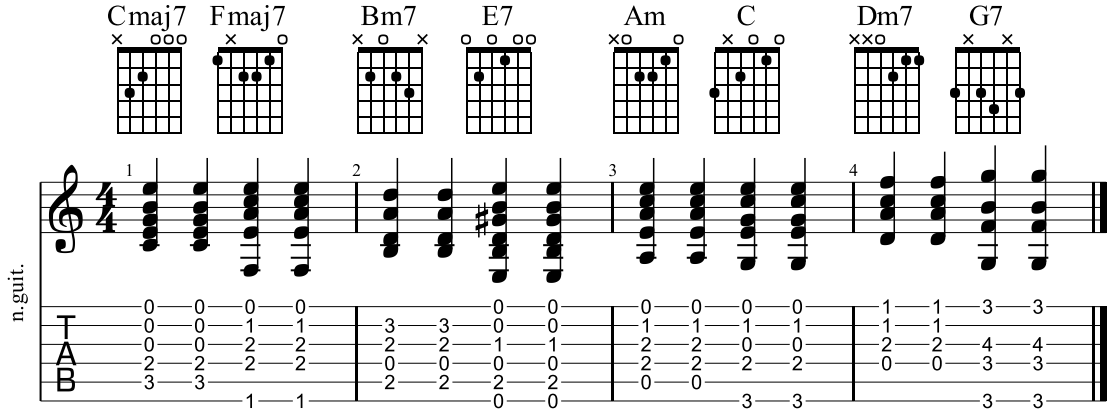


Figure A.11: Rule-based generated tablature of the fourth sample.

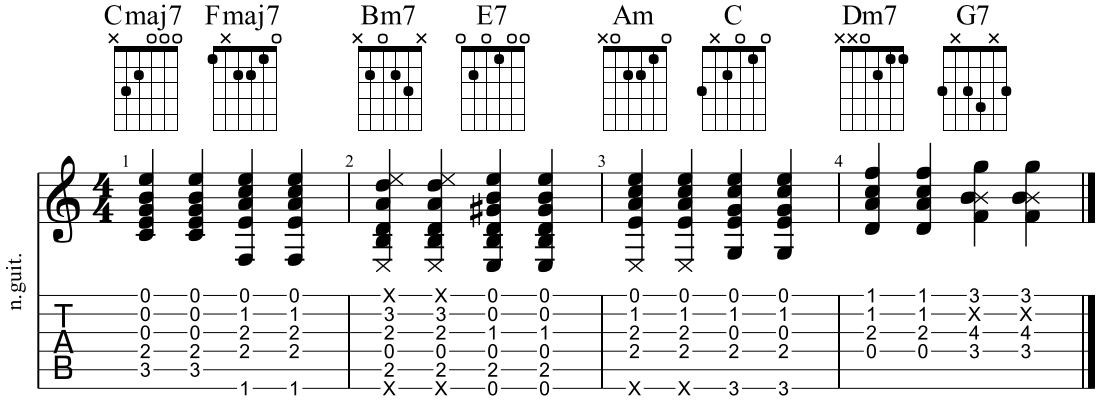


Figure A.12: Transformer generated tablature of the fourth sample.

A.5 Sample 5

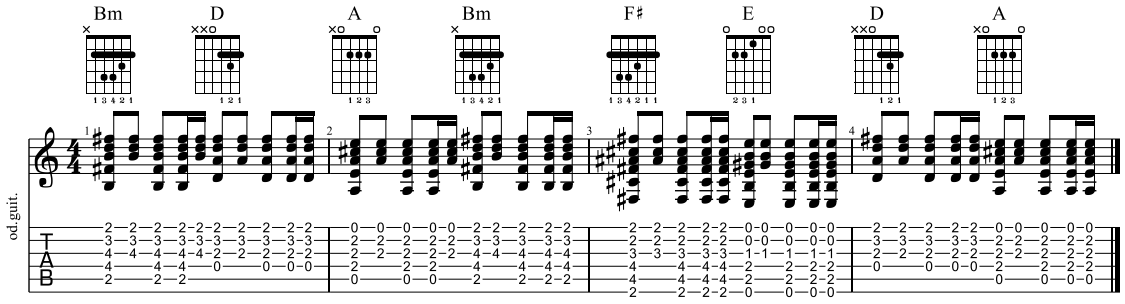


Figure A.13: Reference tablature of the fifth sample.

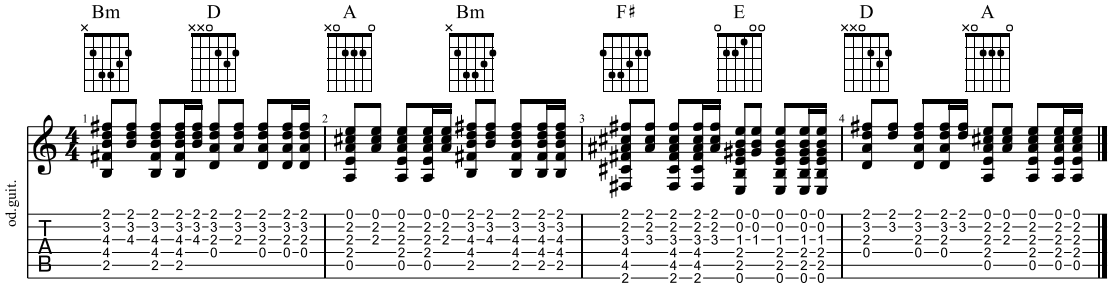


Figure A.14: Rule-based generated tablature of the fifth sample.

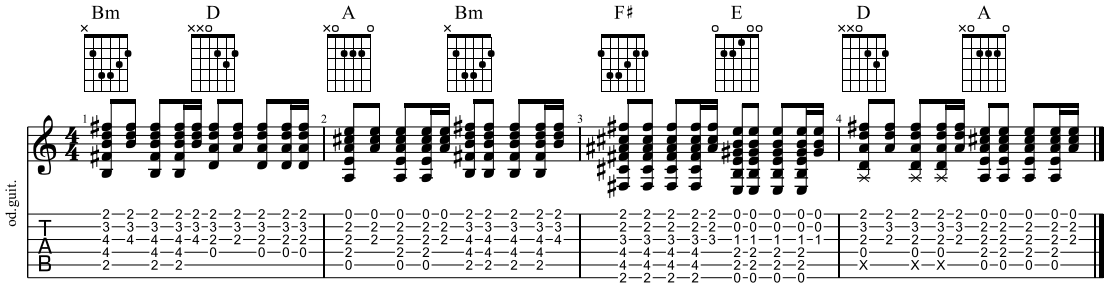


Figure A.15: Transformer generated tablature of the fifth sample.

B

Energy Consumption Considerations

In 2022, researchers from MetaAI published a detailed analysis of the environmental impact of their team developing AI models, through the carbon footprint and energy requirements (C.-J. Wu et al. 2022). In this paper, they thoroughly report on all the steps required to make an AI model usable to Meta's applications. Interestingly, they discuss how the energy consumptions is split between the development steps, from the data processing and initial experiments, to the training and later inference. Because their models are used "trillions of times every day", the inference has the most impact, but they also discuss how extremely small performance gains can strongly increase the energy consumption, something also identified in Douwes et al. (2021). While it is commendable that large companies like Meta transparently share information regarding the energy required for developing their AI models, Schwartz et al. (2020) argues that the number of operations required is a better measure of a model's consumption. Indeed, the energy consumption and the carbon emissions of an AI model are influenced by many factors like the hardware used for the computation, the usage ratio of said hardware, or the energy infrastructure (depending on the ratio of carbon-efficient energy sources). Likewise, the duration of training or the amount of parameters are not efficient measures of performance. However, the number of Floating-Point Operations (FPOs) – i.e. the amount of add or multiply operations – is agnostic to the hardware used for training/inference and accurately reflects the computational load of a model (Schwartz et al. 2020). Nonetheless, reporting the energy required to train AI models allows to compare it with known frames of reference and to remind that DL methods have an energy cost: data centres around the world consume more electricity than a country like France.[1] For this reason, in this section, we discuss the energy consumption of our work, the computational cost of the models developed, and reflect upon possible improvements,[2] to support the need of increased transparency in the MIR community (Holzapfel et al. 2024).

[1] https://www.imf.org/en/Blogs/Articles/2025/05/13/ai-needs-more-abundant-power-supplies-to-keep-driving-economic-growth, accessed in May 2025.

[2] Digital technologies have other impact, like water consumption required for building hardware and mining practices that can be morally questionable. They are not discussed here but we point the interested reader to https://www.ethicalconsumer.org/technology/conflict-minerals-tech-goods-home-appliances (accessed in July 2025), for instance.

Computational load and energy cost of this thesis' work. As discussed in C.-J. Wu et al. (2022), data preparation and early experiments can be a significant part of energy consumption. We started using the hybrid cluster of *Université de Lille* in the second half of this PhD work, and the jobs history can be used for estimating the energy consumption of experiments and models' training. We gathered the history of 456 past jobs, as well as the server node it was ran on. Using the cluster's documentation, we determined the CPU and GPU hardware (if any) on each node, and obtained their Thermal Design Power (TDP)[3] as suggested in Holzapfel et al. (2024). Based on the duration of each job and the TDP values, we obtain a total CPU energy consumption of 271 kWh and 276 kWh for GPUs, so an average per job of 0.6 kWh for each hardware type. Note that those values include many steps of the development of an AI model, like feature extraction and data processing, experiments for training, final training and inference on test sets. Considering all the steps included, those values are comparable to the ones discussed in Holzapfel et al. (2024). Nonetheless, they are fairly important, since the daily electricity used per person in France is around 6 kWh.[4] Besides, it is worth noting that those estimates are only for the second half of this PhD, so doubling the CPU energy consumption (we did not use GPUs in the first half of this PhD) might be closer to the actual value.[5] Assuming a higher bound of 1 MWh for the entirety of this PhD work and 45 g CO_2eq. per kWh[6] in France, we can estimate the corresponding carbon emissions as 45 kgCO_2eq.. This is equivalent to 0.03 of the emissions of a round-trip by plane between Paris and New York, or travelling 414 km by car.[7] Even if it is worth noting that it might be much more in countries where the electricity has a higher carbon intensity, the energy consumption is still high.

When it comes to number of FPOs, values are obtained manually for some architectures, or automatically using the `perf` monitoring tool. Results are summarised Table B.1. The FPOs of some pre-processing tasks are reported to show that even when a model is simple, the data preparation can be costly (even though it can usually be computed only once for any number of inference passes). Some values are also for a single prediction or forward pass, while the actual task might need several. For rhythm guitar continuation for instance (chapter 8), the transformer usually generates a few hundred tokens. Likewise, while estimating the difficulty of learning a new song only requires 40 FPOs, finding the best song requires processing the entire catalogue of over 1000 possibilities (chapter 4). Besides, the number of FPOs does not always directly correlate with the duration of execution (Schwartz et al. 2020), which can be an important aspect in production settings.

[3] All values taken from https://www.techpowerup.com/, accessed in May 2025.

[4] https://www.data.gouv.fr/fr/reuses/consommation-par-habitant-et-par-ville-delectricite-en-france/, accessed in May 2025.

[5] This might actually be a high estimate, since multiple jobs can run at once on the same node of the cluster, and not all jobs used the nodes at full capacity.

[6] Average French carbon intensity between 2010 and 2024 https://analysesetdonnees.rte-france.com/bilan-electrique-2024/emissions#Analysehistorique, accessed in May 2025.

[7] https://impactco2.fr/outils/comparateur, accessed in May 2025

Table B.1: Summary of the number of Floating-Point Operations for the tasks discussed in this thesis.

Model/Program	FPOs
Parsing a `.gpif` file	$\sim 150\,000$
Bends - decision tree prediction	29
Bends - multi-layer perceptron forward pass	9048
Bends - extracting features from one tablature	$\sim 400\,000$
Chord Diagrams - extracting chord diagram information from a tab	$\sim 1 \times 10^6$
Chord Diagrams - multi-Layer perceptron forward pass	50 400
Rhythm Guitar Continuation - training data pre-processing	254×10^6
Rhythm Guitar Continuation - transformer forward pass (one token)	$\sim 342 \times 10^3$
Rhythm Guitar Continuation - baseline forward pass (all output)	590×10^3
Bass Tablature Generation - forward pass (one token)	$\sim 24 \times 10^9$
Difficulty estimation of learning a new song	40
Tablature Difficulty - features extraction from one tablature	$\sim 107 \times 10^6$

Finally, the largest model was used for bass generation and requires several orders of magnitude more FPOs for a single forward pass, around the highest values measured in (Schwartz et al. 2020). This model would thus benefit greatly from performance improvements techniques such as the ones discussed hereafter.

Perspectives for performance improvements. While honing research practices to reduce the amount of costly preliminary experiments is still one of the best options, other alternatives can be studied to reduce the computational cost of AI models. Some of the models presented in this thesis could be considered "frugal" (Bas 2020) because of the little amount of operations they require, or because models are relatively small when compared to standard NLP models. However, there are ways to reduce further the computational load of AI models. Such methods were not applied in this thesis, but are discussed here as possible future improvements. Because the models presented are ultimately designed to be used by many, it is especially important to reduce the cost of inference (C.-J. Wu et al. 2022). For Neural Networks, their size in memory and computation time can often be reduced through *pruning*, a practice consisting in removing weights and parameters from a model if they contribute little to the computation (Blalock et al. 2020). Pruning has been reported to be able to halve the size of a network in memory as well as the computation time, without much loss of performance (Blalock et al. 2020). Another common technique for reducing computational load is *quantisation*. Libraries like `Pytorch` often use 32 bits for storing values (default value also used in this work), but there are times where such a resolution is not needed. Reducing the precision of a DL model can reduce

its size in memory and its computation time by four times or more (Gholami et al. 2021). Applying pruning or quantisation to the bass tablature generation model would also be an improvement worth pursuing, as the transformer used is large and require several dozen seconds to generate a tablature, on GPU. *Network distillation* could also be used (Sanh et al. 2020), which consists in training a smaller model from a pre-trained larger one. Distillation has been reported to speed generation up to 18 times in some cases (Novack et al. 2025). It might also be worth reflecting on the amount of data used for training models because, while music is complex to replicate by digital models, not all songs might be relevant for the task at hand. For instance, a model focusing on singable pop melodies might not require rock, metal, classical, or jazz songs that could be filtered out of a large dataset for faster training.

C

Legal Considerations in Generative Music AI

The core of this PhD work has been to develop AI methods to assist guitarists in their learning and composition practices. However, using AI models in art fields, especially generative ones, has spurred debates on the way copyrighted materials can be used for training (commercial) AI models. The debates mainly focus on two aspects: the way copyrighted materials might be used without the creators' consent;[1] and the tendency of generative AI systems to produce outputs suspiciously similar to said copyrighted materials, often compared to plagiarism (Cooper et al. 2024). After briefly presenting how copyright is defined (in Europe) and how it relates to AI models, those two aspects are discussed in relation to this thesis' contributions.

On copyright legislation. Let us start with the definition of copyright from the World Intellectual Property Organisation (WIPO):[2]

Definition (Copyright) *Copyright (or author's right) is a legal term used to describe the rights that creators have over their literary and artistic works. Works covered by copyright range from books, music, paintings, sculpture, and films, to computer programs, databases, advertisements, maps, and technical drawings.* 📖

With that definition, we can deduce that all data used in this thesis is copyrighted. Besides, copyright in the European Union (EU) holds for 70 years after the death of the creator.[3] The music contained in the mySongBook database, the DadaGP dataset, the songs used by the GSC start-up or the songs of our Tablature Difficulty dataset are thus still copyrighted, apart from a few exceptions of classical or jazz music. In addition to

[1] It is based on this suspicion that major music record companies are suing Suno and Udio, two AI music generation services (RIAA 2024)

[2] https://www.wipo.int/en/web/copyright, accessed in May 2025.

[3] https://europa.eu/youreurope/business/running-business/intellectual-property/copyright/index_en.htm, accessed in May 2025.

the compositions being copyrighted by the original composers, the transcriptions themselves might also be copyrighted by the transcribers, if some creativity is deemed to have happened in the transcription process (Stokes 2019, p. 34).

Copyright is commonly split between *moral rights* and *economic rights*. In WIPO (1979), moral rights are defined as:

Definition (Moral Rights - Berne Convention, Article 6bis) *Independently of the author's economic rights, and even after the transfer of the said rights, the author shall have the right to claim authorship of the work and to object to any distortion, mutilation or other modification of, or other derogatory action in relation to, the said work, which would be prejudicial to his honor or reputation.* 📖

Moral rights allow authors to object to any alteration made to their work, depending on how it is implemented in each country (the EU does not impose rules on those rights). Economic rights are related to the ability to earn money from one's work:[4]

Definition (Economic Rights - EU Copyright Rules) *Rights that enable rightholders to control the use of their works and other protected material and be remunerated for their use. They normally take the form of exclusive rights, notably to authorise or prohibit the making and distribution of copies as well as communication to the public. Economic rights and their terms of protection are harmonised at EU level.* 📖

With such definitions, rightholders have a say in most usage people can make of their creations, but a few exceptions have been made for the modern practice of "text and data mining". In the Directive (EU) 2019/790 on copyright in the Digital Single Market (DSM Directive), the EU allowed public research organisations to use any copyrighted data indefinitely, and access cannot be prevented by rightholders (Dornis et al. 2025).

Use of copyrighted data without consent. Moral rights could be interpreted as a way for creators to refuse that their creations are used for training AI models. Economic rights would also be involved if the resulting model is used in a commercial manner. However, the EU DSM directive explicitly discards those possibilities for Text and Data Mining (TDM) applications. Putting aside the moral consideration of using artists' work for training AI models even if they might object against it, it is therefore legal in most cases. Nevertheless, while some researchers in law believe that the TDM exception also applies to generative AI models (C. J. Craig 2024), others consider this matter unresolved (Dornis et al. 2025). The question of whether the TDM exception can apply to music generative AI might be answered in the near future with justice decisions on GEMA (German "Society for musical performing and mechanical reproduction rights") suing SUNO and OpenAI for copyright infringement.[5]

[4] https://digital-strategy.ec.europa.eu/en/policies/copyright, accessed in May 2025.
[5] https://www.gema.de/en/news/ai-and-music/ai-lawsuit, accessed in May 2025.

In this thesis, studying guitar playing techniques (chapter 6), the most used guitar chord diagrams (chapter 7), or why guitar songs are difficult (chapters 4 and 5) can safely be considered data mining. However, generating bass tablatures (chapter 9) or possible continuation of rhythm guitar tablatures (chapter 8) are clear generative tasks that might not be eligible to the TDM exception and would thus require explicit consent of the rightholders. The DadaGP dataset was used for both those tasks and, as identified in Morreale et al. (2023) and acknowledged by Pedro Sarmento in his PhD thesis (Sarmento 2024, pp. 168–172), tablatures were scraped from public online data without their owner's consent, using them for training generative models is therefore permitted only if the TDM exception holds. Apart from the legal aspects, there are also moral considerations that the MIR community is discussing increasingly, and some researchers are advocating for a change in the way datasets are created and used in the community (W. Chen et al. 2019; Morreale et al. 2023), by refraining from scraping online data and putting artists back in the loop. But the "cultural relativity" of copyright should not be disregarded, and legal and "moral" aspects might not always align (Holzapfel et al. 2018). We agree that the MIR community would benefit from better communicating with artists and assembling datasets where consent has been obtained for as much use-cases as possible. However, we also agree with C. Craig (2022) that training AI models on art works should not be considered possible copyright infringement (regardless of TDM exceptions). Indeed, such AI models only use their training data as a source of information and not "as a work of authorship". Nevertheless, it can happen that a generative AI model replicates part of its training set almost identically, which then raise questions of possible plagiarism.

Risks of Plagiarism. Plagiarism is one type of copyright infringement that has been at heart of many complex music-related lawsuits (Yuan et al. 2023), accusations of alleged plagiarism – or uncredited copying – regularly hitting famous artists, like with Tempo Music Investments v. Miley Cyrus for her song *Flowers* (Snapes 2024). Generative AI models, like human composers to some extent (Shiga 2016), are prone to *memorisation* behaviours (Dornis et al. 2025) where parts of the training data are reproduced identically. Because of this memorisation risk, an AI model could possibly generate plagiarising content. Indeed, even if the training data was acquired and used legally, reproducing identically excerpts of an existing song is akin to plagiarism and thus considered copyright infringement (Cooper et al. 2024), much like how *sampling* can often infringe copyright (Challis 2009). While there is research that aims at limiting such memorisation (Satvaty et al. 2025) or detecting when data replication occurs (Batlle-Roca et al. 2024), we have not conducted such analyses in our work. Replicating the training data is not an issue in specific tasks like the suggestion of bends (chapter 6) or of chord diagrams (chapter 7) but becomes more prevalent when generating full bass tablatures (chapter 9) or even rhythm guitar continuation (chapter 8). The latter task might prove less problematic because of the way strumming and fretting data are separated: since the model only generates picking patterns, memorisation might only occur on that dimension, and picking patterns

are not copyrightable, like chord progressions. However, the bass tablature generation model might be prone to generate bass lines identical or close to existing ones. Those situations are not easy to solve because while we believe that AI models can be creative, we also support C. J. Craig et al. (2021) in saying that an AI cannot claim authorship of its generation. In that case, it is ultimately up to either the user who makes something out of the output, or the model's creators, to detect possible plagiarism. Solutions to assist in the detection of possible plagiarism are still being studied, and might help compare the generation with existing content (Batlle-Roca et al. 2024; de Prisco et al. 2016; Savage et al. 2018). However, the "recognisability" of a music excerpt is highly subjective and so-called "objective" measures might not adapt well to different cultural contexts or music traditions (Holzapfel et al. 2018). For this reason, we believe it is important to develop systems that can help humans quickly compare the generated content to existing songs, but that decisions of possible infringement should never be made automatically to avoid hindering creativity.

D

Guitar, AI, and Artists

With guitar players in mind, a major goal of this thesis has been to provide tools to assist them in specific tasks of their creative process. However, the use of (generative) AI models in music creativity is not always seen positively, as illustrated by a letter signed by 200 artists (Artist Rights Alliance 2024):

> “We call on all AI developers, technology companies, platforms and digital music services to pledge that they will not develop or deploy AI music-generation technology, content or tools that undermine or replace the human artistry of songwriters and artists or deny us fair compensation for our work.”

While the letter argues that “AI has enormous potential to advance human creativity”, the fear or artists being replaced by it (and not being paid in the process) is a core preoccupation.

Impact on guitarist composers Regarding this thesis, we believe little risk of replacing human artists exist. Indeed, what we studied aims at easing some parts of the creative process when using tablatures through controllable tools, rather than automate them entirely. A part of our work focuses on specific aspects of tablatures like suggesting bends (chapter 6) or chord diagrams (chapter 7). Bends are often considered characteristic of an artist’s style (Herbst et al. 2024) and might be considered the results of creative choices (Rhodes 1961). Likewise, choosing chord voicings and corresponding positions on the fretboard are creative decisions (Dickinson 2019; Koozin 2011). However, bends and chord diagrams are not enough to create full songs, and therefore suggesting them automatically cannot “replace human artistry”. Other parts of our work study generative tasks that could replace artistry to some extent, like rhythm guitar tablature continuation (chapter 8) or bass tablature accompaniment generation (chapter 9). The generation of bass tablatures might have the most detrimental effect on human composers, because it could

actually replace bass players in the compositional process. Indeed, it could happen that a guitarist composer uses the system to generate the bass tablature, rather than collaborating with a bass player who might compose their own part. However, WPM is overall a genre that calls for authenticity and is based on live performances (Kelly 2007; McKinna 2014), so bass players would still be expected to perform the generated tablatures live if it ever reached that state. Besides, performers and songwriters/composers are not always the same persons in WPM (Herbst et al. 2025), and composition is not always tackled by all members of a band equally (Thorpe 2007). For this reason, we consider our bass generation system as a possible help to guitarist songwriters who do not play bass, or as a way to generate ideas in collaborative composition phases (Thorpe 2007). Similar comments apply to the rhythm guitar continuation model presented in chapter 8, but it has even less risk to replace artists because the generation requires a prompt as well as a chord progression. Because an artist will only be able to use the model after choosing those elements, some kind of human creativity will always be required. Finally, like for the models that suggest bends or chord diagrams, the generation models we presented do not generate full WPM songs as the resulting score would miss a melody and a drum/percussion track. Overall, the tools proposed focus on specific tasks on the composition process and might be most useful to beginner composers by facilitating some creative decisions that can require time and dedicated skills. In doing so, they might even allow more guitarists to be composers, rather than replacing any of them.

Impact on Guitar Teachers In this thesis, another part of the contributions focuses on assisting guitar learning by recommending new songs (chapter 4) or automatically analysing the difficulty of a tablature (chapter 5). While those tasks are not creative, a risk of replacing guitar teachers might exist. We believe, however, that the risk is minimal because, as L. Green 2002 puts it: "[p]opular musicians acquire some or all of their skills and knowledge informally, [...], and with little help from trained instrumental teachers." In that situation, tools that can help musicians during informal learning are precious (Mesbur 2006; Owens 2017; Ruismäki et al. 2012). Besides, studies suggest that popular musicians tend to learn both informally and with music teachers (L. Green 2002; Hess 2020; Mesbur 2006), so facilitating autonomous informal learning has the potential to increase learners' motivation, without replacing formal teaching and human teachers. L. Green (2002) for instance, observes that popular musicians will often learn in groups or be directed by peers in their practice. Mesbur (2006) discusses how 6 adolescent girls learning WPM were all trained by "musical mentors", often through private lessons. Hess (2020) also conducted interviews with musicians and gathered feedback supporting a desire for more formal education in WPM. Overall, music learning research supports the idea that informal autonomous learning in WPM is not exclusive of possibly formal learning with music teachers. Finally, a music teacher contributes to much more than recommending songs and analysing their difficulty, as they provide learners with technical explanations, real-time feedback on their performance, assistance in learning gestures, etc.