JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Emmanouil Karystinaios

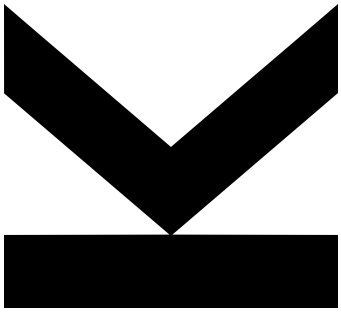
Submitted at
Institute of Computational Perception

Supervisor and
First Evaluator
Gerhard Widmer

Second Evaluator
Markus Neuwirth

October 2024

Symbolic Music Analysis with Graph Neural Networks



Doctoral Thesis

to obtain the academic degree of

Doktor der technischen Wissenschaften

in the Doctoral Program

Technische Wissenschaften

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Emmanouil Karystinaios: *Symbolic Music Analysis with Graph Neural Networks,* , Doctoral Thesis, © March 2020 - October 2024

SUPERVISORS:
Gerhard Widmer
EVALUATORS:
Gerhard Widmer
Markus Neuwirth

LOCATION:
Linz & Vienna

TIME FRAME:
March 2020 - October 2024

ABSTRACT

In recent years, the intersection of artificial intelligence and music informatics has gained significant traction. However, music analysis on symbolic music has not been explored to its full extent. This work investigates the application of Graph Neural Networks (GNNs) to diverse music analysis tasks on digitized classical music scores.

Music analysis, as a scholarly field and as a set of techniques, is crucial for comprehending and appreciating music. It systematically examines elements such as harmony, melody, rhythm, form, and instrumentation, revealing the interplay of compositional techniques and structural elements/patterns. Analyzing music highlights foundational elements essential for creating new compositions.

This thesis examines the effectiveness of Graph Neural Networks (GNNs) on music analysis tasks such as Cadence Detection, Roman Numeral Analysis, Composer Classification, and Voice Separation, focusing on symbolic representations (i.e., musical scores given in some machine-readable encoding). We argue that a graph structure is a more natural representation for modeling a musical score than the feature- or token-based representations that have been used so far.

Our study begins by detailing the intricacies of symbolic music representations and the limitations of existing methods. We explore graph representations for music, enabling graph learning models. The graph emerges as a natural representation that encompasses the mixed hierarchical and sequential nature of a musical score. We propose a new graph model for the score where vertices represent notes and edges capture the relations between them.

In a first step, we experimentally compare graph representations to other modeling approaches such as piano rolls, note arrays, or custom descriptors on a number of different music classification tasks. Next, we design GNN-based machine learning models specifically for music analysis, capable of addressing the unique challenges posed by music data and the targeted application. As a result, we obtain graph-based models that demonstrate improved performance on benchmark datasets for diverse analysis tasks. Furthermore, we develop a new generic graph convolution block based on perception-inspired principles that further improve performance on music understanding tasks. We present a framework for deriving and visualizing explanations of the decisions made by our music-related GNNs. Finally, we develop and publish a dedicated library for symbolic music graph processing in order to reinforce the impact of this work in the research community.

Current AI trends suggest that Large Language Models (LLMs) and transformers have the potential to solve a wide range of tasks in various fields. However, our findings indicate that graphs can be more efficient for music analysis tasks. Thus, the ultimate goal of this thesis is to establish graph-based modeling as a standard approach to computational symbolic music analysis.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor, Professor Gerhard Widmer, for his support and patience throughout the past 4 years. His guidance has helped me develop and realize many ideas while keeping a balance between PhD and normal life.

I would also like to thank my second evaluator Markus Neuwirth for taking the time and effort to review this thesis.

My PhD path has been unique in every way. I recognize that the accomplishments during my journey would not have been there without all the people that surrounded me and supported me. From my colleagues, I would like especially thank Francesco Foscarin not only for copiloting most of the interesting work we have produced but also for motivating and sometimes mentoring me. In addition, I would like to thank Hamid Eghbal-Zadeh for introducing me to Graph Neural Networks and mentoring me through the initial parts of my graph-based research. I would also like to thank my colleagues Luis Carvalho, Lukas Martak, Charles Brazier, and Alessandro Melchiorre for always being there, in particular, during the peculiar Covid times.

Finally, I would like to thank my family who has always supported my decisions and has ever been by my side. But, most of all, I would like to express my gratitude towards my wife, Zitania Badat, because without her none of this would be possible. Her support has been the guiding beacon that helped me to always keep track of my priorities and my focus.

The research reported in this thesis has been carried out at the Institute of Computational Perception (Johannes Kepler University Linz, Austria) and has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreements No. 670035 (project "Con Espressione") and No. 101019375 ("Whither Music?").

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Part I

INTRODUCTION AND BACKGROUND

1 INTRODUCTION

Music is a universally comprehensible language with a remarkable ability to transcend cultural and linguistic boundaries. It encompasses a vast spectrum of styles and genres, from the simple to the intricately complex. Within this diversity lies its power to convey a wide range of sentiments.

Babbit [1] tells us that music can be understood from three different perspectives: the music written by the composer, the music played by the musician, and the music perceived by the listener. Music analysis is a pivotal discipline in unraveling the complexities of musical compositions. Analysts delve into the fundamental components of rhythm, harmony, melody, and beyond, pinpointing their individual significance and collective impact. The analysis process is a step towards understanding compositions, uncovering hidden nuances, and deciphering the intricate interplay between various musical elements.

1.1 THE IMPORTANCE OF MUSIC ANALYSIS

There are numerous theories and methods for music analysis, ranging from holistic approaches to those focusing on specific elements like form, melody, or harmony. For example, structural analysis entails the identification of formal frameworks and sectional divisions within a piece. But even when focusing on particular elements there are many approaches, for example, harmonic analysis can be viewed through the lens of Roman numeral analysis, figured bass or pure chord progression identification.

Other examples include melodic analysis, which examines the structure and development of the melody, and rhythmic analysis, which focuses on meter, tempo, and rhythmic patterns. Textural analysis looks at the spatial distribution of musical elements in polyphonic or homophonic textures, and expressive analysis explores dynamic nuances, articulation, phrasing, and gestures that add depth to a performance.

Historical and contextual analysis situates music within its socio-cultural context, considering the composer's biography, influences, and prevailing aesthetic trends. Holistic frameworks like Schenkerian analysis or the Generative Theory of Tonal Music (GTTM) reduce a piece to its hierarchical elements in terms of rhythm, melody, and harmony.

Music analysis deepens the understanding of music, allowing listeners and musicians to discern a piece's structure and artistic intent. It fosters critical listening and analytical skills, enhancing appreciation and engagement with various musical genres and styles.

For performers, analysis informs interpretation, guiding decisions on tempo, dynamics, phrasing, and other expressive nuances, enriching performances with

depth and authenticity. Composers gain inspiration and learning from analyzing past works, expanding their musical vocabulary, and refining their craft.

Music analysis also contributes to ethnomusicological research, providing insights into the historical, cultural, and stylistic dimensions of music. Through comparative analysis, musicologists can trace the evolution of musical forms, techniques, and genres across different periods and cultures. In short, music analysis is a fundamental tool for understanding music.

1.2 DEEP LEARNING FOR MUSIC ANALYSIS

Musical analysis has been a quintessential part of MIR including tasks such as harmonic analysis, composer classification, motif detection, textural analysis, fugal analysis, cadence detection, voice separation, score reduction, and many others. Furthermore, the datasets for such music analysis tasks have been continuously growing.

Deep learning approaches have been introduced to target many of the aforementioned tasks, focusing on learning from data. However, such data are still scarce and too complex. Therefore, many approaches introduce inductive biases that assist models to better model and understand music.

The inductive biases can be introduced on different levels such as the model-level or at the representation level. The former concerns ways to inform models about certain musical expectations and the latter focuses on how the music is represented before it is fed to a model.

1.3 THE INHERENT ISSUES OF MUSIC REPRESENTATION

Music in its digital form can be represented in many ways. There are two main categories of musical representations, the audio-based representations and the symbolic music representations. For audio, music is typically represented as a waveform or as a transformation of it, such as a spectrogram. Symbolic means that music is associated with symbols corresponding mostly to its written form, i.e. a musical score.

Western classical sheet music is most often represented with staff notation. However, representing this symbolic music form computationally and training algorithms on that representation is not straightforward. Research has been inspired by modeling scores similarly to audio spectrograms or even as language, and more recently as graphs. Each representation comes with its own advantages and drawbacks for designing and training deep learning models.

Pianorolls were originally perforated paper rolls used to operate a player piano, as a physical form of music notation [2]. However pianorolls, in their current digital form, can be viewed as a spectrogram representation of symbolic music containing only the fundamental frequencies of the written notes. This point of view can be advantageous when re-adapting powerful audio-based models. However, pianorolls are sparse data and could also omit important information that inherently appears in a score but not in a pianoroll.

Language-inspired representations often treat symbolic music as language which means they tokenize each symbol. Having a tokenized sequence is a huge advantage because music can then be treated with powerful transformer models that have been dominating the AI-field in the past few years. However, representing music as a series of tokens removes the hierarchical nature of music which is present in the score, particularly for polyphonic music.

Finally, graph-based representation has been a new and relatively unexplored alternative that can in principle capture sequential and hierarchical relations between notes. It is more intuitive from a musical point of view and graph neural networks that can process graph data have been increasingly popular since 2017 [3]. However, the limited application of GNNs on data of vastly different structures creates a still unexplored field with a lot of uncertainty about the effectiveness of GNNs on music analysis.

1.4 MOTIVATION AND VISION

Despite advancements in Music Information Retrieval (MIR) within the symbolic domain and the diversity of algorithms and representations, there remain significant concerns regarding the handling of symbolic music in research. My academic background in musicology, with a particular emphasis on music analysis and contemporary composition, has provided me with extensive exposure to various music analysis theories and representation ambiguities. These experiences have led me to form strong opinions on the appropriate handling of symbolic music representations within deep learning frameworks.

My research pursuits have directed my focus towards graph-based representations. I posit that graphs may offer a more suitable representation for symbolic music, as they can effectively capture the dual organizational structure of polyphonic music—namely, its hierarchical and sequential elements. Viewing music as a graph can facilitate a more intuitive understanding and manipulation of musical structures. Therefore, my objective is to demonstrate the viability of a graph-based approach in addressing or enhancing computational music analysis in symbolic classical music. Ultimately, my goal is to contribute to the standardization of graph-based symbolic music processing.

1.5 ORGANIZATION

This is a *cumulative dissertation*: the main contributions are presented in the form of a set of published papers, reproduced here in only slightly edited form. The thesis is organized it two parts, the first part being introductory and the second part containing part of my published work. In Chapter 2 and Chapter 3, I provide background information on the content of this thesis on two important topics. Chapters 5-11 contain my scientific contributions, each representing a peer-reviewed publication. These papers have been slightly modified for formatting consistency and to correct any errors found in the original versions. Each chapter begins with

the full title, along with the names of co-authors and the conference where the publication appeared.

1.6 OUTLINE

This thesis is divided into two parts. The first part is introductory and discusses introductory notions about symbolic music representations and graph-based deep learning. The second part contains a subset of my published work with every chapter containing a publication. The last chapter presents a reflection on the work done, discusses some potential challenges of the field, and outlines some work for the future to come.

1.6.1 First Part

My work is based on the premise that musical scores can be seen as graphs. The graph emerges as a natural representation that encompasses the mixed hierarchical and sequential nature of a musical score. Elements in the score such as notes, are structured in time but also more than one note can occur at the same time. I propose a new graph model for the score where vertices represent notes and edges capture the relations between them.

Each node in the graph corresponds to one and only one note in the musical score. The graph's edges include multiple temporal relations between nodes in order to capture the temporal evolution of the musical piece. Given such a score graph, I can apply Graph Convolutional Networks to capture intra-dependencies between notes. Graph Convolutional Neural Networks use a principle called message passing in order to update the information of a node in the graph based on the information of its neighbors [4].

Chapter 2: Before diving into the individual contributions of the thesis, I present an overview of music representations, which are crucial for Music Information Retrieval (MIR) and music analysis. These representations, including symbolic formats like piano rolls, tokenized sequences, and graphs, as well as audio representations, significantly impact how musical information is encoded and processed. Visualizing and analyzing the advantages and caveats of each representation is important for the development of better more interpretable models for music analysis.

Chapter 3: Subsequently, I discuss the inner workings of Graph Neural Networks (GNNs). These networks are used to train algorithms on data that can be represented as graphs. I describe some influential and fundamental work in the field of graph deep learning. This work is the foundation of models developed during the thesis and presented in the following chapters. Understanding the theoretical underpinnings of GNNs is crucial for leveraging their potential in music analysis.

1.6.2 Second Part

The second part of the thesis aims to give a clear narrative of this PhD journey, in the form of a series of chapters each of which corresponds to one specific publication.

First, I have a more general look at the comparison of different symbolic representations and corresponding popular model architectures; then I introduce GNN models to tackle analysis tasks such as cadence detection, Roman numeral analysis, and voice separation on symbolic classical music. Subsequently, I investigate how I can visually represent graph-based explanations for music. Then I introduce a perception-inspired graph convolutional block. Next, I present a dedicated graph library for symbolic music processing. Finally, I discuss a wholistic graph-based model for performing music analysis.

Chapter 4: The first chapter of the second part is dedicated to a collaborative paper [7] that systematically evaluates various symbolic music representations, including matrix (piano roll), sequence (tokenized), and graph, for piece-level classification tasks such as composer classification, performer classification, and difficulty assessment. The study explores the capabilities of these representations, with neural architectures such as Convolutional Neural Networks (CNNs) for matrix, Transformers for sequence, and Graph Neural Networks (GNNs) for graph representations. The evaluation is conducted on datasets containing symbolic scores and performances, revealing that graph representations, particularly when incorporating advanced features like hierarchical musical structures, show promising performance and efficiency in training. The findings highlight the importance of choosing appropriate representations based on the specific characteristics of the musical data and the classification tasks, suggesting that graph-based approaches are a valuable direction for future research in Music Information Retrieval (MIR). My contribution for this work was focused on graph representations by providing insights for designing and training graph-based representations and models.

Chapter 5: I present a graph-based model for cadence detection [8]. The identification of cadences is fundamental for understanding the structural integrity of musical compositions. This problem presents a lot of challenges such as the sparsity of annotations and the ambiguity of the definition of cadences. For evaluation, I compare a graph-based model with an SVM classifier method introducing complex cadence-related features [5]. The results demonstrated a promising direction for the application of graph modeling for cadences achieving better results than previous methods on the explored datasets.

Chapter 6: I present ChordGNN, a graph-based approach for automatic Roman Numeral analysis in symbolic music [9]. ChordGNN can transform and learn from note-wise information and produce onset-wise representation which can be particularly useful when annotations are provided at the onset level. ChordGNN outperforms previous state-of-the-art models in Roman Numeral analysis, as demonstrated through experiments on a large dataset of Western classical music. The study also explores variants of the model, such as incorporating NADE and post-processing techniques, and provides a configuration study on multitask learning strategies. ChordGNN achieves higher accuracy in predicting Roman Numerals, offering more coherent and meaningful harmonic analysis of musical pieces.

Chapter 7: As a next step, I attempt to solve voice separation through the application of GNNs [10]. Voice separation is the process of identifying monophonic streams in polyphonic music. This process is essential for dissecting complex musical textures into individual melodies that interact together. In this work, I enrich the graph modeling of the score and approach voice separation as a link prediction

task. I compare our graph-based model with an HHM-based method [6]. The graph-based approach presents a number of advantages such as better performance, reduced computational time and space complexity, and independence on the number of individual monophonic streams.

Chapter 8: I discuss a more recent publication, where I extend the previous work on monophonic voice separation towards homophonic voice separation and staff prediction for symbolic piano music using a GNN approach [11]. This method involves creating an input graph from the score as before, and then predicting voice and staff assignments through an encoder-decoder architecture. The system addresses challenges such as cross-staff voices and polyphonic music, which are difficult tasks in music score engraving. Evaluated on two datasets of different styles, the proposed method shows consistent improvements over previous approaches, effectively separating notes into readable musical scores. The process includes a post-processing phase to ensure valid musical outputs and a visualization tool for displaying the results on musical scores.

Chapter 9: In this work, I investigate how graphs can be used as an explanation means for GNN models [12]. Ergo, I present SMUG-Explain, a framework for generating and visualizing explanations for GNNs applied to musical scores. SMUG-Explain uses post-hoc gradient-based methods to generate explanations, focusing on both the importance of individual note features and subgraphs that significantly influence the model's predictions. The framework features an interactive web interface that visualizes these explanations directly on the musical score, enhancing interpretability. Applied to cadence detection, the framework demonstrates the ability to provide musically meaningful insights into the model's decision process. The evaluation includes quantitative metrics, such as fidelity and characterization scores of the explanations, and qualitative analyses of classical music excerpts, highlighting the practical benefits of this approach in understanding complex musical predictions.

Chapter 10: This chapter introduces MusGConv, a novel graph convolutional block designed specifically for processing musical score data by integrating principles of human musical perception, focusing on pitch and rhythm [13]. MusGConv addresses tasks such as monophonic voice separation, harmonic analysis, cadence detection, and composer identification, framed as node classification, link prediction, and graph classification problems respectively. Experiments show that MusGConv improves performance in three of these tasks without increasing computational cost, demonstrating the benefits of perception-informed processing in graph network applications for music.

Chapter 11: This chapter addresses the lack of a unified framework by offering tools to efficiently process music graphs and train GNNs [14]. Therefore, I introduce GraphMuse, a graph processing framework and library designed for symbolic music tasks using GNNs. Key innovations of GraphMuse include a neighbor sampling technique tailored to the unique properties of musical scores and the integration of hierarchical modeling elements, such as beats and measures, to enhance graph network expressivity. The framework's efficacy is demonstrated through experiments on pitch spelling and cadence detection tasks, showing significant performance improvements over previous methods. The GraphMuse Python library is released

as open-source in Github and it aims to standardize graph-based symbolic music processing and catalyze further advancements in the field.

REFERENCES

[1] Milton Babbit. "The Use of Computers in Musicological Research." In: *Perspectives of New Music* 3.2 (1965), pp. 74–83.

[2] The Pianola Institute. *History of the Pianola – Piano Players*. Accessed: 2024-06-25. n.d. URL: http://www.pianola.org.

[3] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1263–1272.

[5] Louis Bigo, Laurent Feisthauer, Mathieu Giraud, and Florence Levé. "Relevance of musical features for cadence detection." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[6] Andrew McLeod and Mark Steedman. "HMM-based voice separation of MIDI performance." In: *Journal of New Music Research* 45.1 (2016), pp. 17–26.

PUBLICATIONS MAKING UP THIS THESIS

[7] Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer, and Carlos Eduardo Cancino-Chacón. "Symbolic Music Representations for Classification Tasks: A Systematic Evaluation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[8] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[9] Emmanouil Karystinaios and Gerhard Widmer. "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[10] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[11] Francesco Foscarin, Emmanouil Karystinaios, Eita Nakamura, and Gerhard Widmer. "Cluster and Separate: a GNN Approach to Voice and Staff Prediction for Score Engraving." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2024.

[12] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "SMUG-Explain: A Framework for Symbolic Music Graph Explanations." In: *Proceedings of the Sound and Music Computing Conference (SMC)*. 2024.

[13] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Perception-Inspired Graph Convolution for Music Understanding Tasks." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2024.

[14] Emmanouil Karystinaios and Gerhard Widmer. "GraphMuse: A Library for Symbolic Music Graph Processing." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2024.

2 MUSIC REPRESENTATIONS

Music representations are the fundamental component for Music Information Retrieval (MIR) and particularly for musical analysis tasks. Whether the scientific work is focused on audio or symbolic music, the choice of representation is crucial. It determines how musical information is encoded, stored, and processed, directly influencing the efficiency and accuracy of models and machine learning approaches.

2.1 SYMBOLIC VS. AUDIO REPRESENTATIONS

One can consider two primary types of music representations: symbolic and audio. Audio representation captures the sound of music as it is heard. It involves encoding the waveform of audio signals into digital formats such as WAV, MP3, or FLAC. These representations include acoustic information such as timbre, dynamics, and pitch. While audio data is rich and detailed, it is also complex and requires significant computational resources for processing. Features calculated from audio signals such as spectrograms, mel-frequency cepstral coefficients (MFCCs), and chroma features are commonly used to analyze audio data.

Symbolic representation, on the other hand, encodes music in terms of discrete note events and their associated parameters, such as pitch, duration, and onset time, and it is used principally to capture music as it was written. Unlike audio, symbolic representations do not contain information about the actual sound but rather the musical structure and notation. This makes them more compact and easier to manipulate for certain types of musical analysis. Symbolic notation formats include MIDI, MusicXML, MEI, ABC, **kern, and others.

Symbolic representations can be broadly categorized into three main forms: matrix (piano roll), sequence (tokenized sequences), and graph representations. Each form has its unique way of encoding musical information and is suitable for different types of MIR tasks.

2.2 SYMBOLIC MUSIC STORAGE FORMATS

The term symbolic refers to the symbols present in any form of musical score or notation. A musical score can contain a variety of elements other than notes. Such elements may include time signature, key signature, articulation markings, dynamic markings, and many others. Each notation system and music format includes different sets of musical nuances and elements. Should one consider different music cultures and their own notation then the set of individual music elements increases considerably. The nature and diversity of music make the definition of a uniform generalized notation an almost infeasible task. Therefore, in this section, I will focus

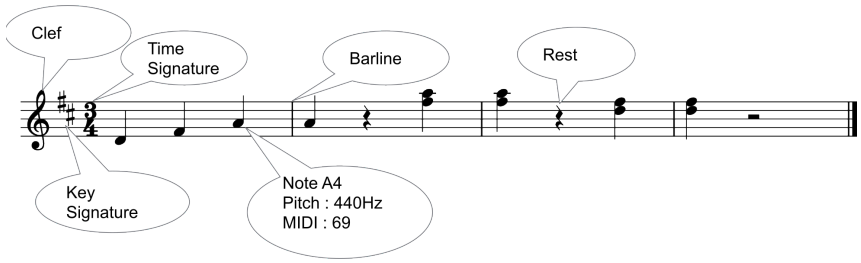
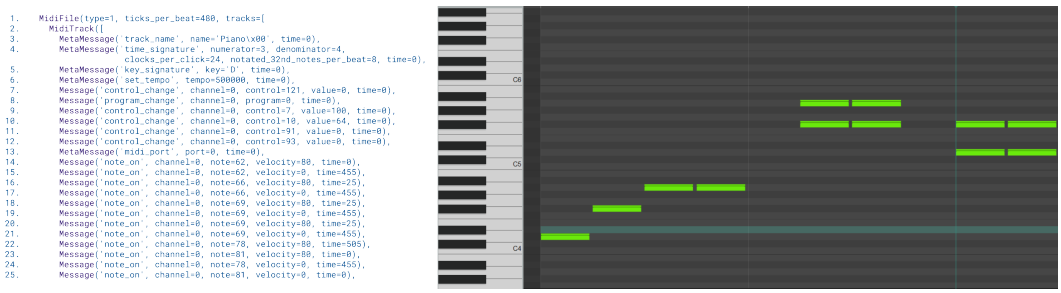


Figure 2.1: A musical score with highlighted elements of the score.



```
MidiFile(type=1, ticks_per_beat=480, tracks=[
  MidiTrack([
    MetaMessage('track_name', name='Piano\x00', time=0),
    MetaMessage('time_signature', numerator=3, denominator=4,
                clocks_per_click=24, notated_32nd_notes_per_beat=8, time=0),
    MetaMessage('key_signature', key='D', time=0),
    MetaMessage('set_tempo', tempo=500000, time=0),
    Message('control_change', channel=0, control=121, value=0, time=0),
    Message('program_change', channel=0, program=0, time=0),
    Message('control_change', channel=0, control=7, value=100, time=0),
    Message('control_change', channel=0, control=10, value=64, time=0),
    Message('control_change', channel=0, control=91, value=0, time=0),
    Message('control_change', channel=0, control=93, value=0, time=0),
    MetaMessage('midi_port', port=0, time=0),
    Message('note_on', channel=0, note=62, velocity=80, time=0),
    Message('note_on', channel=0, note=62, velocity=0, time=455),
    Message('note_on', channel=0, note=66, velocity=80, time=25),
    Message('note_on', channel=0, note=66, velocity=0, time=455),
    Message('note_on', channel=0, note=69, velocity=80, time=25),
    Message('note_on', channel=0, note=69, velocity=0, time=455),
    Message('note_on', channel=0, note=69, velocity=80, time=25),
    Message('note_on', channel=0, note=69, velocity=0, time=455),
    Message('note_on', channel=0, note=78, velocity=80, time=505),
    Message('note_on', channel=0, note=81, velocity=80, time=0),
    Message('note_on', channel=0, note=78, velocity=0, time=455),
    Message('note_on', channel=0, note=81, velocity=0, time=0),
```

Figure 2.2: An example of MIDI messages on the left and the pianoroll on the right.

on representations stemming principally from the Western classical music notation (example shown in Figure 2.1).

Many formats exist for storing and exchanging forms of music notation. In this section, I present some of the most commonly used in MIR today.

2.2.1 MIDI

The Standard MIDI File (SMF), commonly known as MIDI, is one of the oldest and most widely used formats for storing musical data [1]. It uses an 8-bit binary format to store a sequential stream of MIDI events, such as Note-On and Note-Off, which correspond to the pressing and releasing of a key on a keyboard. These events also encode pitch and velocity (the intensity of the note). The MIDI format was created as a communication protocol for digital keyboard players but since its use has been expanded to encode information from any instrument. MIDI events are organized into tracks, with each track representing the notes that an instrument plays. Tracks are streams of events, each preceded by a delta-time value indicating the time before the next event.

MIDI files contain a header that defines the time units and file structure, with two main types: Type 0 (a single track for solo pieces) and Type 1 (multiple tracks for ensemble performances). While MIDI files can effectively represent performance data (e.g. performance on a MIDI capable keyboard), they are more limited in terms of the information they can encode in respect to traditional scores. For instance, they lack discrete note elements, ties, dots, rests, and do not distinguish between different pitch spellings (e.g., D-sharp vs. E-flat). Additionally, all voice events are flattened into a single track, making it less ideal for representing complex scores with more hierarchical organization.

```
**kern
*M4/4
=1
4c 4e 4g 4c
4d 4f 4a 4d
4e 4g 4b 4e
4f 4a 4c 4f
=2
*-
```

Figure 2.3: Example of the Kern format

2.2.2 Text-Based Formats

Text-based formats like ABC [2], MuseData [3], Humdrum **Kern** [4], LilyPond [5], Guido [6], and MusicTeX [7] offer the advantage of being easy to read and manually create. These formats use different methods to balance readability with the ability to encode complex musical elements. However, as the complexity and size of a score increase, these formats become less efficient to parse compared to binary or XML-based formats. In Figure 2.3 we provide a short example of a kern score.

2.2.3 XML-Based Formats

Recent approaches to music notation storage have led to the development of XML-based formats, such as MusicXML [8] and MEI [9]. These formats prioritize a comprehensive description of the score, including graphic specifications for visual rendering, over compactness. MusicXML is widely supported by commercial software and provides detailed, and sometimes redundant, representations. MEI, designed for corpus preservation and analysis, supports a wider range of notations beyond the Western classical notation, and includes unique identifiers for each score element, aiding in visualization and interaction.

2.3 COMPUTATIONAL REPRESENTATIONS OF SYMBOLIC MUSIC

Most symbolic music notation formats attempt to accommodate Western musical notation [10]. However, regardless of whether the format is text-based, xml-based, or binary, it is not a form of representation that could be fed to a learning algorithm without creating an intermediate representation first. Three forms of representations can be identified that are used to train deep-learning models.

2.3.1 Matrix

Matrix representation, commonly known as piano roll, visualizes music as a two-dimensional grid with time on the horizontal axis and pitch on the vertical axis (example in Figure 2.5). Each cell in the matrix indicates whether a note is played

```
<time>
    <beats>3</beats>
    <beat-type>4</beat-type>
    </time>
<clef>
    <sign>G</sign>
    <line>2</line>
    </clef>
</attributes>
<note default-x="110.48" default-y="-45.00">
<pitch>
    <step>D</step>
    <octave>4</octave>
    </pitch>
<duration>1</duration>
<voice>1</voice>
<type>quarter</type>
<stem>up</stem>
</note>
```

Figure 2.4: An example of a MusicXML score.

at a specific time and pitch. This representation is straightforward and intuitive, it is historically derived from the paper rolls of a player piano. It is particularly useful for tasks like automatic music transcription and classification of piece-level attributes such as composer or genre. However, piano rolls often lack detailed information about dynamics, articulation, and other musical nuances.

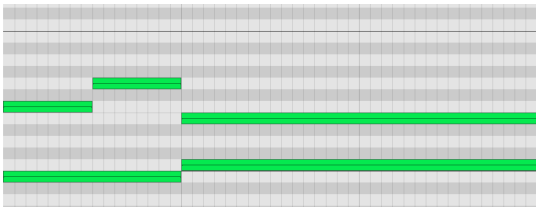


Figure 2.5: Example of a pianoroll segment.

2.3.2 Sequence

Sequence representation treats music as a sequence of events, similar to a string of language tokens (example in Figure 2.6). This approach involves tokenizing musical attributes such as pitch, duration, and onset time into a sequence of discrete symbols. Recent advancements in natural language processing have influenced the development of sequence representations for music, using models like Transformers to capture long-range dependencies and contextual information. Examples of tokenization schemes include MIDILike [11], REMI [12], and CompoundWord [13]. Sequence representations are powerful for generative tasks, such as music composition and arrangement, but can struggle with capturing hierarchical structures inherent in music.

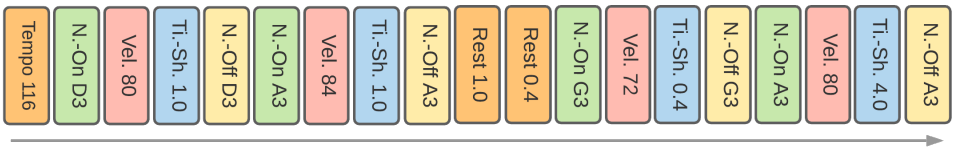


Figure 2.6: Example of Midi Like tokenization of a musical score. Taken from https://miditok.readthedocs.io/en/latest/tokenizations.html

2.3.3 Graph

Graph representation models a musical score as a network of notes (vertices) connected by relationships (edges) based on the temporal relation between the notes (example in Figure 2.7). This approach can capture complex interactions between notes, such as harmonic, melodic, and rhythmic relationships. Graph neural networks (GNNs) are used to process these representations, allowing for sophisticated analysis of musical features. Graph representations are particularly effective for tasks that require an understanding of musical structure and context, such as cadence detection and expressive performance modeling. However, constructing and processing musical graphs can be computationally intensive and requires careful design to avoid over-smoothing and other issues inherent to graph learning.

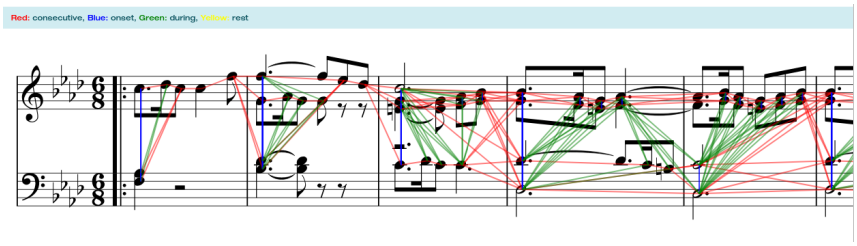


Figure 2.7: Example of a graph representation overlayed on a musical score. Colors indicate the different types of relations.

2.4 LIBRARIES FOR HANDLING MUSIC NOTATION FORMATS

From a computational perspective, various tools and techniques have been developed to encode and decode music scores. These include software packages capable of loading, saving, and extracting information from a wide range of digitized musical notation formats. Using such packages information from the score can be extracted and transformed into a form of representation (e.g. numerical data) that can be subsequently fed to a learning algorithm to solve specific tasks.

Focusing on the libraries that can parse scores and extract information we cite below a few important libraries in the field.

2.4.1 Music21

Music21 is a powerful toolkit for computer-aided musicology, developed at the Massachusetts Institute of Technology (MIT) by Michael Scott Cuthbert and his team

in 2008 [14]. It is a Python-based library that allows people to study large collections of music, generate music, teach music theory, and edit musical notation. Music21 is widely used in academic research, music theory, and educational settings.

2.4.2 Partitura

The Partitura Python package is a toolkit designed for working with symbolic music data, particularly focusing on the analysis, processing, and conversion of musical scores and performances from a computer scientist's perspective [15]. It has a similar scope to other tools like Music21 but with specific features for handling performance data, and it is designed for simplicity first.

2.4.3 PrettyMidi

PrettyMIDI is a Python library designed to handle and process MIDI files in a simple and intuitive way [16]. It is widely used in music information retrieval (MIR), algorithmic composition, and other areas of music technology. Developed by Colin Raffel, PrettyMIDI provides a high-level interface for working with MIDI data, making it easier to analyze, manipulate, and synthesize music.

2.4.4 MidiTok

MidiTok is a Python package for MIDI file tokenization [17]. It tokenizes symbolic music files such as MIDI and ABC. It is designed to convert such formats into sequences of tokens ready to be fed to models such as transformers for MIR tasks.

For symbolic music, the tokenizer can be trained to increase both model performance and efficiency. MidiTok includes many known MIDI Tokenizations. Tokenizers can be further trained with techniques such BPE [18] to reduce the vocabulary of the tokens.

2.5 FOCUS ON SYMBOLIC REPRESENTATION

The path from musical notation to training an algorithm for solving a musical task such as harmonic analysis or composer classification, involves a series of steps. Given a digitized version or format of music, first a digital library should be able to read it and extract information, and then, the extracted information should follow some data type paradigm appropriate to serve as input to learning algorithms.

Symbolic representation, with its many forms, offers distinct advantages for different MIR tasks. Each data type form, i.e., matrix, sequence, and graph, provides unique insights and capabilities, making them valuable tools for researchers and practitioners in the field of symbolic MIR. By leveraging these representations, it is possible to develop more efficient, interpretable, and sophisticated MIR systems that can handle the complexities of musical data.

In summary, understanding the different types of symbolic music representation and their applications is crucial for advancing MIR research. Although audio

representations provide detailed acoustic information, symbolic representations offer compact and structured data that are easier to manipulate for specific analytical tasks. Chapter 2 of this thesis will offer a systematic experimental evaluation of different symbolic representations over a number of different task, in order to obtain more insight into their relative merits.

REFERENCES

[1] David Back. *Standard MIDI-file format specifications*. Acessed August 29, 2024. 1999. URL: http://www.music.mcgill.ca/~ich/classes/mumt306.

[2] Irwin Oppenheim, Chris Walshaw, John Atchley, and Guido Gonzato. *he abc standard 2.0.* Acessed August 29, 2024. 2010. URL: https://abcnotation.com/wiki/abc:standard:v2.0.

[3] Eleanor Selfridge-Field. "Musedata: multipurpose representation." In: *Beyond MIDI: The handbook of musical codes*. Center for Computer Assisted Research in the Humanities. The MIT Press, 1997.

[4] David Huron and Craig Sapp. *The Humdrum Toolkit*. Acessed August 29, 2024. 1993. URL: https://www.humdrum.org/.

[5] Han-Wen Nienhuys and Jan Nieuwenhuizen. "LilyPond, a system for automated music engraving." In: *Proceedings of the xiv colloquium on musical informatics (xiv cim 2003)*. Vol. 1. Citeseer. 2003, pp. 167–171.

[6] Holger H Hoosy, Keith A Hamelz, Kai Renzy, and J urgen Kiliany. "The GUIDO Notation Format A Novel Approach for Adequately Representing Score-Level Music." In: *International Computer Music Conference (ICMC)*. 1998.

[7] Daniel Taupin, Ross Mitchell, and Andreas Egler. "MusiXTEX. Using TEX to write polyphonic or instrumental music." In: *TUGboat* 14.3 (1993), pp. 212–220.

[8] M Good. "An internet-friendly format for sheet music." In: *Proceedings of XML Conference*. 2001.

[9] Perry Roland. "The music encoding initiative (MEI)." In: *Proceedings of the First International Conference on Musical Applications Using XML*. Vol. 1060. Citeseer. 2002, pp. 55–59.

[10] Mark Gotham, Kyle Gulling, and Chelsey Hamm. *Open Music Theory-Version 2.*, 2022.

[11] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. "This Time with Feeling: Learning Expressive Musical Performance." In: *Neural Computing and Applications* 32 (2018), pp. 955–967. URL: https://link.springer.com/article/10.1007/s00521-018-3758-9.

[12] Yu Siang Huang and Yi Hsuan Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions." In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. arXiv: 2002.00212.

[13] Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2021. arXiv: 2101.02402v1.

[14] Michael Scott Cuthbert and Christopher Ariza. "music21: A toolkit for computer-aided musicology and symbolic music data." In: (2010).

[15] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[16] Colin Raffel and Daniel PW Ellis. "Intuitive Analysis, Creation, and Manipulation of MIDI data WITH pretty_midi." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[17] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "MidiTok: A Python package for MIDI file tokenization." In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[18] Nathan Fradet, Nicolas Gutowski, Fabien Chhel, and Jean-Pierre Briot. "Byte Pair Encoding for Symbolic Music." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2001–2020. DOI: 10.18653/v1/2023.emnlp-main.123. URL: https://aclanthology.org/2023.emnlp-main.123.

3 GRAPH NEURAL NETWORKS

3.1 INTRODUCTION TO GRAPH NEURAL NETWORKS

Graphs are the natural representation in many real-world applications such as community detection, drug discovery, recommender systems, and many more. By creating graph data structures it is possible to define elements from the perspective of their relations while capturing structural information that is often lost in other representations. For example, in a social network, a vertex can represent a person and edges can represent their friends or connections within the network. This structure is almost natural for such fields and it can pave the way towards a variety of tasks such as community detection, link prediction, and node classification.

3.1.1 Challenges in Graph Representation

Graph structure is the natural representation for some domains, but learning from graph-structured data is not straightforward. Unlike matrix or grid-structured data such as images or sequences, graphs are not defined by a fixed structure, or even a fixed size. The nodes within a graph do not respect a specific order and each node in the graph can have a varying number of neighbors. This non-Euclidean structure [1] requires non-traditional deep learning techniques. Although explicit connections in a graph can help represent relationships, the complexity of these connections can still make it difficult to capture long-range dependencies and hierarchical structures in a meaningful way.

Standard deep learning approaches cannot effectively deal with the structural complexities of graphs. For instance, the direct application of convolutional neural networks (CNNs) is non-trivial due to the irregular structure of graphs [2], recurrent neural networks are not appropriate since graphs are usually not ordered structures [3], and finally, transformers sometimes struggle with the memory requirements of big social network graphs [4].

3.1.2 Emergence of Graph Neural Networks

Graph Neural Networks (GNNs) have been introduced as a tool for learning on graph-structured data [5]. The main difference between GNNs and other learning techniques is the application of message-passing which allows to learn from the graph structure of the data.

The foundational idea of GNNs is to leverage the graph's structure by aggregating and transforming information from a node's neighbors. This approach enables GNNs to create node embeddings that encapsulate both the features of individual nodes and the structure of the graph. This has been shown to improve performance in various tasks such as node classification, link prediction, and graph classification.

For instance, in node classification, the goal is to predict the label of each node by considering not only the features of the node itself but also the features of its neighboring nodes. This task is common in social network analysis for detecting fraudulent users or in biological networks for predicting protein functions [6].

Similarly, link prediction aims to predict the existence of edges between nodes, which is crucial for applications such as recommending new friends in social networks or identifying potential interactions between proteins in biological networks [7].

Graph classification, on the other hand, involves classifying entire graphs into different categories. This is particularly useful in scenarios like classifying chemical compounds based on their molecular structure or categorizing documents based on their citation networks [5].

3.1.3 Formal Definition of Graphs

A *graph* is composed of two primary components: a set of vertices (or nodes) and a set of edges that connect these vertices. Vertices represent the entities in a system, while edges signify the relationships or interactions between these entities. Formally, a graph G is defined as an ordered pair $G = (V, E)$, where V is a set of vertices, and $E \subseteq V \times V$ is a set of edges, each representing a connection between two vertices. In the case of *undirected graphs*, each edge is an unordered pair u, v, where $u, v \in V$. For *directed graphs*, each edge is an ordered pair (u, v), indicating a directed relationship from vertex u to vertex v. The *degree* of a vertex in an undirected graph is the number of edges incident to it, while in a directed graph, each vertex has both an *in-degree* and an *out-degree*, corresponding to the number of incoming and outgoing edges, respectively.

Moreover, a matrix $X \in \mathbb{R}^{(|V|,k)}$ with node features can be associated with the graph, where every vertex is associated to a vector with k features, in which case we call it an *attributed graph*. Graphs may also be *weighted* or *unweighted*. In a weighted graph, each edge is assigned a scalar value, which might represent aspects like distance or connection strength.

An *adjacency matrix* is a square matrix also used to represent a graph, where the rows and columns correspond to the vertices. Each element in the matrix indicates whether a pair of nodes is connected by an edge. In an unweighted graph, the matrix contains binary values (0 or 1), with a 1 indicating the presence of an edge. In a weighted graph, the matrix entries hold the weight of the edge connecting the nodes. For directed graphs, the matrix is typically asymmetric, reflecting the directionality of the edges. It is common to consider graphs with self-loops, in which case the diagonal of the adjacency matrix is filled with non-zero values.

3.1.4 Graphs and Heterophily

Graphs can be *homogeneous* or *heterogeneous*. In a homogeneous graph, all nodes and edges represent the same type of entities and relationships. An example is a social network graph where all nodes are people and all edges represent the same type of connection.

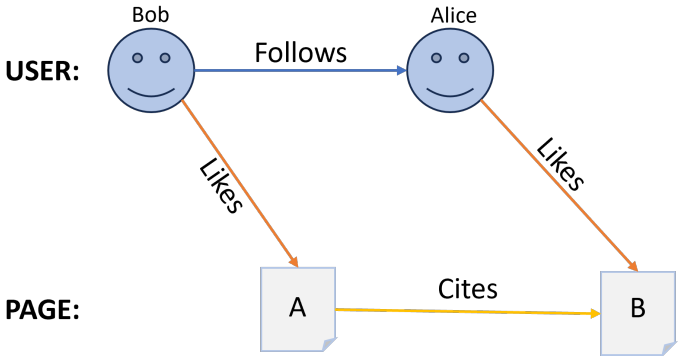


Figure 3.1: An example of a heterogeneous graph. Two types of vertices user and page and three types of edges (likes, follows and cites)

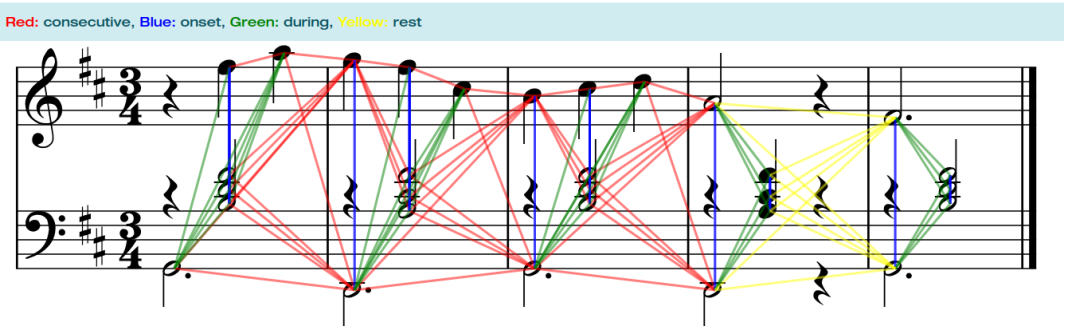


Figure 3.2: A score graph depicted on an exerpt from Eric's Satie Gnosserie No. 1. The different colors of the edges stand for different connection types.

In contrast, a heterogeneous graph contains nodes and edges of different types. For example, a social network can be modeled by a heterogeneous graph, where vertices can be either users or pages, and a user can friend another user or like a page as shown in Figure 3.1.

A *heterogeneous* graph [8] can be formally defined by a quadruplet $G = (V, E, \mathcal{T}, \mathcal{R})$ where V and E are the set of nodes and the set of edges, respectively, $\mathcal{T}$ is the set of node types, and $\mathcal{R}$ is the set of edge types (relations). Each node $v \in V$ is associated with a type $t(v) \in \mathcal{T}$, and each edge $e = (u, v) \in E$ is associated with a relation $r(e) \in \mathcal{R}$.

Heterogeneous graphs can be used to model a variety of scenarios in which the features for each type or relation can vary in dimensionality. In such a setting, the definition of an attributed graph from the previous section can be extended when multiple types of vertices and edges are considered.

The *music score graph* from the previous chapter can also be viewed as a heterogeneous graph where there is a single node type for notes but many relation types for the different edge conditions. A visual example is provided in Figure 3.2. In this musical example, different edge types relate to different kinds of temporal relations between notes: simulteneity (blue), following notes (red), silence (yellow), and notes that occur after another note has started but before it ended. Modelling those four relations, asserts a graph without disconnected components from start to finish of a musical piece.

3.2 THE LEARNING BLOCKS OF GRAPH NEURAL NETWORKS

A Graph Neural Network is a type of feed forward convolutional neural network that takes as input an attributed graph. The output can vary depending on the task, if can be: i) node-level, such as node classifications or embeddings; ii) edge-level, like predicting relationships between nodes or iii) graph-level outputs, such as a single prediction for the entire graph. GNN training works by iteratively updating node embeddings, through message passing, where each node updates its feature representation by aggregating information from its neighboring nodes for each layer of the network. This involves combining the node's own features with those of its neighbors. As the network deepens, the node's representation captures more complex, and more distant relationships within the graph. During training, the GNN optimizes a loss function (e.g., for classification or regression) using gradient-based methods, like backpropagation.

Graph Neural Networks (GNNs) utilize various types of convolutions to process and learn from graph-structured data. These convolutions can be broadly categorized into spectral methods and spatial methods [9]. Each of these methods has distinct characteristics and computational requirements.

Regardless of the convolution method, almost every learning process from graph data is based on the principle of *message passing* to propagate information across the graph. The message passing process is defined by the following steps:

1. **Initialization:** Each node is assigned a feature vector, which can include some important properties. For example, in a musical score, this could include pitch, duration, and onset time for each node/note.

2. **Message Generation:** Each node generates a message to send to its neighbors. The message typically includes the node's current feature vector and any edge features that describe the relationship between the nodes. A message can be for example a linear transformation of the neighbor's node features.

3. **Message Aggregation:** Each node collects messages from its 1-hop neighbors. We define a *l-hop neighborhood* of a node in a graph consists of all nodes that are reachable from the given node within l edges or steps. The aggregation function is usually a permutation invariant function such as sum, mean, or max and it combines these messages into a single vector, ensuring that the node captures information from its entire neighborhood.

4. **Node Update:** The aggregated message is used to update the node's feature vector. This update often involves applying a neural network layer (like a fully connected layer) followed by a non-linear activation function (such as ReLU).

5. **Iteration:** Steps 2–4 are repeated for a specified number of iterations or layers, allowing information to propagate through the graph. With each iteration, nodes incorporate information from progressively larger neighborhoods.

By using this recursive process, a neural network can learn from graph-structured data. The whole process can be summarized in the following equations:

$$\mathbf{h}_v^{(0)} = x_v \tag{3.1}$$

$$\mathbf{m}_v^{(l)} = \text{agreggate}_{u \in \mathcal{N}(v)} M^{(k)}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_u^{(l-1)}, \mathbf{e}_{vu}) \tag{3.2}$$

$$\mathbf{h}_v^{(l)} = \sigma\left(U^{(l)}\left(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l)}\right)\right) \tag{3.3}$$

Where $\mathbf{x}_v$ represents the input node features of node v, $\mathbf{e}_{uv}$ represents the potential edge features of (u, v), $\mathcal{N}(v)$ denotes the neighbors of v, σ is a non-linear activation function, l is the current layer, and U, M are feedforward neural networks [3].

Message passing can be also viewed from a matrix percpective, where messages are just the dot product of features from the previous layer with the adjacency matrix. Then a simplified version of the message passing process can be expressed as:

$$H^{(l)} = \sigma\left(A(H^{(l-1)}M^{(l)})U^{(l)}\right) \tag{3.4}$$

where A is the adjacency matrix, U, M are feedforward neural networks and σ is a non-linear activation function.

3.2.1 Spectral Methods

Spectral methods for graph deep learning are based on the spectral decomposition of the graph Laplacian. The key idea is to use the eigenvalues and eigenvectors of the Laplacian to perform convolution operations in the frequency domain. Spectral methods have been the initial blocks for the kickstart of the graph deep learning field. Therefore, not all work is based on the message passing principle defined earlier although as the field progressed recent spectral approaches are build upon message passing. The graph Laplacian L, is typically defined as:

$$L = D - A \text{ or } L = D^{\frac{1}{2}}AD^{\frac{1}{2}} \text{ or } L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{3.5}$$

where D denotes the degree matrix, i.e. a diagonal matrix with the degree (number of edges) of each node in a graph, A denotes the adjacency matrix of the graph, and I is an identity matrix. The laplacian relates to many useful properties for graphs and it captures key structural information about a graph, including its connectivity and smoothness of signals over nodes.

Some key models for spectral graph neural networks include Spectral CNNs, ChebNet and the Graph Convolutional Network.

SPECTRAL CNNS The first notable spectral-based GNN was proposed by Bruna et al. [10]. The authors proposed convolution on graphs via a spectral decomposition of the graph Laplacian. The spectral convolution of a signal x with a graph filter g can be defined as:

$$\hat{x} = Ug_\theta(\Lambda)U^T x \tag{3.6}$$

where U is the matrix of eigenvectors of the Laplacian, Λ is the diagonal matrix of its eigenvalues, and $g_\theta(\Lambda)$ is a diagonal matrix of spectral filter parameters. However, the computational complexity of this method is high due to the need for eigenvalue decomposition.

CHEBNET Defferrard et al. introduced ChebNet [11], which uses Chebyshev polynomials to approximate the convolution operation, reducing computational complexity and improving efficiency. The Chebyshev polynomials are used to construct the filter:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}) \tag{3.7}$$

where $\theta \in \mathbb{R}^k$ is a vector of learnable Chebyshev coefficients, $\tilde{\Lambda} \in [-1, 1]$ is rescaled from Λ, the Chebyshev polynomials $T_{k+1}(\Lambda) = 2\Lambda T_k(\Lambda) - Tk - 1(\Lambda)$ are recursively defined with $T_0(\Lambda) = 1$ and $T_1(\Lambda) = \Lambda$, and k controls the size of filters, i.e., localized in k-hop neighborhood of a vertex.

GRAPH CONVOLUTIONAL NETWORK (GCN) Kipf and Welling [6] proposed the GCN model, which simplifies the convolution operation using a first-order approximation of spectral convolutions. The convolution operation in GCN is defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}) \tag{3.8}$$

where $\tilde{A} = A + I$ is the adjacency matrix with added self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $H^{(l)}$ is the feature matrix at layer l , $W^{(l)}$ is the weight matrix, and σ is an activation function.

Due to the simplicity, efficacy and low memory requirements of the GCN model, it is regarded one of the foundational blocks of graph convolutional networks and still achieves comparable performance to state-of-the-art approaches on graph benchmarks [12].

3.2.2 Spatial Methods

Spatial methods define convolutions directly on the graph structure, using the "spatial" relationships between nodes These methods aggregate information from a node's neighborhood to update its representation. Some popular spatial methods for graph convolution include GraphSAGE and Graph Attention Networks.

GRAPHSAGE GraphSAGE (Graph Sample and AggregatE) [13] is a popular spatial method that generates node embeddings by sampling and aggregating features from a node's local neighborhood . The GraphSAGE framework defines several aggregation functions, including mean, LSTM, and pooling. The general aggregation operation for a node v is:

$$h_v^{(l+1)} = \sigma\left(W \cdot \text{concat}\left(h_v^{(l)}, \text{agreggate}\left(\{h_u^{(l)}, \forall u \in \mathcal{N}(v)\}\right)\right)\right) \tag{3.9}$$

where $h_v^{(l)}$ is the embedding of node v at layer l, $\mathcal{N}(v)$ is the set of neighbors of v , W is the weight matrix, and σ is an activation function.

GRAPH ATTENTION NETWORK (GAT) Graph Attention Networks (GATs) [14] introduce attention mechanisms to assign different weights to different neighbors,

allowing the model to focus on the most relevant nodes when aggregating information. The attention mechanism is defined as:

$$e_{ij} = \text{LeakyReLU}\left(a^T \left[Wh_i \| Wh_j\right]\right) \tag{3.10}$$

where e_{ij} is the attention coefficient between nodes i and j, a is the attention vector and $\|$ denotes concatenation. The attention coefficients are then normalized using a softmax function. The entire convolutional block is then described by:

$$h_i^{(l+1)} = \sum_{j \in \mathcal{N}(i)} \text{softmax}\left(e_{i,j}\right) W^{(l)} h_j^{(l)} \tag{3.11}$$

Graph Attention Networks (GATs) are one of the most popular graph convolutional blocks and they achieve high accuracy in many tasks. Many variants of GATs have been proposed, such as GATv2 [15].

3.2.3 Convolution in Heterogeneous Graphs

Targeted achitectures for applying graph convolution on heterogeneous graphs have been recently developed, however, any convolutional message-passing network can be adapted for heterogeneous graphs.

To understand convolution in heterogeneous graph we first need to define *relation triplets*. If $\mathcal{T}_N$ is the set of node types in the graph and $\mathcal{R}$ is the set of edge types, the set of relations triplet is composed by elements $(t_u, r, t_v) \in \mathcal{T}_N \times \mathcal{R} \times \mathcal{T}_N$.

GENERIC CASE The process of applying a message-passing network to heterogeneous graphs requires two steps. First, multiply the number of convolutional blocks (a block is one message passing iteration) per layer by the number of different relation triplets present in the input graph, and second, define an aggregation process before iterating the message and aggregation steps. This process is visually depicted in Figure 3.3.

RGCN A Relational Graph Convolutional Network (RGCN) [16] is an extension of the Graph Convolutional Network (GCN) designed to handle graphs with multiple types of relations, making it particularly useful for heterogeneous graphs with a single node type. In an RGCN, each node's representation is updated by aggregating information from its neighbors, taking into account the type of edge (relation) connecting them. This allows the model to learn different transformations for different types of relations.

In an RGCN, the hidden representation $\mathbf{h}_v^{(l)}$ of a node v at layer l is computed as:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_v^r} \frac{1}{c_{v,r}} \mathbf{W}_r^{(l+1)} \mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)}\right) \tag{3.12}$$

where $\mathcal{R}$ is the set of all relation types in the graph, $\mathcal{N}_v^r$ denotes the set of neighboring nodes u of node v that are connected by relation type r, $\mathbf{W}_r^{(l)}$ is the learnable weight matrix associated with relation r at layer $l+1$, $\mathbf{W}_0^{(l)}$ is the weight matrix for the self-loop, which applies to the node's own features at layer $l+1$, $c_{v,r}$

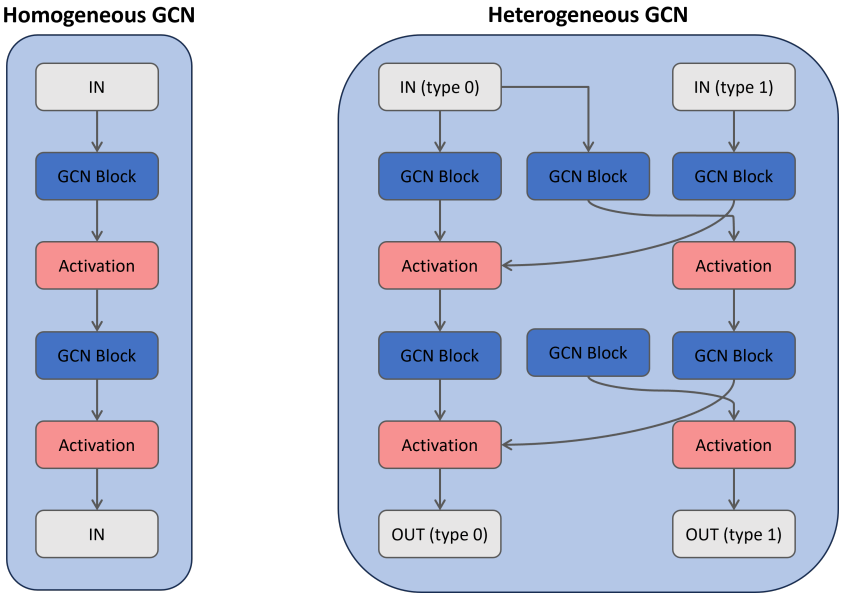


Figure 3.3: Comparing the graph convolution architecture between a homogeneous and a heterogeneous graph model.

is a normalization constant that is typically set to the degree of node v with respect to relation r, ensuring that the contribution of each neighbor is appropriately scaled, $\mathbf{h}_u^{(l)}$ represents the hidden state of the neighboring node u from the previous layer l, and σ is a non-linear activation function.

The use of distinct weight matrices for each relation type allows the RGCN to capture the unique characteristics of each type of connection within the graph, enhancing its ability to model multi-relational data effectively.

THE HETEROGENEOUS GRAPH TRANSFORMER The **Heterogeneous Graph Transformer (HGT)** [17] is a graph neural network architecture designed to model heterogeneous graphs where nodes and edges can have multiple types. HGT leverages attention mechanisms to learn representations of nodes that account for the different types of nodes and relationships in the graph. The key idea is to apply a type-specific attention mechanism that can dynamically focus on different neighbors depending on their types.

For a node v of type t_v, the hidden representation $\mathbf{h}_v^{(l)}$ at layer l is updated using a type-specific attention mechanism. The update rule can be formalized as follows:

$$\mathbf{m}_{uv}^{(l+1),r} = \oplus_{k \in K} \left(h_v^{(l)} W_M^{(l+1,r)} \right) \tag{3.13}$$

$$\mathbf{ff}_{uv}^{(l+1,r)} = \oplus_{k \in K} \text{softmax} \left(\mathbf{k}_u^{(l+1,r,k)} \cdot \mathbf{q}_v^{(l+1,r,k)} * \mu \right) \tag{3.14}$$

$$\mathbf{h}_v^{(l+1)} = W_{up}^{(l+1,t_v)} \sigma \left(\text{aggregate}_{u \in \mathcal{N}_v^{t_u,r}} \mathbf{m}_{uv}^{(l+1,r)} \cdot \mathbf{ff}_{uv}^{(l+1,r)} \right) + \mathbf{h}_v^{(l)} \tag{3.15}$$

where $\mathcal{T}_N$ is the set of node types in the graph and $\mathcal{R}$ is the set of edge types, R stands for the set of relation triplets in the graph $(t_u, r, t_v) \in \mathcal{T}_N \times \mathcal{R} \times \mathcal{T}_N$, $\mathcal{N}_v^{t_u,r}$ represents the set of neighboring nodes u of node v that have type t_u and

are connected by relation r. $\alpha_{vu}^{(l,r)}$ is the attention coefficient between node v and its neighbor u that depends on keys $\mathbf{k}_u^{(l+1,r,k)}$ of the source node and queries of the destination node $\mathbf{q}_v^{(l+1,r,k)}$ which are linear transformations of $h_u^{(l)}$ and $h_v^{(l)}$ accordingly for head k, $\oplus$ concatenates information from each head k up to the total number of heads K, finally σ is a non linear activation functions and μ a normalization term that depends on the degree of v.

The attention mechanism in HGT is inspired by the Transformer architecture, but it is adapted to handle the heterogeneity of the graph. The attention mechanism focuses on the type-specific interactions between nodes and uses different query, key, and value projections depending on the node and edge types.

After multiple layers of HGT, the final node representation $\mathbf{h}_v^{(L)}$ is obtained by stacking and applying the type-specific attention mechanism over L layers. This allows HGT to capture both local and global information across different node and edge types in the graph.

3.2.4 Hybrid Models for Graph Convolution

Many approaches that combine transformers or sequential models with graph convolution have been proposed; however, the most notable work is the "General, Powerful, Scalable Graph Transformer (GPS)" [18] which processes a graph at each layer by simultaneously passing it through both a GNN layer and a Transformer layer. Subsequently, it joins the resulting node embeddings with an MLP. This design enables GPS to combine the GNN's ability to capture structural information from local neighborhoods and the Transformer's ability to capture long-range relationships between nodes. Furthermore, GPS is flexible, allowing the use of any homogeneous GNN and Transformer layer. When the Transformer layer uses a linear attention mechanism, the overall complexity of GPS becomes linear in relation to the number of nodes and edges. Other derivative work has proposed variants of GPS achieving high accuracy in benchmark tasks for graph classification [12]. However, applying hybrid models to larger graphs remains non-trivial.

3.3 TAXONOMY OF GRAPH NEURAL NETWORK TASKS

Graph Neural Networks (GNNs) can be applied to various supervised graph learning tasks, including node classification, link prediction, and graph classification. Each of these tasks has distinct objectives and applications that take advantage of the power of GNNs to process and learn from graph-structured data.

3.3.1 Node Classification

Node classification or, more generally, node property prediction involves predicting the labels or properties of individual nodes within a graph. This task is essential in various applications, such as:

- **Social Network Analysis**: Detecting fraudulent users or predicting user interests based on their connections and interactions.

- **Biological Networks**: Predicting the function of proteins or genes by analyzing their interactions within a biological network.
- **Music Graphs**: Node classification can be used for any music tasks that require prediction at the note level, e.g. pitch spelling (predicting the note name and accidentals of each note).

Several methods have initially been developed for node classification using GNNs, such as GCN and GraphSAGE.

3.3.2 Link Prediction

Link prediction aims to predict the existence of edges between nodes. This task is crucial in applications like:

- **Recommender Systems**: Suggesting new connections or friends in social networks based on existing connections.
- **Biological Networks**: Predicting interactions between proteins or genes to understand biological processes.
- **Music Graphs**: Specific kind of link prediction task could be to ask the question whether notes u, v share the same property, e.g. do they belong to the same voice, octave, chord?

Since the prediction of edges can be viewed as altering the existing graph or creating a new graph, methods for link prediction usually employ graph auto-encoders or graph generative models.

VARIATIONAL GRAPH AUTO-ENCODERS (VGAE) VGAE is a probabilistic model for unsupervised learning of graph-structured data, using a variational auto-encoder framework to generate latent representations of nodes and predict the likelihood of edges between them [19].

GRAPH GENERATIVE MODELS Graph generative models, such as GraphRNN, leverage generative adversarial networks to learn the underlying distribution of graphs and generate new edges or entire graph structures. These models are particularly useful in scenarios where the graph is incomplete or evolving [20].

3.3.3 Graph Classification

Graph classification involves classifying entire graphs into different categories. This is particularly useful in scenarios like:

- **Chemical Compound Classification**: Classifying molecules based on their structure and properties to predict their chemical activity.
- **Document Classification**: Categorizing documents based on their citation networks or content relationships.

- **Music Graphs**: Graph classification can be applied to any problem that searches for a global answer on an entire piece of music, e.g. composer classification.

Since graph classification uses the entire graph to predict a single global label or property and graphs may vary in size, it usually employs some kind of contraction or aggregation of all latent representations of nodes after graph convolution. Many models apply the contractions in many steps to incrementally select the most representative components of the graph. Models such as DiffPool and hierarchical graph convolutional networks are such examples.

DIFFPOOL DiffPool [21] introduces a differentiable pooling mechanism to hierarchically cluster nodes and create a coarsened graph representation. This method enables the GNN to learn hierarchical structures and improves its performance on graph classification tasks.

HIERARCHICAL GRAPH CONVOLUTIONAL NETWORKS Hierarchical graph convolutional networks [22]use multiple layers of graph convolutions and pooling operations to capture information at different levels of granularity. This approach helps in learning both local and global graph features, enhancing the classification accuracy.

3.4 TRAINING TECHNIQUES FOR GRAPH NEURAL NETWORKS

Training Graph Neural Networks (GNNs) involves unique challenges, particularly when dealing with large graphs. This section discusses the differences in training techniques for small and large graphs, highlighting key methods used to handle these challenges.

3.4.1 Training on Small Graphs

Training GNNs on small graphs is relatively straightforward. The entire graph can typically be loaded into memory, allowing for efficient batch training. Standard gradient descent methods can be used to optimize the network parameters.

BATCH TRAINING Batch training involves processing many graphs in a single forward and backward pass. This method is feasible for small graphs where memory constraints are not a major issue. The process includes forward propagation through the network for many graphs simultaneously, computing the loss, and then performing backpropagation to update the weights. The batch size is defined by the number of graphs used for each pass, but the total number of nodes and edges within each batch contributes to the memory limitations of the method.

3.4.2 Sampling Techniques for Large Graphs

Training GNNs on large graphs presents significant challenges due to memory and computational constraints. When a graph is too large to fit into memory, it is impossible to train a model without partitioning the graph. To address this issue, various sampling techniques have been developed that approximate the convolution operations on a smaller subset of the graph.

3.4.2.1 Node-wise Sampling

Node-wise sampling is a specific type of sampling technique which generates node embeddings by sampling and aggregating features from a node's local neighborhood that was introduced in GraphSAGE [13]. GraphSAGE is the first to consider inductive representation learning on large graphs. The node-wise sampling approach in GraphSAGE allows for efficient training on large graphs by sampling a fixed number of neighbors for each node, rather than using the entire neighborhood. This reduces the computational complexity and makes the model scalable.

Another notable approach to node sampling is VR-GCN [23], which aims to reduce the size of the sampled nodes by analyzing the variance of sampling as a means to avoid the neighborhood expansion problem of node-wise sampling. The neighborhood expansion problem in GNNs occurs when, with deeper layers, a node gathers information from too many distant nodes, leading to over-smoothing and loss of useful local details.

3.4.2.2 Layer-wise Sampling

Layer-wise sampling techniques were initially proposed to tackle the neighborhood expansion problem. Instead of sampling nodes and their neighbors collectively across all layers, layer-wise sampling is the process of making sampling decisions for each individual layer of the network.

FASTGCN FastGCN uses importance sampling to approximate the convolution operation, enabling efficient minibatch training on large graphs [24]. The key idea is to sample a subset of nodes and their neighbors, reducing the amount of computation and memory required. The convolution operation in FastGCN is the same as GCN [6] but the sampled adjacency and degree matrices are used instead of the entire degree and adjacency matrix of the graph.

LADIES Layer-Dependent Importance Sampling or LADIES for short [25] is a layer-wise sampling technique that considers previously sampled nodes for calculating layer-dependent sampling probability. Based on the sampled nodes, LADIES chooses their neighborhood nodes and computes their importance probability. Subsequently, it samples a fixed subset of nodes for each batch.

The LADIES sampling process computes dense computation graphs and avoids the oversmoothing problem of deep networks caused by the extreme expansion of the receptive field. Moreover, LADIES achieves low memory cost and reduced time complexity compared to other sampling paradigms, and tackles neighborhood expansion problems by controlling sampling variance.

3.4.2.3 Graph-wise Sampling

Graph-wise sampling focuses on partitioning the input graph into computationally efficient and meaningful subgraphs.

CLUSTER-GCN In Cluster-GCN [26], the graph is first partitioned into clusters, and then each cluster is treated as a mini-batch. This method ensures that the nodes within each cluster are densely connected, which helps in preserving the local structure of the graph. The convolution operation for a node v in Cluster-GCN is:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cap C} \frac{1}{\sqrt{d_v d_u}} W^{(l)} h_u^{(l)}\right)$$

where $\mathcal{N}(v)$ is the set of neighbors of node v, C is the cluster to which node v belongs, d_v and d_u are the degrees of nodes v and u, respectively, $W^{(l)}$ is the weight matrix, and σ is the activation function.

GRAPHSAINT GraphSAINT [27] employs subgraph sampling and training, improving the scalability and efficiency of GNNs for large-scale graphs. It achieves this in two steps. First, eliminate the sampling bias by applying a loss normalization and aggregation normalization, and second, it minimizes the sampling variance by controlling the edge sampling probability during training.

The GraphSAINT framework focuses its sampling strategy on the edges rather than nodes, therefore solving the neighborhood expansion problem while controlling sampling bias and variance.

3.4.3 Music Graphs and Sampling

Music graphs, i.e. graphs created from music scores, do not necessarily correspond to either of the aforementioned categories. A music graph can be rather big with thousands of nodes and edges but at the same time, it is never vastly big such as a social network with billions of nodes and edges. So, what would be the best paradigm for training models on music graphs?

The size of a music graph depends on many music attributes such as orchestration, form, duration of the piece, etc. Therefore, the size of a music graph can vary considerably between pieces. Since the size variation is large, batching many graphs together without applying sampling is a challenge. However, music graphs are small enough to fit individually in memory.

An applicable training paradigm would be to only use individual music graphs without batching or sampling. Nevertheless, a more memory- and time-efficient paradigm should be applicable to best take advantage of the available resources. Graph Sampling techniques are memory- and time- efficient but are they appropriate for music graphs?

From the presented sampling techniques, node-wise sampling stands out due to its simplicity and effectiveness. Node-wise sampling, as used in GraphSAGE [13], offers a balance between efficiency and performance. Furthermore, the nature of music graphs does not present the risk of neighborhood expansion, namely that

with each additional layer, a node aggregates information from progressively larger neighborhoods. Nevertheless, any other graph sampling technique could be applied if no musical constraints are considered.

Music, even when represented as a graph, has a time component with connects its elements together. Therefore, a sampling method that samples arbitrary nodes from music graphs or partitions a music piece based on graph-related conditions might not be entirely appropriate for music. Based on these considerations, we offer a more complete answer and approach to music graph sampling in Chapter 11.

3.5 CHALLENGES AND FUTURE DIRECTIONS

Despite the success of Graph Neural Networks (GNNs) in various applications, several challenges and open research directions remain. Addressing these challenges will further enhance the capabilities and applicability of GNNs.

3.5.1 Deep Architectures

One of the main challenges in GNNs is developing deeper architectures. Current models often use only a few layers, as deeper models tend to suffer from issues such as over-smoothing and vanishing gradients. Over-smoothing occurs when repeated aggregation operations cause node representations to become indistinguishable, even for nodes from different classes. Additionally, deeper models are more prone to overfitting, especially when the amount of labeled data is limited [28].

To address these issues, researchers have proposed several approaches, such as residual connections and skip connections, which help maintain gradient flow and prevent over-smoothing. For example, the Jumping Knowledge Network (JK-Net) aggregates outputs from different layers to adaptively select the most informative representations [29].

3.5.2 Dynamic Graphs

Most GNN models assume static graphs, but many real-world networks are dynamic, with nodes and edges changing over time. Social networks, transportation networks, and communication networks are examples where the graph structure evolves. Developing GNNs that can handle dynamic graphs is crucial for applications requiring real-time analysis and predictions [30].

Dynamic GNNs aim to capture temporal patterns and evolving structures within graphs. Techniques such as Temporal Graph Networks (TGNs) and Dynamic Graph Convolutional Networks (D-GCNs) incorporate time-dependent information to model the changes in the graph over time [31].

3.5.3 Scalability

Scalability is a major concern for GNNs, especially when dealing with large-scale graphs. Efficient sampling techniques and parallelization strategies are crucial for

improving the scalability of GNN models. Although methods like GraphSAGE [13] and FastGCN [24] have made significant progress, there is still a need for more scalable approaches that can handle extremely large graphs with billions of nodes and edges.

Recent advancements such as GraphSAINT [27] and Cluster-GCN [26] have introduced efficient sampling and clustering methods to reduce computational complexity and memory requirements. However, further research is needed to develop models that can scale seamlessly with the growing size of real-world graphs.

3.5.4 Heterogeneous Graphs

Heterogeneous graphs contain multiple types of nodes and edges, representing different entities and relationships within a single graph. These graphs are common in applications such as knowledge graphs, recommender systems, and biological networks. Traditional GNNs, designed for homogeneous graphs, may not effectively capture rich semantic information in heterogeneous graphs [16].

Heterogeneous Graph Neural Networks (HGNNs) extend GNNs to handle heterogeneous graphs by incorporating type-specific transformations and aggregation functions. For example, Relational graph convolutional networks (R-GCNs) use different weight matrices for different types of edges, allowing the model to learn from the various interactions in the graph [16]. Further research is needed to develop more sophisticated HGNNs that can leverage the complex structures and semantics in heterogeneous graphs.

3.5.5 Explainability and Interpretability

A major limitation of deep models and therefore also GNNs is that they are not inherently interpretable. Understanding the decision-making process of GNNs is essential for making transparent, fair and controllable models.

Some methods have been developed to make GNNs more interpretable, such as graph attention networks (GATs) that assign attention weights to different neighbors, highlighting their relative importance [14]. Many post hoc methods for GNN explanations attempt to identify small subgraphs that are most important for making individual predictions. These methods can be categorized in four levels: gradient-based methods, perturbation-based methods, decomposition-based methods, and surrogate methods [32].

Gradient-based methods such as [33, 34] compute target prediction gradients with respect to input characteristics by backpropagation to approximate input importance. Perturbation-based methods such as [35, 36] study the output variations with respect to different input perturbations. Surrogate methods such as [37, 38] employ an interpretable surrogate model to approximate the predictions of the complex deep model for the neighboring nodes of the input example. Decomposition methods such as [39] measure the importance of input features and edges by decomposing the original model predictions. In Chapter 9, we will introduce a framework for applying and visualizing such explanations on music score graphs.

Despite the development of GNNs in terms of explainability, there are still many paths left unexplored.

3.5.6 More Powerful Aggregation Functions

Most of the existing spatial graph convolutional network models are based on neighborhood aggregations. These models have been proved theoretically to be at most as powerful as one-dimensional Weisfeiler-Lehman graph isomorphism tests, which suggests that they can effectively capture the structural properties of graphs. To achieve higher expressive power, research is also focused on investigating the expressiveness of aggregation functions and architectures that go beyond the limitations of current models [40–42].

3.6 CONCLUSION

In summary, while GNNs have achieved remarkable success in various applications, several challenges remain. There are even more challenges when applying GNNs on music but this observation also leaves room for innovation.

In our work, we will address some of these challenges related to music graphs specifically or to general graph deep learning such as scalability, heterophily, explainability. For heterophily, we have developed across our work models, sampling techniques and representations that deep with heterogeous music graphs. For scalability, we have developped in Chapter 11 a library that handles efficiently and intuitively large collections of music score graphs, withing this library we adress specifically issues such as sampling techniques and memory management for music graphs. In, chapter 9, we investigate GNN explainability for music graphs and we develop interpretable visual feedback.

Furthermore, we touch upon issues that are more prominent in music graph learning. In Chapter 4, we outline the advantages of graphs over other architectures and representations for graph-level musical tasks. In Chapter 5, we tackle extreme label imbalances in node classification for cadence detection. In Chapter 6 and in Chapter 11, we experiment with hybrid architectures that combine GNNs with sequential models. In Chapters 7 and 8, we apply algorithmic post hoc methods to improve edge-level tasks for voice separation. Finally, in Chapter 10, we develop dedicated graph convolution techniques that utilize inductive bias to improve performance on graph, edge, and node-level tasks for music.

REFERENCES

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric deep learning: going beyond euclidean data.” In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[2] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. “The emerging field of signal processing on graphs:

Extending high-dimensional data analysis to networks and other irregular domains." In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[3] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022.

[4] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. "Do transformers really perform badly for graph representation?" In: *Advances in Neural Information Processing Systems (NeurIPS* 34 (2021), pp. 28877–28888.

[5] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[6] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[7] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. "Graph convolutional networks: a comprehensive review." In: *Computational Social Networks* 6.1 (2019), pp. 1–23.

[8] Yizhou Sun and Jiawei Han. "Mining heterogeneous information networks: a structural analysis approach." In: *ACM SIGKDD explorations newsletter* 14.2 (2013), pp. 20–28.

[9] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. "A comprehensive survey on deep graph representation learning." In: *Neural Networks* (2024), p. 106207.

[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." In: *Advances in Neural Information Processing Systems (NeurIPS* 29 (2016).

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. "Open graph benchmark: Datasets for machine learning on graphs." In: *Advances in Neural Information Processing Systems (NeurIPS* 33 (2020), pp. 22118–22133.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[14] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[15] Shaked Brody, Uri Alon, and Eran Yahav. "How attentive are graph attention networks?" In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2022).

[16] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer. 2018, pp. 593–607.

[17] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous graph transformer." In: *Proceedings of the web conference 2020*. 2020, pp. 2704–2710.

[18] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. "Recipe for a general, powerful, scalable graph transformer." In: *Advances in Neural Information Processing Systems (NeurIPS* 35 (2022), pp. 14501–14515.

[19] Thomas N Kipf and Max Welling. "Variational graph auto-encoders." In: *NIPS Workshop on Bayesian Deep Learning* (2016).

[20] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. "Graphrnn: Generating realistic graphs with deep auto-regressive models." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5708–5717.

[21] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. "Hierarchical graph representation learning with differentiable pooling." In: *Advances in Neural Information Processing Systems (NeurIPS*. Vol. 31. 2018.

[22] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. "Hierarchical graph pooling with structure learning." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2019.

[23] Jianfei Chen, Jun Zhu, and Le Song. "Stochastic training of graph convolutional networks with variance reduction." In: *International Conference on Machine Learning (PMLR)*. 2018.

[24] Jie Chen, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[25] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. "Layer-dependent importance sampling for training deep and large graph convolutional networks." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2019.

[26] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks." In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 257–266.

[27] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. "GraphSAINT: Graph Sampling Based Inductive Learning Method." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020.

[28] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. "Deepergcn: All you need to train deeper gcns." In: *arXiv preprint arXiv:2006.07739* (2020).

[29] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5453–5462.

[30] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. "Representation learning for dynamic graphs: A survey." In: *Journal of Machine Learning Research* 21.70 (2020), pp. 1–73.

[31] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. "Temporal graph networks for deep learning on dynamic graphs." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2020.

[32] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2022), pp. 5782–5799.

[33] Federico Baldassarre and Hossein Azizpour. "Explainability techniques for graph convolutional networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[34] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10772–10781.

[35] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 32 (2019).

[36] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. "Interpreting graph neural networks for NLP with differentiable edge masking." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021.

[37] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. "Graphlime: Local interpretable model explanations for graph neural networks." In: *IEEE Transactions on Knowledge and Data Engineering* 35.7 (2022), pp. 6968–6972.

[38] Minh Vu and My T Thai. "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 33 (2020), pp. 12225–12235.

[39] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. "Higher-order explanations of graph neural networks via relevant walks." In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7581–7596.

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[41] Lisa Schneckenreiter, Richard Freinschlag, Florian Sestak, Johannes Brandstetter, Günter Klambauer, and Andreas Mayr. "GNN-VPA: A Variance-Preserving Aggregation Strategy for Graph Neural Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2024.

[42] Eran Rosenbluth, Jan Toenshoff, and Martin Grohe. "Some might say all you need is sum." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023.

Part II

INDIVIDUAL CONTRIBUTIONS

4 SYMBOLIC MUSIC REPRESENTATIONS FOR CLASSIFICATION TASKS

Title: Symbolic Music Representations for Classification Tasks: A Systematic Evaluation

Published In Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR), Milano Italy, 2023.

Authors: Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer, Carlos Cancino-Chacon

Contribution: In this work, I was mostly involved in the graph-related part of the experiments and setup by providing insights for designing and training graph-based representations and models. Furthermore, I participated in designing some of the experiments and writing the paper.

Abstract: Music Information Retrieval (MIR) has seen a recent surge in deep learning-based approaches, which often involve encoding symbolic music (i.e., music represented in terms of discrete note events) in an image-like or language-like fashion. However, symbolic music is neither an image nor a sentence, and research in the symbolic domain lacks a comprehensive overview of the different available representations. In this paper, we investigate matrix (piano roll), sequence, and graph representations and their corresponding neural architectures, in combination with symbolic scores and performances on three piece-level classification tasks. We also introduce a novel graph representation for symbolic performances and explore the capability of graph representations in global classification tasks. Our systematic evaluation shows advantages and limitations of each input representation. Our results suggest that the graph representation, as the newest and least explored among the three approaches, exhibits promising performance, while being more light-weight in training.

4.1 INTRODUCTION

The deep learning boom has profoundly impacted MIR, including research involving symbolic music representations (MIDI, scores, etc.). A large body of recent literature focuses on adapting existing architectures from computer vision and natural language processing to the field of symbolic MIR. These approaches often treat music data as an image (piano roll), as a sequence of language tokens, or, more recently, as a graph. However, a piece of music is neither an image nor a sentence or graph, therefore, a critical question still remains open concerning the choice of input representations for symbolic music.

A source of complexity in symbolic music arises from the different modalities of data such as scores and performances. A score contains information about music notation and often includes rich hierarchically structured information such as metrical structure and voicing. Symbolic music performances, on the other hand, such as those recorded on a MIDI-capable instrument, consist of a stream of controller events. Extracting a hierarchical structure from such a stream is not a trivial task [1–3]. Furthermore, such performance data omit some of the rich information that a score provides, such as pitch spelling and articulation markings, but instead, it can include information about expression, timing, local tempo, and performance dynamics.

Recent research has produced relatively large datasets containing scores and performances at the symbolic level, including efforts to align these [4–6]. Motivated by these developments, we present an attempt to shed light on questions revolving around the input representation of symbolic music for deep-learning-based MIR. We formulate an empirical framework where we test multiple input representations, models, and piece-level classification tasks.

In terms of input representations, we investigate piano rolls, tokenized sequences, and graphs. We evaluate multiple models based on these representations on three different tasks: composer classification, performer classification, and (playing) difficulty assessment. Furthermore, having datasets containing both performances and their corresponding scores such as ATEPP and ASAP [4, 5], allows us to apply each combination of representation and task to either score or performance. Our goal is to contribute an experimental overview of different symbolic music representations. The contributions of this work are threefold:

1. We investigate the performance and complexity of matrix, sequence and graph input representations, and their corresponding neural architectures (respectively Convolutional Neural Networks, Transformers, and Graph Neural Networks).

2. We compare the impact that the different information contained in symbolic scores and performances has on different piece-level classification tasks.

3. We introduce a new graph representation for symbolic performances, and explore the capability of graph representations in classification tasks.

4.2 RELATED WORK

The complexity of representing music data has been discussed in the literature [7–9]. Wiggins et al. [10] analyzed the trade-offs of music representation systems with respect to expressive completeness and structural generality. In the age of deep learning, such considerations are still relevant regarding the variety of machine-readable representations such as piano rolls, MIDI-like sequences, NoteTuples, and Musical Spaces [11, 12]. In this section, we focus on three symbolic representations (matrix, sequence, and graphs) and discuss their respective strengths and limitations.

MUSIC AS A MATRIX: Similar to audio spectrograms, a pitch-time representation that is typically used as input to a CNN, the piano roll representation of music

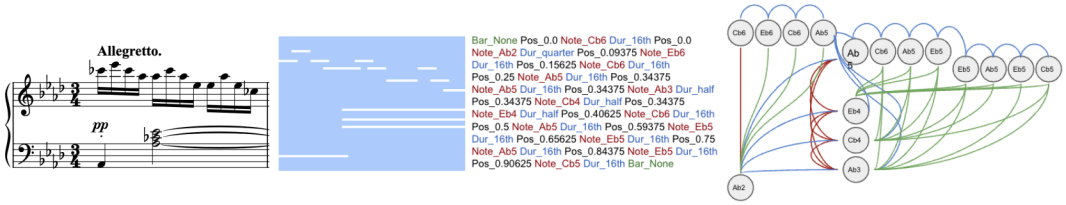


Figure 4.1: Excerpt of Schubert's *Impromptu Op. 90 No.4* and its input visualizations (from left to right): generic matrix, sequence (REMI-like) and graph.

naturally emerges as the symbolic equivalent. Piano rolls have been widely applied in tasks such as automatic music transcription [13, 14], classification of piece-level attributes such as difficulty and composer [15–18], as well as generation of music accompaniment or performed dynamics [19, 20].

A piano roll is a bare-bones representation of symbolic music data, and, therefore, information such as key signatures, articulation annotations, metrical structure, different instrument parts, and voicing structure are not encoded in the representation[11, 21].

MUSIC AS A SEQUENCE: Modeling symbolic music as sequences has a longstanding tradition in MIR. The multiple viewpoint system is a sequence representation that has been widely used for music analysis, generation, and classification [22–25], as well as the basis for cognitively plausible models of expectation [26, 27]. In this system, musical elements are represented by viewpoints [28], which are abstract functions mapping musical events to abstract derived features like pitch, interval, and melodic contour.

With the advances of deep learning-based language models, sequential representation of music as *language tokens* has recently received a lot of attention in sequence-to-sequence generative tasks from automatic orchestration [29] to description-based medley generation [30]. Similar to a stream of MIDI messages, various tokenization schemes encode music features such as pitch, onset time, duration, and velocity sequentially. Besides generation, large-scale pre-training using music sequences has been applied to downstream music understanding tasks [31, 32].

However, tokenized music sequence representations create difficulty for models to learn the dependency of long contexts. Length reduction methods such as Byte Pair Encoding (BPE) [29, 33] aim to address the length overflow problem by replacing the occurrence of frequent subsequences with new tokens.

MUSIC AS A GRAPH: A musical score can also be seen as a graph where notes form the vertices and relations between notes define the edges. Jeong and al. [34] introduced a graph modeling of a musical score for generating expressive performances. Recently, Karystinaios and Widmer [35] presented a new modeling of the score graph based on three different note relations and a Graph Convolutional Network for cadence detection in classical music. A score graph can be homogeneous or heterogeneous, i.e. having one or several types of edges and/or vertices, respectively [36]. We will investigate both heterogeneous and homogeneous score graphs based on the representation used in [35].

Graph Neural Networks have gained popularity in recent years, however, graph learning inherently presents some limitations, such as over-smoothing in deep graph networks [37] and restrictions of Message Passing, where information in graph neu-

ral networks flows only between edge relations predetermined by the representation (in contrast to a Transformer architecture where everything is interconnected [38]).

4.3 METHODOLOGY

In this section, we describe the methodology followed, the corpora used, and the experiments conducted to investigate in-depth the different symbolic representations.

4.3.1 Representation Design

We briefly introduce a formal definition of each representation type, i.e. matrix, sequence, and graph. An example of the three representations is shown in Figure 4.1.

4.3.1.1 Matrix

We define as a matrix representation of music a 2-dimensional array $\mathbf{M} \in \mathbb{N}^{H \times W}$ that depicts musical notes on the time axis, commonly referred to as a piano roll. The vertical axis consists of 128 possible values attributed to the MIDI pitch of note events, where we add three more optional fields for the *una corda*, *sostenuto*, and sustain pedals only applied on the MIDI performances.

In this work, we experimented with multiple channels as used in Onsets and Frames [39]. The onset channel is a binarized roll with activations at onset timestamps, while the frame channel encodes the duration of the note and the velocity of the MIDI event. For scores, the velocity values are substituted by the voice index, i.e. the integer number assigned to a note to indicate the index among the number of independent voices.[1]

4.3.1.2 Sequence

A symbolic music sequence $\mathbf{S} \in \mathbb{N}^{1 \times N}$ is defined by a series of discrete tokens that represent attributes of notes. Vocabularies such as $V_{\texttt{pitch}}, V_{\texttt{TimeShift}}, V_{\texttt{Vel}}$ assign semantic meanings to tokens, and different tokenization schemes translate into different grammars of sequence construction. In this work, we test three popular tokenization schemes: *MIDILike* [40, 41], *REMI* [42], and *CompoundWord* [43] and use the implementation of the MidiTok library [44].

As there is no existing tokenizer for processing scores, we implemented custom MusicXML tokenizers following MidiTok's framework, in the style of *REMI* as well as *CompoundWord*. The major difference is the timing of bars and event positions, as well as the addition of score-specific tokens such as $V_{\texttt{KeySig}}, V_{\texttt{Voice}}$.[2]

Byte Pair Encoding (BPE) is a tokenizer add-on technique that has recently been applied to music sequence learning [33]. It consists of a data compression technique that replaces the most common token subsequences in a corpus with newly created tokens. BPE increases the vocabulary size and shortens the sequence length. We follow the best results from [33] and adopt a BPE with 4 times the

1 This voice information is commonly available in formats such as MusicXML, **Kern, and MEI.
2 Full documentation is provided with our open-source tokenizer in the project repository.

original vocabulary size. On average, this reduced our sequence length between $55 - 65\%$ in both datasets.

4.3.1.3 Graph

A homogeneous score graph G is defined by a tuple (V, E) of vertices and edges. V is the set of notes in a musical score and $E \subseteq V \times V$. Given a score with N notes, we extract a matrix of k-dimensional note-wise features $X \in \mathbb{R}^{N \times k}$ based on features contained in the score or performance. A heterogenous score graph $G = (V, E, \mathcal{R})$ also includes a set of relation types $\mathcal{R}$ such that for every edge $e \in E$, e is of type $r \in \mathcal{R}$ if a condition defined by r holds. In our work, we consider the following relations between two notes u, v which define the edges $e \in E$:

- u and v have the same onset, i.e. $on(v) = on(u)$, then $r =$ onset;
- The offset of u is the onset of v, i.e. $off(u) = on(v)$, then $r =$ consecutive;
- The onset of u lies between the onset of v and the offset of v, i.e. $on(v) < on(u) \wedge on(u) < off(v)$, then $r =$ overlap.

The above relations only hold in the case of score graphs. To adapt this to performance graphs, we use a window tolerance t_{tol}, such that if two notes $(u, v) \in E$ and:

- $|on(v) - on(u)| < t_{\text{tol}}$, then $r =$ onset;
- $|off(u) - on(v)| < t_{\text{tol}}$, then $r =$ consecutive;
- $on(v) < on(u) \wedge on(u) < off(v)$, then $r =$ overlap.

In our configurations, for all graphs created from performance MIDI, we set $t_{\text{tol}} = 30\,\text{ms}$, a perceptual threshold of expressive timing [45]. In addition to the above relations, we consider the possibility of adding an inversely directed edge for the overlap and the consecutive edge types, and we name the inclusion of such edges *inverse edges*. For a homogeneous graph G_{hom} and heterogeneous graph G_{het}, $e \in G_{\text{hom}} \implies e \in G_{\text{het}}$.

The node features X are divided into two categories, the basic and the advanced features. The basic features are implicitly contained in any score or performance note such as one-hot encoding of pitch class and octave of the note's pitch, and duration information. The advanced features contains articulation, dynamics, and notation information from the *Partitura* python package [46]. The detailed computation of these features can be found in original partitura paper [46] and the basis mixer [47].

4.3.1.4 Information Levels

Given the differences in information captured by symbolic scores and performances (Sec. 4.1), we run experiments with separate levels of used information. For the base comparison experiments, we input the basic level of information that is present in both modalities: pitch, duration and onset. The advanced level of information for performance includes dynamics (MIDI velocity) and pedals, while for score includes the voice index (Sec. 4.3.1) as well as score markings such as articulation and dynamics. The results and comparison of each level of information, also with respect to different tasks, will be discussed in Section 4.4.3.

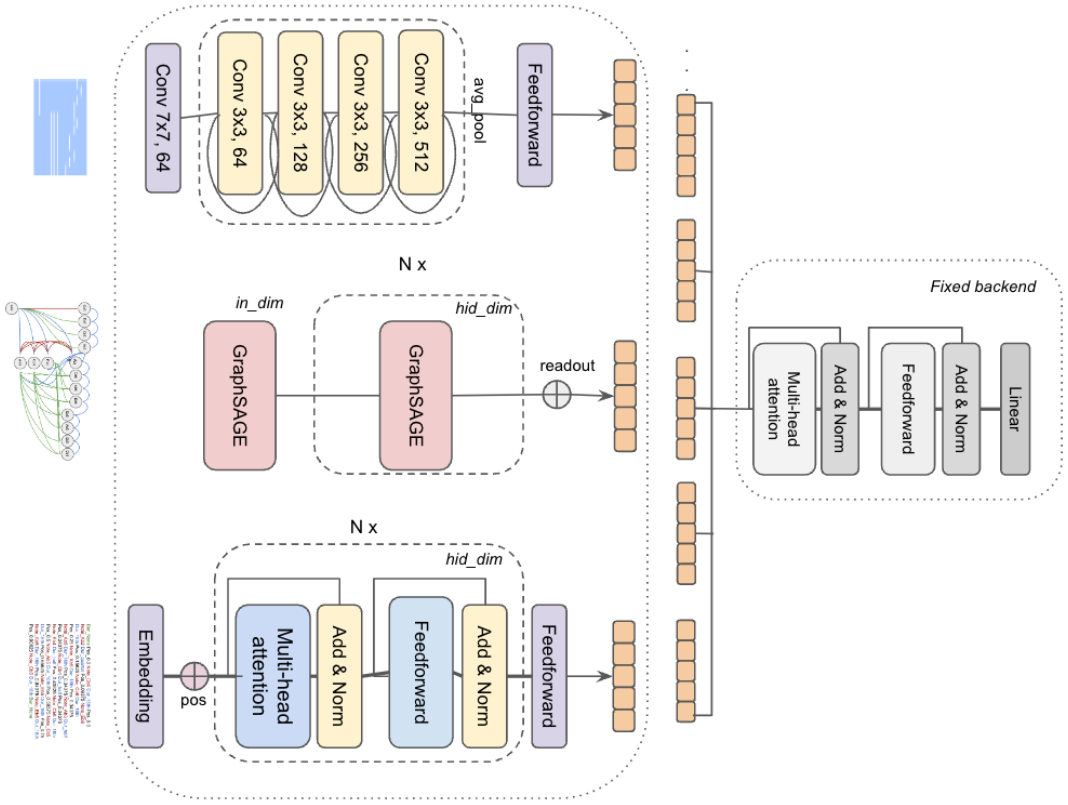


Figure 4.2: Left: front end for three representations, matrix, graph, and sequence, from top to bottom. Right: fixed back end with attention modules.

4.3.2 Modelling Pipelines

In this work, we evaluate the input representations under the same training pipeline of different piece-level classification tasks, as discussed in Section 4.3.3. We split our training architecture into two parts, a front end that projects a window of musical context into a 64-dimensional embedding, and a back end that aggregates the embedding for final prediction. The front end is representation-specific while the back end rests fixed. For a fair comparison, we ensure that the same amount of musical context is given for different front ends to learn. For MIDI performances we fix a window of 60 s, and for symbolic scores, we choose a window of 120 beats given that 120 bpm is a common tempo for music.

For the front end, we employ a commonly used architecture for each respective representation domain:

Matrix: Convolutional neural network based on ResNet [48] blocks with channel numbers adapted to our input.

Sequence: Transformer-encoder [49] front end with positional encoding. Each layer includes multi-head attention with 16 heads followed by an Add & Norm layer. For the combined tokens *CPWord* we add separate embedding layers for each token category in the front end.

Graph: Our graph convolution network (GCN) is built by stacking GraphSAGE blocks [50] followed by a global mean pooling layer. We experiment with both

heterogeneous and homogeneous GraphSAGE. Note that a heterogeneous network has r times more parameters, where r is the number of distinct edge relation types.

For the fixed back end, we used a multi-head attention block with linear projection heads to the desired number of classes, as shown in Figure 4.2. To minimize the impact of model capacity on our comparative discussion, we carried out an ablation study to understand the size of the architecture proportional to each kind of representation (Sec. 4.4.2).

4.3.3 Tasks and Datasets

In this work, we focus on three tasks: composer classification, performer classification, and difficulty assessment. Each one of these tasks is a piece-level task since a label is attributed per piece. The composer classification consists of predicting the composer of the piece. The performer classification involves the prediction of the performer among a list of predefined performers included in the data source. Finally, difficulty assessment involves the prediction of a number between 1-9, with 1 being easy and 9 being hard. The difficulty labels were assembled from Henle Music.[3]

To evaluate the aforementioned tasks, we use two large-scale collections of Western classical piano music that contain corresponding symbolic scores (MusicXML files) and performances (MIDI files), ASAP (1067 performances, 245 scores) and ATEPP (11742 performances, 415 scores). Both datasets contain individual files per movement.

For the composer classification task, we exclude the least populated composer classes for balance in experiments, resulting in 10 classes for the ASAP dataset and 9 classes for the ATEPP dataset. The performer classification task uses MIDI performances of ATEPP with 20 classes. For difficulty, given that both ASAP and ATEPP datasets focus on concert repertoire, the actual classes used range from difficulty 4-9.[4] For all experiments, we use an eight-fold cross-validation evaluation where 85% of our data is used for training and 15% for testing in each fold.

4.3.4 Training

We performed hyperparameter optimization sweeps to determine the optimal learning rate and model hyperparameters. Our convergence criteria include early stopping at the 60 epoch breakpoint with the patience parameter set at 0.005 on the validation accuracy. All our experiments are trained on a single A5000 GPU, and the best models, training logs, and the code is available in the repository.[5]

4.4 EXPERIMENTS AND RESULTS

To evaluate the different representations we performed three experiments. Our first experiment focuses on a detailed comparison of the predictive accuracy of the three

3 Henle Music difficulty labels, https://www.henle.de/en/about-us/levels-of-difficulty-piano/
4 The full distribution of the classes for each task is shown in the supplementary material.
5 https://github.com/anusfoil/SymRep

		ASAP-performance		ASAP-score		ATEPP-performance		ATEPP-score	
		ACC	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*
Matrix	Resl								
	Chnl								
	400	0.59±0.04	0.18±0.02	0.59±0.03	0.18±0.01	0.24±0.05	0.20±0.04	**0.25±0.02**	0.16±0.03
	600	0.62±0.06	0.21±0.03	**0.61±0.07**	**0.19±0.02**	0.28±0.01	**0.22±0.03**	0.24±0.02	0.16±0.04
Sequence	Tokn								
	BPE								
	MidiLike	**0.53±0.05**	**0.16±0.02**	N/A	N/A	0.18±0.04	0.10±0.02	N/A	N/A
	REMI	0.51±0.04	0.15±0.02	0.43±0.04	**0.14±0.01**	**0.23±0.04**	0.10±0.02	0.23±0.04	**0.13±0.02**
Graph	Bi-dir								
		×		×		×		×	
		0.56±0.01	0.17±0.02	0.51±0.05	0.16±0.02	0.22±0.02	0.10±0.03	0.23±0.03	0.21±0.05
		0.58±0.03	0.19±0.01	**0.54±0.05**	**0.17±0.02**	**0.27±0.03**	0.13±0.02	**0.29±0.10**	0.18±0.06
		0.62±0.02	**0.21±0.01**	0.50±0.04	0.17±0.01	0.23±0.04	**0.16±0.03**	0.27±0.06	**0.22±0.03**

Table 4.1: Composer classification results for all representations, on all target subsets of our datasets on the composer classification task using only basic level features. For each subset of data, we present the accuracy score and the macro F1 score with 8-fold cross-validation.
See Section 4.4.1 for explanation of the parameters.

representations/architectures applied to the composer classification task, since it is the most well-understood task among the three. The second experiment studies the impact of model capacity (number of trainable parameters) per representation. Our last experiment investigates the effect of different levels of input features (see Section 4.3.1.4) on the three tasks.

4.4.1 Representations for Composer Classification

Our first experiment is a comparative analysis of the three representations on our two datasets, in the domains of both MIDI performance and MusicXML score with basic level features. For each representation group we test different configurations, i.e. for matrix we experiment with the channel (Chnl) and timestep resolution (Resl), for sequence we change the tokenization scheme (Tokn) and apply BPE, and for graph we investigate the effect of homogeneous or heterogeneous graphs (Multi-rel) and the addition of inverse edges (Bi-dir) (see Sec. 4.3.1). In Table 4.1, we present for each data subset the accuracy score and the macro F1 score and their respective standard deviations under 8-fold cross-validation (see Sec. 4.3.3).

In terms of observations per representation, the matrix representation results indicate no significant differences under different experimental configurations. For sequence representations, the *MIDILike* and *REMI* tokenization schemes yield comparable performance. However, our experiments suggest that *CPWord* is a more challenging representation to learn in the same setting. Concerning the BPE technique, no significant difference is observed between results with 4 times the original vocabulary and the non-BPE version.

Our graph-based models exhibit similar performance regardless of the configuration of the graph edges. In particular, the effect of reverse edges is not significant, and homogeneous graph convolution already achieves similar results to heterogeneous graph convolutional models, which indicates that implicit structural information contained in the heterogeneous approach is not strictly necessary for piece-level classification tasks.

Overall, we observe that three representations show small performance differences in given experiments, with the matrix-CNN approach having the overall best metric across the experiment groups and sequence have the worst.

Finally, we would like to discuss the *album effect*, which concerns the tendency of classification models to learn non-intended features, such as acoustic features in pieces of the same album [51]. In our case, this effect concerns different performances of the same piece that may give away cues for classification. Training with the entire corpus of performance MIDI, which involves different interpretations of the same piece, yields an average accuracy of 90% (see supplementary material), which is 30% higher for the `ASAP-perf` group. To address this issue, we fix the splits to only contain unseen pieces in the test set, which reduced the accuracy score gap between performance and score. This issue has often been overlooked in literature [52, 53] and a commonly-used dataset split is not piece-specific [16]. Given the recent development of large score-performance datasets, we wish to establish a scientifically correct evaluation split taking into consideration the *piece effect*.

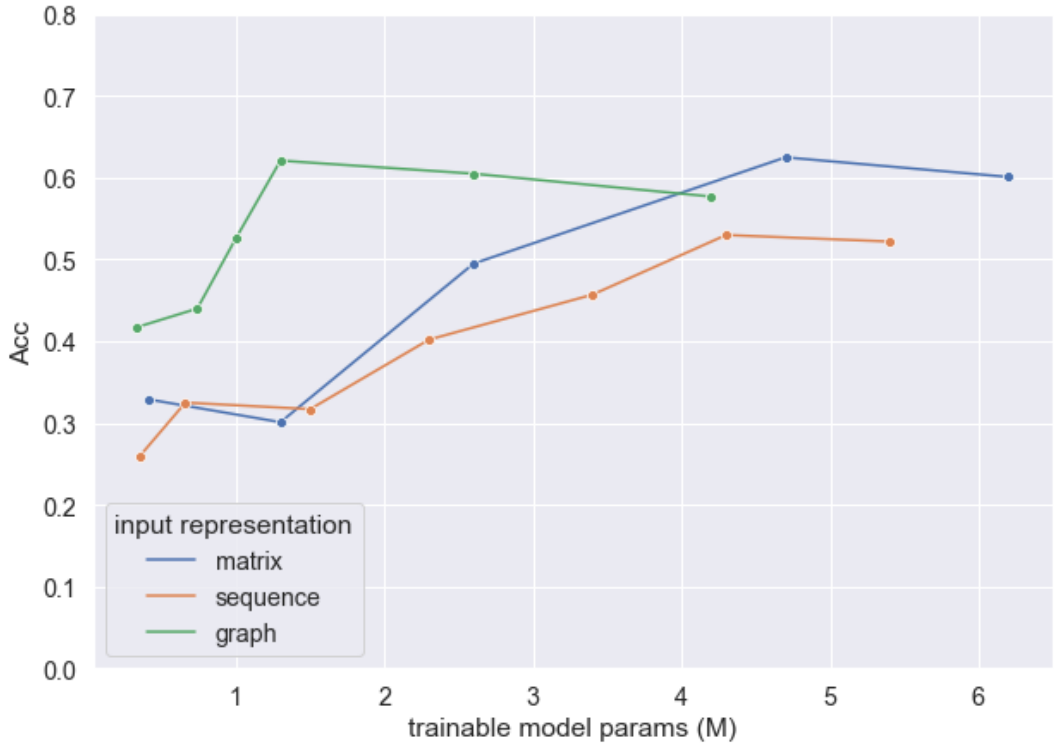


Figure 4.3: Model capacity vs. macro F1 score for each representation approaches on the `ASAP-composer` task.

4.4.2 Complexity

In our second experiment we investigate the impact of model capacity for each representation on the composer classification task using the ASAP dataset. We experiment with different hidden dimensions h and the number of layers N on each architecture corresponding to each of the three representations (Sec 4.3.2), and show our results in Figure 4.3. Overall, we observe that the GCN achieves its best performance using 1.3M parameters, while architectures for matrix and sequence achieve a similar accuracy at around three times the number of parameters.

Another observation concerns the use of large models for piece-level classification tasks on symbolic data. Large convolution models such as ResNet-18/34/50 [16] are substantially over-parametrized, as our results suggest we can achieve similar results using a reduced version of ResNet-8, using less than half the parameters of the smallest used ResNet architecture. Similar observations can be made for transformers, where scaling the model beyond 4.3M parameters does not further improve the performance. Our most efficient transformer encoder consists of 4 layers of attention modules with a hidden dimension of 256, significantly less than transformers used in previous related work [33].

Finally, we note one aspect of our results after scaling our graph network. While *oversmoothing* [37] (features of graph vertices converging to the same value) is a well-known challenge to train deep GCN, our best performing model is a relatively deep and narrow network consisting of 5 layers with a hidden dimension of 64. One possible interpretation is that convergence of node features does not complicate training in the graph-level classification context.

Composer			Performer		
perf	score	perf (ATEPP)	perf	score	Difficulty
Matrix	basic feats	**0.625**	**0.364**	0.403	**0.420**
	advanced feats	0.618	0.342	**0.411**	0.415
Sequence	basic feats	**0.530**	0.287	**0.438**	**0.368**
	advanced feats	0.513	**0.292**	0.426	0.349
Graph	basic feats	**0.607**	0.305	**0.373**	0.361
	advanced feats	0.598	**0.323**	0.356	**0.405**

Table 4.2: Accuracy of three identification tasks on the ASAP dataset, with basic or higher-level features.

4.4.3 Comparison of Feature Levels and Tasks

As discussed in Section 4.3.1.4, we are also interested in understanding the impact of different levels of features on the three classification tasks. With this motivation, we performed our third set of experiments, where we adopted the best configuration of models explored in experiment 1 (see Section 4.4.1). We report the accuracy results in Table 4.2.

Our results indicate that MIDI performances and MusicXML scores have similar capabilities for distinguishing composers and difficulty. Furthermore, matrix and sequence approaches exhibit better results when learning with performances compared to scores. For the difficulty classification task, in particular, all three representations achieved approximately 40% accuracy on the 6 difficulty levels. Performer classification is more challenging since the difference lies in the timing nuances and dynamic changes instead of the pitch information, which are more prominent in our input representations. In the 20-way classification, our approaches generally achieved around 30% accuracy.

Our observations suggest that the addition of advanced features has a variable impact on the representations. Interestingly, the addition of advanced features does not improve the training from sequence representations in most experiments, which can possibly be explained by the increase in vocabulary size and relative sparsity of such information. Graph structures benefit from the addition of voice edges, especially in the representation of scores, where the performance boosts for both composer and difficulty classification. Notably, the `graph-score` with advanced features configuration achieved the best result in score-based composer classification, when jointly compared with Table 4.1.

4.4.4 Transformer vs. GNN: Are We Learning the Same Set of Musical Edges?

A transformer can be seen as a special case of Graph Neural Networks [38]. Assuming a fully connected graph where vertices are tokens in a sequence, we can draw parallels between a GCN and learned attention in a transformer block.

Therefore, we examine attention weights between `NoteOn` tokens in an effort to understand how our graph representation of the score relates to the sequence-based representation. For all pairs of `NoteOn` tokens from music sequences, we output their attention values and compute the correlation with the aggregated adjacency matrix (with all musical edges constructed in Sec. 4.3.1). Across the test set of

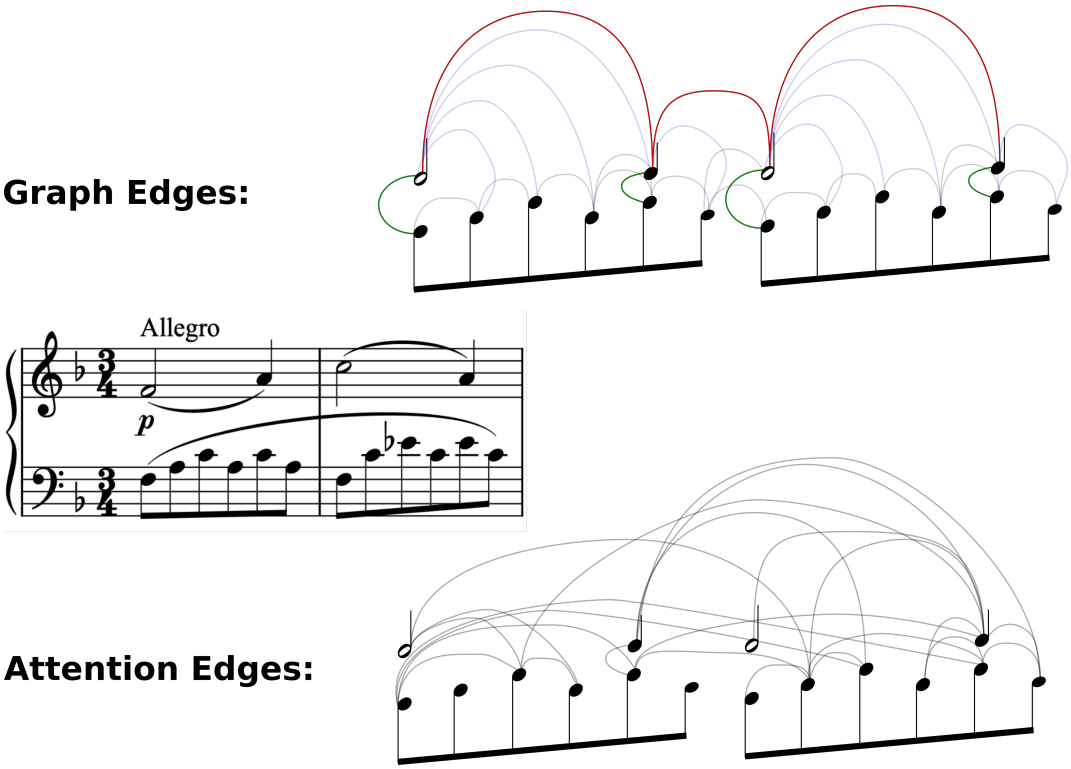


Figure 4.4: Visualization of graph edges (all edge types aggregated) and the attention among `NoteOn` tokens for the first measures of *Mozart Piano Sonata No.12, 1st mvt*.

ASAP composer classification on scores, there is a weak positive correlation, with Pearson's value of 0.212.

In Figure 4.4, we visualize two measures of music with its constructed graph edges, and the attention across `NoteOn` tokens. We can observe some structural similarities, especially the overlap pattern in both measures, but overall the learned attention spans are much more global while graph edges connect nodes within a local range.

4.5 DISCUSSION AND FUTURE WORK

In this paper, we presented a series of systematic experiments to investigate the impact of symbolic representations for three piece-level tasks. In terms of simple *classification performance*, we found that for a given task, different representations showed small performance differences, but no clear pattern of superiority emerged. The matrix results were marginally better on average, and usually more robust to hyper-parameter changes. More advanced features were beneficial only for certain tasks and representations.

The *graph representation*, as the newest and least explored among the three approaches, exhibits promising performance, while being more light-weight (in terms of required model complexity – cf. Fig. 4.3). We observe that homogeneous graphs produce comparable results to heterogeneous graphs for our piece-level classifi-

cation tasks, and deep GCNs perform better despite over-smoothing. As graphs are arguably a more natural representation for structured artifacts such as musical scores, we believe that they should merit more detailed studies in the future.

Our model complexity experiments demonstrated that commonly used architectures in the literature are larger than necessary for our tasks, as the same results can be achieved with smaller architectures (Section 4.4.2). Furthermore, we discussed the *album effect* in score-performance datasets, where multiple interpretations of the same composition may cause information leakage. Our results indicate the profound impact of the album effect, and we introduce new evaluation splits to guard against this effect.

4.6 ACKNOWLEDGEMENTS

This work is supported by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, funded by UK Research and Innovation [grant number EP/S022694/1], also by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme, grant agreement No. 101019375 (*Whither Music?*).

REFERENCES

[1] Lele Liu, Qiuqiang Kong, GV Morfi, Emmanouil Benetos, et al. "Performance MIDI-to-score conversion by neural beat tracking." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[2] David Temperley. "A Unified Probabilistic Model for Polyphonic Music Analysis." In: *Journal of New Music Research* 38.1 (2009), pp. 3–18. DOI: 10.1080/09298210902928495. eprint: https://doi.org/10.1080/09298210902928495. URL: https://doi.org/10.1080/09298210902928495.

[3] David Temperley. *The Cognition of Basic Musical Structures*. MIT Press, 2004.

[4] Huan Zhang, Jingjing Tang, Syed Rafee, Simon Dixon, and George Fazekas. "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[5] Silvan David Peter, Carlos Eduardo Cancino-Chacón, Francesco Foscarin, Andrew Philip McLeod, Florian Henkel, Emmanouil Karystinaios, and Gerhard Widmer. "Automatic Note-Level Score-to-Performance Alignments in the ASAP Dataset." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* (2023).

[6] Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter, Carlos Cancino-Chacón, Maarten Grachten, and Gerhard Widmer. "The match file format: Encoding Alignments between Scores and Performances." In: *Proceedings of the Music Encoding Conference (MEC)*. Halifax, Canada, 2022.

[7] Iannis Xenakis. *Formalized Music: Thoughts and Mathematics in Composition*. 1992. ISBN: 0-945193-01-7.

[8] Mitch Harris, Alan Smaill, and Geraint Wiggins. "Representing Music Symbolically." In: *IX Colloquio di Informatica Musicale (Venice)*. 1991. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.473.

[9] Milton Babbit. "The Use of Computers in Musicological Research." In: *Perspectives of New Music* 3.2 (1965), pp. 74–83.

[10] Geraint Wiggins, Eduardo Miranda, Alan Smaill, and Mitch Harris. "A Framework for the Evaluation of Music Representation Systems." In: *Computer Music Journal* 17.3 (1993), pp. 31–42. URL: https://about.jstor.org/terms.

[11] Christian Walder. "Modelling symbolic music: Beyond the piano roll." In: *Journal of Machine Learning Research*. Vol. 63. 2016, pp. 174–189. arXiv: 1606.01368.

[12] Mathieu Prang. "Representation learning for symbolic music." PhD thesis. IRCAM, 2021. URL: https://hal.archives-ouvertes.fr/tel-03329980.

[13] Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. "Score-informed transcription for automatic piano tutoring." In: *European Signal Processing Conference (EUSIPCO)*. 2012.

[14] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution piano transcription with pedals by regressing onset and offset times." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3707–3717.

[15] Youssef Ghatas, Magda Fayek, and Mayada Hadhoud. "A hybrid deep learning approach for musical difficulty estimation of piano symbolic music." In: *Alexandria Engineering Journal* 61.12 (2022), pp. 10183–10196. ISSN: 11100168. DOI: 10.1016/j.aej.2022.03.060.

[16] Sunghyeon Kim, Hyeyoon Lee, Sunjong Park, Jinho Lee, and Keunwoo Choi. "Deep Composer Classification Using Symbolic Representation." In: *International Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD)*. 2020. arXiv: 2010.00823. URL: http://arxiv.org/abs/2010.00823.

[17] Gissel Velarde, Tillman Weyde, Carlos E. Cancino-Chacón, David Meredith, and Maarten Grachten. "Composer recognition based on 2D-filtered piano-rolls." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[18] Francesco Foscarin, Katharina Hoedt, Verena Praher, Arthur Flexer, and Gerhard Widmer. "Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022. URL: https://api.semanticscholar.org/CorpusID:251881711.

[19] Hao Wen Dong, Wen Yi Hsiao, Li Chia Yang, and Yi Hsuan Yang. "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2018. ISBN: 9781577358008. DOI: 10.1609/aaai.v32i1.11312. arXiv: 1709.06298. URL: https://salu133445.github.io/musegan/.

[20] Sam van Herwaarden, Maarten Grachten, W de Haas, and W. Bas de Haas. "Predicting expressive dynamics in piano performances using neural networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[21] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – A Survey*. 2017. ISBN: 9783319701622. arXiv: 1709.01620. URL: http://arxiv.org/abs/1709.01620.

[22] Darrell Conklin and Ian H. Witten. "Multiple viewpoint systems for music prediction." In: *Journal of New Music Research* 24.1 (1995), pp. 51–73. DOI: 10.1080/09298219508570672. eprint: https://doi.org/10.1080/09298219508570672. URL: https://doi.org/10.1080/09298219508570672.

[23] Darrell Conklin. "Multiple Viewpoint Systems for Music Classification." In: *Journal of New Music Research* 42.1 (2013), pp. 19–26. DOI: 10.1080/09298215.2013.776611. eprint: https://doi.org/10.1080/09298215.2013.776611. URL: https://doi.org/10.1080/09298215.2013.776611.

[24] Raymond P. Whorley and Darrell Conklin. "Music Generation from Statistical Models of Harmony." In: *Journal of New Music Research* 45.2 (2016), pp. 160–183. DOI: 10.1080/09298215.2016.1173708. eprint: https://doi.org/10.1080/09298215.2016.1173708. URL: https://doi.org/10.1080/09298215.2016.1173708.

[25] Darrell Conklin. "Chord sequence generation with semiotic patterns." In: *Journal of Mathematics and Music* 10.2 (2016), pp. 92–106. DOI: 10.1080/17459737.2016.1188172. eprint: https://doi.org/10.1080/17459737.2016.1188172. URL: https://doi.org/10.1080/17459737.2016.1188172.

[26] Marcus T. Pearce. "Statistical learning and probabilistic prediction in music cognition: Mechanisms of stylistic enculturation." In: *Annals of the New York Academy of Sciences* 1423.1 (2018), pp. 378–395.

[27] Marcus Pearce. "The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition." PhD thesis. UK: City University of London, 2005.

[28] Darrell Conklin and Ian H. Witten. "Multiple Viewpoint Systems for Music Prediction." In: *Journal of New Music Research* 24.1 (1995), pp. 51–73. ISSN: 17445027. DOI: 10.1080/09298219508570672.

[29] Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. "Symphony Generation with Permutation Invariant Language Model." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[30] Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023.

[31] Mikaela Keller, Gabriel Loiseau, and Louis Bigo. "What Musical Knowledge Does Self-Attention Learn?" In: *Proceedings of the 2nd Workshop on NLP for Music and Spoken Audio (NLP4MusA)*. 2021, pp. 6–10. URL: https://aclanthology.org/2021.nlp4musa-1.2.

[32] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie Yan Liu. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. 2021. ISBN: 9781954085541. DOI: 10.18653/v1/2021.findings-acl.70. arXiv: 2106.05630.

[33] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "Byte Pair Encoding for Symbolic Music." In: 2023. arXiv: 2301.11975. URL: http://arxiv.org/abs/2301.11975.

[34] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[35] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[36] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. "A survey of heterogeneous information network analysis." In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 17–37.

[37] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. "DeepGCNs: Can GCNs go as deep as CNNs?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00936. arXiv: 1904.03751. URL: https://sites.google.com/view/deep-gcns.

[38] Petar Veličković. "Everything is Connected: Graph Neural Networks." In: *Artificial Intelligence (AI) Methodology in Structural Biology* (2023). ISSN: 1879033X. DOI: 10.1016/j.sbi.2023.102538. arXiv: 2301.08210. URL: http://arxiv.org/abs/2301.08210.

[39] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. "Onsets and frames: Dual-objective piano transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018, pp. 50–57. ISBN: 9782954035123. DOI: 10.5281/zenodo.1492341. arXiv: 1710.11153.

[40] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. "This time with feeling: learning expressive musical performance." In: *Neural Computing and Applications* 32.4 (2018), pp. 955–967. ISSN: 14333058. DOI: 10.1007/s00521-018-3758-9. arXiv: 1808.03715.

[41] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. "Music Transformer." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019. arXiv: 1809.04281. URL: http://arxiv.org/abs/1809.04281.

[42] Yu Siang Huang and Yi Hsuan Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions." In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. arXiv: 2002.00212.

[43] Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2021. arXiv: 2101.02402v1.

[44] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "Miditok: a Python Package for Midi File Tokenization." In: *International Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD)*. 2021. ISBN: 9783319701622.

[45] Werner Goebl. "Melody lead in piano performance: Expressive device or artifact?" In: *The Journal of the Acoustical Society of America* 110 (2001), p. 641. DOI: 10.1121/1.1376133. URL: https://asa.scitation.org/doi/10.1121/1.1376133.

[46] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[47] Carlos E. Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. "Computational Models of Expressive Music Performance: A Comprehensive and Critical Review." In: *Frontiers in Digital Humanities* 5.October (2018), pp. 1–23. ISSN: 2297-2668. DOI: 10.3389/fdigh.2018.00025.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017. arXiv: 1706.03762.

[50] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[51] Arthur Flexer. "A closer look on artist filters for musical genre classification." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2007. ISBN: 9783854032182.

[52] Gianluca Micchi. "A neural network for composer classification." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) Late-Breading Demo (LBD)*. 2018.

[53] Qiuqiang Kong, Keunwoo Choi, and Yuxuan Wang. "Large-Scale MIDI-Based Composer Classification." In: *arXiv*. 2020. arXiv: arXiv:2010.14805v1.

5 CADENCE DETECTION

Title: Cadence Detection in Symbolic Classical Music using Graph Neural Networks.

Published: in Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR), Bengaluru India, 2022.

Authors: Emmanouil Karystinaios, Gerhard Widmer

Abstract: Cadences are complex structures that have been driving music from the beginning of contrapuntal polyphony until today. Detecting such structures is vital for numerous MIR tasks such as musicological analysis, key detection, or music segmentation. However, automatic cadence detection remains challenging mainly because it involves a combination of high-level musical elements like harmony, voice leading, and rhythm. In this work, we present a graph representation of symbolic scores as an intermediate means to solve the cadence detection task. We approach cadence detection as an imbalanced node classification problem using a Graph Convolutional Network. We obtain results that are roughly on par with the state of the art, and we present a model capable of making predictions at multiple levels of granularity, from individual notes to beats, thanks to the fine-grained, note-by-note representation. Moreover, our experiments suggest that graph convolution can learn non-local features that assist in cadence detection, freeing us from the need of having to devise specialized features that encode non-local context. We argue that this general approach to modeling musical scores and classification tasks has a number of potential advantages, beyond the specific recognition task presented here.

5.1 INTRODUCTION

Graph Neural Networks (GNNs) have recently seen staggering successes in various fields. The MIR community has also experienced the influence of GNNs, principally in the field of recommender systems [1]. However, other sub-branches of MIR could potentially enjoy the graph representation and the benefits of graph deep learning.

Modeling musical scores in all their complexity has been challenging, with many approaches resorting to piano rolls [2], note arrays [3], or custom descriptors [4]. In this paper, we present a new representation of the score as a homogeneous graph with note-wise features to model aspects of the score. We use this representation to address the cadence detection task using graph neural networks, treating the task as a node classification problem. More specifically, our contribution is two-fold: a simple graph representation of scores extended with local features, and a Graph

Convolutional Network (GCN) model to tackle heavily imbalanced classification tasks such as Cadence Detection. *Score modeling* itself has two aspects: (1) the construction of the graph, i.e., what are the nodes, and which connections do we define between them; and (2) the choice of score features, and how these relate to their respective graph nodes. The *classification model* is an adapted version of GraphSMOTE [5], a Graph Convolutional Network designed to deal with imbalanced classification problems, which we modified to deal with larger graphs and apply stochastic training. Henceforth, we call this model *Stochastic GraphSMOTE*. We employ this model on top of our score modeling with the intention of solving the Cadence Detection task.

The cadence detection setting is binary, i.e., there is a cadence (maybe of a specific type) or not. The current state of the art [4] uses an Support Vector Machine (SVM) classifier on a set of custom-designed cadence-specific features, based on three defined "cadence anchor points", and performs score/feature modeling and cadence classification at the level of beats. The model was tested on two annotated datasets: 24 Bach fugues and 42 Haydn string quartet expositions. Our new model proposed here will be shown to achieve comparable overall results; however, we will argue that it makes fewer task-related and musical assumptions, resulting in more general applicability. In particular, our empirical results suggest that by providing local features and applying a Graph Neural Network with neighbor convolution, we can learn nonlocal aspects that help improve prediction. This gives a more general approach for a variety of tasks where features are provided at the level of notes, but prediction may be note-wise, onset-wise, or beat-wise.

The rest of the paper is structured as follows. Section 5.2 discusses related work on cadence detection and music score modeling. Section 5.3 describes the score model and the graph construction from the score, section 5.4 introduces the corpora, and section 5.5 presents the proposed learning algorithm. Section 5.6 presents a series of three experiments and also takes a qualitative look at some examples. Finally, section 5.7 summarizes and concludes.

5.2 RELATED WORK

Graphs have emerged as a natural representation of music since the development of Tonnetz by Euler. Since then, there have been various proposals to use graph representations for addressing music analysis and MIR tasks. For instance (to name just two), [6] introduced relational *Klumpenhouwer networks* for music analysis, and [7] used *Tonnetz trajectories* for composer classification. One can distinguish between *heterogeneous* and *homogeneous* graphs [8]. Heterogeneous graphs may have multiple types of edges and nodes, while homogeneous graphs are simpler, containing only a single edge and node type. Recently, the creators of VirtuosoNet, a computational model for generating piano performances, used a heterogeneous graph representation of the score and trained their system using a Graph Neural Network [9]. However, in later publications, they reverted to a model without using graphs which achieved better performance [10]. In the present paper, we wish to show that a simple, homogeneous graph representation can form a natural and general basis for modeling a non-trivial music analysis task.

Automatic *cadence detection* is a challenging task. Although cadences are well established concepts, their definition or annotation in music can cause disagreements among musicologists. Previous work on automatic cadence detection has been done by [11] on Bach fugues and by [12] for a generalized classical music analysis system. A feature-based approach using standard Machine Learning classifiers is presented in [4] which represents the current state of the art. Recently, Sears and Widmer [13] highlighted the difficulty of detecting textbook voice leading schemata that occur near cadences in written music. However, to our knowledge, there exists no method employing deep learning models to solve the task.

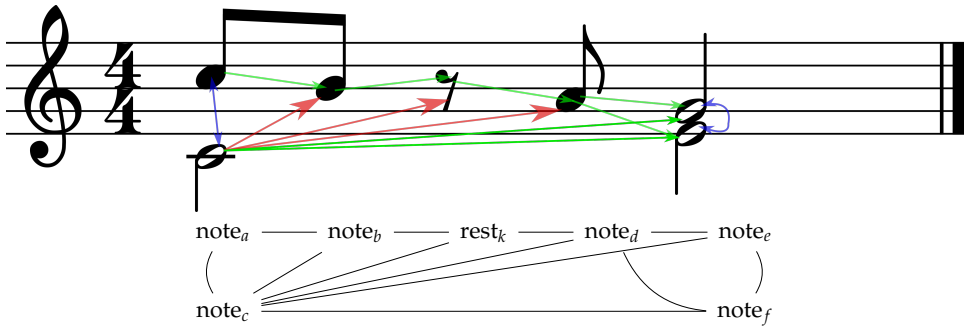


Figure 5.1: Example graph creation from a score following the process described in the text. E_{on} is denoted in blue, E_{cons} in green, and E_{dur} in red. Global attributes such as time and key signatures are added as node features.

5.3 MODELING SCORES AS A GRAPH

We model a score as a graph with individual notes and rests as nodes and simple temporal relations as edges. In addition, each graph node is associated with a vector of feature values that represent some basic properties of a note and its immediate context. Formally, let $G = (V, E)$ be a graph, where V is the set of nodes and $E \subseteq V \times V$ the set of edges and let A be the adjacency matrix of G. Each note and each rest in a score are represented as a node in the graph. We create three types of undirected connections between notes/rests: edges E_{on} between notes that occur on the same onset; edges E_{cons} between consecutive notes, and edges E_{dur} between a note of longer duration and notes whose onsets occur during this time:

$$\begin{aligned} E_{on} &= \{(i,j) \mid on(n_i) = on(n_j)\} \\ E_{cons} &= \{(i,j) \mid on(n_i) + dur(n_i) = on(n_j)\} \\ E_{dur} &= \{[(i,j) \mid on(n_i) + dur(n_i) > on(n_j)] \wedge \\ &\qquad [on(n_i) < on(n_j)]\} \\ E &= E_{on} \cup E_{cons} \cup E_{dur} \end{aligned}$$

where n_i is the i^{th} note. *on* denotes the onset of a note, *dur* the duration. All edges in E are undirected.

5.3.1 Feature Overview

We use three types of features to further describe a note:[1] general-purpose note-level features to describe a note and its immediate rhythmic/melodic context; general graph topology features to capture aspects of local connectivity; and cadence-specific note features inspired by [4]. The third feature category is the only one that is designed with the specific classification target in mind; however, in contrast to [4], we restrict these to only consider the immediate local context of a note instead of using positional features relating to predefined past “cadence anchor points". In this way, we wish to demonstrate the generality of our representation and learning approach, which will hopefully learn more long-distance aspects automatically, as needed.

The first category, *general note-wise features*, is the largest one. For each note in the score, we extract onset time expressed in score-relative beats, duration in beats, and MIDI pitch, using the partitura package [14]. Furthermore, we translate global attributes such as time signature and assign them to each note. Also using partitura, we extract a set of generic note-wise features as defined in [15]. Finally, we extract features summarizing intervallic information at the time of onset of each note. These include *interval vectors*[16] and binary features activated when intervallic content is identical to the interval set corresponding to particular chord types, i.e., major, minor, diminished, etc.

Second, we add *graph-aware features* using the first 20 eigenvectors from the Laplacian of the adjacency matrix [17].

The final category contains *note-wise cadence-related features* similar to those in [4], such as voice leading information and voicing. However, our features are calculated at the note level only, considering the time of onset for each note and its immediate neighbors, such as adjacent past onsets or simultaneous onsets. In particular, we do not use any information about events that occur on previous beats. While these features are more restricted compared to [4] they are also more general, since we make no assumptions on and reference to “cadence anchor points" (e.g., the occurrence of the preceding subdominant and dominant harmony), which in [4] are identified with specialized heuristics. In total, we store 135 features per node.

5.4 PROBLEM SETTING & CORPORA

In this work, we are interested in cadences of the Baroque and Classical periods. The main focus will be on detecting Perfect Authentic Cadences (PAC); where our annotated datasets permit, we will also consider root position Imperfect Authentic Cadences (rIAC) and Half Cadences (HC). The manual annotations in these datasets mark a cadence as occurring on the beat where the final I (i) arrives. Our precise task thus is to predict, for every note of the score, whether this note is contained in a cadence’s arrival beat.

To benchmark our method, we used two datasets also used by Bigo et al. [4], and a third one annotated by Allegraud and al. [18]. The first set contains the 24 fugues

1 Code and a complete specification of all features is available on https://github.com/manoskary/cadet.

Dataset	Pieces	Nodes	Edges	PAC	rIAC	HC
Bach Fugues	24	24,567	229,107	237	78	15
Haydn String Quartets	45	38,661	441,491	434	24	340
Mozart String Quartets	31	68,190	762,796	1,089	-	1,930

Table 5.1: Cadence nodes constitute less than 2% of all nodes.

from Bach's Well-tempered Clavier, Book I. The cadence annotations were presented in [11]. The second dataset contains 45 movement expositions from Haydn string quartets; the cadence annotations were produced by Sears and colleagues [19]. The last dataset contains 31 movements of Mozart string quartets with cadence annotations included. All the scores were retrieved from `http://kern.ccarh.org` and were parsed in python using the partitura package [14].[2]

Cadences occur with low frequency in music. In particular, for the corpora we cover in this paper, cadences of all types combined account for less than 2% of the total notes in the score. Our produced score graphs range from approximately 25k nodes for the Bach fugues all the way to 70k nodes for the Mozart string quartets with more than 750k edges. Table 5.1 gives detailed dataset statistics.

5.5 MODEL

5.5.1 Graph Convolutional Network

The authors of [4] underline the importance of non-fixed positions for the cadence anchor points. We address this by employing a graph convolutional network. Graph Convolution Networks (GCNs) are based on the same principle as CNNs, but in the context of graphs we encounter the message passing concept, meaning convolution occurs only among nodes connected by edges. This theoretically allows local features to connect with distant features of their k-hop neighbors. Therefore, graph representation can learn, using local node information, higher lever information by sampling information from neighbors. Figure 5.2 illustrates the neighbor sampling concept.

For our model, we propose *Stochastic GraphSMOTE*, a Graph Convolutional Network with a built-in graph Auto-Encoder and Synthetic Minority Over-sampling for imbalanced node classification. The model consists of 4 parts, the encoder, a SMOTE layer in the encoder's latent space, the decoder, and the classifier. The structure of the model follows GraphSMOTE [5] but with some major differences, mainly to adapt for stochastic training, which is needed because of the large size of our score graphs.

2 For reproducibility, we provide the generated graphs that were used for training on `https://github.com/manoskary/tonnetzcad`

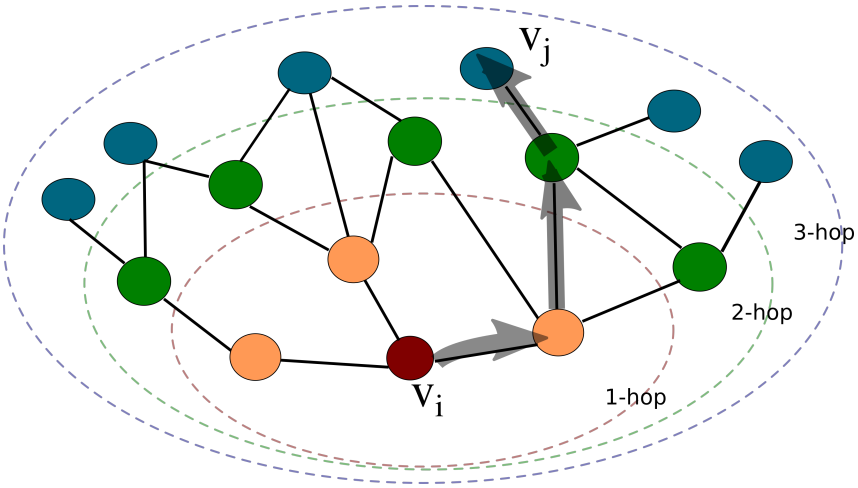


Figure 5.2: Multi-hop Neighborhood sampling. v_j is 3-hop neighbor of v_i. Color cues mark the k-hop neighborhoods occurring within the ellipses. The arrows demonstrate a random walk starting from v_i and ending at v_j.

The encoder applied to a node i is defined as a standard GraphSAGE [20] stack given by:

$$\mathbf{h}_{\mathcal{N}(i)}^{(l+1)} = \text{mean}\left(\{\mathbf{W}_{pool}^{(l+1)} \cdot \mathbf{h}_j^l, \forall j \in \mathcal{N}(i)\}\right)$$
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_{enc}^{(l+1)} \cdot \text{concat}(\mathbf{h}_i^l, \mathbf{h}_{\mathcal{N}(i)}^{l+1})\right)$$
$$\mathbf{h}_i^{(l+1)} = \text{norm}(\mathbf{h}_i^l)$$

where $h_i^{(l)}$ is the hidden representation of node i on layer l, σ is an activation function, norm is a normalization function, $\mathbf{W}$ are learnable weights, and $\mathcal{N}(i) = \{j \mid (i,j) \in E\}$ are the neighbors of node i. Let $B \subseteq V$ a subset of nodes denoting a batch sample. Then, given L the total number of hidden layers, $\mathbf{H}_B^{(enc)} = \{\mathbf{h}_u^{(L+1)} \mid u \in B\}$.

5.5.2 Dealing with Extreme Class Imbalance: Stochastic GraphSMOTE

Since cadences are very sparse, we need to introduce a balancing technique in order to avoid gradient convergence that will result in predicting only the majority class, i.e., absence of cadence. To counter this effect, we introduce a SMOTE layer that is applied in the *latent space* of the encoder. SMOTE generates synthetic samples with the same label as the minority class (see [21] for details). The main novelty of our model is that the SMOTE is performed for each batch separately.

In each batch, we count the occurrence, μ_i, for each of the classes $i \in I$. In the binary setting, let μ_M be the number of samples with the same label as the majority class and μ_m be the number of samples with the same labels as the minority class. By generating $(\mu_M - \mu_m)$ samples with the same label as the minority class, we force a $1:1$ binary class distribution. To generate these samples, in each batch, we randomly select a sample instance of the minority class as an anchor point and gather the k nearest neighbor samples of the same class within the batch. Finally,

μ samples are generated as random linear interpolations between a randomly selected neighbor out of the k, and the selected anchor point in the euclidean space. Performing SMOTE in the latent space assumes that a more appropriate representation for the generation of the synthetic minority samples is learned.

If $\mathbf{H}_B^{(enc)}$ is the hidden representation of the batch sampled nodes after the encoder layer, then $\mathbf{H}_B^{(smote)}$ is the SMOTE upsampling algorithm applied on $\mathbf{H}_B^{(enc)}$. Our Decoder layer is responsible for generating edges within the original nodes of the graph and the synthetic ones, created by SMOTE. The decoder output is described by the following equation:

$$\mathbf{A}_B^{(dec)} = \sigma\left(\mathbf{H}_B^{(smote)} \cdot \mathbf{W}^{(dec)} \cdot \text{transpose}(\mathbf{H}_B^{(smote)})\right)$$
$$\mathbf{A}_B^{(thr)} = \text{hardshrink}\left(\mathbf{A}_B^{(dec)}, \tau\right)$$

where $W^{(dec)}$ are the decoder's learnable weights, σ is a sigmoid activation function, and hardshrink is the hard shrinkage function with threshold τ. $\mathbf{A}_B^{(dec)}$ is the generated adjacency from the decoder and $\mathbf{A}_B^{(thr)}$ is a thresholded adjacency by a factor τ.

We define a regularization loss that aims at constraining the generated adjacency close to the original, defined by:

$$\mathcal{L}_B^{(dec)} = \text{BCE}\left(\mathbf{A}_B^{(dec)}, \mathbf{A}_B\right)$$

where BCE is the binary cross entropy loss, $\mathbf{A}_B^{(dec)}$ is the generated adjacency of the decoder for batch sample B and A_B is the adjacency matrix for batch sample B. Since we learn an edge generator which is good at reconstructing the adjacency matrix using the encoder's latent representations, it should also give adequate edge predictions for synthetic nodes.

The GNN classifier is composed of a GraphSAGE layer [20] with a linear layer on top. By adding a graph convolution layer such as GraphSAGE in the classifier, we can benefit from learning information from the generated adjacency and the neighbors of nodes. The GraphSAGE layer of the classifier is slightly different from the encoder because it performs directly on the generated thresholded adjacency of each batch sample:

$$\mathbf{h}_{\mathcal{N}(i)}^{(clf)} = \text{mean}\left(\mathbf{W}^{(pool)} \cdot \mathbf{A}_B^{(thr)}[i,:] \cdot \mathbf{H}_B^{(enc)}\right)$$
$$\mathbf{h}_i^{(clf)} = \text{norm}\left(\sigma\left(\mathbf{W}^{(clf)} \cdot \text{concat}\left(\mathbf{h}_i^{(enc)}, \mathbf{h}_{\mathcal{N}(i)}^{(clf)}\right)\right)\right)$$
$$\mathbf{h}_i^{(clf)} = \text{softmax}(\mathbf{W}^{(proj)} \cdot \mathbf{h}_i^{(clf)})$$

where $\mathbf{h}_i^{(clf)}$ are the predicted class probabilities of node i, $\mathbf{W}$ are learnable weights, $\mathbf{A}_B^{(thr)}$ is the generated thresholded adjacency from the decoder, $\mathbf{H}_B^{(enc)}$ are the batch encodings of the encoder and $\mathbf{h}_i^{(enc)}$ is the encoder's output for node i. During

training, we use $\mathbf{H}_B^{(smote)}$ and $\mathbf{h}_i^{(smote)}$ respectively instead of $\mathbf{H}_B^{(enc)}$ and $\mathbf{h}_i^{(enc)}$. We define the *total loss* of our model for batch samples B:

$$\mathcal{L}_B^{(tot)} = \mathcal{L}_B^{(CE)} + \gamma * \mathcal{L}_B^{(dec)}$$

where $\mathcal{L}_{CE}$ signifies the cross entropy loss and γ is a hyper parameter.

Our model is trained stochastically, meaning that to create each batch a subset B of nodes are sampled. From these sampled nodes, given a pre-defined depth k, we retrieve the immediate neighbors of every $v \in B$ up to their k-hop neighbors in the graph G. We use neighbor sampling to reduce the cost of retrieving all up to k-hop neighbors of v by defining a maximum number ϕ_l of neighbors per depth layer l.

5.6 EXPERIMENTS

We conduct three main experiments. The first compares our model to the state of the art results in [4], using the same data and train/test setup. The second experiment focuses on multi-class learning of the particular type of cadence using different sets of features, in order to investigate how the model generalizes to a more complex setting and inspect the relevance of different feature sets. The third experiment investigates how neighbor convolution contributes to the model's performance.[3]

We fix our model with a hidden dimension of 256, with $L = 2$ hidden layers with $\phi_1 = 10$ and $\phi_2 = 25$ sampled neighbors for hidden layers 1 and 2 of the encoder, respectively, and one hidden layer of the same dimension for the classifier. The learning rate is set at 0.007, the weight-decay at 0.007, with a batch size of 1024, $k = 3$ for SMOTE, the decoder regularization loss multiplier $\gamma = 0.5$, and adjacency threshold value $\tau = 0.5$.

5.6.1 Quantitative Results

Table 5.2 summarizes the results of the first experiment, comparing our model's performance to the state of the art.[4] The reference model [4] can only classify at the beat level; our representation and classification model are more flexible in this regard, as they have access to, and describe, individual notes. In particular, our model can provide predictions at three different levels, note-wise, onset-wise and beat-wise predictions (the latter two simply by aggregation). In Table 5.2 we present the results of these predictions at all levels, in terms of F1 score. Only beat-wise scores are given for the reference model (taken from [4]). The last two columns of table 5.2 give the recall and precision for the beat-wise prediction. All metrics are presented for the positive, i.e. minority/cadence, class.

Our model matches or slightly surpasses the state of the art in rIAC detection in Bach fugues and on HCs in Haydn string quartets but does not reach the reference model's F1 results in PAC detection. We additionally present a pre-trained version of Stochastic GraphSMOTE, where the network was first trained on additional data

3 All results, experiments, and the trained models are available on https://wandb.ai/melkisedeath/CadenceDetection

4 In accordance with [4], we ignore the HC in Bach and rIAC in Haydn, because of their low numbers.

Dataset	Model	F1 Note	F1 Onset	F1 Beat	Prec. Beat	Recall Beat
Bach Fugues (PAC) (12 fugues)	Bigo et al. model	-	-	**0.80**	**0.89**	0.72
	SGSMOTE	0.85	0.75	0.73	0.70	0.77
	Pretrained SGSMOTE	**0.90**	**0.83**	**0.80**	0.74	**0.89**
Bach Fugues (rIAC) (12 fugues)	Bigo et al. model	-	-	0.68	0.71	0.65
	SGSMOTE	**0.87**	**0.75**	**0.73**	**0.75**	0.72
	Pretrained SGSMOTE	0.87	0.73	0.71	0.62	**0.82**
Haydn String Quartets (PAC) (21 pieces)	Bigo et al. model	-	-	**0.69**	**0.60**	**0.82**
	SGSMOTE	0.77	0.56	0.59	0.47	0.78
	Pretrained SGSMOTE	**0.81**	**0.63**	0.64	0.54	0.78
Haydn String Quartets (HC) (21 pieces)	Bigo et al. model	-	-	0.29	0.19	**0.56**
	SGSMOTE	0.65	0.32	0.30	0.33	0.27
	Pretrained SGSMOTE	**0.69**	**0.44**	**0.41**	**0.41**	0.41

Table 5.2: Results using half of the dataset for training, half for testing. Bach: fugues no.1-12 were used for training, no.13-24 for testing; Haydn: random 21:21 split. The pretrained network was trained on the other dataset, i.e. *Pretrained SGSMOTE* for Bach Fugues was pre-trained on string quartets, etc. Classification is binary, the presented $F1$ scores are for the positive class, i.e., the cadence (PAC: Perfect Authentic Cadence; rIAC: root position Imperfect AC; HC: Half Cadence).

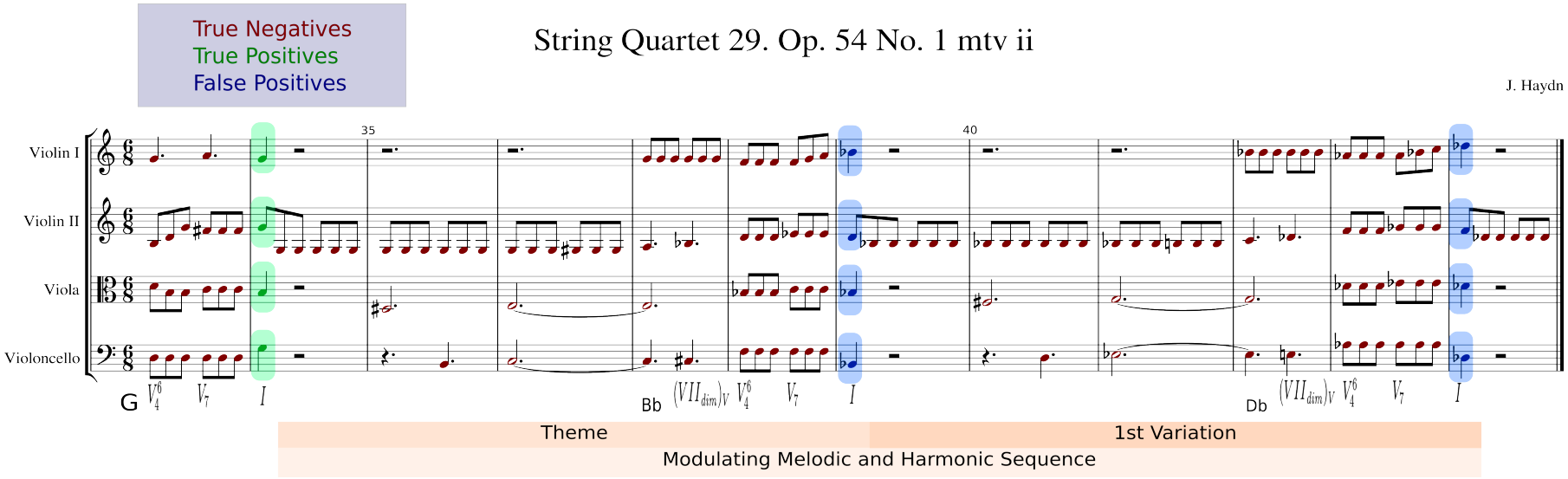


Figure 5.3: Haydn's String Quartet 29. Op.54 No.1 Mvt. II, mm. 33-45. Showing the output of the Stochastic GraphSMOTE Network for PAC prediction. True negatives are marked with red, true positives with green, false positives with blue. A partial analysis shows the chords towards the end of cadences and highlights a modulating sequence where every sequence ends with a cadential pattern, which counts as false positive predictions by the network.

Depth	F1 Note	F1 Onset	F1 Beat
None	0.833	0.671	0.667
1-hop	0.854	0.707	0.701
2-hop	**0.869**	**0.737**	**0.732**
3-hop	0.836	0.706	0.659

Table 5.3: Effect of neighbor convolution depth on PAC prediction in Bach fugues. The F1 Note/Onset/Beat scores presented are binary, i.e., for the PAC class. Depth refers to neighbor convolution depth. *None* means no graph convolution.

Dataset	Features	F1 Note	F1 Beat
Bach Fugues	general	0.602	0.667
(PAC & rIAC)	all	0.653	0.702
Haydn String Quart.	general	0.542	0.610
(PAC & HC)	all	0.648	0.663
Mozart String Quart.	general	0.584	0.569
(PAC & HC)	all	0.588	0.606

Table 5.4: Three-class cadence classification with two different feature sets. Results were obtained by 5 fold cross validation (70% of pieces for training, 10% validation, 20% testing); no pre-training. Feature set *all* contains all features from Section 5.3.1; *general* excludes Category 3 cadence-specific engineered features.

and fine-tuned for the task. Specifically, the network for PAC prediction in Bach was pre-trained on the string quartets and vice versa. Pre-training, and thus the need for additional data, is the price we pay for the generality of the graph representation and the consequent size (number of parameters) of the deep network. Pre-training helps to (markedly) improve the results on HC, catch up with the reference on PAC in Bach, and narrow the gap on PAC in Haydn.

Generally, our results agree with [4] in implying that half cadences (HC) seem significantly harder to identify than authentic cadences, both perfect and imperfect. Another, more specific, observation concerns different ways in which the compared models achieve their overall F1 scores. In the PAC detection tasks, in particular, we observe comparable or higher recall of our model compared to the reference, but lower precision. This observation motivated us to check some of our model's false positive predictions; Section 5.6.2 below will show several instructive examples of 'almost correct' identifications.

The *second experiment* we conducted (Table 5.4) focuses on comparing the relevance of feature groups. For compactness we present here a multi-class classification scenario where we account not only for the existence of a cadence but also for the type of cadence present; that is, we have tree-class problems: no cadence, PAC, or rIAC (Bach) / HC (Haydn, Mozart). We compare two configurations: using all available features (as in the first experiment, feature set *all* in the table), or only feature sets 1 and 2, excluding the cadence-specific features (category 3 in Section 5.3.1; marked *general* in the table). Given this 3-class setting, we chose to report the macro averaged F1 score over all three classes. (Macro averaging was chosen to counter the overwhelming effect of the majority class *no cadence*). The

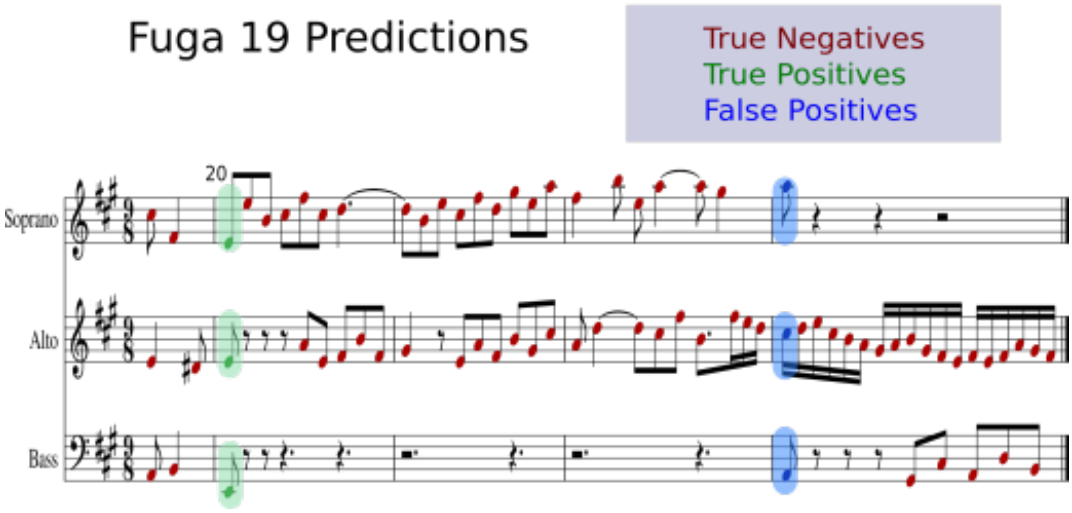


Figure 5.4: Predictions of Stochastic GraphSMOTE for fugue No.19, J.S.Bach, Well-tempered Clavier.

results (see Table 5.4) support the relevance of carefully devised cadence-related features à la [4]. However, also the general-purpose category 1 and 2 features alone support non-trivial cadence recognition and discrimination performance, which implies that the relational graph representation in combination with a convolutional approach manages to enrich highly local features with relevant non-local score context.

To investigate this latter aspect in more detail, we run a *third experiment*, to look at the effect of neighbor convolution depth on the obtainable classification score, again at three prediction granularity levels (note, onset, beat). Convolution depth refers to the number l of hidden layers of the encoder and the subsequent neighbor sampling up to l-hop neighbors. Our results (see Table 5.3) suggest that neighbor convolution clearly contributes to learning non-local features. Best results are achieved when using a convolution depth of 2. Increasing the receptive field beyond that level, we observed some instabilities emerging in the learning model, which could be attributed to the common vanishing gradient problem in deep GCNs [22].

5.6.2 A Qualitative Look

Motivated by the fact that our model, while higher on recall, seems to be lower on precision than the model in [4], we take a closer look at some of the false positives in individual examples. Our findings suggest that many false positive predictions resemble cadences, in terms of tonal structure or implications, and could be considered and annotated as such, but lack some main components.

Figure 5.4 shows an example. The cadence prediction by our model on the downbeat of bar 23 is a false positive, according to the ground truth annotation. However, one could argue that the passage clearly has a cadence-like role, marking the end of the 2nd fugal episode and the return to the original tonality of A major [23].

Another example is the passage discussed in Fig.4 of [4], where a pattern occurs that has all the technical ingredients of a PAC, but was not annotated as such for (debatable) higher-level musicological considerations. Again, our model's PAC prediction there counts as a false positive.

As a final example, consider mm. 33-45 of Haydn's Op.54 No.1, 2nd mvt (Figure 5.3). We observe two false positive beat-wise predictions (8 if we count note-wise) in bars 39 and 44, respectively, following a true PAC on the beginning of bar 34. A harmonic analysis of these bars indicates a proper PAC preparation with text-book voice leading on the cadence arrival point in every occasion. These two false positive PACs form part of a modulating melodic and harmonic sequence; whether to classify them as cadences is a matter of higher-level musicological considerations.

We cite these few qualitative examples in an attempt to show that our prediction model can identify many more cadential patterns than the raw experimental figures suggest, but by design cannot consider high-level musical considerations such as, e.g., whether PAC-like patterns that occur in sequence should count as PACs or not.

5.7 CONCLUSION

We have presented a graph approach to effectively target the cadence detection task on symbolic classical scores. We demonstrated that our Graph Convolutional Network, Stochastic GraphSMOTE, can learn using only local note features, without the need for any musical assumptions about cadence anchor points. Furthermore, our network can produce fine-grained predictions at the level of individual notes.

Future work will address the performance of the model on different tasks, using the same graph representation. We hope to be able to show that this simple but general and natural representation of scores in terms of graphs can support a broad variety of symbolic music analysis and classification tasks.

5.8 ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 ("Whither Music?"), and the Federal State of Upper Austria (LIT AI Lab). The authors would like to thank Dr. Hamid Eghbal-Zadeh for helpful discussions on Graph Neural Networks.

REFERENCES

[1] Filip Korzeniowski, Sergio Oramas, and Fabien Gouyon. "Artist Similarity with Graph Neural Networks." In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. 2021.

[2] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. "The Bach doodle: Approachable music composition with machine learning at scale." In:

Proceedings of the 18th International Society for Music Information Retrieval Conference. 2019.

[3] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling factorized piano music modeling and generation with the MAESTRO dataset." In: *Proceedings of 7th International Conference on Learning Representations*. 2019.

[4] Louis Bigo, Laurent Feisthauer, Mathieu Giraud, and Florence Levé. "Relevance of musical features for cadence detection." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[5] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. "GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks." In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021.

[6] Alexandre Popoff, Moreno Andreatta, and Andrée Ehresmann. "Relational poly-Klumpenhouwer networks for transformational and voice-leading analysis." In: *Journal of Mathematics and Music* 12.1 (2018).

[7] Emmanouil Karystinaios, Corentin Guichaoua, Moreno Andreatta, Louis Bigo, and Isabelle Bloch. "Music Genre Descriptor for Classification Based on Tonnetz Trajectories." In: *Proceedings of Journées Informatiques Musicales*. 2021.

[8] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. "A survey of heterogeneous information network analysis." In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 17–37.

[9] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[10] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam. "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance." In: *Proceedings of the 20th International Society of Music Information Retrieval Conference*. 2019.

[11] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational fugue analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96.

[12] Plácido R Illescas, David Rizo, and José Manuel Inesta Quereda. "Harmonic, melodic, and functional automatic analysis." In: *Proceedings of the International Computer Music Conference*. 2007.

[13] David RW Sears and Gerhard Widmer. "Beneath (or beyond) the surface: Discovering voice-leading patterns with skip-grams." In: *Journal of Mathematics and Music* 15.3 (2021).

[14] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[15] Carlos Eduardo Cancino Chacón. "Computational Modeling of Expressive Music Performance with Linear and Non-linear Basis Function Models." PhD thesis. Johannes Kepler University, Austria, 2018.

[16] Michiel Schuijer. *Analyzing atonal music: Pitch-class set theory and its contexts*. University Rochester Press, 2008.

[17] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. "Benchmarking graph neural networks." In: *arXiv preprint arXiv:2003.00982* (2020).

[18] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[19] David RW Sears, Marcus T Pearce, William E Caplin, and Stephen McAdams. "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models." In: *Journal of New Music Research* 47.1 (2018), pp. 29–52.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[21] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[22] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. "DeepGCNs: Can GCNs go as deep as CNNs?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00936. arXiv: 1904.03751. URL: https://sites.google.com/view/deep-gcns.

[23] *Bach: Prelude and Fugue No.19 in A major, BWV 864 Analysis*. 2018. URL: https://tonic-chord.com/bach-prelude-and-fugue-no-19-in-a-major-bwv-864-analysis/.

6 ROMAN NUMERAL ANALYSIS

Title: Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features.

Published In Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR), Milano Italy, 2023.

Authors: Emmanouil Karystinaios, Gerhard Widmer

Abstract: Roman Numeral analysis is the important task of identifying chords and their functional context in pieces of tonal music. This paper presents a new approach to automatic Roman Numeral analysis in symbolic music. While existing techniques rely on an intermediate lossy representation of the score, we propose a new method based on Graph Neural Networks (GNNs) that enable the direct description and processing of each individual note in the score. The proposed architecture can leverage notewise features and interdependencies between notes but yield onset-wise representation by virtue of our novel edge contraction algorithm. Our results demonstrate that *ChordGNN* outperforms existing state-of-the-art models, achieving higher accuracy in Roman Numeral analysis on the reference datasets. In addition, we investigate variants of our model using proposed techniques such as NADE, and post-processing of the chord predictions. The full source code for this work is available at https://github.com/manoskary/chordgnn

6.1 INTRODUCTION

Automatic Chord Recognition is one of the core problems in Music Information Retrieval. The task consists of identifying the harmonies or chords present in a musical piece. Various methods have been proposed to address this task using either an audio or symbolic representation of the music [1]. In the symbolic domain, most approaches focus on the related and arguably more complex problem of Automatic Roman Numeral Analysis, which is a functional harmony analysis problem that has its roots in musicological research of Western classical music.

Roman Numeral Analysis is a notational system used in music theory to analyze chord progressions and identify the relationship between chords in a given key. In this system, each chord in a piece of music is assigned a Roman numeral based on its position within the key's scale. For example, in the key of C major, the I chord is C major, the IV chord is F major, and the V chord is G major. Roman Numerals are an important tool for understanding and analyzing the harmonic structure of music, and they are a valuable resource for musicians, composers, and arrangers alike.

In Music Information Retrieval, a lot of work has been done to automate Roman Numeral analysis. However, current approaches still face significant challenges. Some of these are related to the large chord symbol vocabulary. A common way to address this problem is to divide a Roman Numeral into several components (e.g., key, degree, inversion) and transform the analysis into a multitask learning scenario. However, multitask approaches themselves face challenges with interdependencies among tasks. Lastly, Roman Numeral analysis faces a score representation problem related to existing models such as CNNs whose inputs must be in fixed-sized chunks. Recent state-of-the-art approaches follow an audio-inspired strategy, dividing a musical score into fixed-length time frames ("windows") which are then processed by a Convolutional Recurrent Neural Network (CRNN). However, such a representation is unnatural for scores and has the added practical disadvantage of being time-limited (for example regarding notes extending beyond the current window) and, due to the fixed-length (in terms of score time) constraint, capturing varying amounts of musically relevant context.

In this paper, we propose a new method for automatic Roman Numeral analysis based on Graph Neural Networks that can leverage note-wise information to address the score representation issue. Our model, *ChordGNN*, builds on top of existing multitask approaches but introduces several novel aspects, including a graph convolutional architecture with an edge contraction pooling layer that combines convolution at the note level but yields the learned representation at the onset level.

Our proposed method, *ChordGNN*, is evaluated on a large dataset of Western classical music, and the experimental results demonstrate that it outperforms existing state-of-the-art methods, in terms of the commonly used Chord Symbol Recall measure. To address the interdependencies among tasks we investigate the effect of post-processing and other proposed techniques such as NADE and gradient normalization. Finally, we look at a qualitative musical example and compare our model's predictions with other state-of-the-art models.

6.2 RELATED WORK

There is a big body of literature covering the topic of Automatic Chord Recognition applied in the audio domain; however, in our work, we focus on the problem of automatic Roman Numeral Analysis in the symbolic domain. It consists of labeling the chords and harmonic progressions in a piece of music using Roman Numerals, where each numeral represents a chord built on a particular scale degree. Numerous approaches have tried to automate Roman Numeral analysis or infer harmonic relations between chords. Notable work includes statistical models such as *Melisma* [2], HMM-based models [3], and grammar-based approaches [4].

In recent years, research has shifted towards a deep learning and data-driven approach. Due to the large vocabulary of possible Roman Numerals, the problem has been divided into several component subtasks, thus resulting in a multitask learning setting [5]. As a multitask problem, a Roman Numeral is characterized by the following components: the primary and secondary degree (as illustrated in Figure 6.1), the local key at the time point of prediction, the root of the chord, the

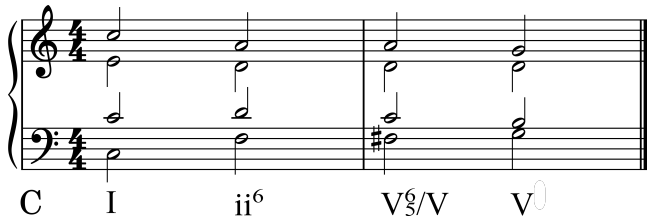


Figure 6.1: A Roman Numeral analysis for two bars for four-part harmony in *C* major. Capital letters stand for major quality and lowercase for minor quality. The third chord has a dominant seven as its primary degree and the dominant of *C* major as its secondary degree. The V_5^6 indicates a major with a seven quality in second inversion. The bass (lowest chord note) of that chord is *F* sharp, the root is *D*, and the local key is *C* major.

inversion of the chord, and the quality (such as major, minor, 7, etc.). Although the root can be derived from the other components, it was pointed out by [6] that redundancy is assisting Roman Numeral analysis systems to learn. An example of Roman Numerals and their components can be viewed in Figure 6.1. Recent state-of-the-art approaches decompose the numeral prediction task to the simultaneous prediction of those 6 components [5–9].

Most deep learning approaches to Roman Numeral analysis are inspired by work in audio classification, cutting a score into fixed-size chunks (in terms of some constant score time unit; e.g., a 32nd note) and using these as input to deep models. Using this quantized time frame representation, [6] introduced a CRNN architecture to predict Roman Numerals. Other work has continued to build on the latter by introducing more tasks to improve performance such as the *AugmentedNet* model [7], or introducing intra-dependent layers to inform in an orderly fashion the prediction of one task with the previously predicted task, such as the model introduced by [8]. Other architectures, such as the CSM-T model, have demonstrated good results by introducing modular networks which treat a score as a sequence of notes ordered first by onset and then by pitch[9].

Should a musicologist perform music analysis on a piece of music, they would consider the individual notes existing in the score. Thus, a time frame representation would come across as unnatural for symbolic music and in particular for such an analysis task. In this paper, we present a method that no longer treats the score as a series of quantized frames but rather as a partially ordered set of notes connected by the relations between them, i.e., a graph. A visual comparison of the two representations is shown in Figure 6.2. Recently, modeling scores as graphs has also been demonstrated to be beneficial for problems such as expressive performance generation [10], cadence detection [11], voice separation [12], or boundary detection [13].

Automatic Roman Numeral analysis, as a multitask problem, is mostly tackled with hard parameter-sharing models. These models share part of the model across all tasks as an encoder, and then the common embeddings are branched to a classification model per task [6–8]. However, some approaches separate tasks from this paradigm to a more modular or soft parameter sharing approach [9].

In the field of multitask learning, a lot of research has been done on the problem of conflicting gradients during backpropagation in hard parameter-sharing models.

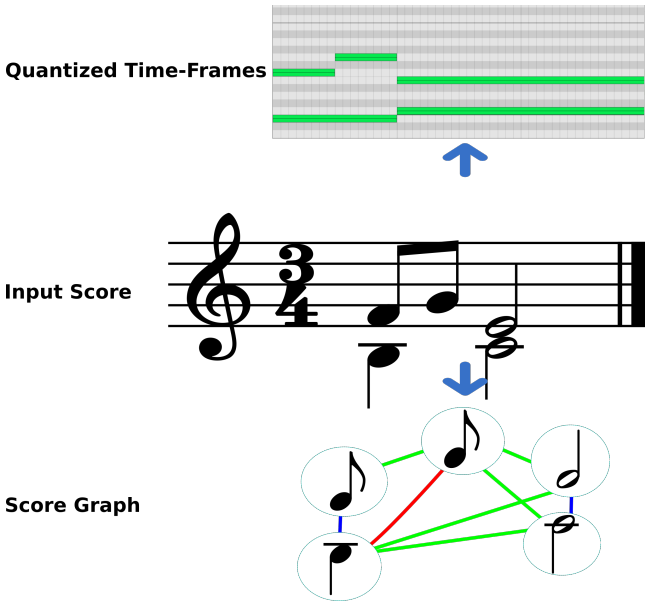


Figure 6.2: Different representations of the score excerpt shown in the middle. Top: quantized time frame representation, bottom: graph representation.

Issues with multi-objective optimization have been early addressed by Zhang et al. [14] and recent solutions have been proposed for the multitask setting in the form of dynamic task prioritization [15], gradient normalization [16], rotation matrices [17], or even game-theoretic approaches [18]. In our work, we experimentally evaluate some of these techniques in the multitask setting to investigate whether Roman Numeral analysis subtasks conflict with each other (see Section 6.5.2).

6.3 METHODOLOGY

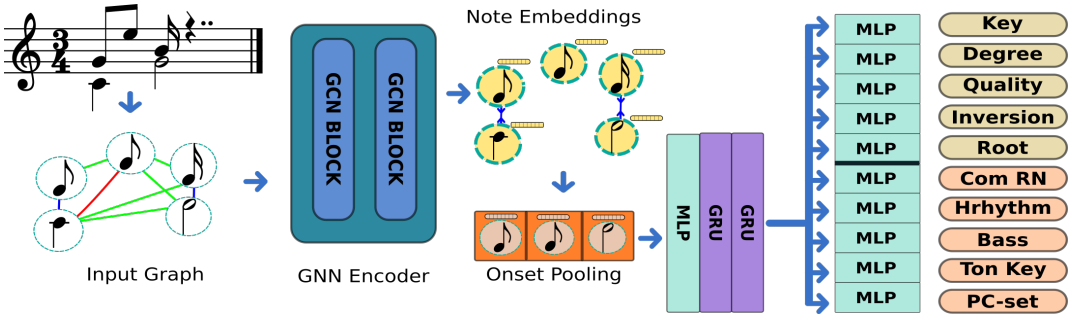


Figure 6.3: The proposed Architecture Chord-GNN

6.3.1 Roman Numeral Analysis

We already discussed, in Section 6.2, how Roman Numeral analysis can be viewed as a multi-task problem. In this section, we describe in detail the additional tasks introduced by [7] that we also use for training and prediction. First, let us assume

that the prediction can be broken down into specific time points, and each time point is attributed to a unique onset in the score.

The Roman Numeral prediction can be viewed as a simultaneous prediction of the local key, degree (primary and secondary), quality, inversion, and root. Each one of these tasks is a categorical, multiclass classification problem. However, [7] indicated that only three tasks would be sufficient for 98% of the Roman Numeral annotations in our dataset (detailed in Section 6.4.1). These three tasks comprise the prediction of a restricted vocabulary of common Roman Numeral symbols in combination with the local key and the inversion. We refer to Roman Numeral prediction involving the 5 tasks as *conventional RN*, and the combined prediction of key, inversion, and restricted RN vocabulary *alternative RN*, as RN_{alt}, in accordance with [7].

Several other tasks have been introduced that have been shown to improve the performance of related models [7]. These include the Harmonic Rhythm, which is used to infer the duration of a Roman Numeral at a given time point; the Tonicization task, a multiclass classification task that refers to a tonicized key implied by the Roman Numeral label and is complementary to the local key; the Pitch Class Sets task, which includes a vocabulary of different pitch class sets, and the Bass task, which aims to predict the lowest note in the Roman Numeral label.

6.3.2 Graph Representation of Scores

Our approach to automatic Roman Numeral analysis no longer treats the score as a sequence of quantized time frames but rather as a graph, which permits us to specify note-wise information such as pitch spelling, duration, and metrical position. We use graph convolution to model interdependencies between notes. We model our score generally following Karystinaios and Widmer [11], but we opt for a heterogeneous graph convolution approach, i.e., including different edge relations/types. Furthermore, we develop an edge contraction pooling layer that learns onset-wise representations from the note-wise embeddings and therefore yields a sequence.

After the edge contraction, we follow [6–8] by adding to the graph convolution a sequence model for the hard-sharing part of our model, and simple shallow multi-layer perceptron heads for each task. In essence, we replace the CNN encoder that works on quantized frames of the score in previous approaches, with a graph convolutional encoder followed by an edge contraction layer. Our proposed architecture is shown in Figure 6.3.

The input to the GNN encoder is an attributed graph $G = (V, E, X)$ where V and E denote its node and edge sets and X represents the node feature matrix, which contains the features of the notes in the score. For our model, we used pitch spelling, note duration, and metrical position features.

Given a musical piece, the graph-building process creates a set of edges E, with different relation types $\mathcal{R}$. A labeled edge (u, r, v) of type r between two notes u, v belongs to E if the following conditions are met:

- notes starting at the same time:
  $on(u) = on(v) \rightarrow r = \text{onset}$

- note starting while the other is sounding: $on(u) > on(v) \wedge on(u) \leq on(v) + dur(v) \rightarrow r = \text{during}$
- note starting when the other ends: $on(u) + dur(u) = on(v) \rightarrow r = \text{follow}$
- note starting after a time frame when no note is sounding: $on(u) + dur(u) < on(v) \wedge \nexists v' \in V,\ on(v') < on(v) \wedge on(v') > on(u) + dur(u) \rightarrow r = \text{silence}$

6.3.3 Model

In this section, we introduce and describe *ChordGNN*, a Graph Convolutional and Recurrent Neural Network. The structure of the network is visually outlined in Figure 6.3. *ChordGNN* uses heterogeneous graphSAGE [19] convolutional blocks defined as:

$$\begin{aligned} \mathbf{h}_{\mathcal{N}_r(v)}^{(l+1)} &= \text{mean}\left(\{\mathbf{h}_u^l, \forall u \in \mathcal{N}_r(v)\}\right) \\ \mathbf{h}_{v_r}^{(l+1)} &= \sigma\left(W \cdot \text{concat}(\mathbf{h}_v^l, \mathbf{h}_{\mathcal{N}_r(v)}^{l+1})\right) \\ \mathbf{h}_v^{(l+1)} &= \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathbf{h}_{v_r}^{(l+1)} \end{aligned} \tag{6.1}$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ and $\mathbf{x}_u$ is the input features for node u, $\mathcal{N}(u)$ are the neighbors of node u, and σ is a ReLU activation function. We name the output representations of all nodes after graphSAGE convolution $H = \{h_u^{(L)} \mid u \in V\}$ where L is the total number of convolutional layers.

Given the hidden representation H of all nodes, and onset edges $E_{\text{On}} = \{(u, v) \mid on(u) = on(v)\}$, the onset edge contraction pooling is described by the following equations: first, we update the hidden representation with a learned weight, $H' = HW^{(\text{cpool})}$. Subsequently we need to unify the representations for every node u, such that $\forall v \in \mathcal{N}_{\text{On}}(v),\ h_u^{(\text{cp})} = h_v^{(\text{cp})}$:

$$h_u^{(\text{cp})} = h_u + \sum_{v \in \mathcal{N}_{\text{On}}(v)} h_v \tag{6.2}$$

where, h_u and h_v belong to H'. Subsequently, we filter the vertices:

$$V' = \{v \in V \mid \forall u \in V,\ (v, u) \in E_{\text{On}} \implies u \notin V'\} \tag{6.3}$$

Therefore, $H^{(cp)} = \{h_u^{(cp)} \mid \forall u \in V'\}$ are the representations obtained. Sorting the representations by the onset on which they are attributed we obtain a sequence $S = [h_{u_1}^{(cp)}, h_{u_2}^{(cp)}, \dots h_{u_k}^{(cp)}]$ such that $on(u_1) < on(u_2) < \cdots < on(u_k)$.

The sequence S is then passed through an MLP layer and 2 GRU layers. This concludes the hard-sharing part of our model. Thereafter, an MLP head is attached per task, as shown in Figure 6.3.

For training, we use the dynamically weighted loss introduced by [20]. The total loss $\mathcal{L}_{tot}$ of our network is calculated as a weighted sum of the individual losses for every task, where the weights are learned during training:

$$\mathcal{L}_{\text{tot}} = \sum_{t \in \mathcal{T}} \mathcal{L}_t * \frac{1}{2\gamma_t^2} + \log(1 + \gamma_t^2) \tag{6.4}$$

where $\mathcal{T}$ is the set of tasks; $\mathcal{L}_t$ is the cross-entropy loss relating to task t; the γ_t are learned scalars that give the weight for each task t; and the log expression is a regularization term [20].

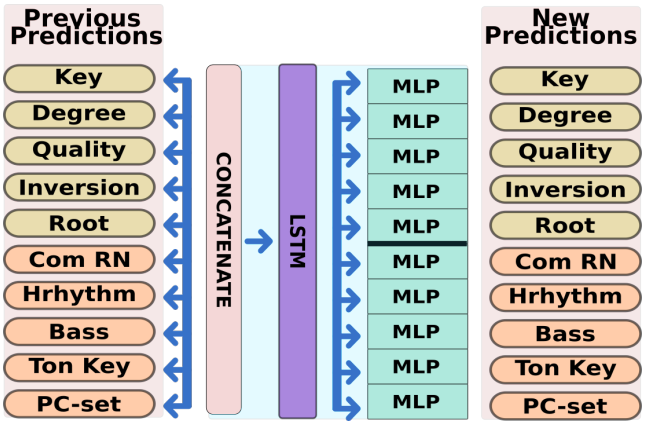


Figure 6.4: Post-processing of Roman Numeral predictions.

6.3.3.1 *Post-processing*

We enhance our model with a post-processing phase after the model has been trained. The post-processing phase combines the logits of all tasks' predictions by concatenating them and, then, feeds them to a single-layer bidirectional LTSM block. Then, again the embeddings of the sequential block are distributed to 11 one-layer MLPs, one for each task. The post-processing block is sketched in Figure 6.4.

6.4 EXPERIMENTS AND CORPORA

In the experiments, we compare our model, *ChordGNN*, with other recent models for automatic Roman Numeral analysis. We run experiments with our model in the exact same way as described in the paper [7], including the specific data splits, so that our results are directly comparable to the figures reported there. A detailed comparison of the results will be given in Table 6.1. Furthermore, we develop variants of our model using proposed techniques such as NADE [8], and post-processing of the chord predictions. We report a configuration study of our model on the use of gradient normalization techniques and NADE that should improve results on Multi-Task learning scenarios and avoid common Multi-Task Learning problems such as conflicting gradients. Lastly, we compare our model with the updated version *v1.9.1* of the state-of-the-art model Augmented-Net [21] and datasets.

	Model	Key	Degree	Quality	Inversion	Root	RN	RN (Onset)	RN_{alt}
BPS	Micchi (2020)	82.9	68.3	76.6	72.0	-	42.8	-	-
	CSM-T (2021)	69.4	-	-	-	75.4	45.9	-	-
	AugNet (2021)	**85.0**	**73.4**	**79.0**	73.4	**84.4**	45.4	-	49.3
	ChordGNN (Ours)	79.9	71.1	74.8	75.7	82.3	46.2	46.6	48.6
	ChordGNN+Post (Ours)	82.0	71.5	74.1	**76.5**	82.5	**49.1**	**49.4**	**50.4**
Full	AugNet (2021)	**82.9**	67.0	**79.7**	78.8	83.0	46.4	-	51.5
	ChordGNN (Ours)	80.9	70.1	78.4	78.8	84.8	48.9	48.4	50.4
	ChordGNN+Post (Ours)	81.3	**71.4**	78.4	**80.3**	**84.9**	**51.8**	**51.2**	**52.9**

Table 6.1: Model comparison on two different test sets, the Beethoven Piano Sonatas (BPS), and the full test set. *RN* stands for Roman Numeral, RN_{alt} for the alternative Roman Numeral computations discussed in Section 6.3.1. $RN(Onset)$ refers to onset-wise prediction accuracy, all other scores use the CSR score (see Section 6.5). Note that model *CSM-T* reports *Mode* instead of *Quality*.

6.4.1 Datasets

For training and evaluation, we combined six data sources into a single "Full" Dataset of Roman Numeral annotations in accordance with [7]: the Annotated Beethoven Corpus (ABC) [22]; the annotated Beethoven Piano Sonatas (BPS) dataset [5]; the Haydn String Quartets dataset (HaydnSun) [23]; the TAVERN dataset [24]; a part of the When-in-Rome (WiR) dataset [25, 26]; and the Well-Tempered-Clavier (WTC) dataset [25] which is also part of the WiR dataset.

Training and test splits for the full dataset were also provided by [7]. It is worth noting that the BPS subset splits were already predefined in [5]. In total, approximately 300 pieces were used for training, and 56 pieces were used for testing, proportionally taken from all the different data sources. We draw a distinction for the BPS test set, which includes 32 Sonata first movements and for which we ran an additional experiment. The full test set also includes the 7 Beethoven piano sonatas.

In addition to the above datasets, we include data augmentations identical to the ones described in [7]: texturization and transposition. The texturization is based on a dataset augmentation technique introduced by [27]. The transposition augmentation boils down to transposing a score to all the keys that lie within a range of key signatures that have up to 7 flats or sharps. It should be noted that the augmentations are only applied in the training split.

For our last experiment (to be reported on in Section 6.5.3 below), we add additional data that were recently introduced by [21]. The additional data include the annotated Mozart Piano Sonatas (MPS) dataset [28] for which we also applied the aforementioned augmentations.

6.4.2 Configuration

For all our experiments, we train our network with the AdamW optimizer. We fix our architecture with a hidden size of 256, a learning rate of 0.0015, a weight decay of 0.005, and a dropout of 0.5 which is applied to each learning block of our architecture.

6.5 RESULTS

As an evaluation metric, we use Chord Symbol Recall (CSR) [29] where for each piece, the proportion of time is collected during which the estimated label matches the ground truth label. We apply the CSR at the 32nd note granularity level, in accordance with [6, 7, 9].

6.5.1 Quantitative Results

In the first experiment, which compares our *ChordGNN* to existing state-of-the-art approaches, we evaluate the full dataset, but also the annotated Beethoven Piano Sonatas (BPS) [5] subset, which many previous approaches had also used. The results are shown in Table 6.1. We present the CSR scores (where they are applicable) for Local Key, Degree, Quality, Inversion, Root, conventional Roman Numeral, and

Variant	RN	RN_{alt}
ChordGNN (Baseline)	46.1 ± 0.003	47.8 ± 0.007
ChordGNN + WLoss	$\mathbf{48.9} \pm 0.001$	$\mathbf{50.4} \pm 0.010$
ChordGNN + Rotograd	45.5 ± 0.003	47.1 ± 0.005
ChordGNN + R-GradN	45.2 ± 0.006	46.7 ± 0.005
ChordGNN + NADE	48.2 ± 0.005	49.9 ± 0.005

Table 6.2: Configuration Study: Chord Symbol Recall on Roman Numeral analysis on the full test set. *RN* stands for Roman Numeral, RN_{alt} refers to the alternative Roman Numeral computations discussed in section 6.3.1. WLoss stands for the dynamically weighted loss described in Section 6.3, and R-GradN stands for Rotograd with Gradient Normalization. Every experiment is repeated 5 times with the same ChordGNN model as Table 1 without post-processing.

Alternative Roman Numeral (see Section 6.3). Furthermore, we include the onset-wise accuracy score for our models' conventional Roman Numeral predictions.

On the BPS subset, we compare our model *ChordGNN* with the Micchi (2020) model [6], the *CSM-T* (2021) model [9] and the *AugmentedNet* 2021 model [7]. Our results on Roman Numeral prediction surpass all previous approaches. Note that the *AugmentedNet* model exhibits higher prediction scores on the individual Key, Degree, Quality, and Root tasks, which are used jointly for the prediction of the Roman numeral. These results indicate that our model obtains more meaningfully interrelated predictions, with respect to the Roman numeral prediction, resulting in a higher accuracy score.

Moreover, we compare *ChordGNN* to *AugmentedNet* on the full test dataset. Our model surpasses *AugmentedNet* with and without post-processing in all fields apart from local key prediction and quality. Our model obtains up to 11.6% improvement in conventional Roman Numeral prediction.

In both experiments, post-processing has been shown to improve both *RN* and RN_{alt}. However, *ChordGNN* without post-processing already surpasses the other models.

6.5.2 Configuration Study

For a systematic study of multitask training, we investigated the effects of extension modules, gradient normalization techniques, and learnable weight loss. In detail, we test 5 configurations using as baseline the *ChordGNN* model (without post-processing) with standard CE loss and no weighing. Furthermore, we test our proposed architecture using the dynamically weighted loss described in Section 6.3.3 (same as the model in Table 6.1), Rotograd [17] and GradNorm [16] for Gradient Normalization, and NADE [8]. The models are run on the Full data set described above and averaged over five runs with random initialization. The results, summarized in Table 6.2, suggest that using the dynamically weighted loss yields better results compared to other methods such as the Baseline or Gradient Normalization techniques. Furthermore, the dynamically weighted loss is comparable to NADE but also more robust on Conventional Roman Numeral prediction on our datasets.

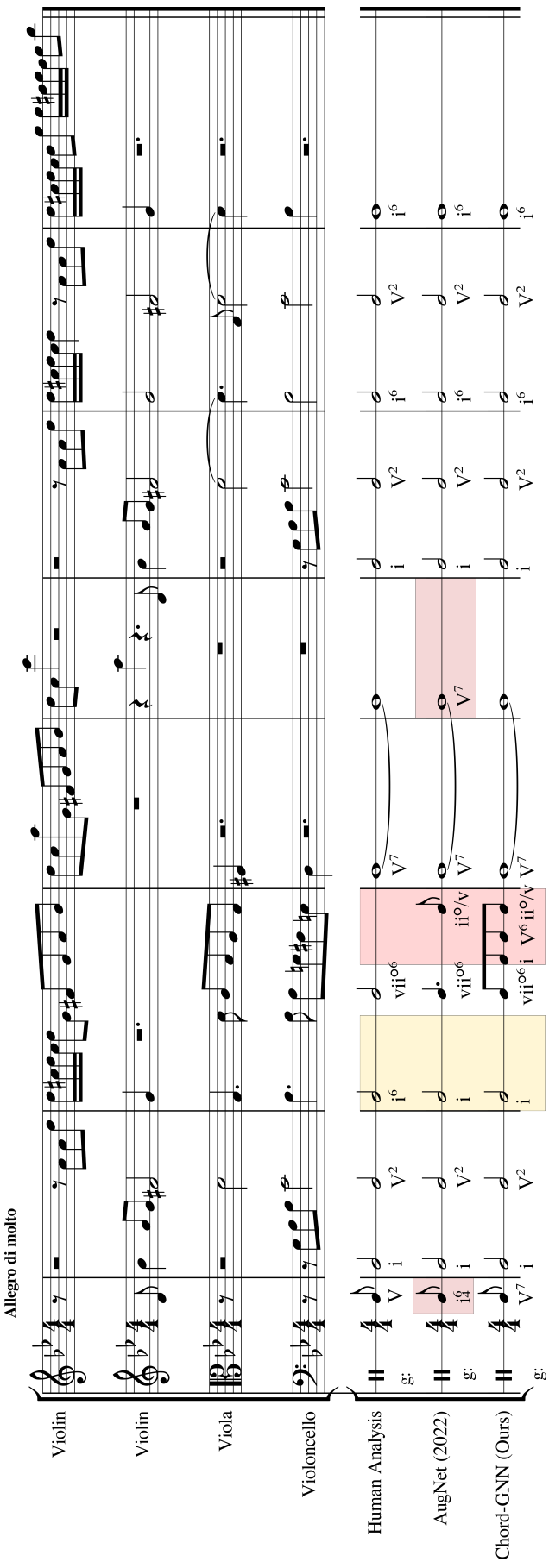


Figure 6.5: A comparison between the human annotation, AugmentedNet, and ChordGNN on a passage of Haydn's string quartet op.20 No.3 movement 4. The red (wrong) markings on Human Analysis and AugNet (2022) are from [21]

6.5.3 Latest developments

Our last experiment focuses on specific developments that have very recently been published in Nápoles López's Ph.D. thesis [21]. In the thesis, three additional tasks, related to predicting the components of a canonical representation of the current chord, as implied by the Roman Numeral, were proposed and the dataset was extended with the Annotated Mozart Piano Sonatas (MPS) corpus [28], as mentioned in Section 6.4.1 above.

To test the relevance of these updates, we trained an adapted version of our model, now with 11+3=14 individual tasks and including the Mozart data. It turns out that the updated model improves significantly in performance, achieving a 53.5 CSR score on conventional Roman Numeral (compare this to row "ChordGNN (Ours)" in Table 6.1). Furthermore, post-processing can improve the results by up to two additional percentage points.[1]

6.5.4 A Musical Example

In Figure 6.5, we look at a comparison between the human annotations, *Augmented-Net* and *Chord-GNN* predictions (The musical excerpt is taken from Nápoles López's thesis [21], and the predictions relate to the new models trained as described in the previous section.). Marked in red are false predictions, and marked in yellow are correct predictions of the model with wrong ground-truth annotations. Both models' predictions are very similar to the human analysis. However, our model correctly predicts the initial pickup measure annotation. In measure 2, the ground truth annotation marks a tonic in first inversion; however, the viola at that point is lower than the cello and therefore the chord is actually in root position. Both models obtain a correct prediction at that point. Subsequently, our model predicts a harmonic rhythm of eighth notes, which disagrees with the annotator's half-note marking. Analyzing the underlying harmony in that passage, we can justify our model's choices.

The human annotation suggests that the entire second half of the 2nd measure represents a vii^{o} chord. However, it should not be in the first inversion, as the cello plays an F# as the lowest note (which is the root of vii^{o}). The AugNet analysis faces the same issue, in contrast with the predictions of ChordGNN. However, there are two conflicting interpretations of the segment. First, the vii^{o} on the third beat is seen as a passing chord between the surrounding tonic chords, leading to a dominant chord in the next measure. Alternatively, the vii^{o} could already be part of a prolonged dominant harmony (with passing chords on the offbeats) leading to the V^{7}. The ChordGNN solution accommodates both interpretations as it doesn't attempt to group chords at a higher level, treating each eighth note as an individual chord rather than a passing event. The other two solutions prefer the second option.

1 Unfortunately, we cannot directly compare these numbers to [21], as their results are not reported in comparable terms.

6.6 CONCLUSION

In this paper, we presented *ChordGNN*, a model for automatic Roman Numeral analysis in symbolic music, based on a note-level, graph-based score representation. We showed that *ChordGNN* improves on other state-of-the-art models, and that post-processing can further improve the accuracy of the predictions. A configuration study suggests that gradient normalization techniques or techniques for carrying prediction information across tasks are not particularly beneficial or necessary for such a model.

Follow-up work will focus on strengthening the robustness of our models by pre-training with self-supervised methods on large corpora. We believe that such pre-training can be beneficial for learning helpful intrinsic musical information. Such a step is crucial since more data improves predictions but Roman Numeral annotations are hard to find or produce. Moreover, we aim to enrich the number of tasks for joint prediction by including higher-level analytical targets such as cadence detection and phrase boundary detection. Finally, we aim to extend our method to the audio domain.

6.7 ACKNOWLEDGEMENTS

We gratefully acknowledge the musical analysis of the vii^{o} passage in Fig. 6.5 (Section 6.5.4) that was offered by an anonymous reviewer, and which we took the liberty of adopting for our text. This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 ("Whither Music?"), and the Federal State of Upper Austria (LIT AI Lab).

REFERENCES

[1] Johan Pauwels, Ken O'Hanlon, Emilia Gómez, Mark Sandler, et al. "20 years of Automatic Chord Recognition from Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[2] David Temperley. *The cognition of basic musical structures*. MIT press, 2004.

[3] Christopher Raphael and Joshua Stoddard. "Functional Harmonic Analysis Using Probabilistic Models." In: *Computer Music Journal* 28.3 (2004), pp. 45–52.

[4] José Pedro Magalhaes and W Bas de Haas. "Functional Modelling of Musical Harmony: an experience report." In: *ACM SIGPLAN Notices* 46.9 (2011), pp. 156–162.

[5] Tsung-Ping Chen, Li Su, et al. "Functional Harmony Recognition of Symbolic Music Data with Multi-task Recurrent Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[6] Gianluca Micchi, Mark Gotham, and Mathieu Giraud. "Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 3.1 (2020), pp. 42–54.

[7] Néstor Nápoles López, Mark Gotham, and Ichiro Fujinaga. "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[8] Gianluca Micchi, Katerina Kosta, Gabriele Medeot, and Pierre Chanquion. "A deep learning method for enforcing coherence in Automatic Chord Recognition." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[9] Andrew Philip McLeod and Martin Alois Rohrmeier. "A modular system for the harmonic analysis of musical scores using a large vocabulary." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[10] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[11] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[12] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[13] Carlos Hernandez-Olivan, Sonia Rubio Llamas, and Jose R. Beltran. *Symbolic Music Structure Analysis with Graph Representations and Changepoint Detection Methods*. 2023. arXiv: 2303.13881.

[14] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Facial Landmark Detection by Deep Multi-task Learning." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

[15] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. "Dynamic Task Prioritization for Multitask Learning." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[16] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018.

[17] Adrián Javaloy and Isabel Valera. "RotoGrad: Gradient Homogenization in Multitask Learning." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022.

[18] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. "Multi-task Learning as a Bargaining Game." In: *Proceedings of the International Conference on Machine Learning (ICML)* (2022).

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[20] Lukas Liebel and Marco Körner. "Auxiliary tasks in multi-task learning." In: *arXiv preprint arXiv:1805.06334* (2018).

[21] Néstor Nápoles López. "Automatic Roman Numeral Analysis in Symbolic Music Representations." PhD thesis. Schulich School of Music McGill University, 2022.

[22] Markus Neuwirth, Daniel Harasim, Fabian C Moss, and Martin Rohrmeier. "The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets." In: *Frontiers in Digital Humanities* 5 (2018), p. 16.

[23] Néstor Nápoles López. "Automatic Harmonic Analysis of Classical String Quartets from Symbolic Score." PhD thesis. Master's thesis, Universitat Pompeu Fabra, 2017.

[24] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. "Theme and Variation Encodings with Roman Numerals (TAVERN): A new data set for symbolic music analysis." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2015.

[25] Mark Gotham, Dmitri Tymoczko, and Michael Scott Cuthbert. "The RomanText Format: A Flexible and Standard Method for Representing Roman Numeral Analyses." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[26] Mark Robert Haigh Gotham and Peter Jonas. "The Openscore Lieder Corpus." In: *Proceedings of the Music Encoding Conference (MEC)*. 2021.

[27] Néstor Nápoles López and Ichiro Fujinaga. "Harmonic Reductions as a Strategy for Creative Data Augmentation." In: *Late-Breaking Demo at International Society for Music Information Retrieval Conference (ISMIR)*. 2020.

[28] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. "The Annotated Mozart Sonatas: Score, Harmony, and Cadence." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4.ARTICLE (2021), pp. 67–80.

[29] Christopher Harte. "Towards automatic extraction of harmony information from music signals." PhD thesis. Queen Mary University of London, 2010.

7 MONOPHONIC VOICE SEPARATION

Title: Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-trajectory Tracking Problem

Published In Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI), Macao, 2023.

Authors: Emmanouil Karystinaios, Francesco Foscarin, Gerhard Widmer

Abstract: This paper targets the perceptual task of separating the different interacting voices, i.e., monophonic melodic streams, in a polyphonic musical piece. We target symbolic music, where notes are explicitly encoded, and model this task as a Multi-Trajectory Tracking (MTT) problem from discrete observations, i.e., notes in a pitch-time space. Our approach builds a graph from a musical piece, by creating one node for every note, and separates the melodic trajectories by predicting a link between two notes if they are consecutive in the same voice/stream. This kind of local, greedy prediction is made possible by node embeddings created by a heterogeneous graph neural network that can capture inter- and intra-trajectory information. Furthermore, we propose a new regularization loss that encourages the output to respect the MTT premise of at most one incoming and one outgoing link for every node, favouring monophonic (voice) trajectories; this loss function might also be useful in other general MTT scenarios. Our approach does not use domain-specific heuristics, is scalable to longer sequences and a higher number of voices, and can handle complex cases such as voice inversions and overlaps. We reach new state-of-the-art results for the voice separation task in classical music of different styles. All code, data, and pretrained models are available on https://github.com/manoskary/vocsep_ijcai2023

7.1 INTRODUCTION

The Multi-Trajectory Tracking (MTT) problem considers an unknown number of moving objects and deals with the task of connecting a sequence of observations, usually points or short tracks in a spatiotemporal space, into accurate long-term trajectories. MTT is a subject of study both in cognitive science and engineering areas and has applications in numerous fields, including guidance systems, surveillance, and threat assessment [1].

Existing approaches are based on dynamic programming algorithms that try to minimize the global cost (or maximize the global probability) of assigning observations to a certain trajectory [1–4]. Approaches based on deep learning have been developed in the related field of Multi-Object Tracking (MOT), which also

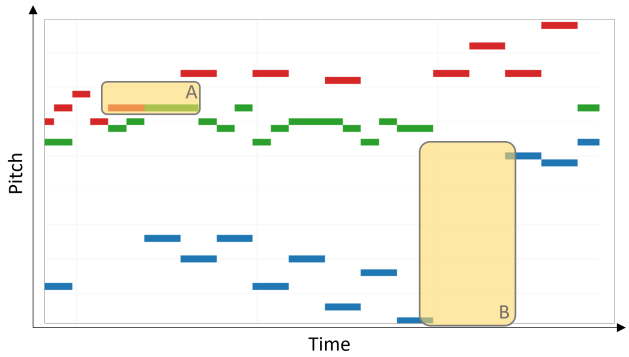


Figure 7.1: Example of multi trajectory following for musical voice separation in a pitch-time space. Different trajectories are highlighted with different colors. Box (A) contains an example of consecutive notes with the same pitch belonging to different voices. Box (B) contains an example of "distant" notes belonging to the same voice. The musical excerpt is taken from Bach's Fugue in C-sharp major, BWV 872, measures 2-3-4.

concerns itself with an object identification step, usually from images or similar data where the object positions are not explicitly encoded. Together with the object detection modules, an MOT system also contains a tracking module, that needs to deal with the MTT problems. However, while MTT systems can rely only on the trajectory shapes [5], MOT systems can also rely on the similarity between the features extracted from the instantaneous state of the objects to compute the trajectory. For example, an MOT system that tracks a red car from video data will extract some features about the car being red that will greatly help distinguish this car from cars of other colours across frames.[1]

In this paper, we will use an MTT approach to model a musical perception task, namely, *voice separation*. Many kinds of music can be seen as a sum or configuration of different *voices* [6], i.e., trajectories of (mostly) nonoverlapping notes (see Figure 7.1) as if they were produced by different voices singing together. Such voices are often not explicitly separated in a score (or a performance on a polyphonic instrument such as the piano), and the task of separating them is a useful step for a number of applications in music information retrieval, such as melody identification [7] and MIDI to score transcription [8].

Symbolic music denotes a set of musical formats (of different degrees of specificity and detail, e.g., MusicXML, **kern, MIDI), which contain explicit information about note pitches, onsets (i.e., starting time), and offsets (i.e., ending times) [9], as opposed to audio files where all this information is mixed in a single acoustic signal.[2] Single notes can therefore be treated as trajectory observations in the pitch-time space, and voice separation can be framed as a trajectory following the problem.

1 A plethora of different terms, including "Multi Target Tracklet Stitching", "Multi Target Tracking", are used in the literature to identify different flavours of related problems, and their usage is not always consistent across different communities. In this paper, we will use only the two terms MTT and MOT with the meaning indicated above.

2 In particular the input to our model is a set of notes with only pitch, onset, and offset (or duration) information. We assume them to be *quantized*, i.e., notes whose onset and offset are aligned to a regular grid. It is irrelevant to our approach whether these notes are obtained from a score, a transcribed performance, a generation algorithm, or other sources.

However, unlike the trajectories of objects that move in space, disentangling voice trajectories presents a set of unique challenges. Voices are not bound to stay in the immediate proximity of their last position and typically contain musical rests, that create "holes" in the trajectory. Note that following the MTT and MOT differences we highlighted above, voice separation cannot be considered an MOT task. The static representative of a voice, i.e., a single note, does not contain any information about the voice to which it belongs. For example, two consecutive notes with the same pitch may belong to different voices (see Figure 7.1). The correct trajectory assignment can be made only by considering the rest of the trajectories.

Existing voice separationalgorithms [10–15] use perceptual principles and domain heuristics (with a few learned parameters in some cases) to compute the probability/cost of assigning a note to a certain voice, and then globally optimize the probabilities/costs for the entire piece. These approaches have a set of intrinsic weaknesses that limit their performance. The first is a problem with generalization since the principles and heuristics employed may not remain valid for different kinds of music. The second is that music is a complex domain full of corner cases, and modelling it with few rules would naturally make them fail in a number of situations. Moreover, the search space of dynamic programming approaches that do global optimization, with an unknown number of voices, scales exponentially with sequence length and the number of voices, making it necessary to work with short sequences or to add extra conditions to limit the search space (very common is to disallow voice crossings, though these can well occur in real music).

In this paper, we overcome these limitations with a technique based on *Graph Neural Networks (GNNs)* that does not rely on any heuristic or domain principle. We frame the voice following as a link prediction problem; we model every note as a vertex in a graph and greedily predict a link between any pair of notes that should be consecutive in the same voice. This process results in a graph in which each fully connected group of nodes corresponds to a different voice. To take advantage of inter- and intra-voice dependencies during the link prediction phase, we build a rich set of edges on top of the node vertices, based on the temporal relations between the corresponding notes in the music piece. We use edge relations to propagate local note information using heterogeneous message passing. Since each link is independently predicted, our output could contain invalid configurations where a note has multiple incoming or outgoing links. Therefore, we also propose a new loss to enforce this number to be a maximum of one. Our model consistently exceeds the state-of-the-art for Voice Separation on a large reference dataset with classical music of different styles. We can further increase the performance by running a polynomial-time global optimization algorithm that ensures that every note has a maximum of one incoming and one outgoing predicted link.

The contributions of this work are as follows.

- a heterogeneous graph neural network approach to producing meaningful contextual features for the MTT task cast as a greedy link prediction problem;
- a new loss function to enforce the MTT constraint of a maximum of one incoming and one outgoing link for every trajectory observation.

- the application of MTT on symbolic music, resulting in a generalizable and scalable approach to the voice separation problem that handles overlaps and voice-switching;
- new state-of-the-art results on a reference dataset with classical music of different styles.

7.2 RELATED WORK

There are a number of approaches that address the problem of separating symbolic music into monophonic voices that are relevant to our work. Duane and Pardo [13] propose the evaluation measure that was used in most of the subsequent research, including ours, and frame the voice separation problem as a set of link prediction problems between each pair of notes. The main conceptual difference between their approach and ours is that we enrich the note features with embeddings computed with a graph neural network, which drastically increases the quality of the predictions. Instead, they run a global optimization algorithm that scales exponentially with the note sequence length. This forces them to restrict the search space by not considering voice-crossings and to target only a few measures each time, stitching together the results afterwards.

The current state-of-the-art results, which we compare to in this paper, were produced by Mcleod and Steedman [15] with an HMM-based method where the probabilities of having a note assigned to a voice are based on Huron's [16] perceptual principles of minimizing the time distance between consecutive notes and the pitch distance in a voice. To restrict the exponential search space for the global solution, they employ a modified Viterbi algorithm where at each step only the two best options are kept.

Notable for an approach which does not require the global optimization process, is the work of Gray and Brunescu [17]. They run a left-to-right algorithm where a neural network greedily predicts, at each step, which existing voice a note should be assigned to (or whether to create a new voice). However, the network is not informed about inter-voice interactions or future voice trajectories, and the manually engineered features they use to help the prediction process are not enough to achieve higher experimental results than McLeod and Steedman. Also worthy of mention is the work of Hsiao and Su [18], which model the score as a graph, and use unsupervised node graph clustering to separate different voices. The limits of this model lie in the fact that the clustering algorithm expects a given number of voices as input, and the heuristic the authors devise to estimate this number assumes the number of voices to be constant during the piece, which is a hypothesis that we are not introducing. Moreover, despite some slightly misleading claims, this approach does not reach new state-of-the-art results. in the case of quantized music, which we assume to be the input of our system.

Another field of research [19–21] targets music that can contain chords (i.e., multiple simultaneous notes) in the same voice. This is a different task (see [22] for a discussion of different types of voice separation problems) and is not the focus of our work. We are also not targeting the problem of voice separation from human performance data that is explored by McLeod and Steedman [15].

Similarly to the above-mentioned work in voice separation, multi-trajectory tracking (MTT) research is based on dynamic programming algorithms that perform global optimization on possible trajectories [1–4]. The field of Multi-Object Tracking (MOT) has received more attention in recent years, with a number of articles using GNNs. Our approach shares some similarities with the work of Brasó and Leal-Taixé [23], Weng et al.[24], and Wang et al.[25], in particular, the formulation of trajectory tracking as a greedy link prediction problem and the use of GNNs to generate relevant features for this prediction. However, our data present a different set of challenges: the absence of useful static features and large temporal and spatial (pitch, in our case) gaps between consecutive observations in the same trajectory. Therefore, while the aforementioned MOT papers use only homogenous graphs (with some minor improvements by Brasó and Leal-Taixé [23] that treat past and future links differently), we use heterogeneous graph neural networks with seven different link types, to create more informative node embeddings.

Finally, some works [24, 25] use the cross-entropy loss applied column-wise and row-wise in the adjacency matrix, to force each node to have a maximum of one incoming and one outgoing link. With the same goal, we propose an alternative loss that gives us better experimental results.

7.3 APPROACH

We model the input of our system, i.e., a set of quantized notes, with pitch, onset, and offset information, with a graph structure, where every note corresponds to a node in the graph. If we consider a set of links that connect only consecutive notes in the same voice (see Figure 7.2, right part), then the voices correspond to connected components in the graph. Such a set of links is the desired output of our system, and the ground truth used for training.

Formally, consider a musical piece as a set V of notes, where the temporal position of the onsets defines a non-strict total order (i.e., multiple notes may have the same onset). Let Θ be a partition of V in disjointed voices θ. Since voices are monophonic, each voice defines a trajectory of length l, $\mathcal{T}_\theta = [v_1^\theta \dots v_l^\theta \mid v \in V]$: a strictly-ordered set that contains notes consecutive in the same voice. We can also view it as a set of pairs of consecutive notes in the same voice:

$$E_{\text{target}} = \{(v_i^\theta, v_{i+1}^\theta) \mid \forall \theta \in \Theta,\ v_i^\theta \in \mathcal{T}_\theta\} \tag{7.1}$$

E_{target}, as a specification of the voices in a piece, defines our ground truth. Our goal is to predict such a set, and we do this by applying a binary classifier to every potential note pair $(u, v) \in V \times V$. We name the predicted set E_{pred}. This process can be seen as predicting whether there is a link between u and v, hence it is usually called *link prediction*.

We model a piece as a heterogeneous graph [26] with different types of relations (or edge types) between notes and learn note embeddings using a GNN. Let $G = (V, E_{in}, \mathcal{R})$ be a graph such that V is the set of notes, E_{in} is the set of edge relations (or typed edges) , and $\mathcal{R}$ is a set of relation types. Every edge relation is defined by a triplet, i.e. (u, r, v) such as $u, v \in V$ and $r \in \mathcal{R}$. In addition, we associate

every note with a vector of k features that describe some intrinsic note properties. We assume all these feature vectors to be collected in a matrix $X \in \mathbb{R}^{|V| \times k}$.

Therefore, our approach to voice separation can be summarized as follows: given a musical piece in symbolic form, we build a heterogeneous graph $G = (V, E_{in}, \mathcal{R})$ and a set of node features X, and we use it to predict a set of links E_{pred} that encode the voice trajectory according to Equation 7.1.

7.3.1 Graph Building

Given a musical piece, the graph-building process creates a set of edges E_{in}, with different relation types $\mathcal{R}$. We follow the work of Karystinaios and Widmer [27] and Jeong et al. [28] but adapt it to our voice separation problem by not using explicit musical rests (which are not present in our input) and considering a dedicated set of relation types that does not include any voice information. Let us consider three functions $on(v)$, $dur(v)$, and $pitch(v)$ defined on a note $v \in V$ that extract the onset, duration, and pitch, respectively.

A labeled edge (u, r, v) of type r between two notes u, v belongs to E_{in} if the following conditions are met:

- notes starting at the same time $on(u) = on(v) \rightarrow r = \text{onset}$
- note starting while the other is sounding $on(u) > on(v) \wedge on(u) \leq on(v) + dur(v) \rightarrow r = \text{during}$
- note starting when the other ends $on(u) + dur(u) = on(v) \rightarrow r = \text{follow}$
- note starting after a time frame when no note is sounding $on(u) + dur(u) < on(v) \wedge \nexists v' \in V,\ on(v') < on(v) \wedge on(v') > on(u) + dur(u) \rightarrow r = \text{silence}$

It is worth noting that, by construction, "onset" is the only undirected relation type, i.e., if (u, onset, v), then (v, onset, u). To keep our graph informed about the temporal evolution of the piece, we want the edges that connect notes at different times to be directed and only point to the future. But we also want the prediction to depend on future context. Therefore, we add in $\mathcal{R}$ three more types corresponding to the inverse of "during", "follow", and "silence", and we add to E_{in} the inverse edges with such types.

7.3.2 Node Features

We build a feature vector for every note $v \in V$. This contains the note's pitch-class and octave, as one-hot vectors of size 12 and 8 respectively, and the duration. The duration is encoded as a single float value $d \in [0, 1]$ computed as

$$d_n = 1 - tanh \frac{dur(v)}{dur(m)}, \tag{7.2}$$

where $dur(m)$ is the duration of the bar to which the note belongs. This information is also available in the symbolic music that we have as input. Normalization with bar duration has the objective of making d independent of the time signature of the

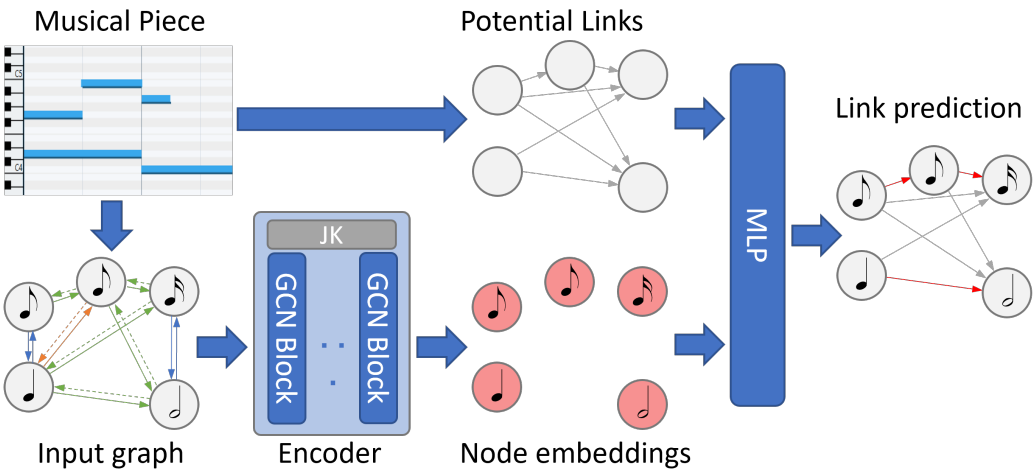


Figure 7.2: The GMTT model architecture for link prediction.

piece, and the *tanh* function gives more resolution for lower note values while still being able to encode high durations. Additionally, we use a positional encoding based on the 20 first eigenvectors from the Laplacian of the adjacency matrix similar to the work of Dwivedi et al. [29].

7.3.3 Model

Our model consists of two parts: a node encoder and a link predictor.

The goal of the node encoder is to project node features X to an embedding space that is enriched with context information. It consists of a series of Residual Gated Convolutions [30] with Jumping Knowledge [31]. To account for the different edge relation types $r \in \mathcal{R}$, we compute independent representations for each type of relation and average them at the end of each convolutional block [32]. More specifically, for every $v \in V$:

$$\mathbf{h}_{v_r}^{(l+1)} = \mathbf{W}_1^{(l)}\mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}_r(v)} \eta_{v,u}^{(l)} \odot \mathbf{W}_2^{(l)}\mathbf{h}_u^{(l)} \tag{7.3}$$

$$\eta_{v,u}^{(l)} = \sigma(\mathbf{W}_3^{(l)}\mathbf{h}_v^{(l)} + \mathbf{W}_4^{(l)}\mathbf{h}_u^{(l)}) \tag{7.4}$$

$$\mathbf{h}_v^{(l+1)} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathbf{h}_{v_r}^{(l+1)} \tag{7.5}$$

where $h_v^{(l)}$ is the embedding of node v for layer l, σ denotes the sigmoid function, and r denotes the relation type. On top of our convolutional blocks, we add Jumping Knowledge, i.e., a bi-directional LSTM that connects the output of every convolutional layer $l \in L$ of the encoder:

$$\mathbf{h}_v^{(jk)} = \sum_{l=1}^{L} \alpha_v^{(l)}\mathbf{h}_v^{(l)} \tag{7.6}$$

where α denotes the attentional weights obtained by the LSTM and $\mathbf{h}_v$ is the node embedding for node v.

The link predictor part of our model is a multilayer perceptron (MLP) that performs the binary classification task of deciding whether two notes should be linked (i.e., be part of E_{pred}). Due to our problem definition, the predictor only considers the links between (u, v) if $on(u) + dur(u) \leq on(v)$. We name the set of potential links Λ. For every potential link $(u, v) \in \Lambda$ we concatenate the embeddings of u and v produced by the encoder and give them as input to the link predictor:

$$\hat{y}_{u,v} = \text{MLP}(\text{concatenate}(h_u^{(jk)}, h_v^{(jk)})) \tag{7.7}$$

where $\hat{y}_{u,v} \in [0, 1]$ denotes the predicted probability of a link from u to v. For links $(u, v) \notin \Lambda$ we set $\hat{y}_{u,v} = 0$. We choose to concatenate the encoder's embeddings instead of taking their product because our links are directed, i.e. $(u, v) \in E_{\text{pred}} \not\Longrightarrow (v, u) \in E_{\text{pred}}$.

After this process, we are left with a probability for each pair (u, v) to be part of E_{pred}. We round the probabilities according to a threshold value τ to obtain hard predictions.

7.3.4 Loss

In the training phase, we use positive and negative link prediction samples by subsampling the negative samples in Λ (i.e. $\{a \in \Lambda \mid a \notin E_{\text{target}}\}$) to match the number of positives. We train our model by minimizing the Binary Cross-entropy:

$$\mathcal{L}_{\text{clf}} = -\sum_{u,v \in \mathcal{D}} (y_{u,v} \log(\hat{y}_{u,v}) + (1 - y_{u,v}) \log(1 - \hat{y}_{u,v}))) \tag{7.8}$$

where $y_{u,v} = 1$ if $(u, v) \in E_{\text{target}}$, 0 otherwise.

Let $\hat{A} \in [0, 1]^{|V| \times |V|}$ be the weighted graph adjacency matrix over V that contains $\hat{y}_{u,v}$ at the corresponding indices. Since all our voices are disjointed, a note u can be connected to at most one note t that occurs before u and at most one note v that occurs after u. This means that, in a perfect prediction scenario, we would have in $\hat{A}$ only one non-zero element for each row and each column (or all zeros if the corresponding note ends or starts a voice, respectively). One can therefore regard the ideal output of our system as the result of a linear assignment problem [33] over the predicted adjacency matrix $\hat{A}$.

In order to drive $\hat{A}$ to be in this format we propose a regularization loss, loosely inspired by [34], in addition to the classification loss. It is defined as follows:

$$\mathcal{L}_{reg}^{(1)} = \|\boldsymbol{\zeta} - \sum_{i \in N} \hat{A}[i, :]\|_2 + \|\boldsymbol{\xi} - \sum_{j \in N} \hat{A}[:, j]\|_2 \tag{7.9}$$

$$\mathcal{L}_{reg}^{(2)} = \|\boldsymbol{\zeta} - \sqrt{\sum_{i \in N} \hat{A}^2[i, :]}\|_2 + \|\boldsymbol{\xi} - \sqrt{\sum_{j \in N} \hat{A}^2[:, j]}\|_2 \tag{7.10}$$

$$\mathcal{L}_{reg}^{(tot)} = \frac{\mathcal{L}_{reg}^{(1)} + \mathcal{L}_{reg}^{(2)}}{N} \tag{7.11}$$

where N is the number of nodes in the graph, ζ is a binary-valued vector of length N with ones only for the nodes that are source nodes of ground truth links in E_{target}, and ξ is also a binary-valued vector, with ones only on the destination node indices of the ground truth links.

Equation 7.9 encodes the linear assignment optimization objective, modified to allow rows and columns with only zeros. Furthermore, we add Equation 7.10, which uses the $L2$ norm of rows and columns, by squaring the positive valued $\hat{A}$. $L2$ and $L1$ are known to have different strengths in minimization problems [35], and we found their sum to yield the best experimental results. Together, 7.9 and 7.10 constitute the regularization loss which is normalized by the order of the graph, i.e. the number of nodes $|V|$, since different musical pieces have a different number of notes. The total loss of the system is then defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{clf}} + \alpha \mathcal{L}_{reg}^{(tot)} \tag{7.12}$$

where α is the regularization loss weight. The regularization loss weight α is initialized to 0 and then it is gradually increased every epoch. Conceptually, during the first epochs of training, the focus is on the classification loss, but as training progresses the focus shifts towards a matrix that also satisfies linear assignment conditions.

7.3.5 Postprocessing

Based on the premise introduced in the previous section, we can view the predicted adjacency matrix $\hat{A}$ as a weighted matrix on which we can apply the Hungarian algorithm [36] to solve the linear assignment problem.

Due to our restriction on the potential links Λ, the lower triangular and the diagonal of the adjacency $\hat{A}$ only contain zeros and should not be the focus of the prediction nor the assignment. Therefore, the linear assignment of our matrix only takes part in the upper triangular part of the predicted adjacency. This formulation simplifies the time complexity of the linear assignment.

Given the number of nodes N, our linear assignment optimization objective is defined as:

$$\begin{aligned} &\text{maximize} \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} \hat{A}[i,j] * B[i,j] \\ &\text{subject to} \sum_{j \in [i+1..N]} B[i,j] = 1 \text{ for } i \in [0..N-1], \\ &\text{and} \sum_{i \in [0..j]} B[i,j] = 1 \text{ for } j \in [1..N] \end{aligned}$$

where $B[i,j] \in \mathbb{B}^{N \times N}$ is a learned binary mask over $\hat{A}$. The updated matrix is given for any two indices i, j by $\hat{A}'[i,j] = \hat{A}[i,j] * B[i,j]$. This matrix contains new link probabilities and, equivalently to the approach without post-processing, we round them according to a threshold value τ to obtain hard predictions.

7.4 EXPERIMENTS

Below, we describe the datasets and the experimental settings.

7.4.1 Datasets and Preprocessing

For training and testing our system, we need sets of quantized notes, with pitch, onset, and offset information, with a ground-truth separation into voices, provided by musical experts. We obtain these from a curated collection of musical scores[3] from different composers and styles. In particular, we use all the 474 pieces from the Symbolic Multitrack Contrapuntal Music Archive (MCMA) (see [37] for a detailed list of the pieces contained), and 662 pieces from the KernScore project `http://kern.humdrum.org/`, in particular from the Bach Chorales, the Haydn string quartets, and the Mozart string quartets, which (mostly) satisfy our assumption of monophonic voices. With a total of 1136 pieces, our dataset constitutes the largest data set publicly available for the voice separation task in symbolic music.

To evaluate our system on different degrees of piece complexity and a variety of musical styles, we run five separate experiments. For each experiment, we fix as a test set a subset of our total data and consider 90% of the remaining pieces for training and 10% for validation. In particular, our five test sets consist of 15 Bach inventions, 15 Bach sinfonias, 12 Bach fugues from WTC I, 12 Bach Fugues from WTC II, and 210 Haydn string quartet movements. This corresponds with the data used by McLeod and Steedman [15].

The pieces are available in different file formats and we use the Python library Partitura [38] to extract the list of notes for each voice. We then preprocess them by removing possible extra notes that would violate the monophonic voice assumption (mostly final chords at the very end of the pieces). In particular, if more than one simultaneous note exists in a single voice, we remove all except the highest. This preprocessing operation removes 0.72% of the total notes in our dataset and was done similarly by Mcleod and Steedman [15].

For practical reasons, during the preprocessing phase, we also produce the set of potential links Λ that our model should predict on. For each note $u \in V$, we restrict the potential links to notes v that are at most 2 measures after. We may thus have $\Lambda \subset E_{\text{target}}$, in which case some links in the ground truth will be assigned a probability 0, independently from any other model choice. This means that if a sufficiently long rest exists between two notes of a voice, this voice will be inevitably split into two voices. It is worth noting that this happens very rarely in our datasets. However, this restriction can be relaxed to a longer duration, or removed completely, at the cost of a higher training and inference time.

The graphs produced by our preprocessing phase contain a total of 867,226 nodes and 5M input edges. The number of potential links is 30M, with 863,277 true links. From the latter, 2,264 links are not considered due to the modelling restrictions aforementioned.

3 Note that a score also contains many other musical and graphical elements, such as rests, slurs, and stem directions, that could help in the voice separation task; we discard these in our application.

	McLeod			GMTT			GMMT+LA		
Datasets	P	R	F1	P	R	F1	P	R	F1
Inventions	0.992	0.991	0.992	0.989	0.997	0.995	0.996	0.995	**0.997**
Sinfonias	0.982	0.982	0.982	0.987	0.989	0.978	0.987	0.982	**0.985**
WTC I	0.964	0.964	0.964	0.949	0.983	0.967	0.980	0.973	**0.976**
WTC II	0.964	0.964	0.964	0.945	0.979	0.962	0.976	0.968	**0.972**
Haydn	0.781	0.781	0.781	0.787	0.929	0.850	0.883	0.860	**0.872**

Table 7.1: Main results comparing the State-of-the-art on Voice separation with our approach. P stands for Precision, R stands for Recall, and F1 for F1-score. All the presented metrics are binary (only for the positive class, i.e. links). (+LA) stands for linear assignment postprocessing.

7.4.2 Main Experiment

For our main experiment on the five test sets mentioned above, we use the AdamW optimizer, with learning rate 0.003 and weight decay 0.005. We fix the other parameters of our model to 3 convolutional layers, a convolutional embedding size of 128, i.e. $h_u^{(l)} \in \mathbb{R}^{128}$ for a node u and layer l, and prediction threshold $\tau = 0.5$. The link predictor (MLP) part of our model has the same hidden size and number of layers.

We evaluate our model GMTT with and without post-processing (i.e., applying the Hungarian algorithm to filter the model's predictions) and compare it with the current state-of-the-art voice separation method of McLeod & Steedman [15]. The results are given in terms of recall, precision, and F1-score, calculated between the predicted links E_{pred} and ground truth E_{target}.

The results in Table 7.1 show that GMTT without post-processing is roughly on par with the SOTA, except for the Haydn String Quartet test set, for which we achieve significantly better results. This is an important result since in contrast to all the other approaches in the literature, our system is able to produce high-quality results for the problem of voice separation by only performing local/greedy predictions on single links. These performances are the result of embeddings that were generated by a graph neural network, which provided rich contextual information to the link predictor.

Linear assignment post-processing slightly reduces the recall, but considerably increases the precision, finally producing a higher F1-score. This means that, while our system comes very close to respecting the constraints of having only one incoming edge and one outgoing link for every note, especially thanks to our proposed regularization loss (see Section 7.4.3 for a discussion on this), it still predicts some invalid configurations.

Overall, our improvement over the previous state-of-the-art approach is particularly significant for the Haydn String Quartets collection, which is also the most complex collection to separate. In Section 7.5 below, we conduct a qualitative analysis of an individual example and discuss the musical elements that make this collection so challenging.

7.4.3 Ablation Studies

We perform several ablation studies to understand how our design choices impact the model performance. For each experiment, we change one element of our architecture; if the element is useful, we expect a reduction in F1-score. We fix the hyperparameters of our model to the ones in Section 7.4.2. A summary of the ablation studies can be seen in Table 7.2 and the single experiments are discussed below.

EFFECT OF HETEROPHILY We tested our model using heterogeneous and homogeneous graph convolutional blocks. Leveraging the heterogeneous relations of the score graph has consistently improved results on voice separation, as can be seen by comparing row 1 (*Homogeneous*) in Table 7.2 to row 5 (GMTT).

Models	Haydn	WTC II
Homogeneous	0.809 ± 0.012	0.943 ± 0.007
SageConv Block	0.828 ± 0.005	0.944 ± 0.002
No regularization	0.720 ± 0.021	0.856 ± 0.049
Fixed regularization	0.652 ± 0.18	0.942 ± 0.015
GMMT	$\mathbf{0.850} \pm 0.001$	$\mathbf{0.962} \pm 0.001$

Table 7.2: Ablation experiments, all the scores presented are binary F1-scores without postprocessing (i.e., LA). *Homogeneous* denotes homogeneous graph message passing, *SageConv* denotes the GraphSage convolutional block, *No regularization* means a regularization weight $\alpha = 0$, *Fixed Regularization* has $\alpha = 1$, and *GMMT* is the model from Table 1, for comparison.

EFFECT OF CONVOLUTIONAL BLOCK We experimented with two types of Convolutional Blocks, standard Graph Sage and Residual Gated Convolution. Residual Gated Convolution consistently outperforms Graph Sage convolution in our datasets (row 2 vs. row 5).

EFFECT OF REGULARIZATION LOSS The regularization loss with gradual weighting tends to stabilize training, resulting in more consistent results. Instead, using $\alpha = 1$ we may obtain results close to the final GMMT model using epoch weighting, but the results vary across repetitions of the run (row 4 in Table 7.2). Using no regularization results in a high number of false positive predictions, which eventually drops the overall F1-score (row 3). This clearly testifies to the ability of the regularisation loss to enforce row-column voice link sparsity. Our preferred model (row 5) has the lowest standard deviation and the best average performance across runs.

7.5 DISCUSSION

In this section, we investigate what makes voice separation on string quartets so challenging and the reasons for our model's better performance. Compared to our other musical corpora, string quartets have a much different orchestration, which may include a bigger instrument range, more frequent and longer rests, accompaniment split to multiple voices (for example between the cello and the viola), octave doubling, and big jumps in pitch within one voice/instrument line. Some of these differences can be attributed to the fact that the individual voices in a string quartet are played by different instruments, which makes it easier for listeners to keep track of voices and gives the composer more freedom in composing a texture that is still comprehensible.

In Figure 7.3, we provide an example – bars 25-26 of Haydn's String Quartet Op17 No2, 1st movement – that demonstrates some of these properties together with our model's prediction and the ground truth. In this two-bar excerpt, we notice several challenging properties, such as octave jumps within the same voice, rests in between notes of the same voice, and voice crossings – i.e., the 1st violin (top voice or yellow colour in the pianoroll) starts lower than the viola and the 2nd violin

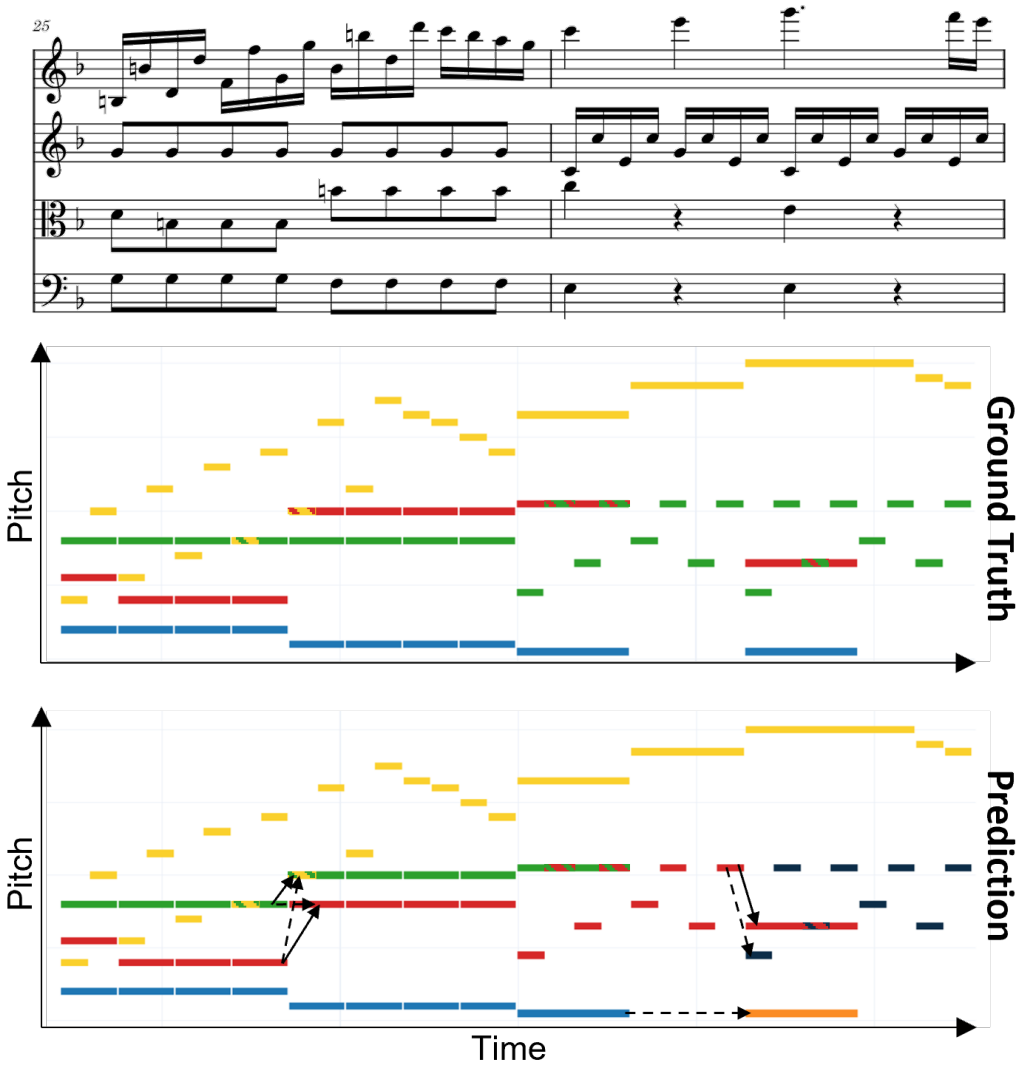


Figure 7.3: Ground truth and prediction for Haydn String Quartet Op17 No2, 1st mvt., bars 25-26. False negative errors are highlighted with dashed arrows and false positive with solid arrows.

but ends more than one octave higher than any other voice by measure 26. This passage is a real challenge for the voice separation algorithm, and it is solved in a very reasonable way by our context-aware network.

A closer look at our model's errors reveals three faulty sections. The first occurs between the viola and the 2nd violin on the third quarter beat of bar 25, where we observe a rhythmically identical voice crossing (red and green streams). This is indeed hard to discern without additional knowledge; human listeners would likely parse this correctly based on the specific sound (timbre) of the instruments, performing something more akin to Multi-Object Tracking (MOT), in our terminology. Another voice crossing mistake occurs between the same two voices on beat 3 of bar 26, where we incorrectly predict the start of a new voice (which is then continued in black). Finally, we notice a rest discontinuity mistake on the cello (bottom; blue and orange lines) on the 3rd beat of bar 26. These examples test the limits of our network and also raise a perceptual question, whether a human analyst would be able to

correctly assess these voice assignments from this bare representation, without additional perceptual clues. We leave this as an open question and a starting point for discussions on possibly more “natural" definitions of the voice separation task.

7.6 CONCLUSION AND FUTURE WORK

In this paper, we present a novel method for the perceptual problem of voice separation from symbolic music that achieves new state-of-the-art performance. We propose a formulation as a multi-trajectory tracking problem, and an end-to-end approach, based on heterogeneous neural networks, that does not rely on any heuristic or musical assumption that may be correct only for limited kinds of music. That allows us to handle traditionally complex corner cases such as voice inversions and overlaps. Furthermore, our approach can find a global voice separation solution on the entire piece, without pruning any potentially relevant option, with a complexity that is independent of the number of voices and scales polynomially with the number of notes in a piece. We also study the problem of reducing the dependence on any postprocessing algorithm and consider only the local, greedy predictions made by the neural network. We propose a new regularization loss that drastically improves our results in this setting, and may be useful for other general MTT scenarios.

Our work can be extended in a number of directions. We plan to address voice separation for unquantized MIDI, i.e., musical pieces obtained by a human recording; this would let us explore how much the expressive timing and intensity deviations introduced by the performer correlate with voice information. Also, we want to relax the monophonic voice assumption to target pieces where multiple simultaneous notes (i.e., chords) can be present in the same voice. Finally, we aim at developing a voice separation method from raw audio, which could also contain relevant information for the voice separation task that is not available in the symbolic format. In the case of multiple instruments performing multiple voices, this would blend into the field of instrument/source separation.

ACKNOWLEDGEMENTS

This work was supported by the European Research Council (ERC) under the EU’s Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab).

CONTRIBUTION STATEMENT

Equal contribution among the two first authors.

REFERENCES

[1] Lodewyk Van der Merwe and Pieter De Villiers. "Comparative investigation into Viterbi based and multiple hypothesis based track stitching." In: *IET Radar, Sonar & Navigation* 10.9 (2016), pp. 1575–1582.

[2] Mei Han, Wei Xu, Hai Tao, and Yihong Gong. "An algorithm for multiple object trajectory tracking." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE. 2004.

[3] Chee-Yee Chong, Greg Castanon, Nathan Cooprider, Shozo Mori, Ravi Ravichandran, and Robert Macior. "Efficient multiple hypothesis tracking by track segment graph." In: *Proceedings of the International Conference on Information Fusion*. IEEE. 2009.

[4] Gregory Castnnón and Lucas Finn. "Multi-target tracklet stitching through network flows." In: *Proceedings of the Aerospace Conference*. IEEE. 2011.

[5] Christopher Shooner, Srimant P Tripathy, Harold E Bedell, and Haluk Öğmen. "High-capacity, transient retention of direction-of-motion information for multiple moving objects." In: *Journal of Vision* 10.6 (2010), pp. 1–20.

[6] Edward Aldwell, Carl Schachter, and Allen Cadwallader. *Harmony and voice leading*. Cengage Learning, 2018.

[7] Xichu Ma, Xiao Liu, Bowen Zhang, and Ye Wang. "Robust Melody Track Identification in Symbolic Music." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[8] Francesco Foscarin. "The Musical Score: a challenging goal for automatic music transcription." PhD thesis. Paris, CNAM, 2020.

[9] Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter, Carlos Cancino-Chacón, Maarten Grachten, and Gerhard Widmer. "The match file format: Encoding Alignments between Scores and Performances." In: *Proceedings of the Music Encoding Conference (MEC)*. Halifax, Canada, 2022.

[10] Elaine Chew and Xiaodan Wu. "Separating voices in polyphonic music: A contig mapping approach." In: *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*. Springer. 2004.

[11] Søren Tjagvad Madsen and Gerhard Widmer. "Separating voices in MIDI." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2006.

[12] David Temperley. "A probabilistic model of melody perception." In: *Cognitive Science* 32.2 (2008), pp. 418–444.

[13] Ben Duane and Bryan Pardo. "Streaming from MIDI using constraint satisfaction optimization and sequence alignment." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2009.

[14] Anna Jordanous. "Voice separation in polyphonic music: A data-driven approach." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2008.

[15] Andrew McLeod and Mark Steedman. "HMM-based voice separation of MIDI performance." In: *Journal of New Music Research* 45.1 (2016), pp. 17–26.

[16] David Huron. "Tone and voice: A derivation of the rules of voice-leading from perceptual principles." In: *Music Perception* 19.1 (2001), pp. 1–64.

[17] Patrick Gray and Razvan C Bunescu. "A Neural Greedy Model for Voice Separation in Symbolic Music." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[18] Yo-Wei Hsiao and Li Su. "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 285–292.

[19] Emilios Cambouropoulos. "Voice separation: theoretical, perceptual and computational perspectives." In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*. Citeseer. 2006.

[20] Jürgen Kilian and Holger H Hoos. "Voice Separation-A Local Optimization Approach." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2002.

[21] Dimitris Rafailidis, Emilios Cambouropoulos, and Yannis Manolopoulos. "Musical voice integration/segregation: VISA revisited." In: *Proceedings of the Sound and Music Computing Conference (SMC)*. 2009.

[22] Emilios Cambouropoulos. "Voice and stream: Perceptual and computational modeling of voice separation." In: *Music Perception* 26.1 (2008), pp. 75–94.

[23] Guillem Brasó and Laura Leal-Taixé. "Learning a neural solver for multiple object tracking." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[24] Xinshuo Weng, Ye Yuan, and Kris Kitani. "Ptp: Parallelized tracking and prediction with graph neural networks and diversity sampling." In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4640–4647.

[25] Yongxin Wang, Kris Kitani, and Xinshuo Weng. "Joint object detection and multi-object tracking with graph neural networks." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[26] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." In: *IEEE Data Engineering Bulletin* 40.3 (2017), pp. 52–74.

[27] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[28] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[29] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. "Benchmarking graph neural networks." In: *arXiv preprint arXiv:2003.00982* (2020).

[30] Xavier Bresson and Thomas Laurent. "An Experimental Study of Neural Networks for Variable Graphs." In: *Proceedings of the International Conference on Learning Representations*. 2018.

[31] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018.

[32] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer. 2018, pp. 593–607.

[33] Rainer E Burkard and Eranda Cela. "Linear assignment problems and extensions." In: *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[34] He Liu, Tao Wang, Congyan Lang, Songhe Feng, Yi Jin, and Yidong Li. "GLAN: A Graph-based Linear Assignment Network." In: *arXiv preprint arXiv:2201.02057* (2022).

[35] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[36] David F Crouse. "On implementing 2D rectangular assignment algorithms." In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696.

[37] Anna Aljanaki, Stefano Kalonaris, Gianluca Micchi, and Eric Nichols. "MCMA: A Symbolic Multitrack Contrapuntal Music Archive." In: *Empirical Musicology Review* 16.1 (2021), pp. 99–105.

[38] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

8 POLYPHONIC VOICE SEPARATION

Title: Cluster and Separate: a GNN Approach to Voice and Staff Prediction for Score Engraving.

Published In Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR), San Francisco USA, 2024.

Authors: Francesco Foscarin*[1], Emmanouil Karystinaios*, Eita Nakamura, Gerhard Widmer

Abstract: This paper approaches the problem of separating the notes from a quantized symbolic music piece (e.g., a MIDI file) into multiple voices and staves. This is a fundamental part of the larger task of music score engraving (or score typesetting), which aims to produce readable musical scores for human performers. We focus on piano music and support homophonic voices, i.e., voices that can contain chords, and cross-staff voices, which are notably difficult tasks that have often been overlooked in previous research. We propose an end-to-end system based on graph neural networks that clusters notes that belong to the same chord and connects them with edges if they are part of a voice. Our results show clear and consistent improvements over a previous approach on two datasets of different styles. To aid the qualitative analysis of our results, we support the export in symbolic music formats and provide a direct visualization of our outputs graph over the musical score. All code and pre-trained models are available at https://github.com/CPJKU/piano_svsep

8.1 INTRODUCTION

The musical score is an important tool for musicians due to its ability to convey musical information in a compact graphical form. Compared to other music representations that may be easier to define and process for machines, for example, MIDI files, the musical score is characterized by how efficiently trained musicians can read it.

An important factor that affects the readability of a musical score for instruments that can produce more than one note simultaneously, is the separation of notes into different *voices* (see Figure 8.1). This division may follow what a listener perceives as independent auditory streams [1], which can also be reflected in a clearer visual rendition of a musical score [2]. A similar point can be made for the division into multiple staves (generally 2) for instruments with a large pitch range, such as piano, organ, harp, or marimba. We will consider in this paper piano music.

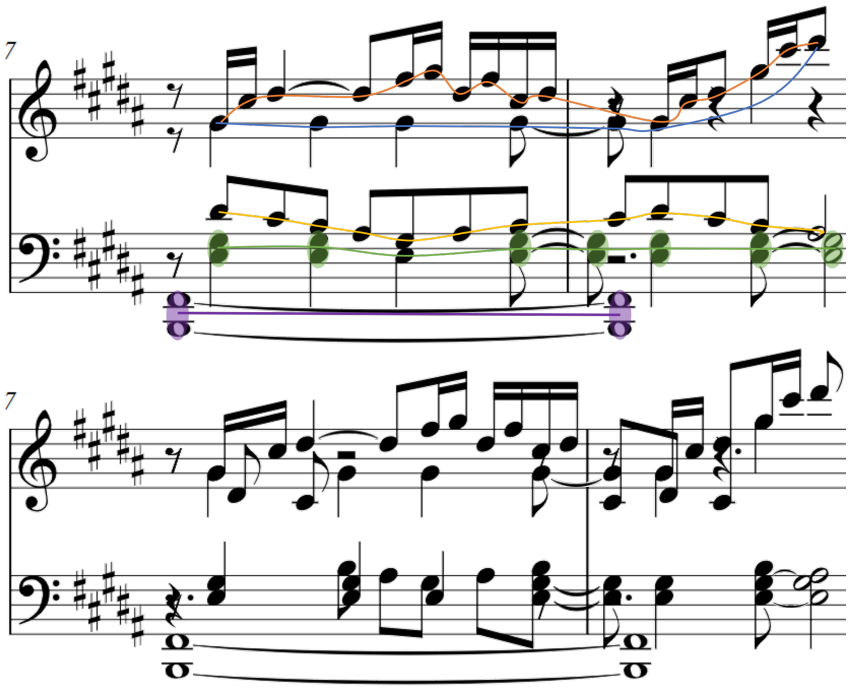


Figure 8.1: Comparing different voice/staff assignments for two bars from C. Debussy's Estampes - Pagodes. (top) original; voices can be inferred from the beam grouping and (horizontal lines connecting notes), rests, and stem sharing, and are colored for clarity. (bottom) hard-to-read rendition with voice and staff assigned according to heuristics we propose as a baseline.

The term voice is frequently used to describe a sequence of musical notes that do not overlap, which we call a *monophonic voice*. However, this definition may be insufficient when considering polyphonic instruments. The voices could contain *chords*, which are groups of *synchronous notes* (i.e., notes with the same onset and offset) and are perceived as a single entity [3]. We name a voice that can contain chords a *homophonic voice*. Note that notes partially overlapping notes cannot be part of a homophonic voice.

Music encoded in MIDI (or similar) formats, even when containing quantized notes, time signature, or bar information, often does not contain voice and staff information. The same can be said for the output of music generation [4], transcription [5], or arranging [6] systems. Therefore, such music cannot be effectively converted into a musical score, to be efficiently read and played by human musicians.[2] The tasks of producing voice and staff information from unstructured symbolic music input are called *voice separation* (or voice segregation in some papers [3]) and *staff separation*, respectively.

Most of the existing approaches to voice separation have focused only on music with monophonic voices [7–12], which is not sufficient for our goal of engraving[3] piano music. The task of homophonic voice separation is much harder to solve: the presence of chords within voices makes the space of solutions grow much bigger;

2 Voice and staff separation are only two of the multiple elements, such as pitch spelling, rhythmic grouping, and tuplet creations, which need to be targeted by a score engraving system, but we will only focus on the former two in this paper.

3 "score engraving" and "score typesetting" are used interchangingly.

and the choice of the "true voice separation" can be ambiguous, with multiple valid alternatives among which experts may disagree.

The existing approaches to homophonic voice separation can be divided into two groups: the first [1, 3, 5, 13] use dynamic programming algorithms based on a set of heuristics, which makes for systems that are controllable and interpretable, but also hard to develop and tune. Such systems are often prone to fail on exceptions and corner cases that are present in musical pieces. The second group of approaches [14–17] applies deep learning models to predict a voice label for each note. Such an approach creates two fundamental issues: i) the necessity of setting a maximum number of voice labels, and ii) a (highly) unbalanced ratio of occurrence of some voice labels. Moreover, all these approaches assume that a voice cannot move between the two staves, which is not true for complex piano pieces.

In this work, we propose a novel system for homophonic voice separation that can efficiently and effectively assign notes to voices and staves for polyphonic music engraving. Efficiency is ensured by a graph neural network (GNN) encoder, which can create input embeddings with a relatively small number of parameters. Effectiveness is targeted by approaching voice prediction not as a note labeling, but as an edge prediction problem [12], which solves the maximum voice number and the label imbalance problems presented above. Our system predicts staff and voice separately and does not make any assumption on the number of voices; therefore it can deal with cross-staff voices and complex corner cases. We avoid the problem of ground truth ambiguity since we focus specifically on voice separation for musical score engraving, therefore we can extract the (unique) ground truth directly from digitized musical scores.

We evaluate our system on two piano datasets of different difficulty levels, one containing popular, the other classical music. A comparison with a baseline and the approach of Shibata et al. [5] shows a consistent improvement in performance on both datasets. Finally, we develop a visualization tool to display the input and output of our system directly on the musical score, and discuss some predictions and comments on homophonic voice separation.

8.2 RELATED WORK

The most influential work for this paper is the monophonic voice separation system by Karystinaios et al. [12]. Similarly, we consider voice separation a edge prediction task and use a similar score-to-graph routine and the same GNN encoder. Differently from that work, we consider homophonic voices and staves and, therefore, we extend the model formulation, the deep learning architecture, and the postprocessing routine to deal with this information.

Shibata et al. [5] developed a voice and staff separation technique applied after music transcription to quantized MIDI files. It works in two stages: first, an HMM separates the notes of the two hands (which will then be used as staff), and then a dynamic programming algorithm that maximizes the adherence to a set of heuristics is applied to separate voices in the two hands independently. We compare against this method since it is the most recent approach focusing specifically on homophonic voice separation.

There are some approaches based on neural networks [14–17], but they never perform this task in isolation, but rather in combination with other tasks such as symbolic music transcription, full scorification, and automatic arrangement. This means that they can only train on a much smaller dataset and a comparison would not be fair.

All the approaches mentioned before, except [12], perform voice separation as a label prediction task, which is problematic, as discussed in the introduction, due to the label imbalance and choice of the maximum number of voices. The former is particularly problematic for the neural network approaches.

8.3 METHODOLOGY

Our system inputs data in the form of a set of quantized notes (e.g., coming from a quantized MIDI or a digitized musical score), each characterized by pitch, onset, and offset. This information is modeled as a graph, which we call *input graph*, and then passed through a GNN model to predict an *output graph* containing information about voices, staves, and chord groupings. We remind the reader that in our 'homophonic voice' scenario, chords are groups of synchronous notes that belong to the same voice.

8.3.1 Input Graph

From the set of quantized notes representing a musical piece we create a directed heterogeneous graph [18] $G_{in} = (V, E_{in}, \mathcal{R}_{in})$ where each node $v \in V$ corresponds to one and only one note, and the edges $e \in E_{in}$ of type $r \in \mathcal{R}_{in}$ model temporal relations between notes [12]. $\mathcal{R}_{in}$ includes 4 types of relations: onset, during, follow, and silence, corresponding, respectively, to two notes starting at the same time, a note starting while the other is sounding, a note starting when the other ends, and a note starting after a time when no note is sounding. We also create the inverse edges for during, follows, and silence relations. Each node corresponds to a vector of features: one of the 12 note pitch classes[4] (C, C#, D, etc.), the octave in $[1, \ldots, 7]$, the note duration, encoded as a float value $d \in [0, 1]$ computed as the ratio of the note and bar durations, passed through a tanh function to limit its value and boost resolution for shorter notes, as proposed in [12]. We don't consider grace notes in our system, and we remove them from the input notes.

8.3.2 Output Graph

The output graph $G_{out} = (V, E_{out}, \mathcal{R}_{out})$ has the same set V of nodes as the input graph, but a staff number in $\{0, 1\}$ is assigned to every node. There are two edge types in E_{out}: chord and voice, i.e. $\mathcal{R}_{out} = \{\text{"chord"}, \text{"voice"}\}$.

Voice edges [8, 12] are an alternative in the literature to the more straightforward approach of predicting a voice number for every note; the usage of voice edges

4 We don't consider tonal pitch classes [19] since they are not specified in MIDI files which we assume to be our input.

has the advantage of enabling a system to work with a non-specified number of voices, and avoiding the label imbalance problem for high voice numbers. Voice edges are directed edges that connect consecutive notes (without considering rests) in the same voice. Formally, let $u_1, u_2 \in V$ be two notes in the same voice then $(u_1, \text{"voice"}, u_2) \in E_{out}$ if and only if $\text{offset}(u_1) \leq \text{onset}(u_2)$ and $\nexists\, u_3 \in V$ within the same voice such that $\text{offset}(u_1) \leq \text{onset}(u_3) < \text{onset}(u_2)$.

The previous definition also holds in our setting with homophonic voices. Let us extend the definition of *chord* (a set of synchronous notes) to include the limit case of a single note. Two chords are consecutive if any two notes, respectively, from the first and second chords are consecutive. In the case of two consecutive chords with m and n notes in the same voice, there will be $m * n$ voice edges.

Chord edges are undirected and connect all notes that belong to the same chord without self-loops, so for a n-note chord, there are $n(n-1)$ edges. They serve to unambiguously identify which notes together form a single chord.

The same output graph can be created from an already properly engraved score. To obtain the graph we only need to draw the true voice edges between consecutive notes in the same voice within a bar and for chord edges we draw the chord ground truth between synchronous notes with the same voice number assignment. This graph can subsequently serve as the ground truth during training.

8.3.3 Problem Simplification

In this section, we apply some obvious musical constraints to reduce computation and memory usage in our pipeline, without impacting the results. Let us first focus on *chord edge* prediction. Given the simple constraint that all notes of a chord must start and end simultaneously, we can restrict the chord edge prediction process to only consider pairs of sychronous notes (same onset and offset values) as candidates. We do this by creating a set of *chord edge candidates* Λ_c which are calculated automatically and associated with our input graph. Only notes connected by such candidate edges will be considered in the chord prediction part of the model (see next section).

The same reasoning can be applied to the voice edges, by creating a set of *voice edge candidates* Λ_v such that $\forall u_1, u_2 \in V,\ (u_1, \text{"voice"}, u_2) \in \Lambda_v$ only when $\text{offset}(u_1) > \text{onset}(u_2)$. Another step can be taken towards reducing the number of candidates in the set Λ_v by incorporating some musical engraving considerations.

The separation of notes in multiple voices does not have to be consistent in the whole score, but only within each bar, to produce the intended visual representation. There are no graphical elements that show if two notes in different bars are or are not in the same voice[5]. Music engraving software does not force users to use consistent voices across bars. This can be often observed in digitized musical scores where music motives that belong to the same voice, are assigned different voices in different bars. Such observations have motivated projects such as the Symbolic Multitrack Contrapuntal Music Archive [20] that explicitly encode a "global" voice number.

5 This may change for cross-bar beamings, but they are very rarely used in standard music notation (there are no occurrences in our datasets) and therefore we do not consider them in this work.

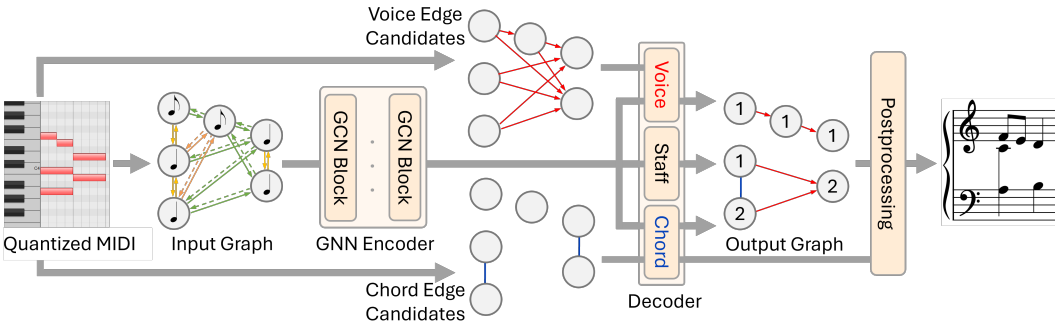


Figure 8.2: Our Architecture. For simplification, we display the output graph as having "hard" voice predictions, while these are probabilities over voice candidates.

Since cross-bar consistency is not necessary for our goal of engraving (and is often wrongly annotated in our data) we limit the *voice edge candidates* Λ_v to contain only pairs of notes that occur within the same bar. This design choice is also reflected in our evaluation, i.e. we do not evaluate how the voices propagate across bars, but only within each bar. Note that this process is different from processing each bar independently since our network (detailed in the next section) considers music content across bars.

8.3.4 Model

We design an end-to-end model (see Figure 8.2) that receives an input graph as described in Section 8.3.1 and produces an output graph as in Section 8.3.2. The model is organized as an encoder–decoder architecture.

The encoder receives an input graph created from a quantized MIDI score and passes it through three stacked Graph Convolutional Network (GCN) blocks to produce a node embedding for each note. We use the heterogeneous version of the Sage convolutional block [18] with a hidden size of 256; the update function for each node u is described by:

$$\begin{aligned} \mathbf{h}_{\mathcal{N}(u)}^{(l+1)} &= \sum \left(\{ \mathbf{h}_v^l, \forall v \in \mathcal{N}(u) \} \right) \\ \mathbf{h}_u^{(l+1)} &= \sigma \left(\mathbf{W} \cdot \text{concat}(\mathbf{h}_u^l, \mathbf{h}_{\mathcal{N}(u)}^{l+1}) \right) \end{aligned} \tag{8.1}$$

where $\mathcal{N}(u)$ are the neighbors of node u, σ is a non-linear activation function, $\mathbf{W}$ is a learnable weight matrix.

The decoder consists of three parts that all use the same node embedding as input: i) a staff predictor; ii) a voice edge predictor; and iii) a chord clustering (i.e., a chord edge predictor). The *staff predictor* is a 2-layer Multi-Layer Perceptron (MLP) classifier that produces probabilities for each graph node (i.e., each note) to belong to the first or second staff. The *voice edge predictor* receives the embeddings of pairs of notes connected by edge candidates and produces a probability for each pair to be in the same voice. It works by concatenating the pairs of note embeddings and applying a 2-layer MLP. The final decoder part, *chord clustering*, receives the embeddings of pairs of notes connected by chord edge candidates (i.e., pairs of

synchronous notes) and produces the probability for a pair to be merged into a chord. This is achieved by computing the cosine similarity between the elements of the pair. This process forces the node embeddings created by the decoder to be similar to each other for notes of the same chord, which helps the voice predictor produce consistent voice edge probabilities for notes of the same chord. We apply a threshold to pass from probabilities to decisions on which notes to cluster.

The complete model contains $\sim$ 3M parameters and we train it end-to-end with the (unweighted) sum of three *Binary Cross Entropy* loss functions, one for each task.

8.3.5 Postprocessing

A straightforward approach to deciding whether to connect two notes with a voice edge would be to threshold the predicted voice edge probabilities. However, even when using edge and chord candidates, we could still produce three kinds of invalid output: (1) multiple voices merging into one voice, (2) one voice splitting into multiple voices, and (3) notes in the same chord that are not in the same voice. To eliminate these issues, we add a postprocessing phase that accompanies our model and guarantees a valid output according to music engraving rules.

The first step, which we call *chord pooling*, merges all nodes that belong to the same chord to a single new "virtual node". This is done by looking for the connected components considering only chord edges in the output graph, then *pooling* in a single node all original nodes in each connected component, creating a new node which has as incoming and outgoing voice edges all edges entering and exiting the original nodes, respectively. If multiple edges collapse in one edge (e.g. in the case of two consecutive chords in the same voice), the new edge has a probability that is the average of the corresponding edge probabilities.

After the first step, we are left with monophonic streams, which could still exhibit problems (1) and (2). We can solve this with the technique proposed in [12] for monophonic voices, i.e. by framing the voice assignment problem as a linear assignment problem [21] over the adjacency matrix obtained by the updated edge candidates Λ'_v. We follow the linear assignment step by unpooling or unmerging the nodes that were previously pooled, in this way, obtaining the original nodes again. During unpooling, the incoming edges and outgoing edges of the "virtual nodes" are reassigned to each original node, thus resolving problem (3).

The complete postprocessing method is depicted in Figure 8.3. It is worth noting that the staff labels are not considered during the postprocessing phase, and we copy them unchanged to the postprocessed output graph.

8.3.6 Evaluation

We evaluate the predicted voice assignments with the metric proposed by Hiramatsu et al. [15], which formalizes the metric of McLeod and Steedman [22]. This is a version of the F1-score for voice separation [8] which is adapted to work on homophonic voices, by reducing the importance of notes if they are part of a chord. This is important since chords create many voice edges (e.g., two 4-note chords in the same voice are connected by 16 edges), which could potentially overshadow the importance of edges in monophonic voices (or voices with fewer/smaller chords).

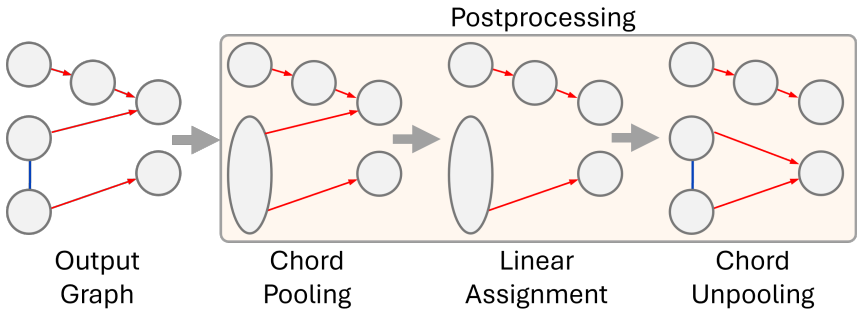


Figure 8.3: Output graph postprocessing. We do not display the predicted staff labels.

Formally, the homophonic voice F1-score $F1$ is calculated as:

$$P = \frac{\sum_{i<j} a_{ij}\hat{a}_{ij}/\hat{w}_i}{\sum_{i<j} \hat{a}_{ij}/\hat{w}_i}, \quad R = \frac{\sum_{i<j} a_{ij}\hat{a}_{ij}/w_i}{\sum_{i<j} a_{ij}/w_i}$$
$$F1 = \frac{2PR}{P+R} \tag{8.2}$$

where $i < j$, in the sum, considers all pair of notes i, j such that $\text{offset}(i) < \text{onset}(j)$; a_{ij}, $\hat{a}_{ij}$ are equal to 1 or 0 if a voice edge exists or not in the ground truth and predictions, respectively; and w_i and $\hat{w}_i$ are the number of notes that are chorded together with the note i in the ground truth and predictions, respectively. Unlike [15], we consider only notes j in the same bar of i, for the reasons presented in Section 8.3.3. We evaluate the staff prediction part of our model with binary accuracy, and we assess chord prediction with the F1 score computed on the chord edges.

8.3.7 From Network Prediction to Readable Output

The computation of voice and staff numbers is sufficient for the system evaluation, but not for producing a usable tool, which we are interested in in this paper. The missing step, to be described in this section, is the integration of the network predictions into a readable musical score. To achieve this integration we need to undertake two essential steps: beam together notes within the same voice, and infill rests to "fill holes" within each voice.

For the first step, we proceed according to the rules of engraving [2]. We beam two consecutive notes (or chords) in the same voices if their duration is less than a quarter note (excluding ties) unless they belong to different beats. Following the music notation convention we consider the compound time signatures, i.e., $\frac{6}{x}, \frac{9}{x}, \frac{12}{x}$ to have, respectively, 2,3, and 4 beats. When confronted with tied notes, the algorithm prioritizes producing notations with the fewest number of notes, an heuristic with promotes easier-to-read notation [23].

The second step consists of introducing rests so that each voice fills the entire bar and can be correctly displayed. Some rests could be set as invisible to improve the graphical output when their presence and duration are easy to assume from other score elements, but we display all of them for simplicity. As for the notes, we

Dataset	J-pop Dataset			DCML Romantic Corpus		
	Staff Acc	Chord F1	Voice F1	Staff Acc	Chord F1	Voice F1
Baseline	89.9	86.9	85.4	80.7	65.2	78.2
Shibata et al. [5]	92.8	-	92.2	88.5	-	84.9
GNN wo Chord wo Post	**96.5** $\pm$ 0.1	-	95.2 $\pm$ 1.9	**91.5** $\pm$ 0.1	-	87.2 $\pm$ 3.3
GNN wo Post	96.3 $\pm$ 0.1	**94.9** $\pm$ 0.1	95.7 $\pm$ 0.4	91.0 $\pm$ 0.1	**79.5** $\pm$ 0.4	88.9 $\pm$ 0.4
GNN	96.3 $\pm$ 0.1	**94.9** $\pm$ 0.1	**96.6** $\pm$ 0.1	91.0 $\pm$ 0.1	**79.5** $\pm$ 0.4	**89.9** $\pm$ 0.2

Table 8.1: Metrics for our the J-Pop and DCML test sets. "GNN" denotes our method, without postprocessing ("GNN wo Post"), and without both postprocessing and chord prediction parts ("GNN wo Chord wo Post"). All GNN model runs are repeated 5 times: $\pm$ refers to the standard deviation of results across runs.

choose the rest types (with eventual dots) to minimize the number of rests in the score.

The two steps described above cover common cases and produce a complete score in MEI format [24]. However, the score export is still a prototype, since developing one that is robust against all corner cases is an extremely complex task, and is outside the scope of this paper. Since score output problems may obscure the output of our system, we also develop a graph visualization tool. Both the input and output graphs (including the candidate edges) can be displayed on top of the musical score in an interactive web-based interface based on Verovio [25]. Some examples of the output graph visualization are in Figure 8.4.

8.4 EXPERIMENTS

We train our model with the ADAM optimizer with a learning rate of 0.001 and a weight decay of $5 * 10^{-4}$ for 100 epochs. For a quantitative evaluation, we compare our results with those of a baseline algorithm and the method proposed by Shibata et al. [5], on two rather diverse datasets.

Our baseline algorithm assigns all notes under C4 (middle C) to the second staff and the rest to the first. Then it groups all synchronous notes (per staff) as chords. Finally, it uses the time and pitch distances between the candidate pairs of notes as weights to be minimized during the linear assignment process (the same as we use in our postprocessing) which creates the voice edges.

8.4.1 Datasets

We use two piano datasets of different styles and difficulties to evaluate our system under diverse conditions. The ability to handle complex corner cases should not reduce the performance on easier (and more common) pieces.

The *J-Pop* dataset contains pop piano scores introduced by [5]. Most of the scores consist of accompaniment chords on the lower staff and some simple melodic lines on the upper staff. The dataset contains 811 scores; we randomly sampled 159 (roughly 20%) of these for testing and used the rest for training and validation.

The *DCML Romantic Corpus* is more challenging. It was created by [26] and contains piano pieces from the 17th to 20th centuries with some virtuosic quality. It includes characteristics such as cross-staff beaming, a high number of voices, challenging voicing, etc. Similarly to the pop dataset we randomly sample 77 out of the 393 scores (approx. 20%) and use the rest for training and validation.

The *J-Pop* dataset is available in MusicXML format, while the *DCML Romantic Corpus* is in Musescore file format. We use Musescore to convert DCML files to MusicXML and load them with the Python library Partitura [27] to extract the note list.

8.4.2 Results

Our model aims to be generic across a variety of music, therefore we train a single model on the joined training set of pop and classical scores, not two individual ones. The rules that govern the handling of voices may be fundamentally different in the two datasets, but we assign to the model the task of handling these differences. This approach ensures better future scalability on bigger and more diverse datasets. We compute the metrics separately on the test part of our two datasets.

Table 8.1 reports results for three versions of our graph-based model: the complete model from Section 8.3, a variant without postprocessing, and a variant without chord prediction and postprocessing (our postprocessing technique cannot be run without the chord prediction task, since it pools nodes that belong to the same predicted chord). The method of Shibata requires the specification of the number of voices per staff. For compactness, we report only the results with one voice per staff (2 voices total); the results degrade by increasing the number of voices.

Our results show that even our system without pooling and without postprocessing obtains consistently better results than both Shibata et al. [5] and our baseline. Interestingly, the chord prediction task improves the Voice F1 results even when the post-processing is not used; this confirms the benefits of multi-task training, and of enforcing notes of the same chord to have similar representations in the hidden space, with cosine similarity, to predict coherent voice edges. However, we observe a reduction in staff accuracy, probably for the same reason, since the same hidden representation is also used to predict chords, making it harder (though not impossible) to split notes of the same chord in different staves. When the full system is used, there are further improvements in Voice F1.

We are also evaluate our system on the bar-level and study performances for music excerpts of varying difficulties. We compute the voice F1 score for each bar and average them based on the number of voices in the ground truth. We compare with Shibata et al. [5] with 1 & 2 voices per staff (vps). Table 8.2 shows the results for the DCML Romantic Corpus. Both our model and [5] perform best with 2 voices, the most common number in our dataset. Interestingly, Shibata et al. approach with 2 vps never outperforms vps 1, not even when the target number of voices is 3 or 4, a situation that vps 1 cannot handle. This can be explained by the fact that Shibata et al. parameters were tuned on a simpler dataset, and accepting more voices creates more errors than benefits. Setting vps > 2 consistently degraded the performances, probably also for similar reasons.

8.4.3 Qualitative Analysis

Let us take a closer look into the predictions of our deep-learning approach (GNN) on the excerpt of Figure 8.4 produced by our visualization tool. Our approach captures correctly the cross-staff voice in the first two bars, while such a situation causes performance degradation for all other voice separation approaches that don't support it. We observe some disagreements with the original score in Measure 3: our model predicts a single chord (instead of splitting across the staff) containing all the synchronous syncopated quarter notes, and also mispredicts the staff for the first D#4 note. A more in-depth study of why this happens is not trivial, as neural

#Voices	#Bars	**GNN**	**[5] 1vps**	**[5] 2vps**
1	322	96.6	88.3	87.9
2	4576	94.1	89.3	88.1
3	2464	89.0	84.2	81.5
4	719	81.6	80.5	75.1
5	99	81.6	76.7	73.7
6	17	78.4	68.9	61.6

Table 8.2: Voice F1 score aggregated by bars with the same number of voices in the ground truth, on the DCML Romantic Corpus. Shibata et al. [5] is used with 1 and 2 voices per staff (vps).

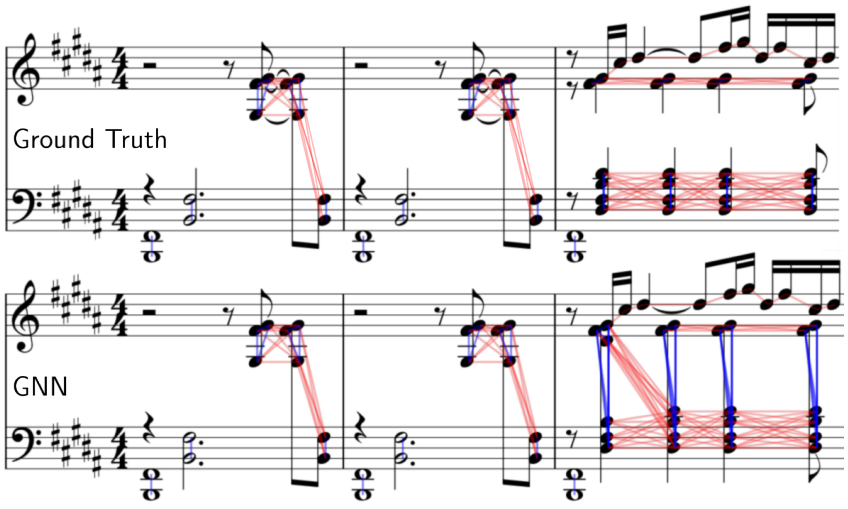


Figure 8.4: Comparison of voice and staff assignment between the original score and our method on the first bars of C. Debussy's Estampes-Pagodes. Voice edges (red) and chord edges (blue) are drawn for the original score (Ground Truth) and our proposed approach (GNN).

networks are not interpretable. This is a drawback compared to heuristic-based separation techniques.

Synchronous notes with the same pitch are problematic. Our system can predict different voices for these notes, while Shibata et al. always predict them as a chord in the same voice, and this reduces the performances for pieces that contain a lot of them, like Schumann Kinderszenen Op.15. For fairness, we should note that we should expect the output of a music transcription system to only contain one of these notes, instead of multiple like in our current input. An enhancement of our system would then be able to receive a single note as input, assign multiple voices to it (with multiple incoming and outgoing edges) and then split it into multiple notes. Another current limitation of our system is the missing support for grace notes, which in the actual version are ignored and removed from the input.

8.5 CONCLUSION AND FUTURE WORK

This paper presented a novel graph-based method for homophonic voice separation and staff prediction in symbolic piano music. Our experiments highlight our system's effectiveness compared to previous approaches. Notably, we obtained consistent improvements over two datasets of different styles with a single model.

Future work will focus on integrating grace notes and the possibility of multiple voices converging on a single note. We aim to create a framework that produces complete engravings from quantized MIDI, including the prediction of clef changes, beams, pitch spelling, and key signatures.

8.6 ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab).

REFERENCES

[1] Emilios Cambouropoulos. "Voice and stream: Perceptual and computational modeling of voice separation." In: *Music Perception* 26.1 (2008), pp. 75–94.

[2] Elaine Gould. *Behind bars: the definitive guide to music notation*. Faber Music Ltd, 2016.

[3] Dimos Makris, Ioannis Karydis, and Emilios Cambouropoulos. "VISA3: Refining the voice integration/segregation algorithm." In: *Proceedings of the Sound and Music Computing Conference*. 2016.

[4] Yu Siang Huang and Yi Hsuan Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions." In:

Proceedings of the 28th ACM International Conference on Multimedia. 2020. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. arXiv: 2002.00212.

[5] Kentaro Shibata, Eita Nakamura, and Kazuyoshi Yoshii. "Non-local musical statistics as guides for audio-to-score piano transcription." In: *Information Sciences* 566 (2021), pp. 262–280.

[6] Moyu Terao, Eita Nakamura, and Kazuyoshi Yoshii. "Neural Band-to-Piano Score Arrangement with Stepless Difficulty Control." In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[7] Elaine Chew and Xiaodan Wu. "Separating voices in polyphonic music: A contig mapping approach." In: *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*. Springer. 2004.

[8] Ben Duane and Bryan Pardo. "Streaming from MIDI using constraint satisfaction optimization and sequence alignment." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2009.

[9] Patrick Gray and Razvan C Bunescu. "A Neural Greedy Model for Voice Separation in Symbolic Music." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[10] Andrew McLeod and Mark Steedman. "HMM-based voice separation of MIDI performance." In: *Journal of New Music Research* 45.1 (2016), pp. 17–26.

[11] Yo-Wei Hsiao and Li Su. "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 285–292.

[12] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[13] Jürgen Kilian and Holger H Hoos. "Voice Separation-A Local Optimization Approach." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2002.

[14] Masahiro Suzuki. "Score Transformer: Generating Musical Score from Note-level Representation." In: *Proceedings of the 3rd ACM International Conference on Multimedia in Asia*. 2021, pp. 1–7.

[15] Yuki Hiramatsu, Eita Nakamura, and Kazuyoshi Yoshii. "Joint Estimation of Note Values and Voices for Audio-to-Score Piano Transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 278–284.

[16] Lele Liu, Qiuqiang Kong, GV Morfi, Emmanouil Benetos, et al. "Performance MIDI-to-score conversion by neural beat tracking." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[17] Jingwei Zhao, Gus Xia, and Ye Wang. "Q&A: Query-Based Representation Learning for Multi-Track Symbolic Music re-Arrangement." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[18] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." In: *IEEE Data Engineering Bulletin* 40.3 (2017), pp. 52–74.

[19] Francesco Foscarin, Nicolas Audebert, and Raphaël Fournier-S'Niehotta. "PKSpell: Data-driven pitch spelling and key signature estimation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[20] Anna Aljanaki, Stefano Kalonaris, Gianluca Micchi, and Eric Nichols. "MCMA: A Symbolic Multitrack Contrapuntal Music Archive." In: *Empirical Musicology Review* 16.1 (2021), pp. 99–105.

[21] Rainer E Burkard and Eranda Cela. "Linear assignment problems and extensions." In: *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[22] Andrew McLeod and Mark Steedman. "Evaluating automatic polyphonic music transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018, pp. 42–49.

[23] Francesco Foscarin, Florent Jacquemard, and Philippe Rigaux. "Modeling and learning rhythm structure." In: *Sound and Music Computing Conference (SMC)*. 2019.

[24] Perry Roland. "The music encoding initiative (MEI)." In: *Proceedings of the First International Conference on Musical Applications Using XML*. Vol. 1060. Citeseer. 2002, pp. 55–59.

[25] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. "Verovio: A library for Engraving MEI Music Notation into SVG." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[26] Johannes Hentschel, Yannis Rammos, Fabian C Moss, Markus Neuwirth, and Martin Rohrmeier. "An annotated corpus of tonal piano music from the long 19th century." In: *Empirical Musicology Review* 18.1 (2023), pp. 84–95.

[27] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

9 SYMBOLIC MUSIC GRAPH EXPLANATIONS

Title: SMUG-Explain: A Framework for Symbolic Music Graph Explanations

Published In Proceedings of the Sound and Music Conference (SMC), Porto Portugal, 2024.

Authors: Emmanouil Karystinaios, Francesco Foscarin, Gerhard Widmer

Abstract: In this work, we present Score MUsic Graph (SMUG)-Explain, a framework for generating and visualizing explanations of graph neural networks applied to arbitrary prediction tasks on musical scores. Our system allows the user to visualize the contribution of input notes (and note features) to the network output, directly in the context of the musical score. We provide an interactive interface based on the music notation engraving library Verovio. We showcase the usage of SMUG-Explain on the task of cadence detection in classical music. All code is available on https://github.com/manoskary/SMUG-Explain.

9.1 INTRODUCTION

In recent years, Graph Neural Networks (GNNs) have emerged as a method for processing musical scores in Music Information Research (MIR) applications, such as cadence detection[1], expressive performance rendering[2], optical music recognition [3], music generation [4], voice separation[5], and Roman numeral analysis [6]. Like the majority of deep learning-based approaches, GNNs are not intrinsically interpretable, thus making it impossible to inspect the system to reveal potential issues of the system itself and the data it uses or to gain knowledge about the specific task [7]. One could modify the system to make it more interpretable, but this often leads to a reduction in performance. A popular alternative is the so-called *post-hoc* method which aims to explain already trained deep models.

In the field of MIR, multiple explanation techniques have been proposed in recent years [8–12], but they all target systems which use matrix-like inputs, such as spectrograms or pianorolls. In this paper, instead, we focus on the explainability of MIR systems that use graphs as input and process musical scores. We argue that graph explanations for scores are more musically interpretable since they point to individual note elements and their neighborhood in the score. We experiment with various post-hoc gradient-based explanation methods for GNNs from the literature [13–19]. The generated explanations are quantitatively evaluated by verifying that our explanations satisfy the sufficiency and necessity conditions from a GNN point of view [18]. We name our framework Score MUsic Graph (SMUG)-Explain.

Once the explanations are produced, there remains the question of how to present them to the user in an effective way. We display them directly on the musical score, to promote a clear and intuitive relation between the system input, the output and the explanations. Moreover, our use case is more complex than the (more common) explanation of global classifiers (i.e., systems that predict one label for each input excerpt). We want to be able to target systems that predict labels for multiple elements in the input, e.g., for every note or for every time step. This calls for an interactive interface that enables users to focus on specific subsets of predictions. We start with a web-based graphical rendition of a digitized musical score, based on the engraving library Verovio [20]. The user can interact with individual notes to trigger the visualization of explanations of the target GNN model. Specifically, each note is associated with the subgraph that most contributes to the underlying model's prediction. Furthermore, for each selected note, the user can visualize its feature importance, i.e. the relative importance of input features of the notes involved in the explanation.

We showcase SMUG-Explain on the cadence detection model of Karystinaios et al. [1], by comparing different explanation techniques, and by a qualitative analysis of three music excerpts. GNNs excel at capturing intricate relationships and patterns within graph-structured data. In music, that could be related to voice-leading, harmonic relations, melodic patterns, and other elements that are implicitly modeled by the score graph. From a musicological point of view, some parallelism can be found between our explainer and an analyst who highlights the voice leading and most important notes that contribute to the analysis assessment. In particular, some analysis methodologies, such as Schenkerian analysis or GTTM [21], also create graph structures between notes. In the second part of the paper, we attempt to shed some insight into more musical interpretations of graph explanations. To achieve this, we apply our framework to several music excerpts and comment on the generated explanations. We believe that explanations of deep analytical models might contain some musical pointers that can correlate with expert musical analyses.

9.2 PRELIMINARY CONCEPTS

In this section, we present a common task-independent approach (as emerging from recent papers on the topic) to modelling musical scores with GNNs, then we give some information on explainability techniques for GNNs and their evaluation.

9.2.1 GNN-based Approaches on Musical Scores

The fundamental idea of GNN-based approaches to musical scores is to model a musical score as a graph where notes are the vertices and edges model the temporal relation between the notes.[1] The most common approach [1, 2, 5, 6] to create a graph from a musical score considers four types of edges (see Figure 9.1 for visualization on the score):

1 More elements could be used as vertices, such as rests, bar lines, and dynamics symbols, but these are not commonly used and we do not consider them in this paper.

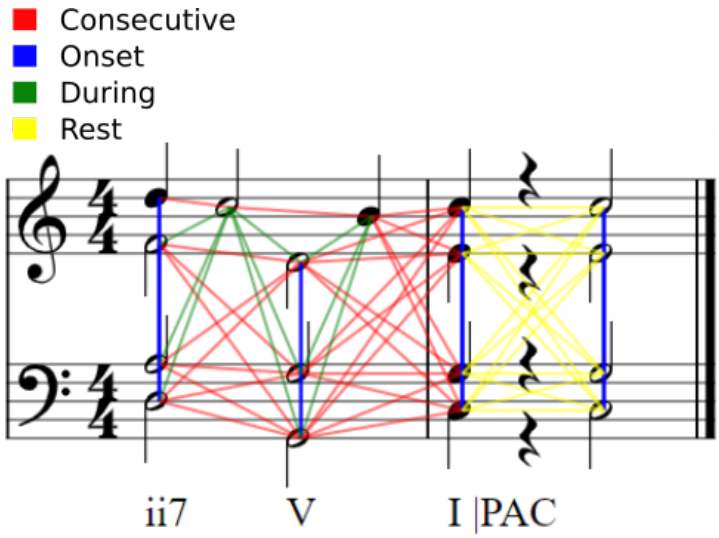


Figure 9.1: An example of a score graph depicting the different graph edge types in different colours.

- *onset edges*: connect notes that share the same onset;
- *consecutive edges* (or *next edges*): connect a note x to a note y if the offset of x corresponds to the onset of y;
- *during edges*: connect a note x to a note y if the onset of y falls within the onset and offset of x;
- *rest edges* (or *silence edges*): connect the last notes before a rest to the first ones after it.

The GNN can treat these four edge types with a single representation [1], thus considering a *homogeneous* input graph, or can treat them as different representations per edge type, i.e. as a *heterogeneous* graph.

Adopting the latter approach, a score graph is represented as an attributed heterogeneous graph $G = (V, E, \mathcal{R}, X)$, where V is the set of nodes representing the notes in a score. E is the set of typed edges with elements of the form (v, τ, u) where, $v, u \in V$ and $r \in \mathcal{R}$ is a relation type. Finally, $X \in \mathcal{R}^{V \times k}$ is a feature matrix such that every node u has its corresponding feature vector $x_u \in X$. Furthermore, we additionally can construct an adjacency matrix $A \in V \times V$.

In our work, we apply the GraphSAGE convolutional block [22]. For a node v the features message passing process per layer l is described as follows:

$$\begin{aligned} h_{\mathcal{N}(v)}^{(l+1)} &= \text{aggregate}\left(\{h_j^l, \forall j \in \mathcal{N}(v)\}\right) \\ h_v^{(l+1)} &= \sigma\left(W \cdot \text{concat}(h_v^l, h_{\mathcal{N}(v)}^{l+1})\right) \\ h_v^{(l+1)} &= \text{norm}(h_v^{(l+1)}) \end{aligned} \tag{9.1}$$

Where W is a learnable weight, $\mathcal{N}(v)$ is the set of neighbors of v, aggregate is a permutation invariant aggregation function such as mean, sum, or max, and σ is an activation function such as ReLU.

9.2.2 Explainability and Graphs

As explained in the introduction, we are interested in *post-hoc* methods, i.e., explainability techniques that work on pre-trained models. Within the post-hoc realm, the dichotomy of model-aware and model-agnostic explanations emerges. Model-aware methods dissect model parameters for insights, while model-agnostic approaches treat the model as a black box, perturbing inputs to unveil the significance of elements in the output. In this work, we try both categories but find the model-aware techniques to work better for our case.

In terms of explanation evaluation, post-hoc GNN explanations can be measured using the *fidelity metric* [19]. The fidelity metric measures the impact of the generated explanatory subgraph on the initial prediction, achieved either by exclusively presenting the subgraph to the model (fidelity-) or by excluding it from the entire graph (fidelity+). These fidelity scores capture the ability of an interpretable model to replicate the intrinsic logic of the natural phenomenon or the GNN model.

When it comes to describing post-hoc explanations, we can identify two types of explanations based on their fidelity scores: *necessary* and *sufficient*. A sufficient explanation can be used on its own to reproduce the model's prediction, and it gets a near-zero negative fidelity score. However, a sufficient explanation is not necessarily unique. On the flip side, a necessary explanation is crucial – removing it from the initial graph changes the model's prediction, like a counterfactual explanation. This type of explanation earns a positive fidelity score close to 1. The ideal situation is found when an explanation is both necessary and sufficient. Amara et al. [18] propose to balance the sufficiency and necessity requirements with the *characterization score* which is the weighted harmonic mean of the positive and negative fidelities. Therefore a characterization score close to 1 suggests clear, comprehensive, and informative insights into the model's decisions.

In this work, we use the characterization score as a means to evaluate different explanation techniques and select the most fitting. It has to be noted that the evaluation of explainability techniques is a particularly complex field, and many approaches have been proposed and then put into question by subsequent research [23–25]. Be that as it may, in this work, it is assumed that the characterization score is a suitable metric to measure the quality of graph explanations.

9.3 OUR APPROACH

In this section, we detail the cadence detection model that was trained and used to showcase the explanations, then we describe our framework, and finally, we focus on the choice of explanation techniques.

9.3.1 Cadence Detection Model

To showcase our framework, we chose to use a slightly modified version of the cadence detection model introduced by Karystinaios and Widmer [1]. We extended the model to use as input a heterogeneous score graph as described in Section 9.2.1. Furthermore, we extended the prediction capabilities of the model from binary

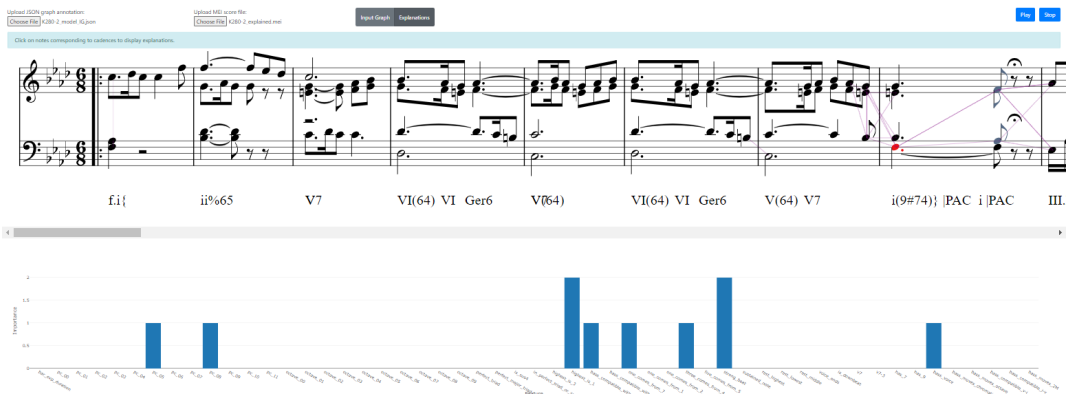


Figure 9.2: A demonstration of the SMUG-Explain Web interface. In this example, we view the first bars of Mozart's Piano Sonata K280 2^{nd} mvt. It includes a Roman numeral analysis and the cadence label predicted by our model at the top. The purple dashed lines are the produced explanation for the note highlighted in red. Note the vertical connection line in the very first bar, which is also a part of this explanation. At the bottom, we can view the feature importance for the explained note.

(i.e. no-cad or PAC) to multiclass cadence prediction, covering PAC, IAC, and HC. Moreover, we modified the architecture by adding an onset regularization module that sums the latent representations (after the GNN encoder) of all the notes that occur on a distinct onset of the score to every note that shares this onset. In summary, our cadence detection model consists of a graph convolutional encoder, an onset regularization module, an embedded SMOTE layer for training, and a shallow MLP classifier.

During training, the graph input is passed first through the graph encoder. The obtained node embeddings are then grouped by onset based on the score information, and their representations are averaged together. Following this step, embedded SMOTE is applied to balance the number of cadence classes compared to the notes not having cadence labels in the score. However, when doing inference, the latter synthetic oversampling step is skipped. Finally, the oversampled embeddings are given as input to a shallow 2-layer MLP classifier that predicts the cadence type.

We trained our model with a joined corpus of cadence annotations from the DCML corpora [2], the Bach fugues from the well-tempered clavier No. 1 [26], the annotated Mozart string quartets [27], and the annotated Haydn string quartets [28]. Our joined corpus makes for 590,149 individual notes and 17,188 cadence annotations. We train our model on 90% of the data and evaluate on 10% using a random split. Our model reaches a mean F-score of 59% on the test set. Note that these results cannot be directly compared with [1], since we use a different (bigger) dataset and perform multiclass prediction.

9.3.2 The SMUG-Explain Framework

Our framework has two main functionalities: generating explanations and making them visually interpretable.

2 https://github.com/DCMLab/dcml_corpora

The first step involves importing musical scores, creating the graph structure, and running the explanation techniques. The score import uses the Python library Partitura [29] which supports a variety of score formats, such as MEI, MusicXML, Kern, or (quantized) MIDI. The graph creation is based on previous work [5, 6] and outputs a graph in the widely used Pytorch Geometric format [30] to favour reusability and extensibility of our frameworks.

For the explanation part, we follow a standardized GNN model explanation pipeline: the explainer receives a pre-trained GNN model that performs node-level classification and produces an explanation of the cadence label prediction for a specific note, at two levels: (a) by quantifying the relative importance of the various features of the note; and (b) by identifying an explanation subgraph consisting of the notes and their relations that seem most important for the prediction. For producing an explanation subgraph, the explainer computes importance masks over all edges. These importance masks reflect an importance score for each one of the edges in the input graph that is used to filter which (most important) edges belong to the explanation subgraph. The importance masks can be soft (i.e. continuous numbers between 0 and 1) or hard (i.e. binary). Furthermore, a maximum number of edges can be imposed per edge type to limit the explanation to stay within a certain graph size. It has been shown that hard masks tend to increase the necessity and sufficiency of explanations [18]. Therefore, in our application, we use a hard top-k method for each one of our edge types, where k is set to 10. Likewise, for producing the feature importance, the explainer produces masks over all nodes and their features and it keeps the k most important by summing the masks along the feature dimension.

The interface of the SMUG-Explain framework is a web-based interface implemented in HTML and Javascript. The core of the interface is the score engraving library Verovio [20] which outputs an SVG representation from a musical score file. We then extend this score representation with the target deep learning model output (i.e., predicted cadences in our case) and with the information produced by the explainers. From the latter, we display in particular the edges between different notes that contribute to the explanation and the feature importance for each note that corresponds to a cadence prediction (see the next section for details). To make this part possible we need a one-to-one mapping between elements in Python and elements in the SVG generated image. Verovio preserves the mapping between note-ids in the musical score file and the SVG image (if they exist), but only the MEI format contains note-ids. Therefore we extended Partitura with an MEI export function, and we support Roman numeral analysis and cadence name exports.

The final step for an effective interface consists of making it interactive. The user can click on single notes, and this will trigger the visualization of the corresponding explanation edges and feature importance. Moreover, the user can switch back and forth between the visualization of the input edges (from the graph presented in Section 9.2.1) and the explanations, and listen to an audio rendition of the musical score. The interface is shown in Figure 9.2.

9.3.3 Choice of Explainability Techniques

To investigate and evaluate the explanations produced by our framework we test several explanation algorithms such as Saliency, Integrated Gradients, Deconvolution, and Guided Backpropagation. Saliency gauges node and edge importance by weighing each node after calculating the gradient of the output concerning input features [16]. Integrated Gradient tackles the saturation issue of gradient-based methods like Saliency by accumulating gradients along the path from a baseline input (zero-vector) to the current input [31]. Deconvolution computes the gradient of the target output but overrides ReLU function gradients, only propagating non-negative gradients [14]. Guided Backpropagation follows a similar approach, backpropagating only non-negative gradients [13].

Each of those methods was evaluated on model-level explanations, i.e. the explanation algorithm computes its losses with respect to the model output. We use the characterization score as defined in [18] (see Section 9.2.2). We fix the positive and negative fidelity weights to 0.5 and apply k-top hard masks for edges and nodes. The results are shown in Table 9.1. Our findings suggest that for this task the Integrated Gradients method better captures explanations, in some cases achieving a perfect characterization score of 1.

We note that most methods tested on our application are gradient-based. Some perturbation-based methods for generation GNN explanations such as the GNNExplainer [32], Occlusion [17], or the GraphMaskExplainer [33], might be suitable for our application, however, they are not yet adapted to work with heterogeneous graphs.

As a general remark, we would like to underline that typically graph-dedicated explainers are more biased toward generating compact explanation subgraphs. However, this bias might be less suitable for music where some analysis methods suggest that important elements of a piece might not be directly interconnected.

Pieces/Model	SAL	GBP	DC	IG
WTC-I Fuga 1	0.0588	0.4706	0.2941	**1.0**
WTC-I Fuga 2	0.0	0.0435	0.0435	**0.9130**
WTC-I Fuga 5	0.0	0.0667	0.0	**0.8667**
Mozart K280-2	0.0	0.3125	0.3438	**0.9375**
Chopin Op.48	0.0	0.1875	0.1250	**0.8125**
Mozart K331	0.0	0.0	0.0625	**1.0**

Table 9.1: Characterization score for the model explanations of cadences per piece. The four methods are mentioned in Section 9.3.3. SAL stands for Saliency, GBP for Guided Backpropagation, DC for Deconvolution and IG for Integrated Gradients. Highlighted values indicate the highest (best) explanations in terms of characterization score.

9.4 QUALITATIVE ANALYSIS

In this section, we perform a qualitative analysis of a diverse set of scores and comment on the graph explanations produced by our system on the task of cadence classification. Apart from the explanation metrics described in Section 9.3.3 we believe that explanation should be also motivated in musical terms. Therefore, we go into depth about some individual explanations of cadence predictions, targeting both true positive and false positive predictions.[3]

We showcase explanations in four classical music excerpts ranging from the Baroque to the Romantic era. Furthermore, we consult expert analyses in terms of harmony, voice leading, and Schenkerian analysis and attempt to interpret the produced explanations. Naturally, we assert that the explanations discussed should be necessary and sufficient. We remind the reader that the generated explanations do not explain the working mechanics of cadences but rather present insights into the model's decision process. In other words, we are not answering the question "why is there a cadence here?", but "why does the model think there is a cadence here?".

All of the below explanations are included in our publicly available code, including the entire pieces. We invite the reader to explore our interface and test our framework.

9.4.1 Mozart Piano Sonata K280 Mov. 2

Our first example (see Figure 9.2) focuses on an excerpt from Mozart's Piano Sonata K280 in F major, from the second movement. The harmonic analysis of this segment was provided by Hentschel et al. [34]. We focus on the perfect authentic cadence arriving on the second beat of measure eight and signaling the end of the first phrase. The harmonic analysis indicates a textbook preparation of the cadence with the German augmented sixth chord on the second beat of measure 6, then the dominant with a tonic in second inversion resolving to a dominant seventh. Finally, the tonic bass arrives on the first beat of measure 8 but with sustained soprano, alto, and tenor voices which themselves resolve on the second beat of the same measure.

We take a closer look at the explanation subgraph of the bass of the tonic (F_4) which arrives earlier than the cadential arrival point of the soprano. However, the explanation highlights the descending Urlinie melody containing the $\hat{3}, \hat{2}, \hat{1}$ in the top voice, capturing the first appearance of the $\hat{3}$ on the relative strong beat of the sixth measure (sub-dominant space), whilst considering the overarching bass arpeggiation of the $i - V - i$. Interestingly, the explanation subgraph also contains the first chord of the piece (see faint vertical line in bar 1), indicating that the model considers some information regarding the key of the piece. It's important to highlight that the cadential ground truth obtained from [34] designates notes with the cadence label at the cadential arrival point of the soprano. However, we argue that including the first appearance of the bass note on the strong beat of the same measure is a crucial addition to the cadence annotation ground truth

3 One could also focus on no-cadence predictions (and false negatives), but we are not considering these in this section, since the "absence of a specific evidence for a cadence" may result in less musicologically interesting graph patterns.

from a musicological standpoint. This addition is accurately captured by our model. Upon analyzing the predictions, we observe that the model correctly identifies the individual notes involved in the PAC in measure 8, namely the *F*4 on the first beat and the soprano, tenor, and alto resolutions on the second beat (remember that our model predicts cadence labels for individual notes, not score onsets).

Furthermore, we deduce from the explanation subgraph that the next notes after the cadence are also crucial for the model's prediction, as they point towards an ending of a phrase and a new rhythmical and harmonic idea further on.

9.4.2 Bach WTC Fugue

For this example, we explore the predictions and explanations of cadences in the fifth Fuga in D major from J.S. Bach's Well-tempered Clavier. We believe that the contrapuntal nature of this piece could provide insightful hints about voice leading. Our analysis is complemented by a Schenkerian analysis conducted by Marlowe [35], and we enhance clarity by performing a Roman numeral analysis on the explanatory excerpt (refer to Fig. 9.3).

Our specific focus centers on the explanation of a false positive IAC (Imperfect Authentic Cadence) prediction on the downbeat of the fifth measure. The ground-truth cadence annotations were provided by [26]. The cadence annotations, following the ground truth, were originally provided by Giraud (2015). IAC labels were annotated on the downbeats of measures 3 and 6, both corresponding to the dominant of the D major (i.e. the tonic). Our model accurately predicts these labels but also anticipates an additional IAC label on the downbeat of measure 5, which is not present in the ground truth annotations. Consequently, we investigate what led to this prediction.

We observe that the explanation subgraph for the highlighted red note at the top of Figure 9.3 contains, as expected, the edges between the dominant and the subdominant. However, the interesting part of the subgraph takes place in bars 2-3 of the score. The high voice contains the descending melodic line towards the leading tone which then re-appears right before measure 5, and the highlighted red note subsequently resolves it. The lower voice on measure 2 focuses around the E note which would be a fifth from the dominant A. Therefore, the subgraph outlines a movement ii-V-I for the bass.

The middle-ground Schenkerian analysis displayed in the bottom part of Figure 9.3 provides some similar observations. We observe an oscillating bass from the tonic to the dominant and over again. This oscillating bass is captured in the cadence annotations by the two cadences on the dominant. It could be argued that the falsely predicted cadence on the tonic strengthens the analysis assumption of the oscillating bass.

In terms of feature importance, we see that characteristics of imperfect cadence are activated such as the existence of a perfect major triad, the highest note being a transposed third interval for the bass, the existence of a leading tone resolving, and the presence of a dominant seventh chord before. Naturally, such characteristics are not exclusively present in the event of cadences but in this case, they seem to influence the model's prediction.

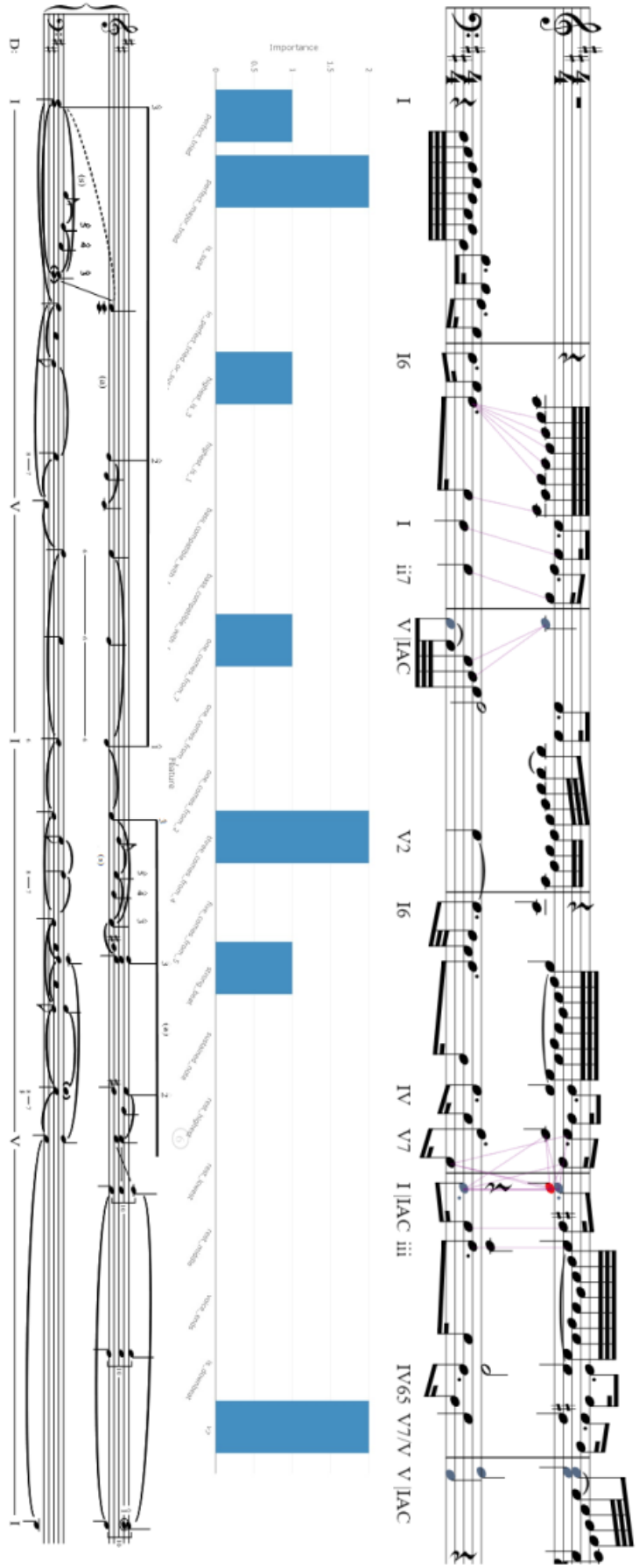


Figure 9.3: The first bars of the Fuga No. 5 of the Well-Tempered Clavier book No. 1. On the top the score and the explanation of the wrong prediction of the highlighted note in red. In the middle, the feature importance is visualized for the highlighted note. On the bottom, a Schenkerian analysis of the segment by [35]

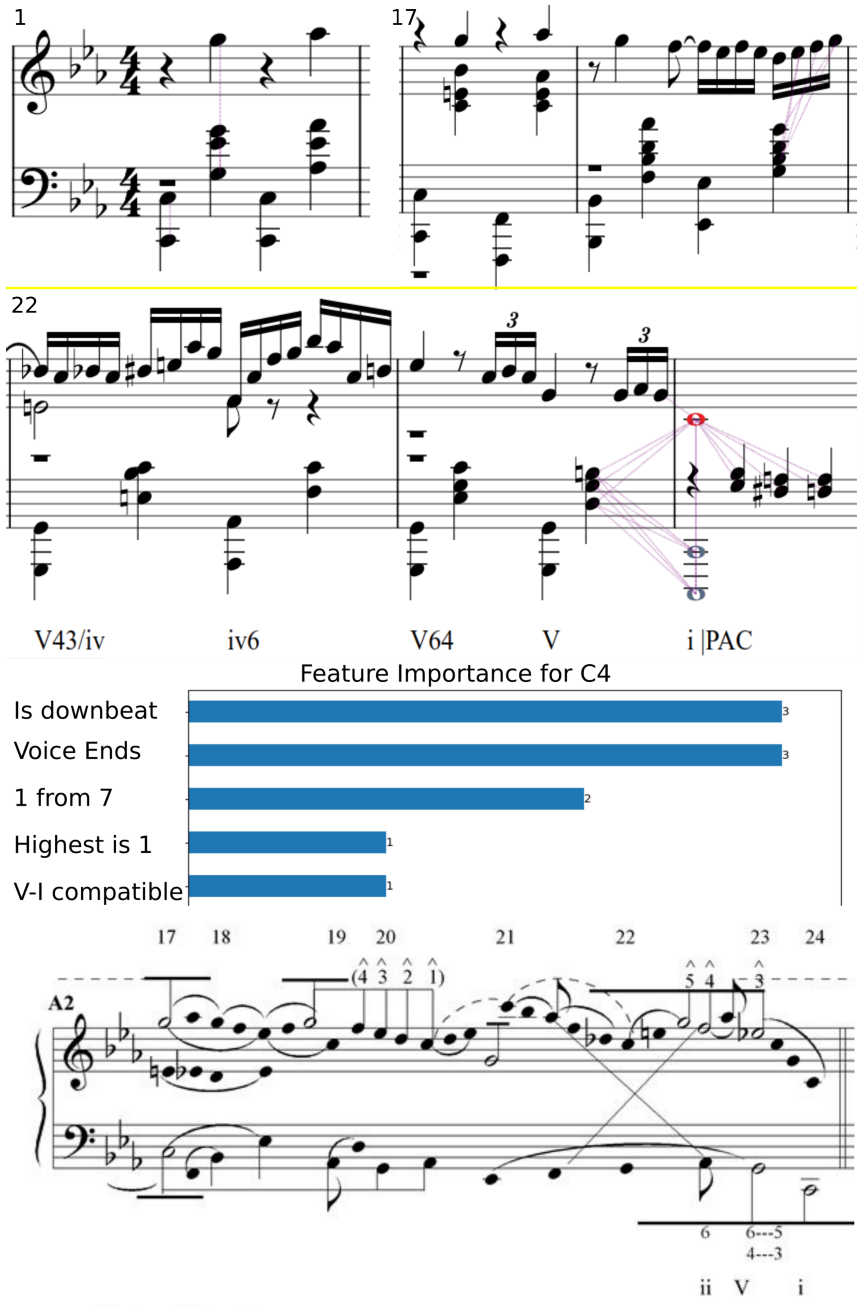


Figure 9.4: Excerpt of Nocturne Op. 48, no. 1 in C minor by F. Chopin. Top: excerpts of the explanation for Cadence on measure 24. Bar numbers are notated to the top left of each score segment. Middle: Feature importance for the highlighted C4 note in red. Bottom: Middleground voice leading analysis (from [36]).

9.4.3 Chopin Nocturne in C minor op. 48

For our last example, we chose a passage with more ambiguous cadential and harmonic elements: bars 22-24 from Chopin's Nocturne op. 48 no 1, in C minor. For this piece, we consult a Schenkerian analysis by Swinkin [36]. In particular, we look closely at the perfect authentic cadence that arrives in measure 24. Unlike previous examples, this cadence does not have the textbook voice leading and harmonic elements that distinctly define PACs but, nevertheless, carries a cadential character.

Our model correctly identifies the PAC that arrives on the downbeat of measure 24. To provide support for our analysis and commentary, we perform a Roman numeral analysis on the segment depicted in Figure 9.4. From the model's explanation subgraph, we see that once again the explanation contains the first chord of the piece, further strengthening our assumption that the model is gathering information about the key to inform its predictions. However, the rest of the generated explanation subgraph is mostly compact and local, focusing only on the last notes before the cadence and the ones after it. The chordal content of the segment contains the supported harmony for a perfect authentic cadence (PAC) preparation. But, since the cadence does not follow the typical voice-leading patterns usually involved, it seems that the model does not need to go towards far neighborhoods in the graph to infer the necessary context for its prediction.

Interestingly, the model includes in the explanation subgraph a part of measure 18. The connected notes in measure 18 correspond to a part of the descending Urline that is present in the voice-leading analysis excerpt in the Figure, however, they are not in order. This could be a connection with what one could view as a "sustained" high G in the melody line which slowly descends towards G an octave lower before the cadence. That being said, it is rather ambiguous how the part in measure 18 affects the model's prediction.

9.5 CONCLUSION AND FUTURE WORK

This paper presented the SMUG-Explain framework for generating and visualizing graph explanations on musical scores. We showcased the framework on a cadence detection model, compared different explanation techniques, and gave some qualitative insights into the explanations.

Future work will focus on developing new explanation techniques dedicated to musical score graph data, and on testing our results with user-based evaluations. The final objective would be to produce a musicologically trustworthy and user-friendly framework that can support expert analysts to produce more effective musical analyses. Furthermore, we aim to invest efforts into making the system more accessible by releasing an online server-based version of our interface, which can be used to produce predictions for some reference models and tasks and explain them, without the need to run any code locally.

9.6 ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab).

REFERENCES

[1] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[2] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[3] Arnau Baró, Pau Riba, and Alicia Fornés. "Musigraph: Optical Music Recognition Through Object Detection and Graph Neural Network." In: *International Conference on Frontiers in Handwriting Recognition*. Springer. 2022, pp. 171–184.

[4] Emanuele Cosenza, Andrea Valenti, and Davide Bacciu. "Graph-based Polyphonic Multitrack Music Generation." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[5] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[6] Emmanouil Karystinaios and Gerhard Widmer. "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[7] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[8] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. "Local Interpretable Model-agnostic Explanations for Music Content Analysis." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 537–543.

[9] Saumitra Mishra, Emmanouil Benetos, Bob L. Sturm, and Simon Dixon. "Reliable Local Explanations for Machine Listening." In: *Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN*. IEEE, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207444.

[10] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. "audioLIME: Listenable Explanations Using Source Separation." In: *Proceedings of the International Workshop on Machine Learning and Music, MML*. 2020, pp. 20–24.

[11] Alessandro B. Melchiorre, Verena Haunschmid, Markus Schedl, and Gerhard Widmer. "LEMONS: Listenable Explanations for Music recOmmeNder Systems." In: *Advances in Information Retrieval: Proceedings of the European Conference on IR Research, ECIR*. Vol. 12657. Springer, 2021, pp. 531–536.

[12] Francesco Foscarin, Katharina Hoedt, Verena Praher, Arthur Flexer, and Gerhard Widmer. "Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022. URL: https://api.semanticscholar.org/CorpusID:251881711.

[13] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net." In: *Workshop at International Conference on Learning Representations (ICLR)*. 2015.

[14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 818–833.

[15] Erik Strumbelj and Igor Kononenko. "An efficient explanation of individual classifications using game theory." In: *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[16] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." In: *Workshop at International Conference on Learning Representations (ICLR)*. 2013.

[17] Lukas Faber, Amin K. Moghaddam, and Roger Wattenhofer. "When comparing to ground truth is wrong: On evaluating GNN explanation methods." In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 332–341.

[18] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks." In: *In Learning on Graphs Conference (Proceedings of Machine Learning Research)* 198 (2022).

[19] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2022), pp. 5782–5799.

[20] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. "Verovio: A library for Engraving MEI Music Notation into SVG." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[21] Fred Lerdahl and Ray S Jackendoff. *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.

[22] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[23] Katharina Hoedt, Verena Praher, Arthur Flexer, and Gerhard Widmer. "Constructing adversarial examples to investigate the plausibility of explanations in deep audio and image classifiers." In: *Neural Computing and Applications* 35.14 (2023), pp. 10011–10029.

[24] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. "Sanity Checks for Saliency Maps." In: *Neural Information Processing Systems*. 2018.

[25] G. Yona and Daniel Greenfeld. "Revisiting Sanity Checks for Saliency Maps." In: *Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4)*. 2021.

[26] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational fugue analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96.

[27] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[28] David RW Sears, Marcus T Pearce, William E Caplin, and Stephen McAdams. "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models." In: *Journal of New Music Research* 47.1 (2018), pp. 29–52.

[29] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[30] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric." In: *Workshop on Representation Learning on Graphs and Manifolds at International Conference on Learning Representations (ICLR)*. 2019.

[31] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328.

[32] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 32 (2019).

[33] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. "Interpreting graph neural networks for NLP with differentiable edge masking." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021.

[34] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. "The Annotated Mozart Sonatas: Score, Harmony, and Cadence." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4.ARTICLE (2021), pp. 67–80.

[35] Sarah Marlowe. "Schenkerian Analysis of Fugue: A Practical Demonstration." In: *Journal of Music Theory Pedagogy* 33.1 (2019), p. 6.

[36] Jeffrey Swinkin. "Schenkerian analysis, metaphor, and performance." In: *College Music Symposium*. Vol. 47. JSTOR. 2007, pp. 76–99.

10 GRAPH CONVOLUTION FOR MUSIC

Title: Perception-Inspired Graph Convolution for Music Understanding Tasks.

Published In Proceedings of the the 33rd International Joint Conference on Artificial Intelligence (IJCAI), Jeju Island South Korea, 2024.

Authors: Emmanouil Karystinaios, Francesco Foscarin, Gerhard Widmer

Abstract: We propose a new graph convolutional block, called *MusGConv*, specifically designed for the efficient processing of musical score data and motivated by general perceptual principles. It focuses on two fundamental dimensions of music, pitch and rhythm, and considers both relative and absolute representations of these components. We evaluate our approach on four different musical understanding problems: monophonic voice separation, harmonic analysis, cadence detection, and composer identification which, in abstract terms, translate to different graph learning problems, namely, node classification, link prediction, and graph classification. Our experiments demonstrate that *MusGConv* improves the performance on three of the aforementioned tasks while being conceptually very simple and efficient. We interpret this as evidence that it is beneficial to include perception-informed processing of fundamental musical concepts when developing graph network applications on musical score data. All code and models are released on https://github.com/manoskary/musgconv.

10.1 INTRODUCTION

Music data can be represented in computer applications in multiple formats, two popular ones being audio and symbolic representations. The first encodes a measure of the pressure/intensity of the sound wave over time, while the second explicitly encodes discrete musical events such as notes and rests (see [1] for an overview of different symbolic music formats). Due to this higher-level information, symbolic representations are generally considered better inputs/outputs for music analysis and generation tasks in the Music Information Research (MIR) field. Moreover, any musical task that starts from a musical score (sheet music) or from MIDI files is naturally in the symbolic domain.

In the MIR literature, symbolically encoded music is typically handled with techniques heavily inspired by computer vision (CV) or natural language processing (NLP) research. In the first case, music is represented in a so-called "piano roll" format (first developed in MIDI sequencers) and treated as a raster image, with the X axis being time, and the Y axis pitch. The pitch values are typically encoded as MIDI pitch, i.e., with integers in $[0 - 127]$, which covers all notes that can be played

by common well-tempered instruments (a range that is larger than the range of the piano, with 88 notes), and the time resolution is a parameter that is usually set to the expected shortest note duration. In its simpler version, each element of the 2D matrix is set to 1 if a note is sounding at the corresponding pitch and time, or 0 otherwise. The downside of this approach is that it creates a very large and very sparse input matrix since only a few notes will play at any time. The other common approach is treating music with sequential models from NLP research. Although different tokenization techniques have been proposed [2], it is easy to argue that music does not fit well into strictly sequential models, since more than one note can sound simultaneously, notes can partially overlap, and generally, the pitch-temporal relations between notes hold important musical information.

A few recent works [3–7] started to explore the use of graphs, and Graph Neural Networks (GNNs), to represent and process symbolic music. Significant advances have been reported on various musical tasks [6, 8]. However, the components that these papers use are "borrowed" from GNN research with other types of data.

We argue that this can lead to suboptimal results. As a solution, we design MusGConv, a new graph convolutional block specifically dedicated to music data, that is founded on fundamental music perception principles. We test our approach on four musical tasks: voice separation, composer classification, Roman numeral analysis, and cadence detection. This selection of tasks allows us to cover three major graph neural network classes of problems: graph classification, link prediction, and node classification. We compare our results with the state-of-the-art graph models for these tasks and show that the use of MusGConv leads to better results overall. Moreover, its simple design enables this performance boost without adding any additional computational cost.

10.2 PERCEPTUAL AND MODELING CONSIDERATIONS

We base our research on the perceptual principles of two fundamental musical dimensions: pitch and rhythm. Cognitive studies show that people are not very sensitive to the absolute pitch of individual notes and perceive mainly the distance between pitches [9]. Thus, we perceive the same musical pattern if it is shifted higher or lower in frequency. This is called *relative pitch perception* and has been formalized in music theory through measures of pitch distance called *intervals*, which are the basis of all musical concepts that involve combinations of pitches, such as chords and harmony. The position (and duration) of notes in time is also not important in itself but only relative to the position (and duration) of the other notes. This temporal organization is called *rhythm* and is typically composed of patterns that tend to be periodic and organized at different hierarchical levels.

While these principles are simple, producing relative features to use as input for deep learning systems is not an easy task. For the pitch representation, considering intervals between consecutive notes instead of absolute pitches, as proposed for example in the IDyOM framework [10, 11], poses the problem of defining a note order. This is trivial for monophonic melodies but becomes problematic for polyphonic music, where multiple notes can fully or partially overlap in time. Figure 10.1 (c) exemplifies a possible ordering rule, but such rules may not be generally valid for

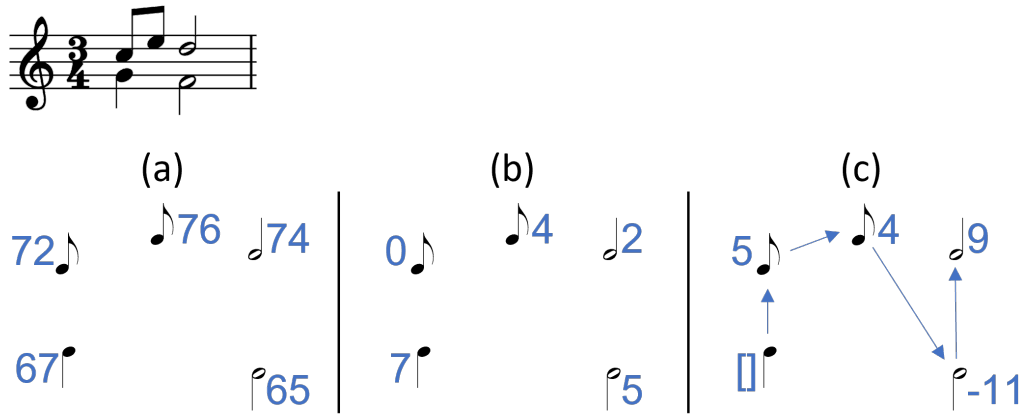


Figure 10.1: Three alternative representations of note pitches in a musical excerpt: (a) absolute representation in terms of MIDI pitch; (b) relative pitch distance (ignoring the octave) in semitones relative to the fundamental pitch specified by the key signature (here: C); (c) relative pitch distance in semitones from the closest preceding note; in case of chords the order is defined from bottom to top.

different pieces and contexts or in general yield very different interval sequences for very similar patterns (e.g., if we remove the F), thus not helping with learning a general representation. Inserting interval information after the music has been tokenised in a sequential representation [12], creates similar ordering problems.

Another strategy is to rely on music theory and consider pitch distances relative to the fundamental note defined by the key of the piece (see Fig. 10.1 (b). This has a musicological limitation, as the key signature of a piece may not change when there are temporary modulations to other keys, and a practical one, as keys are not always notated in musical datasets and MIDI files. Not to mention all the music that falls outside the classic tonal framework, for which the definition of a key is not even meaningful. The most common alternative, e.g., [13, 14], has been the use of data augmentation via transposition, assuming that if the network sees transpositions of the same piece, it will learn patterns that are general across all transpositions. However, this is far from ideal since the network will need to store similar patterns redundantly, thus making inefficient use of network capacity and drastically increasing training time.

For this reason, we believe that designing relative features to input to our system is not a viable option for general music modelling. We explore a different path, which is to customize the working mechanism of our network to take into account the relative perceptual properties of music through a *dedicated message-passing mechanism* that computes pairwise pitch and time representations. This is weakly related to recent work on audio representations of music that aims at learning transposition-invariant features [15–17] and tempo-invariant rhythmic patterns [18]. Other related work targets graphs whose edges encode geometric information [19–21], intending to build representations that are invariant to operations such as translation. The ideas from these last approaches (with minor modifications) are also beneficial for musical tasks, but we show that our music-specific approach outperforms them.

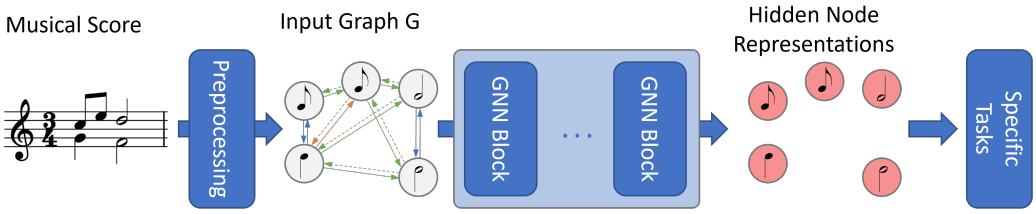


Figure 10.2: General architecture of our pipeline. The first part that produces the hidden node representation is common among all tasks; the last module is task-specific.

Up to this point, we have highlighted the importance of relative pitch and time representations. However, their absolute values could also be important depending on the task at hand. For example, instruments would peculiarly change their timbre as they approach very low or very high notes in their range, which is recognisable by a listener. In tonal music, the absolute pitch of notes defines a key signature which could be relevant in the composer classification task (specific keys can have very specific meanings to composers, and within a musical tradition). The same goes for absolute time positions, for example, with patterns happening at the beginning or end of a piece. Finally, the output of our network may need to be an absolute pitch (for example, in the Roman numeral analysis task we describe later), and therefore we need to retain this information in the network.

This need for considering both absolute and relative pitch and time positions motivates the design of our new convolutional block, to be described in detail in Section 10.4 below.

10.3 GRAPH APPROACHES TO MUSICAL TASKS

In this section, we describe existing graph modelling approaches to the four musical tasks we use to evaluate our proposal. They all have a common pipeline which involves building a graph from a given musical score (see Figure 10.2) and using a series of convolutional blocks to produce context-aware hidden representations for each node. We start by describing the graph-building procedure and a generic graph convolutional block; we then proceed by detailing the tasks and the specific network components used to target them.

10.3.1 Graph from Musical Scores

A graph is defined as a set of nodes V and a set of edges E, where each edge $(u, v) \in E$ connects the nodes $v, u \in V$. We extend this definition by considering labelled edges which we model as a triple (u, r, v), where $u, v \in V$ and $r \in \mathcal{R}$ is a relation type. A graph with edges of multiple types (we don't use different types of nodes in this work) is called *heterogeneous*. Moreover, we consider an *attributed* graph, i.e., a graph where each node is described by a set of features, which we group in the columns of the matrix X. Our attributed heterogeneous graph is defined as $G = (V, E, X)$.

We create such a directed graph from a musical score following the work of Karystinaios et al. [6]. Each node $v \in V$ corresponds to one and only one note in the musical score. $\mathcal{R}$ includes 4 types of relations: onset, during, follow, and silence, corresponding, respectively, to two notes starting at the same time, a note starting while the other is sounding, a note starting when the other ends, and a note starting after a time when no note is sounding. The inverse edges for during, follows, and silence relations are also created.

The feature matrix X is composed of the following features extracted from each note of the score: the pitch class, i.e., one of the 12 note names (C, C#, D, D#, etc.), the octave in $[1, \ldots, 7]$, the note duration, encoded as a single float value $d \in [0, 1]$ computed as the ratio of the note and bar durations, passed through a tanh function to limit its value and give more resolution to shorter notes, as proposed by Karystinaios et al. [6]. For the task of Roman Numeral Analysis and Cadence Detection, we add additional specialized features to be consistent with the approach in the literature we consider in our evaluation [5, 8].

10.3.2 Graph Convolution Operation

We now present a generic graph convolutional block, to simplify the description of our music-dedicated approach in the next section. Given an attributed homogeneous graph G, a graph convolution block that updates the representation of node u for layer $l+1$ can be described as:

$$\boldsymbol{h}_u^{(l+1)} = \psi\left(\boldsymbol{h}_u^{(l)}, \underset{v \in \mathcal{N}(u)}{\text{aggregate}}\left(\{\boldsymbol{\eta}_{vu}^{(l)}\}\right)\right), \tag{10.1}$$

$$\boldsymbol{\eta}_{vu}^{(l)} = \phi\left(\boldsymbol{h}_v^{(l)}, \boldsymbol{h}_u^{(l)}\right) \tag{10.2}$$

$$\boldsymbol{h}_u^{(l+1)} = \sigma\left(\boldsymbol{h}_u^{(l+1)}\right) \tag{10.3}$$

where aggregate$(\cdot)$ denotes a differentiable, permutation invariant aggregation function, e.g., sum, mean, etc.; ϕ and ψ are called edge operation and node operation, respectively, and denote differentiable learnable functions such as concatenation, sum, or multiplication, followed by a linear transformation; σ denotes a non-linear function, $\mathcal{N}(u)$ denotes the neighbours of u; $\boldsymbol{h}_u^{(l)}$ is the hidden representation of node u at layer l.

Furthermore, if we want to leverage edge features, Equation 10.2 becomes:

$$\boldsymbol{\eta}_{vu}^{(l)} = \phi\left(\boldsymbol{h}_v^{(l)}, \boldsymbol{h}_u^{(l)}, \boldsymbol{e}_{vu}\right) \tag{10.4}$$

where, $\boldsymbol{e}_{uv}$ are features of the directed edge connecting node v with node u.

When G is heterogeneous, the function $\mathcal{N}(u)$ in Equation 10.1 is modified as proposed in [22] to return only the neighbours nodes which are connected with an edge of type r. Equations 10.1 and 10.3 are then computed $|\mathcal{R}|$ times and the results $\mathbf{h}_{\mathbf{r}u}^{(l)}$ aggregated in an unique node latent representation $\boldsymbol{h}_u^{(l)}$ as:

$$\boldsymbol{h}_u^{(l)} = \underset{r \in \mathcal{R}}{\text{aggregate}}\left(\{\mathbf{h}_{\mathbf{r}u}^{(l)}\}\right) \tag{10.5}$$

where aggregate$(\cdot)$ denotes a differentiable, permutation invariant aggregation function such as sum or mean.

We now turn to describe the tasks that we will use for evaluation and the task-specific part of our graph model.

10.3.3 Monophonic Voice Separation

Voice separation is the task of segmenting a symbolic music piece into an unknown number of individual monophonic note streams according to musical and perceptual criteria. Duoane and Pardo [23] framed the problem as a link prediction task in which two notes are linked if they are consecutive in the same voice. Karystinaios et al. [6] proposed a GNN-based model that reached new state-of-the-art results.

Following their approach, we perform this task by adding a link predictor module, consisting of an MLP which takes as input the hidden representations of nodes and for each pair of nodes performs a binary classification between the "linked" and "not-linked" classes.

The evaluation metric is the *binary F1 score*, i.e. the F1 score for the positive class which represents the true links in the ground truth. Karystinaios et al. also consider the F1 score after a postprocessing phase, but we don't use it to keep the number of metrics of reasonable size, and because, as they report, postprocessing increases the metric in a way which does not always correlate perfectly with the network performance.

10.3.4 Composer Classification

Composer classification from a symbolic musical score is the task of identifying the composer of the score from a list of composers. In the graph problem taxonomy, this falls in the category of global graph classification tasks. Following the work of Zhang et al. [7], we perform this task by adding a global mean pooling layer to our architecture, which averages the latent representations built by our GNN blocks, followed by an MLP which predicts probabilities over composer classes. The composer classification predictions are evaluated in terms of *classification accuracy*.

10.3.5 Roman Numeral Analysis

Roman numeral analysis is a branch of analytical musicology whose goal is to infer the underlying harmony and chord progressions from a musical score. The result is a set of complex composite labels (the Roman numerals) which annotate music at the onset level, i.e., for every score position that corresponds to one or more note onsets. Related work [24–27] frames RN analysis as a multi-task classification problem where every label is broken down into 5 components (degree, inversion, root, key, and quality) which are predicted by different classifiers in hard-parameter-sharing setting. Of these components, degree, inversion, and quality are transposition invariant and the rest depend on the absolute pitches in the input. A recent approach [8] considers a graph input and obtains new state-of-the-art results.

Following this work, we perform this task by adding to our general architecture an onset edge pooling layer that contracts the latent representations from the note-wise level to the onset-wise, i.e., it creates a single vector per unique onset. All vectors are then ordered by time position and fed into a GRU layer whose output is finally used by the aforementioned MLP classifiers. In the taxonomy of graph problems, RN analysis falls in between node classification and subgraph classification because of the effect of the edge pooling layer.

The evaluation score is the so-called *Chord Symbol Recall (CSR)*, i.e., the ratio of the total duration of segments where prediction equals annotation vs. the total duration of annotated segments [28].

10.3.6 Cadence Detection

Cadence detection is a music analysis task that consists of detecting cadences, i.e. phrase endings with a strong and specific melodic-harmonic closure effect, in a musical score. Cadences are important both musicologically and perceptually and it is known that they relate to particular voicings and chord progressions; however, their automatic predictions remain particularly challenging due to the high number of exceptions and corner cases. A recent graph approach [5] framed the problem as a multiclass node classification scenario, by predicting the presence of a cadence and its type for each note.

Following that work, we use a graph autoencoder architecture and the latent synthetic oversampling technique SMOTE [29] to balance the heavily unbalanced class labels. For the same reason, the reported evaluation metric is the *macro F1 score*.

10.4 OUR APPROACH: MUSGCONV

Similarly to previous works, we use stacked graph convolutions to create a hidden representation of notes that is then used as input for specific music tasks (see Figure 10.2). In our proposed approach, we replace the graph convolutional blocks in the stack with our novel convolutional block (see Figure 10.3).

In terms of the notation introduced in the general description of the graph convolution in Section 10.3.2, our convolutional block is characterised by two core contributions: a way to build the edge features e, and the choice of the edge operation ϕ in Equation 10.4.

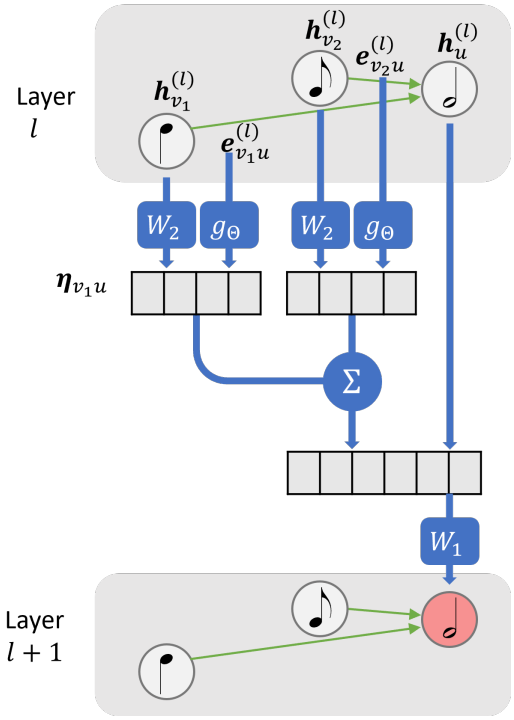


Figure 10.3: Visualization of update for node u in our MusGConv block (considering only one edge type), corresponding to Eqns. 10.11 and 10.8.

10.4.1 Edge Features Computation

For each edge between nodes u, v, we consider three edge features: $e_{vu}^{\text{onset}}, e_{vu}^{\text{dur}}, e_{vu}^{\text{pitch}}$ each of them encoded as a single scalar corresponding to the distance between onset, duration, and pitch, respectively.

$$\begin{aligned} e_{vu}^{\text{onset}} &= |on(u) - on(v)| \\ e_{vu}^{\text{dur}} &= |dur(u) - dur(v)| \\ e_{vu}^{\text{pitch}} &= |pitch(u) - pitch(v)| \end{aligned} \tag{10.6}$$

This roughly corresponds to the computation of distance in papers that deal with geometrical data [21] except that in their case it is a multidimensional space distance, while we compute a set of one-dimensional distances since it makes no sense to mix duration, pitch and onset information. To keep these values in a convenient numerical range, we normalise each feature with ℓ_2 normalisation over all edges in a batch.

Additionally, we inform the network about the *pitch-class interval* (PCInt), i.e., the distance between notes without considering the octave, or the interval direction. This can be seen as a relative version of the *chroma feature*, which is commonly used in MIR tasks related to the harmonic content of the music. This integer in $[0, \ldots, 11]$ is passed through an *embedding layer*, i.e., a learnable look-up table which maps these integers to points $\boldsymbol{e}_{vu}^{\text{PCInt}}$ in a continuous multidimensional space.

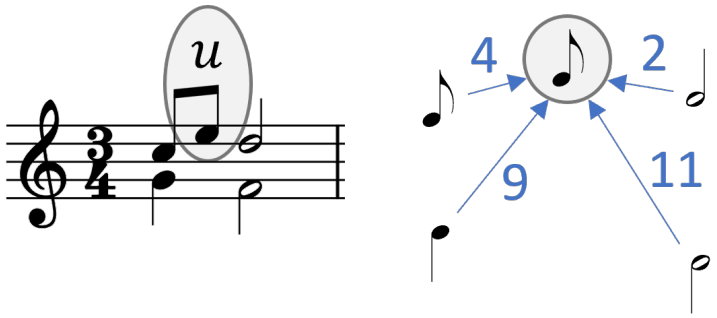


Figure 10.4: Relative Pitch features e_{vu}^{pitch} for the highlighted note u.

For each edge, all the aforementioned edge features are concatenated in a single vector:

$$\boldsymbol{e}_{uv}^{(0)} = \text{cat}\left(e_{vu}^{\text{onset}}, e_{vu}^{\text{dur}}, e_{vu}^{\text{pitch}}, \boldsymbol{e}_{vu}^{\text{PCInt}}\right) \tag{10.7}$$

10.4.2 Edge Operation

Our second contribution, needed to properly leverage the edge feature information, is a new formulation of the message passing paradigm (see Figure 10.3 for a graphical representation). The new edge operation defines the edge operation ϕ in Equation 10.4 as follows:

$$\boldsymbol{\eta}_{vu}^{(l)} = \text{cat}\left(\boldsymbol{W}_2^{(l)}\boldsymbol{h}_v^{(l)}, g_{\boldsymbol{\Theta}}^{(l)}\left(\boldsymbol{e}_{vu}^{(l)}\right)\right) \tag{10.8}$$

where $g_{\boldsymbol{\Theta}}$ denotes a simple two-layer MLP with a Relu activation and layer-wise normalization.

Note how we concatenate the edge features here, while other approaches that deal with edge features tend to apply a permutation-invariant operation such as multiplication or addition [21, 30]. In this way, the transposition- and time-invariant information carried by the edge features is treated in the same way as the node feature when computing the pairwise representation η_{uv}. This is inspired by musicological considerations: we don't want to weight/modify the absolute representation according to the relative representation, but rather to just use it as input, as is done by cognitively plausible musical models [10, 11]. This is similar to what we discuss in Section 10.1 but, by using MusGConv, we no longer have the problem of setting a (debatable) order of the input notes, since for each node, we will consider the relative features according to every other node connected to it (see Figure 10.4).

We consider two variants of our system which differ in the edge features which are passed to layers after the first. The first variant, named MusGConv uses the absolute difference of the node hidden embeddings, i.e. $\forall l > 0$,

$$\boldsymbol{e}_{vu}^{(l)} = |\boldsymbol{h}_v^{(l)} - \boldsymbol{h}_u^{(l)}| \tag{10.9}$$

The second variant, named MusGConv(+EF), uses the edge hidden representation from the previous layer as the edge features of the next layer:

$$\mathbf{e}_{vu}^{(l+1)} = g_{\Theta}^{(l)}\left(\mathbf{e}_{vu}^{(l)}\right) \tag{10.10}$$

10.4.3 Node Operation

To complete the explanation of our convolutional block, we specify the specific computation for the generic node operation ψ introduced in Equation 10.1:

$$\boldsymbol{h}_u^{(l+1)} = \boldsymbol{W}_1^{(l)}\text{cat}\left(\boldsymbol{h}_u^{(l)}, \sum_{v \in \mathcal{N}(u)} \boldsymbol{\eta}_{vu}^{(l)}\right) \tag{10.11}$$

The aggregation function is a sum, and we choose to concatenate the hidden representation of u with the messages η_{vu} from the other nodes. For heterogeneous convolutions (i.e. Equation 10.5), we use $\text{mean}(\cdot)$ as the aggregation function. Such choices are not motivated by musical reasoning, and experimenting with other operations could be an interesting direction, but are out of the scope of this paper.

10.5 DATA

Musical score graph datasets are different from common datasets in the graph-related literature regarding the number and size of graphs. Node classification and link prediction datasets often only consist of a single huge graph, coupled with a sampling strategy to obtain subgraphs to train and evaluate the Graph Neural Network [31, 32]. On the other hand, graph classification datasets often consist of a large number of small graphs, with less than 50 nodes [33]. Musical score graphs are neither small nor extremely large and may vary significantly in size; a Bach Chorale may have $\sim$ 100 notes whereas a Beethoven Sonata might have more than 5000 (with every note corresponding to a node in the graph).

Moreover, popular sampling strategies, such as node-wise sampling, subgraph sampling, random-walk sampling, and spectral sampling, may yield musically problematic note configurations, for example, by segregating notes that are played at the same time while grouping in the same subgraph notes that are very far apart. This is problematic, especially for musically-local tasks such as voice separation or Roman numeral analysis, where the system cannot be expected to produce a meaningful result, and could even learn to perform the task in the wrong way if some notes (and onsets for the roman numeral analysis) are missing.

In the previous works on graph scores that we are considering, the problem is avoided by training on mini-batches which consist of single pieces. However, this is not an efficient solution since it always leaves a big part of memory unused, thus unnecessarily prolonging the training time and decreasing the variability in the batch. We describe the new sampling mechanism we use in the following section, and then move on to detailing the datasets used in our experiments.

10.5.1 Data Sampling

While our nodes can be ordered by multiple features, the organizational aspect that is most prominent from a perceptual point of view is time. Indeed, people would still recognize a music piece if it is segmented over the time axis, while, for example, considering only pitches in a certain interval could lead to meaningless results. There is also perceptual evidence that the offset time of a note is much less salient than the onset time, especially for percussive instruments (including the piano) whose sound naturally decays over time [34]. Therefore, when we create our graphs from a musical score, we set the node order first by the absolute time of onset and then by pitch.

Once this ordering is set (and having defined an index function $ind(\cdot)$ that given a node returns its index in this ordering) we randomly sample a subgraph of size $K > 1$ from a piece with N notes, with the following procedure. If $N > K$ we select the nodes u with $ind(v) \leq ind(u) \leq ind(v) + N$, where v is a random node which satisfies the inequality $ind(v) < K - N$. If $N \leq K$ we select all nodes in the piece. Note that we can still have the problem of segregating notes from the same chord, but this can now only happen at the temporal boundaries of our subgraph, limiting its impact on the network.

We can then create a batch consisting of B subgraphs of at most size N. Our batching mechanism uses the approach of Hamilton et al. [31], i.e., all graphs are joined together in a single batched graph that will contain B disjoint subgraphs.

10.5.2 Datasets

We use four distinct datasets for our four tasks.

10.5.2.1 Voice Separation

The Graph Voice Separation dataset was introduced by Karystinaios et al. [6]. This dataset contains graph data created from five collections: 370 Bach Chorales, 48 Preludes and 48 Fugues from the Bach Well-Tempered Clavier (Books I and II), 15 Bach Inventions, 15 Bach Sinfonias, and 210 movements from Haydn String Quartets. It contains in total 726,246 nodes and 3,408,679 edges from 1,054 unique score graphs. Karystinaios et al. only test on single collections to understand the differences in performance for different composing styles. To have a single general performance indicator, we introduce a new data split that uses 70% of the data for training, 10% for validation, and 20% for testing. This split preserves the percentage of pieces in each collection and is independent of the size of each score graph.

10.5.2.2 Composer Clasification

For composer classification, we use the scores from the DCML corpora dataset[1]. The dataset includes 10 composers, for a total of 419 scores, from where we build score graphs with collectively 710,240 nodes and 3,924,655 edges. We create a random data split with 70% of the data for training, 10% for validation, and 20% for testing, which preserves the percentage of composers in each set.

1 https://github.com/DCMLab/dcml_corpora

10.5.2.3 Roman Numeral Analysis

The Roman numeral analysis dataset with data augmentation was introduced by Lopez et al. [27]. We use their dataset (with augmentations on the train set). The created graphs collectively contain 8,968,413 nodes, 38,390,729 edges, and 5,096,853 unique onset positions from 7,988 scores (after data augmentation).

10.5.2.4 Cadence Detection

For the Cadence Detection task, we use four distinct annotated datasets, the Mozart Piano Sonatas [35], Haydn String Quartets [36], Mozart String Quartets [37], and Bach WTC Fugues [38]. The created graph collectively contains 300,602 nodes and 1,392,753 edges from 153 scores. We create a random data split with 70% of the data for training, 10% for validation, and 20% for testing.

10.6 EXPERIMENTS

Our model for each of the four tasks is built on the respective current state-of-the-art model presented in Section 10.3. From here on forward, we refer to the previous state-of-the-art architecture as *baseline* model, and to the same architecture with the convolutional blocks replaced with our new ones, as *MusGConv model*. These original models serve as the baselines in the following experiment. We follow the implementations of the publicly available code with no major modifications, except for the sampling technique that we highlighted before. For each task, we consider the main evaluation metric proposed in the original paper (presented in Section 10.3).

All experiments for a certain task are run with the best hyperparameter setting specified in the respective papers; this includes 2 GNN layers, and the convolutional blocks being SageConv [31] for all tasks except the voice separation where ResGatedGraphConv [39] is used. We use a fixed training, validation, and test split for each task, and every experiment is run 10 times with different NN initialisations on a single GPU. We used one GTX 1080 Ti GPU with 11 GB of VRAM.

10.6.1 Main Results

The goal of our main experiment is to quantitatively verify whether the use of MusGConv can improve the results on the four tasks compared to the respective baseline, i.e., the architectures used in the corresponding state-of-the-art approaches. Each experiment is run 10 times with the fixed task-specific data split (as described in the previous section) and different random seeds. We report ASO significance [40] with a confidence level $\alpha = 0.05$ and $\epsilon_{\text{min}} < 0.1$.

The results, summarised in Table 10.1, show that MusGConv(+EF) produces statistically significant better results for the Voice Separation and Cadence Detection task. In the Composer Classification task, the best-performing model is MusGConv, while MusGConv(+EF) yields worse results than the baseline. We suspect that being this a global graph classification task, the edge features propagation is harder to

	Voice Separation (Link Prediction)	Composer clf (Graph Classification)	RNA (Node Classification)	Cadence Detection (Node Classification)
Previous SOTA Arch	0.8111 ± 0.058	0.4288 ± 0.031	0.3221 ± 0.010	0.4065 ± 0.011
MusGConv	0.8142 ± 0.035	$\mathbf{0.5233} \pm 0.032$	0.3126 ± 0.015	0.4126 ± 0.016
MusGConv(+EF)	$\mathbf{0.8436} \pm 0.032$	0.3939 ± 0.018	0.3177 ± 0.010	$\mathbf{0.4295} \pm 0.009$

Table 10.1: Experimental comparison with previous SOTA models. The evaluation metric varies for the different tasks (see the corresponding subsections in Section 10.3). Marked in bold are the best results when they are statistically significant.

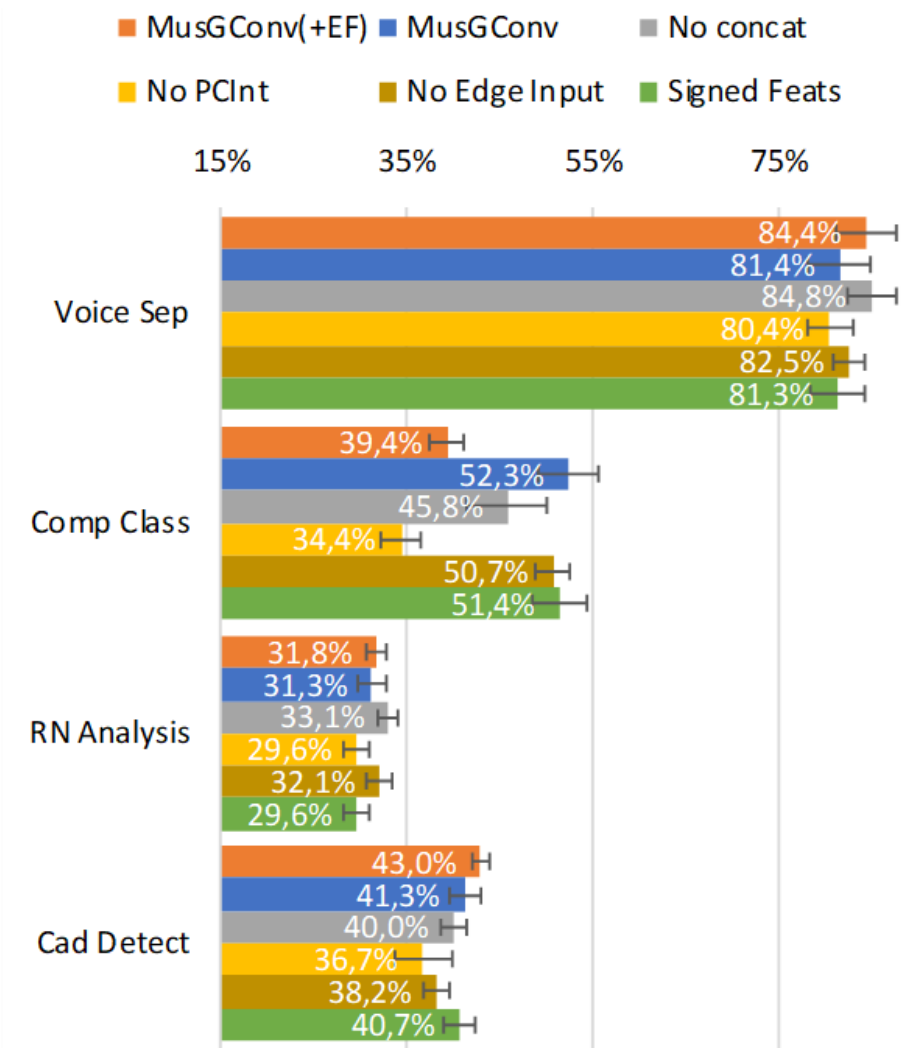


Figure 10.5: Ablation studies.

train, and the simpler mechanism employed in MusGConv is a better choice. There is no statistically significant difference in the performance on the Roman Numeral Analysis task. We found two potential explanations for this behaviour: first, the RNA model is very complex, with multiple components which could hide the effect of a modification on the graph encoder. Moreover, the RNA dataset is augmented with transpositions in all keys, and therefore having transposition-invariant features may only yield minimal (if any) advantage. An experiment without augmentation is not possible, since the output of the RNA model depends on the absolute pitches at the input, and the not-augmented dataset is very small.

Regarding execution time, for each task, we compute the ratio between the average time of the 10 runs for baseline and the 10 runs with MusGConv and MusGConv(+EF). Aggregated across all tasks, this ratio has an average of 1 ± 0.05. Thus, the usage of MusGConv has a minimal impact on the final execution time.

10.6.2 Ablation Studies

We conduct four ablation studies to explore different model variations in terms of architecture and selection of edge features. The results are reported in Figure 10.5, where we also include the results for MusGConv and MusGConv(+EF) from our main experiment (see previous section) for comparison. For some variations we consider, there is not a clear winner, meaning that, while the usage of relative features as edges is beneficial overall, different versions of our system can perform better on different musical tasks.

NO CONCATENATION. We change the feature aggregation function ϕ (Eq. 10.2) from a concatenation (Eq. 10.8) to a multiplication, to mimic the operation used in convolutional blocks that deal with edge features, e.g., [19–21]). The results show that this degrades the performance for the composer classification and cadence detection tasks while improving RN analysis and Voice Separation.

NO EDGE INPUT. In this study, we ignore our manually built edge features from Section 10.4.1 and use node feature difference (see Eq. 10.9) as edge features for all blocks (including the first). This is similar to the edge features employed in EdgeConv [20] (though it has to be noted that our message passing is different from EdgeConv). This degrades the model performance on all tasks but RN.

NO PCINT. We quantify the effect of the PCInt edge feature as a much more music-specific edge feature, compared to the "more standard" feature distances. We observe that the usage of this feature improves all tasks.

SIGNED FEATURES. We evaluate the use of features obtained by removing the absolute value operation in Eq. 10.6. Indeed this is a more informative input; for example, with the absolute value we encode the difference between two note durations, but we lose the information about which is longer. On the other hand, it increases the input numerical range, and one could argue that the network already has access to absolute features, and PCInt and edge type encode similar information. The results show that using signed features is not beneficial.

10.7 CONCLUSION AND FUTURE WORK

This paper has presented a graph convolution block dedicated to music understanding tasks. Its working mechanism is inspired by perceptual considerations and permits the propagation of transposition-invariant and relative timing information in the message-passing process. More generally, our work enables an elegant and efficient way to use pairwise note features, which have been long studied and employed in monophonic music, for polyphonic music processing. Specifically, our approach can be summarized in two core contributions: pitch and time pairwise functions as edge features, and a new way of aggregating this information inside the convolutional block. The design of this block is kept simple to give us a performance advantage without increasing computation time. We experimentally

verify the validity of our proposition on four rather diverse musical tasks covering three graph-related problems: graph classification, node classification, and link prediction.

As future work, it would be interesting to evaluate MusGConv on other kind of music and to investigate the impact of other pairwise note functions, like the one proposed in the cognitively plausible music model IDyOM [10, 11].

ACKNOWLEDGEMENTS

This work was supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab).

REFERENCES

[1] Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter, Carlos Cancino-Chacón, Maarten Grachten, and Gerhard Widmer. "The match file format: Encoding Alignments between Scores and Performances." In: *Proceedings of the Music Encoding Conference (MEC)*. Halifax, Canada, 2022.

[2] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "MidiTok: A Python package for MIDI file tokenization." In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[3] Yo-Wei Hsiao and Li Su. "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 285–292.

[4] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[5] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[6] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[7] Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer, and Carlos Eduardo Cancino-Chacón. "Symbolic Music Representations for Classification Tasks: A Systematic Evaluation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[8] Emmanouil Karystinaios and Gerhard Widmer. "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[9] Diana Deutsch. *Psychology of music*. Elsevier, 2013.

[10] Marcus T. Pearce. "Statistical learning and probabilistic prediction in music cognition: Mechanisms of stylistic enculturation." In: *Annals of the New York Academy of Sciences* 1423.1 (2018), pp. 378–395.

[11] Marcus Pearce. "The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition." PhD thesis. UK: City University of London, 2005.

[12] Mathieu Kermarec, Louis Bigo, and Mikaela Keller. "Improving Tokenization Expressiveness With Pitch Intervals." In: *Late-Breaking Demo Session of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[13] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. "Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music." In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 276–280.

[14] Francesco Foscarin, Nicolas Audebert, and Raphaël Fournier-S'Niehotta. "PKSpell: Data-driven pitch spelling and key signature estimation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[15] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. "Learning Transposition-Invariant Interval Features from Symbolic Music and Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[16] Anders Elowsson and Anders Friberg. "Modeling Music Modality with a Key-Class Invariant Pitch Chroma CNN." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[17] Stefan Lattner, Monika Dörfler, and Andreas Arzt. "Learning Complex Basis Functions for Invariant Representations of Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[18] Bruno Di Giorgi, Matthias Mauch, and Mark Levy. "Downbeat Tracking with Tempo-Invariant Convolutional Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[19] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point Convolutional Neural Networks by Extension Operators." In: *ACM Transactions on Graphics* 37.4 (2018).

[20] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds." In: *ACM Transactions on Graphics* 38.5 (2019), 146:1–146:12.

[21] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. "E(n) Equivariant Graph Neural Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021.

[22] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. "Modeling Relational Data with Graph Convolutional Networks." In: *Proceedings of the Semantic Web International Conference, ESWC*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607.

[23] Ben Duane and Bryan Pardo. "Streaming from MIDI using constraint satisfaction optimization and sequence alignment." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2009.

[24] Tsung-Ping Chen, Li Su, et al. "Functional Harmony Recognition of Symbolic Music Data with Multi-task Recurrent Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[25] Gianluca Micchi, Mark Gotham, and Mathieu Giraud. "Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 3.1 (2020), pp. 42–54.

[26] Andrew Philip McLeod and Martin Alois Rohrmeier. "A modular system for the harmonic analysis of musical scores using a large vocabulary." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[27] Néstor Nápoles López, Mark Gotham, and Ichiro Fujinaga. "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[28] Christopher Harte. "Towards automatic extraction of harmony information from music signals." PhD thesis. Queen Mary University of London, 2010.

[29] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[30] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for pre-training graph neural networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[31] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[32] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. "Layer-dependent importance sampling for training deep and large graph convolutional networks." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2019.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[34] Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription*. Springer, 2006, p. 456.

[35] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. "The Annotated Mozart Sonatas: Score, Harmony, and Cadence." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4.ARTICLE (2021), pp. 67–80.

[36] David RW Sears, Andreas Arzt, Harald Frostel, Reinhard Sonnleitner, and Gerhard Widmer. "Modeling harmony with skip-grams." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2017.

[37] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[38] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational Fugue Analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96. DOI: 10.1162/COMJ_a_00300. URL: https://hal.science/hal-01113520.

[39] Xavier Bresson and Thomas Laurent. "Residual Gated Graph ConvNets." In: *Proceedings of the International Conference on Learning Representations*. 2017.

[40] Eustasio Del Barrio, Juan A Cuesta-Albertos, and Carlos Matrán. "An optimal transportation approach for assessing almost stochastic order." In: *The Mathematics of the Uncertain*. Springer, 2018, pp. 33–44.

11 A LIBRARY FOR SYMBOLIC MUSIC GRAPH PROCESSING

Title: GraphMuse: A Library for Symbolic Music Graph Processing.

Published In Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR), San Francisco USA, 2024.

Authors: Emmanouil Karystinaios, Gerhard Widmer

Abstract: Graph Neural Networks (GNNs) have recently gained traction in symbolic music tasks, yet a lack of a unified framework impedes progress. Addressing this gap, we present GraphMuse, a graph processing framework and library that facilitates efficient music graph processing and GNN training for symbolic music tasks. Central to our contribution is a new neighbor sampling technique specifically targeted toward meaningful behavior in musical scores. Additionally, GraphMuse integrates hierarchical modeling elements that augment the expressivity and capabilities of graph networks for musical tasks. Experiments with two specific musical prediction tasks – pitch spelling and cadence detection – demonstrate significant performance improvement over previous methods. Our hope is that GraphMuse will lead to a boost in, and standardization of, symbolic music processing based on graph representations. The library is available at https://github.com/manoskary/graphmuse

11.1 INTRODUCTION

Symbolic music processing entails the manipulation of digital music scores, encompassing various formats such as MusicXML, MEI, Humdrum, **kern, and MIDI. Unlike audio-based representations, symbolic formats offer granular information on note elements, including onset, pitch, duration, and other musical attributes like bars and time signatures.

While prior research in symbolic music processing often adopted techniques from the image processing [1–3] or natural language processing [4–6] domains, recent attention has shifted towards graph-based models, which could presumably better capture the dual sequential and hierarchical nature of music. Graph Neural Networks (GNNs) have been showcased as potent tools for diverse symbolic music tasks, including cadence detection[7], optical music recognition [8], music generation [9], Roman numeral analysis [10], composer classification [11], voice separation[12], and expressive performance rendering[13]. However, a standardized framework for constructing and processing music graphs has not yet been introduced to the field. To address this challenge, we developed GraphMuse, a Python-based framework to efficiently and effectively process information from

musical scores, construct musically meaningful graphs, and facilitate the training of graph-based models for symbolic music tasks.

A key innovation of our work lies in the introduction of a new sampling technique tailored to specific properties of music while maintaining efficient and robust training of GNNs. Additionally, GraphMuse integrates within the graphs and models hierarchical elements that augment the capabilities of graph networks for musical tasks.

We evaluate our framework on pitch spelling and cadence detection tasks, comparing it against existing state-of-the-art methods. Through the synergistic utilization of our framework's components, we achieve a significant performance increase compared to the previous methods. Our overarching objective is to establish a standardized framework for graph processing in symbolic music analysis, thus catalyzing further progress in the field.

Altogether, our contributions are three-fold: i) We provide a structured, generic, and flexible framework for graph-based music processing; ii) we release an open source *Python* library that comes with it; iii) we achieve performance improvements in a principled way by focusing on the design of the individual parts of the framework.

11.2 PROCESSING MUSIC SCORES WITH GNNS

In this section, we describe existing graph modeling approaches for musical scores. They all have a common pipeline which involves building a graph from a given musical score (see Figure 11.1) and using a series of convolutional blocks to produce context-aware hidden representations for each node. We start by describing the graph-building procedure and a generic graph convolutional block; we then take a detailed look at the problem of graph sampling, which will motivate a new sampling procedure that will be presented in the next section.

11.2.1 Preprocessing: Constructing Graphs from Scores

A score graph can be represented as a heterogenous attributed graph. A heterogeneous graph has a type associated with each node and edge in the graph [14]. An attributed graph has an associated feature vector for each node in the graph [15]. Therefore, a heterogenous attributed graph is defined by a quintuple $G = (V, E, X, \mathcal{A}, \mathcal{R})$, together with the mappings $\phi : V \to \mathcal{A}$ and $\psi : E \to \mathcal{R}$, where V is the set of nodes, E is the set of edges, $X \in V \times \mathbb{R}^k$ the feature matrix $\mathcal{A}$ is the node types and $\mathcal{R}$ is the edge types. ϕ maps each node to its type and ψ maps its each edge to its corresponding type.

We create such a graph from a musical score by following previous work [10–13]. Each node $v \in V$ corresponds to one and only one note in the musical score. $\mathcal{R}$ includes 4 types of relations: onset, during, follow, and silence, corresponding, respectively, to two notes starting at the same time, a note starting while the other is sounding, a note starting when the other ends, and a note starting after a time when no note is sounding. The inverse edges for during, follows, and silence relations are also created.

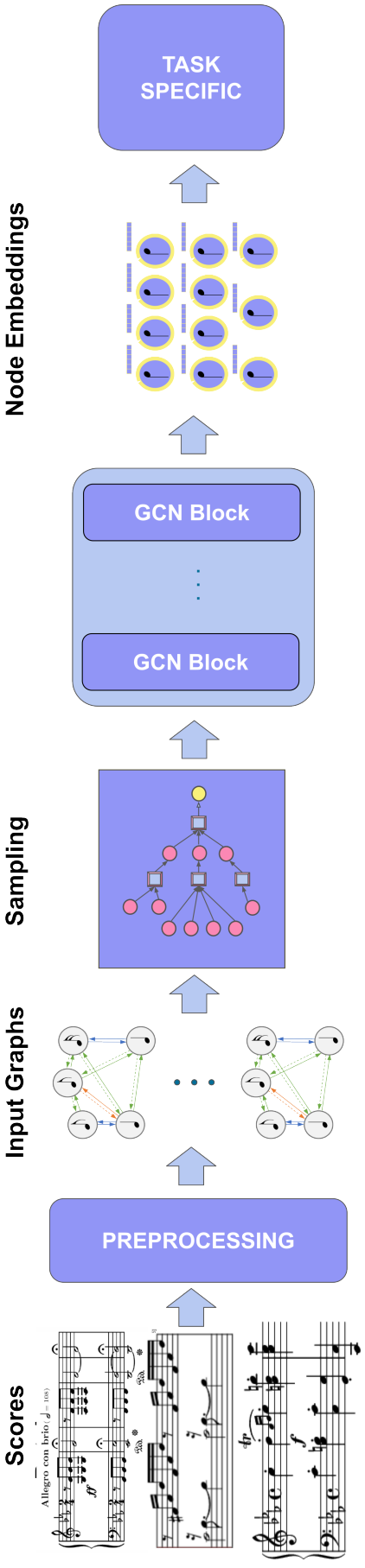


Figure 11.1: The general graph processing/training pipeline for symbolic music scores involves several steps: i) Preprocess the database of scores to generate input graphs; ii) Sample the input graphs to create memory-efficient batches; iii) Form a batch as a new graph with nodes and edges from various input graphs; iv) Sample a subset of nodes (target nodes) and their neighbors from the input graphs; v) Update the target nodes' representations through graph convolution to create node embeddings; vi) Use these embeddings for task-specific applications. Note that target nodes may include all or a subset of batch nodes depending on the sampling strategy.

Formally, let us consider three functions $on(v)$, $dur(v)$, and $pitch(v)$ defined on a note $v \in V$ that extract the onset time, duration, and pitch, respectively. A typed edge (u, r, v) of type $r \in \mathcal{R}$ between two notes $u, v \in V$ belongs to E if the following conditions are met:

- $on(u) = on(v) \rightarrow r = \text{onset}$
- $on(u) > on(v) \wedge on(u) \leq on(v) + dur(v) \rightarrow r = \text{during}$
- $on(u) + dur(u) = on(v) \rightarrow r = \text{follow}$
- $on(u) + dur(u) < on(v) \wedge \nexists v' \in V,\ on(v') < on(v) \wedge on(v') > on(u) + dur(u) \rightarrow r = \text{silence}$

$\mathcal{A}$ in the literature usually only includes a single type, i.e. the note type ν. However, we extend this definition in Section 11.3.1.

11.2.2 Encoding: Graph Convolution

Graph convolution and message passing are core operations in graph neural networks (GNNs) for learning node representations. In graph convolution, in its simplest form, each node aggregates messages from its immediate neighbors by computing a weighted sum of their features:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(l)} \mathbf{h}_u^{(l)}\right) + \mathbf{h}_v^{(l)}\right) \tag{11.1}$$

where $\mathbf{h}_v^{(l)}$ is the representation of node v at layer l, $\mathcal{N}(v)$ denotes the neighbors of node v, $\mathbf{W}^{(l)}$ is a learnable weight, and σ is a non-linear activation function. Through successive iterations of message passing and aggregation, each node refines its representation by incorporating information from increasingly distant nodes in the graph, ultimately enabling the network to capture complex relational patterns and dependencies within the graph data.

In the context of music, graph convolution can be understood as a method for defining a note not only by its own characteristics and properties but by also considering the characteristics of its neighboring notes within the musical graph. In this work, as well as previous graph-based work on music [7, 10, 11] the preferred graph convolutional block is *SageConv* taken from one of the first and fundamental works in graph deep learning [16].

11.2.3 Sampling: Handling Graph Data for Training

In an ideal world without computing resource considerations, we can imagine a training pipeline that receives an entire graph as input to a graph convolutional model. Assuming that we have the resources and time to perform such a task the process is easy to grasp. All nodes of the graph are updated in a single step based on their neighbors as described in the previous section.

However, the graph world presents us with several complexity issues. Graph datasets in the wild typically come in two forms: i) a (possibly large) collection of

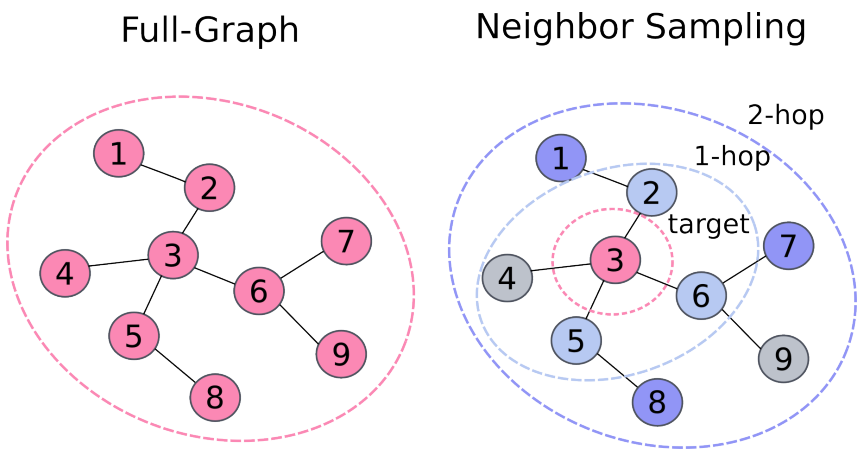


Figure 11.2: Full graph vs neighbor sampling. The pink-colored nodes are selected for convolution by message passing. With neighbor sampling, the pink node is the one whose representation is ultimately updated after convolution (however, for the blue nodes also take part in the convolution process as its context).

small graphs, each containing maybe fewer than 50 nodes[15]; ii) a single large-scale graph such as a social network [17], a recommender system [18], or a knowledge graph [19]. The previous naive scenario presents a time-efficiency and computation waste bottleneck for the former and a memory insufficiency issue for the latter. To mitigate these issues, in the former case one can batch many small graphs together to maximize the available resources and reduce the computation time, then the full graphs can be updated during convolution within each batch.

Training Graph Convolutional Networks (GCNs) for large-scale graphs is a bit more complicated. Such graphs can be exceptionally large – for example, the 2019 Facebook social network boasted 3.51 billion users[1]. To train models with such graphs we need to devise a sampling algorithm to derive subgraphs in steps [16, 20–22]. Such an algorithm may, for example, choose a subset of nodes across the graph and perform random walks to fetch a subset of the k-hop neighbors for the sampled nodes [16]. This process, called *neighbor sampling* or *node-wise sampling*, is shown in Figure 11.2 and compared to the full-graph process.

Musical score graphs fall in between the two scenarios, varying notably in size. For instance, a Bach Chorale might contain 100 notes, while a Beethoven Sonata could exceed 5000 notes, with each note corresponding to a graph node. Furthermore, a musical dataset may contain many such graphs. Therefore the question arises how to efficiently train models on music graph datasets.

Since music graphs are not uniform enough to be batched together like small graph datasets, we investigate the suitability of neighbor sampling methods for music graph processing, taking into account special properties relevant in music. Standard neighborhood sampling would sample notes across different scores and fetch neighbors for those notes, creating a subgraph that can maximize the use of the available resources during training.

However, music has a specific coherence, in both the horizontal (time) and vertical (chords, harmonies) dimensions, which makes sampling approaches from the literature[22] not appropriate for music. Specifically, sampling and updating/encoding

1 https://zephoria.com/top-15-valuable-facebook-statistics

single notes without simultaneously doing so also to notes in their local context makes it difficult to learn properties that persist in time (such as local key or a harmonic function). In this work, we address this issue by presenting a simple and musically intuitive sampling process for graphs that efficiently creates batches containing musically related notes which, as experiments will show, can notably improve the learning results.

11.2.4 Task-specific Modeling

Finally, the node embeddings created by the graph convolutional encoder serve as input to task-specific models that solve some specific prediction or recognition task. In a graph context, we distinguish, at an abstract level, between node classification, link prediction, and entire graph classification tasks. Examples of node classification tasks can be found in [7] which takes the embeddings from the GCN encoder and employs an edge decoder coupled with a graph convolution classifier for cadence prediction labels; and in [10], which forwards the embeddings to sequential layer and then MLP classifiers to perform Roman Numeral Analysis. In [12], musical voice separation is framed as a link prediction task; the node embeddings are input to a pairwise edge similarity encoder to predict link probabilities between notes in the same voice. An example of a graph classification task can be found in [11] where the embeddings are aggregated and passed through a shallow MLP for composer classification.

Naturally, task-specific models will not be part of the generic graph processing pipeline and library which we publish with this paper.

11.3 METHODOLOGY

In this section, we discuss our approach to addressing the different components of the pipeline shown in Figure 11.1. In particular, we explain the preprocessing procedure for creating score graphs, we detail our strategy for musically intuitive graph sampling, and finally, we discuss model variants that are made possible by the previous steps of the pipeline.

11.3.1 Preprocessing

The central activity in the preprocessing step is the creation of graphs from musical scores. In our library, we extend the conventional graph creation process by introducing hierarchical musical dimensions (beats and measures), in order to enhance the score graphs' representational capacity. More specifically, we enrich the node type set $\mathcal{A}$ (defined in Section 11.2.1) with two additional types β and μ for beats and measures respectively. The process involves detecting beats and measures within the musical score, generating edges (of type *connect* to every beat from each note falling within its temporal boundaries, and repeating this process for measures. Additional edges of type *next* are drawn between consecutive beats and measures to enrich the connectivity and contextual understanding within the

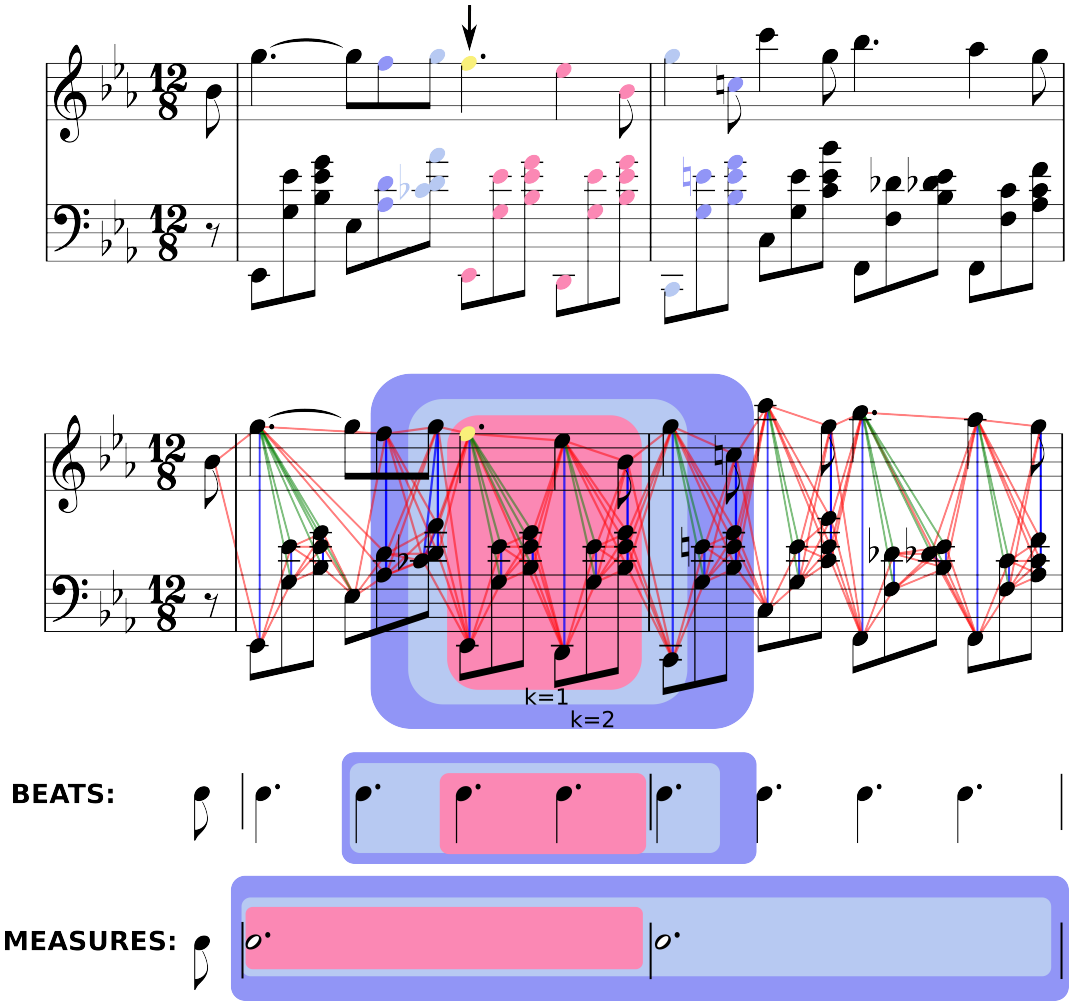


Figure 11.3: Sampling process per score. Top: sampled notes and their neighbors; middle: score graph and sampling process; bottom: sampling process for beats and measures. A randomly selected note (in yellow) is first sampled. The boundaries of the target notes are then computed with a budget of 15 notes in this example (pink and yellow notes). Then the k-hop neighbors are fetched for the targets (light blue for 1-hop and darker blue for 2-hop). The k-hop neighbors are computed with respect to the input graph (depicted with colored edges connecting noteheads in the figure). We can also extend the sampling process for the beat and measure elements (introduced in Section 11.3.1). Note that the k-hop neighbors need not be strictly related to a time window.

graph. Furthermore, we aggregate features from constituent notes through the *connect* edges via message passing to equip each beat and measure with informative attributes by computing the mean vector of their note features.

The inclusion of beat and measure node elements, as well as the creation of inverse edges, are made optional, ensuring compatibility with diverse research needs and avoiding imposing rigid structures onto the graph-based music processing framework.

We prioritize the efficiency and speed of the graph creation process by transitioning the graph creation implementation to C code, leveraging its performance benefits, and establishing a Python binding for seamless integration into our workflow. Recognizing the temporal nature of musical elements, such as notes, beats, and measures, we refine our neighbor searching windows accordingly, optimizing computational efficiency.

11.3.2 Sampling

We discussed general neighbor sampling for large-scale graphs in Section 11.2.3 and some problems related to graph-structured music data. In this section, we elaborate on our musically informed sampling process for music graphs, which enables the training of the models outlined in the subsequent sections. In this process, we aim to sample sections of scores and employ neighbor sampling to fetch the neighbors of notes within those sections.

Indeed while our nodes could be ordered in various ways, the most perceptually significant aspect is time organization. Recognizably, individuals can still identify a musical piece when segmented along the time axis, whereas focusing solely on pitch intervals may be challenging. Moreover, perceptual research indicates that the commencement time of a note holds greater salience than its offset time, particularly for percussive instruments like the piano, where the sound naturally fades over time [1]. Hence, when constructing graphs from musical scores, we prioritize node arrangement based on absolute onset time followed by pitch.

Our initial limitations are mostly related to memory usage. To limit our memory we need to predefine three initial arguments: i) the size of each target subgraph S from every score, ii) the number B of scores in each batch, and iii) the number of hops and neighbors for each hop (similar to node-wise sampling techniques). In each batch, we update the representation of our target nodes which is essentially the size of $S \times B$.

Once the ordering is set and the three arguments are defined we can initiate the process of sampling a subgraph, as shown in Figure 11.3. First, we sample a random note from the graph of each score. Next, we correct the position of the note by searching for any vertical neighbors (same onset value notes and potentially different pitch). Then we extend to S notes to the right where S indicates a predefined maximum subgraph size. We also correct the rightmost boundaries to include or exclude vertical neighbors for the last onset always respecting the aforementioned size S. Once this process is completed we obtain the target nodes per score within the batch. These are the nodes whose representation we want to update at the end of the graph convolutional process.

However, since graph convolution is performed recursively we need to fetch the k-hop neighbors for each one of the target nodes where k indicates the depth of the GCN. For this step, we can consult the literature [16] and perform neighbor sampling to fetch the k-hop neighbors. This process is repeated for B different scores. Finally, the B score subgraphs of size at most S each are first joined together and then fed to the model.

During this process, we can keep information about the target nodes and the size of each score subgraph, which could allow us to design more creative models that can exploit this information. Such models are presented in the next section. Moreover, we adopt a potential approach for hierarchical graphs by also extending the sampling for beat and measure nodes as shown in Figure 11.3.

11.3.3 Model Designs

In this section, we explore various model designs for the graph-based encoder in our processing pipeline (Figure 11.1). Designing such an encoder involves addressing two fundamental questions: the selection of graph convolutional blocks and the selective exploitation of information from the input graph.

The first question, regarding graph convolutional blocks, remains open-ended, offering numerous possibilities for exploration and customization. In its current version, *GraphMuse* offers the options of convolutional blocks on a per-node or per-edge type basis. We suggest that graph-attention networks may offer promising avenues, particularly for hierarchical elements such as beats or measures.

In response to the second question, we devise a series of models by selectively incorporating or excluding elements from the input graph. Our foundational model, termed *NoteGNN*, exclusively utilizes note elements and their corresponding edges. This model serves as the basis for further extension. For instance, we expand upon *NoteGNN* to construct *BeatGNN*, which incorporates beat elements (see Section 11.3.1 above) alongside notes. Similarly, we develop *MeasureGNN* by integrating measures into the encoding process. When all note, measure, and beat elements are included, the resultant model is denoted as *MetricalGNN*.

Furthermore, we explore the possibility of hybridizing model types, such as combining GNNs with sequential models. This hybridization is facilitated by the sampling process that organizes notes in onset order, allowing for the batch to be unfolded by score. Consequently, the same batch can be processed through both GNN and sequential models simultaneously. Specifically, we employ a graph encoder and a sequential encoder in parallel – in our case we use a stack of 2 bidirectional GRU layers. The GRU stack receives the unfolded batch of size (B, S, K) where B is the number of scores within the batch, S is the number of sampled target nodes for each score order by onset and then by pitch, and K is the number of node features. The embeddings of both encoders are concatenated together and an additional linear layer is applied to project them to the required dimension.

This architecture, which we call *HybridGNN* in our experiments, combines the strengths of both GNNs and sequential models, resulting in better performance as shown in our experiments.

11.3.4 The Library

The components discussed in the preceding section have been implemented and made available in an open-source Python library called GraphMuse. This library follows a similar philosophy as PyTorch and PyTorch Geometric, comprising models and graph convolutional blocks, loader pipelines, data pipelines, and related utilities. GraphMuse is built upon and thus requires PyTorch and PyTorch Geometric. The loaders and models provided by GraphMuse are fully compatible with those of PyTorch Geometric. For musical input and output, GraphMuse is compatible with Partitura [23], a Python library for symbolic music processing. GraphMuse can be obtained from `anonymous.com`

11.4 EVALUATION

To evaluate our framework, we perform experiments on two tasks, cadence detection and pitch spelling. We put to the test both the models discussed as well as the sampling process. For pitch spelling, we compare our models to the previous sequential state-of-the-art model, PKSpell [24] and the GraphSAGE variant of our note-level model. For cadence detection, we compare our models to the previous state-of-the-art model by Karystinaios and Widmer [7] which is also graph-based and follows a GraphSAGE sampling strategy. For both tasks, we perform ablations by removing the hierarchical elements such as beat and measure nodes and edges, or incorporating hybrid models.

11.4.1 Pitch Spelling

Previous work on Pitch Spelling set the state-of-the-art by using a sequential model [24]. The task of pitch spelling tackles in parallel key signature estimation and pitch spelling estimation per note, however, the key signature is a global attribute usually set for the whole piece although it can sometimes change midway. The previous architecture uses a GRU encoder for pitch spelling and then infuses the logits together with the latent representation to another GRU layer for the key signature prediction.

For our approach, we use a GNN encoder as described in Section 11.3.3 followed by two classification heads for key and pitch spelling respectively. We train and evaluate all models on the ASAP dataset [25] using a random split with 15% of the data for testing and the 85% for train and validation as described in [24].

11.4.2 Cadence Detection

For the cadence detection model, we chose to use a modified version of the cadence detection model originally proposed in [7]. Our considerations were based on a more efficient training process, and the integration of our pipeline possibilities. The model was expanded to accept a heterogeneous score graph as input, as described in Section 11.2.1. Additionally, we enhanced the model's predictive capabilities from binary (no-cad or PAC) to multiclass cadence prediction, encompassing PAC, IAC,

and HC labels. Furthermore, we refined the architecture by incorporating an onset regularization module, which aggregates the latent representations (post-GNN encoder) of all notes occurring at a distinct onset within the score and assigns them to every note sharing that onset.

In the training phase, the input graph first undergoes processing through the graph encoder. The resulting node embeddings are then grouped based on onset information extracted from the score, and their representations are averaged. Subsequently, embedded SMOTE [26] is applied to balance the distribution of cadence classes compared to the notes lacking cadence labels in the score. However, during inference, this synthetic oversampling step is omitted. Finally, the oversampled embeddings are fed into a shallow 2-layer MLP classifier to predict the cadence type.

We trained our model with a joined corpus of cadence annotations from the DCML corpora[2], the Bach fugues from the well-tempered clavier Book No.1 [27], the annotated Mozart string quartets [28], and the annotated Haydn string quartets [29]. Our joined corpus makes for 590,149 individual notes and 17,188 cadence annotations. We use 80% of the data for training and validation and test on 20% using a random split. Note that these results cannot be directly compared with [7] since we use a different (bigger) dataset and perform multiclass prediction.

11.4.3 Experiments

11.4.3.1 Configuration

The configuration used for training all pitch spelling graph models trained with our sampling technique has a batch size B fixed at 300, i.e. sampling from 300 scores at each training step, and the size of the target nodes of the sampled subgraph S is also set to 300. For the respective cadence graph models, $B = 200$ and $S = 500$. For all graph models (including GraphSAGE), we use three heterogeneous SageConv layers [16] in the encoder with a hidden size of 256 and a dropout of 0.5. The neighbor sampling corresponding to each one of the layers fetches at most 3 neighbors per sampled node per relation. We train all models with the Adam optimizer with a learning rate of 10^{-3} and a weight decay of $5 * 10^{-4}$. All models have been trained on a *GTX 1080 Ti*. Every experiment is repeated four times with different random seeds.

11.4.3.2 Results

Table 11.1 presents the results of experiments experiments conducted on the two tasks. The metrics used for evaluation are Accuracy (A) for pitch spelling and key recognition, and the macro F1 score ($F1$) for cadence detection. Note that the model employed on the GraphSAGE methods and the model NoteGNN are virtually the same apart from the sampling strategy with which they were trained.

For the pitch spelling task, we can observe that the actual pitch spelling accuracy (A-Pitch) of all proposed models surpasses both the PKSpell and GraphSAGE methods. Across all models, the MetricalGNN achieves the highest accuracy of

2 https://github.com/DCMLab/dcml_corpora

Task	Pitch Spelling		Cadence
	A-Pitch	A-Key	F1-Cad
PKSpell	94.8 ± 0.5	$\underline{69.9} \pm 1.6$	-
GraphSAGE	93.6 ± 0.1	43.3 ± 0.1	53.5 ± 0.8
NoteGNN	94.9 ± 0.1	69.3 ± 7.0	55.3 ± 0.9
BeatGNN	95.1 ± 0.2	68.7 ± 1.1	$\underline{57.4} \pm 1.2$
MeasureGNN	$\underline{95.4} \pm 0.3$	69.5 ± 7.2	57.0 ± 1.0
MetricalGNN	$\mathbf{95.6} \pm 0.1$	64.4 ± 5.3	55.8 ± 0.6
HybridGNN	$\underline{95.4} \pm 0.2$	$\mathbf{72.6} \pm 2.8$	$\mathbf{58.6} \pm 0.7$

Table 11.1: Results on the two tasks, in terms of accuracy (A) and F1 score, respectively. Values in bold are the best score per metric; underlined values are the second best. All runs are repeated 4 times. $\pm$ indicates standard deviation.

95.6%, closely followed by BeatGNN and MeasureGNN with accuracies of 95.1% and 95.4%, respectively. These results indicate the benefits of incorporating hierarchical musical elements such as beats and measures. However, it is worth noting that while MetricalGNN achieves the highest accuracy, it is closely followed by the hybrid model, HybridGNN, which achieves an accuracy of 95.4%, suggesting that competitive performance can also be achieved by mixing model types.

Focussing on the key estimation subtask (A-Key) of pitch spelling we notice that the PKSpell model achieves a very good key accuracy of 69.9%, closely followed by the MeasureGNN model and only surpassed by the Hybrid model. We attribute the effectiveness of key detection of a sequential model such as PKSpell to the persistence of the key label across elements of the sequence. Therefore, a hybrid model in this case seems to be able to adapt to the diversity of labels for pitch spelling and uniformity of labels for key estimation.

In the cadence detection task, we evaluate the results using the macro F1 score to account for the overwhelming presence of non-cadence nodes, as instructed by [7]. We observe that GraphSAGE, the previously used technique for training, obtains the lowest F1 score and it is surpassed by all the proposed GNN-based models trained with the new sampling method.

Among our GNN models, BeatGNN and HybridGNN achieve the highest scores of 57.4% and 58.6%, respectively, closely followed by MeasureGNN. In this case, the MetricalGNN model surprisingly does not achieve such a good score even though it includes both measure and beat elements. However, it still performs better than the NoteGNN and the GraphSAGE method.

Overall, the results demonstrate the efficacy of GNN-based models trained using our new sampling method. Incorporating hierarchical elements such as beats and measures improves both pitch spelling and cadence detection tasks. Additionally, the hybrid approach of combining GNNs with sequential models produces promising results.

11.5 CONCLUSION

In this paper, we introduced GraphMuse, a framework and Python library for symbolic graph music processing. We designed a specialized sampling process for musical graphs and demonstrated our pipeline's effectiveness through experiments on pitch spelling and cadence detection. Our results show that carefully designed GNN architectures, especially those incorporating hierarchical elements like beats and measures, can lead to better performance. Finally, hybrid models that integrate GNNs with sequential models yield further performance improvements.

Future research will focus on refining GNN-based models in music processing, adding more tasks, and exploring novel architectures. This includes investigating advanced graph convolutional blocks, other sampling techniques, and attention mechanisms to enhance model performance.

11.6 ACKNOWLEDGEMENTS

The authors would like to thank Nimrod Varga for his contribution to accelerating graph creation by adapting it to C code. This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab).

REFERENCES

[1] Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. "Score-informed transcription for automatic piano tutoring." In: *European Signal Processing Conference (EUSIPCO)*. 2012.

[2] Gissel Velarde, Tillman Weyde, Carlos E. Cancino-Chacón, David Meredith, and Maarten Grachten. "Composer recognition based on 2D-filtered piano-rolls." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[3] Néstor Nápoles López, Mark Gotham, and Ichiro Fujinaga. "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[4] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "MidiTok: A Python package for MIDI file tokenization." In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[5] Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023.

[6] Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. "Symphony Generation with Permutation Invariant Language Model." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[7] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[8] Arnau Baró, Pau Riba, and Alicia Fornés. "Musigraph: Optical Music Recognition Through Object Detection and Graph Neural Network." In: *International Conference on Frontiers in Handwriting Recognition*. Springer. 2022, pp. 171–184.

[9] Emanuele Cosenza, Andrea Valenti, and Davide Bacciu. "Graph-based Polyphonic Multitrack Music Generation." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[10] Emmanouil Karystinaios and Gerhard Widmer. "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[11] Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer, and Carlos Eduardo Cancino-Chacón. "Symbolic Music Representations for Classification Tasks: A Systematic Evaluation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[12] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[13] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[14] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. "Heterogeneous graph attention network." In: *The world wide web conference*. 2019, pp. 2022–2032.

[15] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[17] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems." In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 974–983.

[19] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer. 2018, pp. 593–607.

[20] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. "Layer-dependent importance sampling for training deep and large graph convolutional networks." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2019.

[21] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. "GraphSAINT: Graph Sampling Based Inductive Learning Method." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020.

[22] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. "Sampling methods for efficient training of graph convolutional networks: A survey." In: *IEEE/CAA Journal of Automatica Sinica* 9.2 (2021), pp. 205–234.

[23] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[24] Francesco Foscarin, Nicolas Audebert, and Raphaël Fournier-S'Niehotta. "PKSpell: Data-driven pitch spelling and key signature estimation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[25] Silvan David Peter, Carlos Eduardo Cancino-Chacón, Francesco Foscarin, Andrew Philip McLeod, Florian Henkel, Emmanouil Karystinaios, and Gerhard Widmer. "Automatic Note-Level Score-to-Performance Alignments in the ASAP Dataset." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* (2023).

[26] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[27] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational fugue analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96.

[28] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[29] David RW Sears, Marcus T Pearce, William E Caplin, and Stephen McAdams. "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models." In: *Journal of New Music Research* 47.1 (2018), pp. 29–52.

12 CONCLUSION & FUTURE WORK

12.1 OVERVIEW

This thesis has explored the application of graph-based music representations together with Graph Neural Networks (GNNs) for the automatic music analysis of classical music scores, addressing a range of tasks such as cadence detection, voice separation, and Roman numeral analysis. My research suggests that viewing musical scores as graphs offers significant advantages in capturing the dual hierarchical and sequential structures inherent in polyphonic music.

By introducing new graph models and customized GNN blocks tailored for musical data, I have shown that it is possible to surpass previous methods in accuracy, efficiency, and scalability in automatic music analysis. The development of ChordGNN, MusGConv, MetricalGNN, and other graph-based models highlights the power of graph representations in tasks such as cadence detection, voice separation, Roman numeral analysis, and more. These models not only improved performance but also offered a more intuitive understanding of musical structures, facilitating better interpretability and manipulation of musical data.

Furthermore, I introduced frameworks for explainability on music graphs such as the SMUG-Explain framework which offers an easy visualization of graphs and their explanations making more accessible to music analysts who want to understand and explore how graph-based music analysis models learn. With the introduction of the GraphMuse library for symbolic music graph processing, I have provided the research community with practical resources to continue innovation and exploration in the field.

My work has moved towards the standardization of graph-based symbolic music processing, demonstrating its potential to address complex musical tasks effectively. The successful integration of GNNs in music analysis sets a promising direction for future research, where more sophisticated models and techniques can be developed to further improve our understanding and processing of music.

In conclusion, this thesis establishes a solid foundation for the use of graph-based representations in symbolic music analysis, offering a new perspective that aligns more closely with the intrinsic properties of musical data. The findings and tools developed during my Ph.D. have offered advances to the state-of-the-art in individual music analysis tasks but also contributed to the field of MIR by introducing many open-source resources.

Overall, my work has aimed to contribute to symbolic-based MIR by presenting graph-based methods as a viable option for certain music analysis tasks. While the results are encouraging, they are by no means definitive and much remains to be done to unfold the full potential of these graph-based approaches.

12.2 FUTURE DIRECTIONS

Future research in the field of graph-based symbolic music processing must address several challenges to unlock its full potential. Scalability and computational efficiency remain significant hurdles. As musical compositions grow more complex and datasets expand, the computational resources required for training and inference increase substantially. Optimizing these models to reduce computational costs without sacrificing accuracy is essential for their broader application.

Another challenge lies in the integration of graph-based approaches with other modalities such as audio signals and visual elements like sheet music. Music is inherently multi-modal, and developing frameworks that can effectively combine these diverse data types will be crucial for comprehensive music analysis. Additionally, musical interpretation often involves a degree of ambiguity and subjectivity, as different musicians and analysts may interpret the same piece differently. Creating models that can accommodate such variability and provide multiple plausible interpretations is a complex but necessary task.

Ensuring that graph-based models can generalize across various musical genres and styles is also critical. While current research has primarily focused on Western classical music, there is a vast array of musical traditions worldwide. Adapting these models to different musical contexts will require further exploration. Moreover, the success of deep learning models heavily relies on the availability and quality of annotated data. In symbolic music, creating high-quality annotations for complex tasks is labor-intensive and requires expert knowledge. Addressing data scarcity and improving annotation quality will be vital for the advancement of this field.

Future directions for research include exploring more sophisticated graph representations that capture additional musical attributes such as dynamics, articulation, and phrasing. These enhancements could lead to more nuanced and accurate models that better reflect the complexity of musical scores. Combining graph-based approaches with other machine learning techniques, such as sequence models and convolutional networks, could yield more powerful and flexible models, leveraging the strengths of different architectures for improved performance across various musical tasks.

Developing interactive and real-time applications that utilize graph-based music analysis could potentially impact music education, composition, and performance. Such tools could provide immediate feedback and insights to musicologists and computer scientists working with music, deepening their understanding. Enhancing the explainability and interpretability of graph-based models could also be important for their adoption in musicology and education. Future research should focus on methods that provide clear and musically meaningful explanations for the model's decisions.

Cross-disciplinary collaborations with experts in musicology, cognitive science, and computer science can lead to innovative approaches and a deeper understanding of music analysis. Such collaborations can address complex questions about musical perception, cognition, and creativity, resulting in more holistic and robust models. Continuing the development and dissemination of open-source tools and frameworks will be vital for the growth of this research. Providing accessible and

well-documented resources can encourage wider adoption and experimentation, fostering a collaborative research environment.

In summary, while significant progress has been made, numerous challenges and exciting opportunities lie ahead. Addressing these challenges through innovative research and collaboration will be essential for advancing the field and unlocking the full potential of computational music analysis.

COLLECTIVE BIBLIOGRAPHY

[B1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. "Sanity Checks for Saliency Maps." In: *Neural Information Processing Systems*. 2018.

[B2] Edward Aldwell, Carl Schachter, and Allen Cadwallader. *Harmony and voice leading*. Cengage Learning, 2018.

[B3] Anna Aljanaki, Stefano Kalonaris, Gianluca Micchi, and Eric Nichols. "MCMA: A Symbolic Multitrack Contrapuntal Music Archive." In: *Empirical Musicology Review* 16.1 (2021), pp. 99–105.

[B4] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[B5] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Learning Sonata Form Structure on Mozart's String Quartets." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2.1 (2019), pp. 82–96.

[B6] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks." In: *In Learning on Graphs Conference (Proceedings of Machine Learning Research)* 198 (2022).

[B7] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point Convolutional Neural Networks by Extension Operators." In: *ACM Transactions on Graphics* 37.4 (2018).

[B8] Milton Babbit. "The Use of Computers in Musicological Research." In: *Perspectives of New Music* 3.2 (1965), pp. 74–83.

[B9] *Bach: Prelude and Fugue No.19 in A major, BWV 864 Analysis*. 2018. URL: https://tonic-chord.com/bach-prelude-and-fugue-no-19-in-a-major-bwv-864-analysis/.

[B10] David Back. *Standard MIDI-file format specifications*. Acessed August 29, 2024. 1999. URL: http://www.music.mcgill.ca/~ich/classes/mumt306.

[B11] Federico Baldassarre and Hossein Azizpour. "Explainability techniques for graph convolutional networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[B12] Arnau Baró, Pau Riba, and Alicia Fornés. "Musigraph: Optical Music Recognition Through Object Detection and Graph Neural Network." In: *International Conference on Frontiers in Handwriting Recognition*. Springer. 2022, pp. 171–184.

[B13] Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. "Score-informed transcription for automatic piano tutoring." In: *European Signal Processing Conference (EUSIPCO)*. 2012.

[B14] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. "Simple online and realtime tracking." In: *Proceedings of the IEEE international conference on image processing (ICIP)*. IEEE. 2016.

[B15] Louis Bigo, Laurent Feisthauer, Mathieu Giraud, and Florence Levé. "Relevance of musical features for cadence detection." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[B16] Augustin Bouquillard and Florent Jacquemard. "Engraving Oriented Joint Estimation of Pitch Spelling and Local and Global Keys." In: *International Conference on Technologies for Music Notation and Representation (TENOR)*. 2024.

[B17] Guillem Brasó and Laura Leal-Taixé. "Learning a neural solver for multiple object tracking." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[B18] Xavier Bresson and Thomas Laurent. "Residual Gated Graph ConvNets." In: *Proceedings of the International Conference on Learning Representations*. 2017.

[B19] Xavier Bresson and Thomas Laurent. "An Experimental Study of Neural Networks for Variable Graphs." In: *Proceedings of the International Conference on Learning Representations*. 2018.

[B20] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – A Survey*. 2017. ISBN: 9783319701622. arXiv: 1709.01620. URL: http://arxiv.org/abs/1709.01620.

[B21] Shaked Brody, Uri Alon, and Eran Yahav. "How attentive are graph attention networks?" In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2022).

[B22] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric deep learning: going beyond euclidean data." In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[B23] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[B24] Rainer E Burkard and Eranda Cela. "Linear assignment problems and extensions." In: *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[B25] Emilios Cambouropoulos. "Voice separation: theoretical, perceptual and computational perspectives." In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*. Citeseer. 2006.

[B26] Emilios Cambouropoulos. "Voice and stream: Perceptual and computational modeling of voice separation." In: *Music Perception* 26.1 (2008), pp. 75–94.

[B27] Carlos Cancino-Chacón, Silvan Peter, Patricia Hu, Emmanouil Karystinaios, Florian Henkel, Francesco Foscarin, Nimrod Varga, and Gerhard Widmer. "The ACCompanion: Combining Reactivity, Robustness, and Musical Expressivity in an Automatic Piano Accompanist." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[B28] Carlos Cancino-Chacón, Silvan Peter, Emmanouil Karystinaios, and Gerhard Widmer. "Towards Quantifying Differences in Expressive Piano Performances: Are Euclidean-like Distance Measures Enough?" In: *the 18th Rhythm Production and Perception Workshop (RPPW 2021)*. 2021.

[B29] Carlos Cancino-Chacón, Silvan David Peter, Emmanouil Karystinaios, Francesco Foscarin, Maarten Grachten, and Gerhard Widmer. "Partitura: A Python Package for Symbolic Music Processing." In: *Proceedings of the Music Encoding Conference (MEC)*. 2022.

[B30] Carlos E. Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. "Computational Models of Expressive Music Performance: A Comprehensive and Critical Review." In: *Frontiers in Digital Humanities* 5.October (2018), pp. 1–23. ISSN: 2297-2668. DOI: 10.3389/fdigh.2018.00025.

[B31] Gregory Castnnón and Lucas Finn. "Multi-target tracklet stitching through network flows." In: *Proceedings of the Aerospace Conference*. IEEE. 2011.

[B32] Carlos Eduardo Cancino Chacón. "Computational Modeling of Expressive Music Performance with Linear and Non-linear Basis Function Models." PhD thesis. Johannes Kepler University, Austria, 2018.

[B33] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[B34] Jianfei Chen, Jun Zhu, and Le Song. "Stochastic training of graph convolutional networks with variance reduction." In: *International Conference on Machine Learning (PMLR)*. 2018.

[B35] Jie Chen, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[B36] Tsung-Ping Chen, Li Su, et al. "Functional Harmony Recognition of Symbolic Music Data with Multi-task Recurrent Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[B37] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018.

[B38] Elaine Chew and Xiaodan Wu. "Separating voices in polyphonic music: A contig mapping approach." In: *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*. Springer. 2004.

[B39] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks." In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 257–266.

[B40] Chee-Yee Chong, Greg Castanon, Nathan Cooprider, Shozo Mori, Ravi Ravichandran, and Robert Macior. "Efficient multiple hypothesis tracking by track segment graph." In: *Proceedings of the International Conference on Information Fusion*. IEEE. 2009.

[B41] Darrell Conklin. "Multiple Viewpoint Systems for Music Classification." In: *Journal of New Music Research* 42.1 (2013), pp. 19–26. DOI: 10.1080/09298215.2013.776611. eprint: https://doi.org/10.1080/09298215.2013.776611. URL: https://doi.org/10.1080/09298215.2013.776611.

[B42] Darrell Conklin. "Chord sequence generation with semiotic patterns." In: *Journal of Mathematics and Music* 10.2 (2016), pp. 92–106. DOI: 10.1080/17459737.2016.1188172. eprint: https://doi.org/10.1080/17459737.2016.1188172. URL: https://doi.org/10.1080/17459737.2016.1188172.

[B43] Darrell Conklin and Ian H. Witten. "Multiple Viewpoint Systems for Music Prediction." In: *Journal of New Music Research* 24.1 (1995), pp. 51–73. ISSN: 17445027. DOI: 10.1080/09298219508570672.

[B44] Darrell Conklin and Ian H. Witten. "Multiple viewpoint systems for music prediction." In: *Journal of New Music Research* 24.1 (1995), pp. 51–73. DOI: 10.1080/09298219508570672. eprint: https://doi.org/10.1080/09298219508570672. URL: https://doi.org/10.1080/09298219508570672.

[B45] Emanuele Cosenza, Andrea Valenti, and Davide Bacciu. "Graph-based Polyphonic Multitrack Music Generation." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[B46] David F Crouse. "On implementing 2D rectangular assignment algorithms." In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696.

[B47] Michael Scott Cuthbert and Christopher Ariza. "music21: A toolkit for computer-aided musicology and symbolic music data." In: (2010).

[B48] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." In: *Advances in Neural Information Processing Systems (NeurIPS* 29 (2016).

[B49] Eustasio Del Barrio, Juan A Cuesta-Albertos, and Carlos Matrán. "An optimal transportation approach for assessing almost stochastic order." In: *The Mathematics of the Uncertain*. Springer, 2018, pp. 33–44.

[B50] Diana Deutsch. *Psychology of music*. Elsevier, 2013.

[B51] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. "Theme and Variation Encodings with Roman Numerals (TAVERN): A new data set for symbolic music analysis." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2015.

[B52] Bruno Di Giorgi, Matthias Mauch, and Mark Levy. "Downbeat Tracking with Tempo-Invariant Convolutional Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[B53] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. "Towards Graph Pooling by Edge Contraction." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[B54] Hao Wen Dong, Wen Yi Hsiao, Li Chia Yang, and Yi Hsuan Yang. "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2018. ISBN: 9781577358008. DOI: 10.1609/aaai.v32i1.11312. arXiv: 1709.06298. URL: https://salu133445.github.io/musegan/.

[B55] Ben Duane and Bryan Pardo. "Streaming from MIDI using constraint satisfaction optimization and sequence alignment." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2009.

[B56] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. "Benchmarking graph neural networks." In: *arXiv preprint arXiv:2003.00982* (2020).

[B57] Anders Elowsson and Anders Friberg. "Modeling Music Modality with a Key-Class Invariant Pitch Chroma CNN." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[B58] Lukas Faber, Amin K. Moghaddam, and Roger Wattenhofer. "When comparing to ground truth is wrong: On evaluating GNN explanation methods." In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 332–341.

[B59] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric." In: *Workshop on Representation Learning on Graphs and Manifolds at International Conference on Learning Representations (ICLR)*. 2019.

[B60] Christoph Finkensiep and Martin Alois Rohrmeier. "Modeling and inferring proto-voice structure in free polyphony." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 189–196.

[B61] Arthur Flexer. "A closer look on artist filters for musical genre classification." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2007. ISBN: 9783854032182.

[B62] Francesco Foscarin. "The Musical Score: a challenging goal for automatic music transcription." PhD thesis. Paris, CNAM, 2020.

[B63] Francesco Foscarin, Nicolas Audebert, and Raphaël Fournier-S'Niehotta. "PKSpell: Data-driven pitch spelling and key signature estimation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[B64] Francesco Foscarin, Katharina Hoedt, Verena Praher, Arthur Flexer, and Gerhard Widmer. "Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022. URL: https://api.semanticscholar.org/CorpusID:251881711.

[B65] Francesco Foscarin, Florent Jacquemard, and Philippe Rigaux. "Modeling and learning rhythm structure." In: *Sound and Music Computing Conference (SMC)*. 2019.

[B66] Francesco Foscarin, Emmanouil Karystinaios, Eita Nakamura, and Gerhard Widmer. "Cluster and Separate: a GNN Approach to Voice and Staff Prediction for Score Engraving." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2024.

[B67] Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter, Carlos Cancino-Chacón, Maarten Grachten, and Gerhard Widmer. "The match file format: Encoding Alignments between Scores and Performances." In: *Proceedings of the Music Encoding Conference (MEC)*. Halifax, Canada, 2022.

[B68] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "MidiTok: A Python package for MIDI file tokenization." In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[B69] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "Miditok: a Python Package for Midi File Tokenization." In: *International Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD)*. 2021. ISBN: 9783319701622.

[B70] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. "Byte Pair Encoding for Symbolic Music." In: 2023. arXiv: 2301.11975. URL: http://arxiv.org/abs/2301.11975.

[B71] Nathan Fradet, Nicolas Gutowski, Fabien Chhel, and Jean-Pierre Briot. "Byte Pair Encoding for Symbolic Music." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2001–2020. DOI: 10.18653/v1/2023.emnlp-main.123. URL: https://aclanthology.org/2023.emnlp-main.123.

[B72] Youssef Ghatas, Magda Fayek, and Mayada Hadhoud. "A hybrid deep learning approach for musical difficulty estimation of piano symbolic music." In: *Alexandria Engineering Journal* 61.12 (2022), pp. 10183–10196. ISSN: 11100168. DOI: 10.1016/j.aej.2022.03.060.

[B73] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1263–1272.

[B74] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational Fugue Analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96. DOI: 10.1162/COMJ_a_00300. URL: https://hal.science/hal-01113520.

[B75] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational fugue analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96.

[B76] Werner Goebl. "Melody lead in piano performance: Expressive device or artifact?" In: *The Journal of the Acoustical Society of America* 110 (2001), p. 641. DOI: 10.1121/1.1376133. URL: https://asa.scitation.org/doi/10.1121/1.1376133.

[B77] M Good. "An internet-friendly format for sheet music." In: *Proceedings of XML Conference*. 2001.

[B78] Mark Gotham, Kyle Gulling, and Chelsey Hamm. *Open Music Theory-Version 2.*, 2022.

[B79] Mark Gotham, Dmitri Tymoczko, and Michael Scott Cuthbert. "The RomanText Format: A Flexible and Standard Method for Representing Roman Numeral Analyses." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[B80] Mark Robert Haigh Gotham and Peter Jonas. "The Openscore Lieder Corpus." In: *Proceedings of the Music Encoding Conference (MEC)*. 2021.

[B81] Elaine Gould. *Behind bars: the definitive guide to music notation*. Faber Music Ltd, 2016.

[B82] Patrick Gray and Razvan C Bunescu. "A Neural Greedy Model for Voice Separation in Symbolic Music." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[B83] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. "Dynamic Task Prioritization for Multitask Learning." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[B84] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2017.

[B85] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." In: *IEEE Data Engineering Bulletin* 40.3 (2017), pp. 52–74.

[B86] Mei Han, Wei Xu, Hai Tao, and Yihong Gong. "An algorithm for multiple object trajectory tracking." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE. 2004.

[B87] Mitch Harris, Alan Smaill, and Geraint Wiggins. "Representing Music Symbolically." In: *IX Colloquio di Informatica Musicale (Venice)*. 1991. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.473.

[B88] Christopher Harte. "Towards automatic extraction of harmony information from music signals." PhD thesis. Queen Mary University of London, 2010.

[B89] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[B90] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. "audioLIME: Listenable Explanations Using Source Separation." In: *Proceedings of the International Workshop on Machine Learning and Music, MML*. 2020, pp. 20–24.

[B91] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. "Onsets and frames: Dual-objective piano transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018, pp. 50–57. ISBN: 9782954035123. DOI: 10.5281/zenodo.1492341. arXiv: 1710.11153.

[B92] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling factorized piano music modeling and generation with the MAESTRO dataset." In: *Proceedings of 7th International Conference on Learning Representations*. 2019.

[B93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[B94] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. "The Annotated Mozart Sonatas: Score, Harmony, and Cadence." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4.ARTICLE (2021), pp. 67–80.

[B95] Johannes Hentschel, Yannis Rammos, Fabian C Moss, Markus Neuwirth, and Martin Rohrmeier. "An annotated corpus of tonal piano music from the long 19th century." In: *Empirical Musicology Review* 18.1 (2023), pp. 84–95.

[B96] Carlos Hernandez-Olivan, Sonia Rubio Llamas, and Jose R. Beltran. *Symbolic Music Structure Analysis with Graph Representations and Changepoint Detection Methods*. 2023. arXiv: 2303.13881.

[B97] Sam van Herwaarden, Maarten Grachten, W de Haas, and W. Bas de Haas. "Predicting expressive dynamics in piano performances using neural networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[B98] Yuki Hiramatsu, Eita Nakamura, and Kazuyoshi Yoshii. "Joint Estimation of Note Values and Voices for Audio-to-Score Piano Transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 278–284.

[B99] Katharina Hoedt, Verena Praher, Arthur Flexer, and Gerhard Widmer. "Constructing adversarial examples to investigate the plausibility of explanations in deep audio and image classifiers." In: *Neural Computing and Applications* 35.14 (2023), pp. 10011–10029.

[B100] Holger H Hoosy, Keith A Hamelz, Kai Renzy, and J urgen Kiliany. "The GUIDO Notation Format A Novel Approach for Adequately Representing Score-Level Music." In: *International Computer Music Conference (ICMC)*. 1998.

[B101] Yo-Wei Hsiao and Li Su. "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 285–292.

[B102] Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2021. arXiv: 2101.02402v1.

[B103] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. "Open graph benchmark: Datasets for machine learning on graphs." In: *Advances in Neural Information Processing Systems (NeurIPS* 33 (2020), pp. 22118–22133.

[B104] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for pre-training graph neural networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[B105] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous graph transformer." In: *Proceedings of the web conference 2020*. 2020, pp. 2704–2710.

[B106] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. "The Bach doodle: Approachable music composition with machine learning at scale." In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. 2019.

[B107] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. "Music Transformer." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019. arXiv: 1809.04281. URL: http://arxiv.org/abs/1809.04281.

[B108] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. "Graphlime: Local interpretable model explanations for graph neural networks." In: *IEEE Transactions on Knowledge and Data Engineering* 35.7 (2022), pp. 6968–6972.

[B109] Yu-Siang Huang and Yi-Hsuan Yang. "Pop Music Transformer: Beat-Based Modeling and Generation of Expressive Pop Piano Compositions." In: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 1180–1188. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. URL: https://doi.org/10.1145/3394171.3413671.

[B110] Yu Siang Huang and Yi Hsuan Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions." In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. arXiv: 2002.00212.

[B111] David Huron. "Tone and voice: A derivation of the rules of voice-leading from perceptual principles." In: *Music Perception* 19.1 (2001), pp. 1–64.

[B112] David Huron and Craig Sapp. *The Humdrum Toolkit*. Acessed August 29, 2024. 1993. URL: https://www.humdrum.org/.

[B113] Plácido R Illescas, David Rizo, and José Manuel Inesta Quereda. "Harmonic, melodic, and functional automatic analysis." In: *Proceedings of the International Computer Music Conference*. 2007.

[B114] Tobias Isenberg, André Miede, and Sheelagh Carpendale. "A Buffer Framework for Supporting Responsive Interaction in Information Visualization Interfaces." In: *Proceedings of the Fourth International Conference on Creating, Connecting, and Collaborating through Computing (C^5 2006)*. IEEE, 2006, pp. 262–269. ISBN: 978-0-7695-2563-1.

[B115] Adrián Javaloy and Isabel Valera. "RotoGrad: Gradient Homogenization in Multitask Learning." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022.

[B116] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam. "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance." In: *Proceedings of the 20th International Society of Music Information Retrieval Conference*. 2019.

[B117] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.

[B118] Anna Jordanous. "Voice separation in polyphonic music: A data-driven approach." In: *Proceedings of the International Computer Music Conference (ICMC)*. 2008.

[B119] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. "A comprehensive survey on deep graph representation learning." In: *Neural Networks* (2024), p. 106207.

[B120] Kamil Kamiński, Jan Ludwiczak, Maciej Jasiński, Adriana Bukala, Rafal Madaj, Krzysztof Szczepaniak, and Stanisław Dunin-Horkawicz. "Rossmann-toolbox: a deep learning-based protocol for the prediction and design of cofactor specificity in Rossmann fold proteins." In: *Briefings in Bioinformatics* 23.1 (2022).

[B121] Emmanouil Karystinaios, Francesco Foscarin, Florent Jacquemard, Masahiko Sakai, Satoshi Tojo, and Gerhard Widmer. "8+ 8= 4: Formalizing Time Units to Handle Symbolic Music Durations." In: *proceedings of the 16th International Symposium on Computer Music Multidisciplinary Research*. 2023.

[B122] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Musical Voice Separation as Link Prediction: Modeling a Musical Perception Task as a Multi-Trajectory Tracking Problem." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[B123] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "Perception-Inspired Graph Convolution for Music Understanding Tasks." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2024.

[B124] Emmanouil Karystinaios, Francesco Foscarin, and Gerhard Widmer. "SMUG-Explain: A Framework for Symbolic Music Graph Explanations." In: *Proceedings of the Sound and Music Computing Conference (SMC)*. 2024.

[B125] Emmanouil Karystinaios, Corentin Guichaoua, Moreno Andreatta, Louis Bigo, and Isabelle Bloch. "Music Genre Descriptor for Classification Based on Tonnetz Trajectories." In: *Proceedings of Journées Informatiques Musicales*. 2021.

[B126] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B127] Emmanouil Karystinaios and Gerhard Widmer. "Cadence Detection in Symbolic Classical Music using Graph Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B128] Emmanouil Karystinaios and Gerhard Widmer. "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[B129] Emmanouil Karystinaios and Gerhard Widmer. "GraphMuse: A Library for Symbolic Music Graph Processing." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2024.

[B130] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. "Representation learning for dynamic graphs: A survey." In: *Journal of Machine Learning Research* 21.70 (2020), pp. 1–73.

[B131] Mikaela Keller, Gabriel Loiseau, and Louis Bigo. "What Musical Knowledge Does Self-Attention Learn?" In: *Proceedings of the 2nd Workshop on NLP for Music and Spoken Audio (NLP4MusA)*. 2021, pp. 6–10. URL: https://aclanthology.org/2021.nlp4musa-1.2.

[B132] Mathieu Kermarec, Louis Bigo, and Mikaela Keller. "Improving Tokenization Expressiveness With Pitch Intervals." In: *Late-Breaking Demo Session of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B133] Jürgen Kilian and Holger H Hoos. "Voice Separation-A Local Optimization Approach." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2002.

[B134] Sunghyeon Kim, Hyeyoon Lee, Sunjong Park, Jinho Lee, and Keunwoo Choi. "Deep Composer Classification Using Symbolic Representation." In: *International Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD)*. 2020. arXiv: 2010.00823. URL: http://arxiv.org/abs/2010.00823.

[B135] Thomas N Kipf and Max Welling. "Variational graph auto-encoders." In: *NIPS Workshop on Bayesian Deep Learning* (2016).

[B136] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[B137] Phillip B Kirlin and Paul E Utgoff. "VOISE: Learning to Segregate Voices in Explicit and Implicit Polyphony." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2005.

[B138] Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription*. Springer, 2006, p. 456.

[B139] Qiuqiang Kong, Keunwoo Choi, and Yuxuan Wang. "Large-Scale MIDI-Based Composer Classification." In: *arXiv*. 2020. arXiv: arXiv:2010.14805v1.

[B140] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution piano transcription with pedals by regressing onset and offset times." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3707–3717.

[B141] Filip Korzeniowski, Sergio Oramas, and Fabien Gouyon. "Artist Similarity with Graph Neural Networks." In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. 2021.

[B142] Peter van Kranenburg and Folgert Karsdorp. "Cadence Detection in Western Traditional Stanzaic Songs using Melodic and Textual Features." In: *Proceedings of the 15th International Society of Music Information Retrieval Conference*. 2014.

[B143] Ulrich Lampe, Markus Kieselmann, André Miede, Sebastian Zöller, and Ralf Steinmetz. "A Tale of Millis and Nanos: On the Accuracy of Time Measurements in Virtual Machines." In: *Proceedings of the Second European Conference on Service-Oriented and Cloud Computing (ESOCC 2013)*. Springer, 2013, pp. 172–179. ISBN: 978-3-642-40650-8.

[B144] Ulrich Lampe, Qiong Wu, Ronny Hans, André Miede, and Ralf Steinmetz. "To Frag Or To Be Fragged – An Empirical Assessment of Latency in Cloud Gaming." In: *Proceedings of the Third International Conference on Cloud Computing and Services Science (CLOSER 2013)*. 2013, pp. 5–12. ISBN: 978-898-8565-52-5.

[B145] Stefan Lattner, Monika Dörfler, and Andreas Arzt. "Learning Complex Basis Functions for Invariant Representations of Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[B146] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. "Learning Transposition-Invariant Interval Features from Symbolic Music and Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018.

[B147] Fred Lerdahl and Ray S Jackendoff. *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.

[B148] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. "DeepGCNs: Can GCNs go as deep as CNNs?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00936. arXiv: 1904.03751. URL: https://sites.google.com/view/deep-gcns.

[B149] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. "Deepergcn: All you need to train deeper gcns." In: *arXiv preprint arXiv:2006.07739* (2020).

[B150] Lukas Liebel and Marco Körner. "Auxiliary tasks in multi-task learning." In: *arXiv preprint arXiv:1805.06334* (2018).

[B151] He Liu, Tao Wang, Congyan Lang, Songhe Feng, Yi Jin, and Yidong Li. "GLAN: A Graph-based Linear Assignment Network." In: *arXiv preprint arXiv:2201.02057* (2022).

[B152] Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. "Symphony Generation with Permutation Invariant Language Model." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B153] Lele Liu, Qiuqiang Kong, GV Morfi, Emmanouil Benetos, et al. "Performance MIDI-to-score conversion by neural beat tracking." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B154] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. "Sampling methods for efficient training of graph convolutional networks: A survey." In: *IEEE/CAA Journal of Automatica Sinica* 9.2 (2021), pp. 205–234.

[B155] Xichu Ma, Xiao Liu, Bowen Zhang, and Ye Wang. "Robust Melody Track Identification in Symbolic Music." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B156] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. "Path Integral Based Convolution and Pooling for Graph Neural Networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 33 (2020), pp. 16421–16433.

[B157] Søren Tjagvad Madsen and Gerhard Widmer. "Separating voices in MIDI." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer. 2006.

[B158] José Pedro Magalhaes and W Bas de Haas. "Functional Modelling of Musical Harmony: an experience report." In: *ACM SIGPLAN Notices* 46.9 (2011), pp. 156–162.

[B159] Dimos Makris, Ioannis Karydis, and Emilios Cambouropoulos. "VISA3: Refining the voice integration/segregation algorithm." In: *Proceedings of the Sound and Music Computing Conference*. 2016.

[B160] Sarah Marlowe. "Schenkerian Analysis of Fugue: A Practical Demonstration." In: *Journal of Music Theory Pedagogy* 33.1 (2019), p. 6.

[B161] Andrew McLeod and Mark Steedman. "HMM-based voice separation of MIDI performance." In: *Journal of New Music Research* 45.1 (2016), pp. 17–26.

[B162] Andrew McLeod and Mark Steedman. "Evaluating automatic polyphonic music transcription." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2018, pp. 42–49.

[B163] Andrew Philip McLeod and Martin Alois Rohrmeier. "A modular system for the harmonic analysis of musical scores using a large vocabulary." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[B164] Alessandro B. Melchiorre, Verena Haunschmid, Markus Schedl, and Gerhard Widmer. "LEMONS: Listenable Explanations for Music recOmmeNder Systems." In: *Advances in Information Retrieval: Proceedings of the European Conference on IR Research, ECIR*. Vol. 12657. Springer, 2021, pp. 531–536.

[B165] Gianluca Micchi. "A neural network for composer classification." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) Late-Breading Demo (LBD)*. 2018.

[B166] Gianluca Micchi, Mark Gotham, and Mathieu Giraud. "Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* 3.1 (2020), pp. 42–54.

[B167] Gianluca Micchi, Katerina Kosta, Gabriele Medeot, and Pierre Chanquion. "A deep learning method for enforcing coherence in Automatic Chord Recognition." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[B168] André Miede. "Theses and other Beautiful Documents with `classicthesis`." In: *TUGboat – The Communications of the TEX Users Group* 31.1 (2010), pp. 18–20. ISSN: 0896-3207.

[B169] André Miede, Gökhan Şimşek, Stefan Schulte, Daniel F. Abawi, Julian Eckert, and Ralf Steinmetz. "Revealing Business Relationships – Eavesdropping Cross-organizational Collaboration in the Internet of Services." In: *Proceedings of the Tenth International Conference Wirtschaftsinformatik (WI 2011)*. Vol. 2. 2011, pp. 1083–1092. ISBN: 978-1-4467-9236-0.

[B170] Saumitra Mishra, Emmanouil Benetos, Bob L. Sturm, and Simon Dixon. "Reliable Local Explanations for Machine Listening." In: *Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN*. IEEE, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207444.

[B171] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. "Local Interpretable Model-agnostic Explanations for Music Content Analysis." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 537–543.

[B172] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[B173] Nicola J Müller, Pablo Sánchez, Jörg Hoffmann, Verena Wolf, and Timo P Gros. "Comparing State-of-the-art Graph Neural Networks and Transformers for General Policy Learning." In: (2024).

[B174] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. "Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music." In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 276–280.

[B175] Néstor Nápoles López. "Automatic Harmonic Analysis of Classical String Quartets from Symbolic Score." PhD thesis. Master's thesis, Universitat Pompeu Fabra, 2017.

[B176] Néstor Nápoles López. "Automatic Roman Numeral Analysis in Symbolic Music Representations." PhD thesis. Schulich School of Music McGill University, 2022.

[B177] Néstor Nápoles López and Ichiro Fujinaga. "Harmonic Reductions as a Strategy for Creative Data Augmentation." In: *Late-Breaking Demo at International Society for Music Information Retrieval Conference (ISMIR)*. 2020.

[B178] Néstor Nápoles López, Mark Gotham, and Ichiro Fujinaga. "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2021.

[B179] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. "Multi-task Learning as a Bargaining Game." In: *Proceedings of the International Conference on Machine Learning (ICML)* (2022).

[B180] Markus Neuwirth, Daniel Harasim, Fabian C Moss, and Martin Rohrmeier. "The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets." In: *Frontiers in Digital Humanities* 5 (2018), p. 16.

[B181] Han-Wen Nienhuys and Jan Nieuwenhuizen. "LilyPond, a system for automated music engraving." In: *Proceedings of the xiv colloquium on musical informatics (xiv cim 2003)*. Vol. 1. Citeseer. 2003, pp. 167–171.

[B182] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. "This Time with Feeling: Learning Expressive Musical Performance." In: *Neural Computing and Applications* 32 (2018), pp. 955–967. URL: https://link.springer.com/article/10.1007/s00521-018-3758-9.

[B183] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. "This time with feeling: learning expressive musical performance." In: *Neural Computing and Applications* 32.4 (2018), pp. 955–967. ISSN: 14333058. DOI: 10.1007/s00521-018-3758-9. arXiv: 1808.03715.

[B184] Irwin Oppenheim, Chris Walshaw, John Atchley, and Guido Gonzato. *he abc standard 2.0.* Acessed August 29, 2024. 2010. URL: https://abcnotation.com/wiki/abc:standard:v2.0.

[B185] Johan Pauwels, Ken O'Hanlon, Emilia Gómez, Mark Sandler, et al. "20 years of Automatic Chord Recognition from Audio." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2019.

[B186] Marcus Pearce. "The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition." PhD thesis. UK: City University of London, 2005.

[B187] Marcus T. Pearce. "Statistical learning and probabilistic prediction in music cognition: Mechanisms of stylistic enculturation." In: *Annals of the New York Academy of Sciences* 1423.1 (2018), pp. 378–395.

[B188] Marcus Thomas Pearce. "The construction and evaluation of statistical models of melodic structure in music perception and composition." PhD thesis. City University London, 2005.

[B189] Silvan David Peter, Carlos Eduardo Cancino-Chacón, Francesco Foscarin, Andrew Philip McLeod, Florian Henkel, Emmanouil Karystinaios, and Gerhard Widmer. "Automatic Note-Level Score-to-Performance Alignments in the ASAP Dataset." In: *Transactions of the International Society for Music Information Retrieval (TISMIR)* (2023).

[B190] Silvan David Peter, Carlos Eduardo Cancino-Chacón, Emmanouil Karystinaios, and Gerhard Widmer. "Sounding Out Reconstruction Error-Based Evaluation of Generative Models of Expressive Performance." In: *Proceedings of the 10th International Conference on Digital Libraries for Musicology*. 2023, pp. 58–66.

[B191] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10772–10781.

[B192] Alexandre Popoff, Moreno Andreatta, and Andrée Ehresmann. "Relational poly-Klumpenhouwer networks for transformational and voice-leading analysis." In: *Journal of Mathematics and Music* 12.1 (2018).

[B193] Mathieu Prang. "Representation learning for symbolic music." PhD thesis. IRCAM, 2021. URL: https://hal.archives-ouvertes.fr/tel-03329980.

[B194] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. "Verovio: A library for Engraving MEI Music Notation into SVG." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[B195] Dimitris Rafailidis, Emilios Cambouropoulos, and Yannis Manolopoulos. "Musical voice integration/segregation: VISA revisited." In: *Proceedings of the Sound and Music Computing Conference (SMC)*. 2009.

[B196] Colin Raffel and Daniel PW Ellis. "Intuitive Analysis, Creation, and Manipulation of MIDI data WITH pretty_midi." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[B197] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. "Recipe for a general, powerful, scalable graph transformer." In: *Advances in Neural Information Processing Systems (NeurIPS* 35 (2022), pp. 14501–14515.

[B198] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. "Asap: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2020.

[B199] Christopher Raphael and Joshua Stoddard. "Functional Harmonic Analysis Using Probabilistic Models." In: *Computer Music Journal* 28.3 (2004), pp. 45–52.

[B200] Martin Rohrmeier and Fabian C Moss. "A formal model of extended tonal harmony." In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. 2021.

[B201] Perry Roland. "The music encoding initiative (MEI)." In: *Proceedings of the First International Conference on Musical Applications Using XML*. Vol. 1060. Citeseer. 2002, pp. 55–59.

[B202] Eran Rosenbluth, Jan Toenshoff, and Martin Grohe. "Some might say all you need is sum." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023.

[B203] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. "Temporal graph networks for deep learning on dynamic graphs." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2020.

[B204] Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023.

[B205] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. "E(n) Equivariant Graph Neural Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021.

[B206] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer. 2018, pp. 593–607.

[B207] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. "Interpreting graph neural networks for NLP with differentiable edge masking." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021.

[B208] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. "Modeling Relational Data with Graph Convolutional Networks." In: *Proceedings of the Semantic Web International Conference, ESWC*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607.

[B209] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. "Higher-order explanations of graph neural networks via relevant walks." In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7581–7596.

[B210] Lisa Schneckenreiter, Richard Freinschlag, Florian Sestak, Johannes Brandstetter, Günter Klambauer, and Andreas Mayr. "GNN-VPA: A Variance-Preserving Aggregation Strategy for Graph Neural Networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2024.

[B211] Michiel Schuijer. *Analyzing atonal music: Pitch-class set theory and its contexts*. University Rochester Press, 2008.

[B212] David RW Sears, Andreas Arzt, Harald Frostel, Reinhard Sonnleitner, and Gerhard Widmer. "Modeling harmony with skip-grams." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2017.

[B213] David RW Sears, Marcus T Pearce, William E Caplin, and Stephen McAdams. "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models." In: *Journal of New Music Research* 47.1 (2018), pp. 29–52.

[B214] David RW Sears and Gerhard Widmer. "Beneath (or beyond) the surface: Discovering voice-leading patterns with skip-grams." In: *Journal of Mathematics and Music* 15.3 (2021).

[B215] Eleanor Selfridge-Field. "Musedata: multipurpose representation." In: *Beyond MIDI: The handbook of musical codes*. Center for Computer Assisted Research in the Humanities. The MIT Press, 1997.

[B216] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. "A survey of heterogeneous information network analysis." In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 17–37.

[B217] Kentaro Shibata, Eita Nakamura, and Kazuyoshi Yoshii. "Non-local musical statistics as guides for audio-to-score piano transcription." In: *Information Sciences* 566 (2021), pp. 262–280.

[B218] Christopher Shooner, Srimant P Tripathy, Harold E Bedell, and Haluk Öğmen. "High-capacity, transient retention of direction-of-motion information for multiple moving objects." In: *Journal of Vision* 10.6 (2010), pp. 1–20.

[B219] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[B220] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." In: *Workshop at International Conference on Learning Representations (ICLR)*. 2013.

[B221] D. Sleator. *Introduction to the Melisma System*. URL: https://www.link.cs.cmu.edu/melisma/intro.html.

[B222] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net." In: *Workshop at International Conference on Learning Representations (ICLR)*. 2015.

[B223] Erik Strumbelj and Igor Kononenko. "An efficient explanation of individual classifications using game theory." In: *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[B224] Yizhou Sun and Jiawei Han. "Mining heterogeneous information networks: a structural analysis approach." In: *ACM SIGKDD explorations newsletter* 14.2 (2013), pp. 20–28.

[B225] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328.

[B226] Masahiro Suzuki. "Score Transformer: Generating Musical Score from Note-level Representation." In: *Proceedings of the 3rd ACM International Conference on Multimedia in Asia*. 2021, pp. 1–7.

[B227] Jeffrey Swinkin. "Schenkerian analysis, metaphor, and performance." In: *College Music Symposium*. Vol. 47. JSTOR. 2007, pp. 76–99.

[B228] Daniel Taupin, Ross Mitchell, and Andreas Egler. "MusiXTEX. Using TEX to write polyphonic or instrumental music." In: *TUGboat* 14.3 (1993), pp. 212–220.

[B229] David Temperley. *The cognition of basic musical structures*. MIT press, 2004.

[B230] David Temperley. "A probabilistic model of melody perception." In: *Cognitive Science* 32.2 (2008), pp. 418–444.

[B231] David Temperley. "A Unified Probabilistic Model for Polyphonic Music Analysis." In: *Journal of New Music Research* 38.1 (2009), pp. 3–18. DOI: 10.1080/09298210902928495. eprint: https://doi.org/10.1080/09298210902928495. URL: https://doi.org/10.1080/09298210902928495.

[B232] David Temperley. *The Cognition of Basic Musical Structures*. MIT Press, 2004.

[B233] Moyu Terao, Eita Nakamura, and Kazuyoshi Yoshii. "Neural Band-to-Piano Score Arrangement with Stepless Difficulty Control." In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[B234] The Pianola Institute. *History of the Pianola – Piano Players*. Accessed: 2024-06-25. n.d. URL: http://www.pianola.org.

[B235] Srimant Tripathy and Christina J Howard. "Multiple trajectory tracking." In: *Scholarpedia* 7.4 (2012).

[B236] Hsin-Yi Tsai, Melanie Siebenhaar, André Miede, Yu-Lun Huang, and Ralf Steinmetz. "Threat as a Service? Virtualization's Impact on Cloud Security." In: *IEEE IT Professional* 14.1 (2012), pp. 32–37. ISSN: 1520-9202.

[B237] Reinier de Valk, Tillman Weyde, Emmanouil Benetos, et al. "A machine learning approach to voice separation in lute tablature." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2013.

[B238] Lodewyk Van der Merwe and Pieter De Villiers. "Comparative investigation into Viterbi based and multiple hypothesis based track stitching." In: *IET Radar, Sonar & Navigation* 10.9 (2016), pp. 1575–1582.

[B239] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017. arXiv: 1706.03762.

[B240] Gissel Velarde, Tillman Weyde, Carlos E. Cancino-Chacón, David Meredith, and Maarten Grachten. "Composer recognition based on 2D-filtered piano-rolls." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2016.

[B241] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[B242] Petar Veličković. "Everything is Connected: Graph Neural Networks." In: *Artificial Intelligence (AI) Methodology in Structural Biology* (2023). ISSN: 1879033X. DOI: 10.1016/j.sbi.2023.102538. arXiv: 2301.08210. URL: http://arxiv.org/abs/2301.08210.

[B243] Minh Vu and My T Thai. "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 33 (2020), pp. 12225–12235.

[B244] Christian Walder. "Modelling symbolic music: Beyond the piano roll." In: *Journal of Machine Learning Research*. Vol. 63. 2016, pp. 174–189. arXiv: 1606.01368.

[B245] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. "Heterogeneous graph attention network." In: *The world wide web conference*. 2019, pp. 2022–2032.

[B246] Yongxin Wang, Kris Kitani, and Xinshuo Weng. "Joint object detection and multi-object tracking with graph neural networks." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[B247] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds." In: *ACM Transactions on Graphics* 38.5 (2019), 146:1–146:12.

[B248] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. "Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[B249] Xinshuo Weng, Ye Yuan, and Kris Kitani. "Ptp: Parallelized tracking and prediction with graph neural networks and diversity sampling." In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4640–4647.

[B250] Raymond P. Whorley and Darrell Conklin. "Music Generation from Statistical Models of Harmony." In: *Journal of New Music Research* 45.2 (2016), pp. 160–183. DOI: 10.1080/09298215.2016.1173708. eprint: https://doi.org/10.1080/09298215.2016.1173708. URL: https://doi.org/10.1080/09298215.2016.1173708.

[B251] Geraint Wiggins, Eduardo Miranda, Alan Smaill, and Mitch Harris. "A Framework for the Evaluation of Music Representation Systems." In: *Computer Music Journal* 17.3 (1993), pp. 31–42. URL: https://about.jstor.org/terms.

[B252] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022.

[B253] Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, Stan Li, et al. "GraphMixup: Improving Class-Imbalanced Node Classification on Graphs by Self-supervised Context Prediction." In: *arXiv preprint arXiv:2106.11133* (2021).

[B254] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[B255] Iannis Xenakis. *Formalized Music: Thoughts and Mathematics in Composition*. 1992. ISBN: 0-945193-01-7.

[B256] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[B257] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018.

[B258] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5453–5462.

[B259] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. "Do transformers really perform badly for graph representation?" In: *Advances in Neural Information Processing Systems (NeurIPS* 34 (2021), pp. 28877–28888.

[B260] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems." In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 974–983.

[B261] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS* 32 (2019).

[B262] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. "Hierarchical graph representation learning with differentiable pooling." In: *Advances in Neural Information Processing Systems (NeurIPS*. Vol. 31. 2018.

[B263] G. Yona and Daniel Greenfeld. "Revisiting Sanity Checks for Saliency Maps." In: *Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4)*. 2021.

[B264] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. "Graphrnn: Generating realistic graphs with deep auto-regressive models." In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5708–5717.

[B265] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2022), pp. 5782–5799.

[B266] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 818–833.

[B267] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. "GraphSAINT: Graph Sampling Based Inductive Learning Method." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020.

[B268] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie Yan Liu. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. 2021. ISBN: 9781954085541. DOI: 10.18653/v1/2021.findings-acl.70. arXiv: 2106.05630.

[B269] Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer, and Carlos Eduardo Cancino-Chacón. "Symbolic Music Representations for Classification Tasks: A Systematic Evaluation." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2023.

[B270] Huan Zhang, Jingjing Tang, Syed Rafee, Simon Dixon, and George Fazekas. "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[B271] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. "Graph convolutional networks: a comprehensive review." In: *Computational Social Networks* 6.1 (2019), pp. 1–23.

[B272] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Facial Landmark Detection by Deep Multi-task Learning." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

[B273] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. "Hierarchical graph pooling with structure learning." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*. 2019.

[B274] Jingwei Zhao, Gus Xia, and Ye Wang. "Q&A: Query-Based Representation Learning for Multi-Track Symbolic Music re-Arrangement." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[B275] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. "GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks." In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021.

[B276] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. "Layer-dependent importance sampling for training deep and large graph convolutional networks." In: *Advances in Neural Information Processing Systems (NeurIPS*. 2019.