

University of Warsaw

Faculty of Mathematics, Informatics and Mechanics

Mateusz Szymański

Student no. 457163

Deep Learning-Based Performance MIDI to Score Automatic Transcription

Master's thesis

in MACHINE LEARNING

Supervisors:

dr hab. Marek Cygan

Instytut Informatyki

Wydział Matematyki, Informatyki i Mechaniki

Uniwersytet Warszawski

dr inż. Mateusz Modrzejewski

Instytut Informatyki

Wydział Elektroniki i Technik Informacyjnych

Politechnika Warszawska

Warsaw, December 2024

Abstract

The primary objective of this thesis is to explore, evaluate, and extend recent developments in the automatic transcription of performance MIDI recordings into musical scores, a subtask within the broader domain of *Automatic Music Transcription* (AMT). AMT aims to extract symbolic representations of music from raw audio signals, providing a bridge between musical performance and formalized notation.

This work focuses on enhancing the methodology introduced in the state-of-the-art study, *Performance MIDI-to-Score Conversion by Neural Beat Tracking* [LKMB22]. This approach combines Convolutional Recurrent Neural Networks with dynamic programming approach to convert performance MIDI recordings into musical scores.

The thesis includes a detailed analysis of the model’s architecture, behavior, and limitations, incorporating insights from custom explainable artificial intelligence (XAI) techniques developed to analyze model decisions. These analyses identified model shortcomings, including undesired dependencies on certain features, and informed strategies to enhance robustness. Additionally, ablation studies were conducted to evaluate the impact of the robustness enhancement on the model’s performance.

To expand the capabilities of the baseline model, this work integrates an additional sub-model for predicting note dynamics. A novel extension to the MV2H score evaluation metric (*Multi-pitch detection, Voice separation, Metrical alignment, Note Value detection, and Harmonic Analysis*) is proposed, augmenting it with *Dynamic detection*, resulting in the MV2HD metric.

The study also compares the baseline architecture with two promising alternatives: *Transformers* and *Temporal Convolutional Networks*. While the base model demonstrated superior performance in most tasks, these architectures offer valuable insights into the trade-offs between accuracy, complexity, and computational efficiency.

This thesis contributes to the ongoing research in AMT by enhancing model robustness, incorporating dynamics transcription, and proposing new evaluation methodologies, laying the groundwork for further advancements in performance-to-score conversion.

Keywords

deep learning, music information retrieval, automatic music transcription, performance MIDI, score generation

Thesis domain (Socrates-Erasmus subject area codes)

11.4 Sztuczna inteligencja

Subject classification

I. Computing Methodologies

I.2. Artificial Intelligence

I.2.7. Natural Language Processing

Tytuł pracy w języku polskim

Automatyczna transkrypcja wykonan MIDI do partytur oparta na głębokim uczeniu

Contents

Introduction	15
1. Representation of Music Information	17
1.1. Audio Signals	18
1.1.1. Waveform	18
1.1.2. Spectrogram	19
1.2. Symbolic Representation	20
1.2.1. Sheet Music	20
1.2.2. Musical Instrument Digital Interface	22
2. Automatic Music Transcription	25
2.1. Music Information Retrieval	25
2.1.1. Examples of Problems in Music Information Retrieval	25
2.1.2. Sources of Music Information	26
2.1.3. Music Facets	26
2.2. Automatic Music Transcription	27
2.2.1. Application of Music Transcription	28
2.2.2. Levels of Music Transcription	28
2.2.2.1. Frame-level Transcription	28
2.2.2.2. Note-level Transcription	28
2.2.2.3. Stream-level Transcription	29
2.2.2.4. Notation-level Transcription	30
2.2.3. Stages of Music Transcriptions	30
2.2.4. Performance MIDI to Score Transcription	31
2.2.4.1. Performance MIDI Encoding	32
2.2.4.2. Music Score Encoding	33
2.3. Challenges in the Transcription Task	34
2.3.1. Common Transcription Problems	34
2.3.1.1. Tempo	35

2.3.1.2.	Note Duration	35
2.3.1.3.	Dynamics	35
2.3.1.4.	Harmonic Equivalence	36
2.3.1.5.	Voicing	36
2.3.2.	Score-Informed MIDI files	37
2.3.3.	Summary of Steps	38
2.3.4.	Limitations	38
3.	MIDI-to-Score Transcription Evaluation	41
3.1.	Manual Attribution	42
3.2.	Binary and Multiclass Classifier Metrics	43
3.2.1.	Cross-entropy Loss	43
3.2.2.	Accuracy, Precision and Recall	44
3.2.3.	F_1 score	44
3.3.	MUSTER: Music Score Transcription Error Rates	45
3.3.1.	Note Alignment	45
3.3.2.	Rhythm Correction Cost	46
3.3.3.	Metrics	47
3.3.4.	Limitations	47
3.4.	MV2H Metric	48
3.4.1.	Components of the MV2H Metric	48
3.4.1.1.	Multi-pitch Detection	48
3.4.1.2.	Voice Separation	49
3.4.1.3.	Metrical Alignment	49
3.4.1.4.	Note Value Detection	49
3.4.1.5.	Harmonic Analysis	50
3.4.2.	Disjoint Penalty Principle	50
3.4.3.	Limitations	51
3.5.	Score Similarity	51
3.5.1.	Note Alignment	51
3.5.2.	Comparison	52
3.5.3.	Limitations	52
4.	Overview of Existing Methods for MIDI to Score Conversion	55
4.1.	A Brief History	55
4.2.	Modern Formulation of the Problem	56
4.3.	Hidden Markov Model	57
4.3.1.	Hidden Markov Model for Beat Quantization	58

4.3.2.	Optimal Sequence of States	59
4.3.3.	Training	60
4.3.4.	Limitations	60
4.3.5.	Polyphonic Extension	60
4.4.	Dynamic Programming Approach	60
4.4.1.	Simultaneity Quantization	61
4.4.2.	Tatum Segmentation	61
4.4.3.	Note Onset Quantization	62
4.4.4.	Limitations	62
5.	Performance MIDI-to-Score Conversion by Neural Beat Tracking	65
5.1.	Beat Tracking	65
5.1.1.	In-note Beat Prediction	66
5.1.2.	Out-of-note Beat Prediction	66
5.2.	Input Data Encoding	68
5.3.	Score Elements Assignment	69
5.4.	Score Generation	69
5.5.	Training and Evaluation	70
5.5.1.	Datasets	70
5.5.1.1.	Classical Piano MIDI Database	71
5.5.1.2.	Augmented MIDI Aligned Piano Sounds	72
5.5.1.3.	Aligned Scores and Performances	72
5.5.2.	Data Augmentation	73
5.5.3.	Training and Evaluation	73
5.5.3.1.	Feature Preparation	73
5.5.3.2.	Training	74
5.5.3.3.	Evaluation	74
5.6.	Model Performance	74
5.6.1.	Validation Metrics	74
5.6.2.	MV2H Metric	75
5.7.	Case Studies	76
5.7.1.	Unrecognized Pickup Measures	76
5.7.2.	Hand Part Misalignments	77
5.7.3.	Meter Misalignment	77
5.8.	Robustness Analysis	78
5.8.1.	<i>Ceteris Paribus</i> Analysis	79
5.8.1.1.	Perturbations	79
5.8.1.2.	Results	80

5.8.1.3.	Hand Part Mitigation Strategies	80
5.9.	Local Feature Importance	81
5.9.1.	LIME-Based Analysis for Hand Part Assignment	82
5.9.1.1.	Observations and Limitations	83
5.9.2.	Key Signature Attribution via Note Omission	83
6.	Experiments and Improvements	87
6.1.	Ablation Studies	87
6.1.1.	Beat Tracking	88
6.1.2.	Hand Part Model	89
6.1.3.	Key Signature Model	89
6.1.4.	Time Signature Model	90
6.2.	Transformers	90
6.2.1.	Vanilla Transformer	90
6.2.2.	Results	92
6.2.3.	Beat Model	92
6.2.4.	Hand Part Model	93
6.2.5.	Key Signature Model	94
6.2.6.	Time Signature Model	94
6.2.7.	Future Work	95
6.3.	Temporal Convolutional Network	95
6.3.1.	Dilated Convolution	96
6.3.2.	Temporal Convolutional Network	96
6.3.3.	Model Architecture	97
6.3.4.	Results	97
6.3.4.1.	Beat Model	97
6.3.4.2.	Hand Part Model	98
6.3.4.3.	Key Signature Model	98
6.3.4.4.	Time Signature Model	99
6.4.	Dynamics	99
6.4.1.	Architecture	101
6.4.2.	Challenges	101
6.4.2.1.	Dynamics Data	101
6.4.2.2.	Matching Algorithm	102
6.4.3.	Subjectivity of Dynamics Markings	102
6.4.4.	Evaluation	103
6.4.4.1.	MV2HD	103
6.4.4.2.	Results	103

7. Conclusions	105
7.1. Robustness Analysis	105
7.2. Feature Importance Methods	106
7.3. Experiments	106
7.3.1. Transformers	106
7.3.2. Temporal Convolutional Networks	107
7.3.3. Dynamics	107

List of Figures

1.1.	An example of a sampling a sine wave with different sampling rates.	18
1.2.	An illustration of bit depth affecting the quality of an audio.	19
1.3.	A comparison of two different representations of the same sound.	20
1.4.	First bars of Frédéric Chopin’s <i>Prelude Opus 28, No. 4</i>	21
1.5.	A piano keyboard layout with 49 keys.	23
2.1.	A frame-level and note-level transcription of a vocal audio sample.	29
2.2.	An example of an Automatic Music Transcription task on a concrete piano recording.	32
2.3.	An example of the imported performance MIDI of Chopin’s <i>Sonata No. 2, Op. 35, 2nd movement</i> “as it is”.	35
2.4.	Four independent voices in the Fugue No.1 in C Major, BWV846 by J.S. Bach.	37
2.5.	An actual recording of a simple melody.	37
3.1.	Scaling and shift operations that recover the rhythm structure.	46
4.1.	Performance MIDI to score system proposed by Cogliati et al.	56
4.2.	System Diagram.	61
5.1.	In-note beat prediction model.	67
5.2.	The initial architecture of the model.	70
5.3.	<i>Sevilla</i> by Isaac Albéniz.	77
5.4.	Claude Debussy’s <i>Clair de Lune</i>	78
5.5.	Impact of data augmentation on hand part model robustness.	81
5.6.	Results of the <i>ceteris paribus</i> experiment.	84
5.7.	Hand part assignment errors after velocity perturbations.	85
5.8.	Velocity influence on hand part assignment.	86
5.9.	Key signature alignment for G major scale.	86
6.1.	The Transformer architecture.	91
6.2.	The Transformer Encoder block.	92

6.3. Dilated convolution.	96
6.4. The Temporal Convolutional Block.	98
6.5. Extended model architecture with dynamics module.	101

List of Tables

2.1. The dynamics table.	36
5.1. Statistics of the dataset used for training.	71
5.2. Annotation structure for the subdatasets.	71
5.3. Example of performance MIDI annotation in TSV format.	73
5.4. Validation metrics of the hand part model.	74
5.5. Validation metrics of the time signature model.	75
5.6. Validation metrics of the key signature model.	75
5.7. Validation metrics of the beat quantization model.	75
5.8. MV2H metric evaluation on the test set.	76
5.9. The average errors of certain perturbations (in percent).	80
5.10. The average errors for the hand part model.	81
6.1. Feature omission study.	88
6.2. Ablation study for data augmentation.	88
6.3. Ablation study for the hand part model.	89
6.4. Ablation study for the key signature model.	89
6.5. Ablation study for the time signature model.	90
6.6. Transformer results for the beat model.	93
6.7. Transformer results for the hand part model.	94
6.8. Transformer results for the key signature.	94
6.9. Transformer results for the time signature.	95
6.10. Temporal Convolutional Network results for the beat model.	99
6.11. Temporal Convolutional Network results for the hand part model.	99
6.12. Temporal Convolutional Network results for the key signature model.	100
6.13. Temporal Convolutional Network results for the time signature model.	100
6.14. Dynamics TSV annotations for the ASAP dataset.	102
6.15. Results for the dynamics model.	104

List of Algorithms

4.1. Viterbi algorithm.	59
4.2. Dynamic programming tatum segmentation.	63
5.1. Out-of-note beat prediction.	68

Introduction

The aim of this thesis is to explore the topic of automatic transcription of performance MIDI into musical scores. Despite the rapid advancements in machine learning methods across various music domains, particularly music generation, we believe that the problem of automatic score generation remains underexplored.

The discussion begins with an introduction to the necessary concepts and theoretical frameworks in Chapter 1. The problem of automatic transcription is placed within the broader context of *Automatic Music Transcription*, which is itself a subset of the broader domain of *Music Information Retrieval*. Chapter 2 explores a variety of transcription tasks, along with an overview of the challenges and difficulties associated with automating music transcription.

We delve into how the quality of transcriptions can be measured and the challenges inherent in their evaluation. Three sets of metrics are presented, each representing a distinct approach to assessing transcription quality: MUSTER, MV2H, and *Score Similarity*. This topic is covered in Chapter 3.

In Chapter 4, we present brief history of the developments in the field, provide the modern formulation of the problem, and analyze two earlier methods of performance MIDI to score conversion: one based on Hidden Markov Models, and another using dynamic programming. These methods were the main academically documented approaches to the problem. We detail the concepts and algorithms behind them and discuss their limitations.

The core of the thesis focuses on the state-of-the-art model for MIDI transcription, *Performance MIDI-to-Score Conversion by Neural Beat Tracking* [LKMB22], which combines convolutional recurrent networks with dynamic programming for beat prediction. This model was the first to successfully utilize machine learning methods for (almost) complete MIDI to score transcription. We provide a detailed discussion of the model’s architecture, performance results, and observed behaviors in Chapter 5. As a novel approach, using custom explainable machine learning tools, we analyze the model’s undesired behaviors in depth and introduce methods to better understand its individual decisions. We also highlight several common transcription problems generated by the model. Insights gained from this analysis were used to improve the model’s robustness to specific transformations, e.g. velocity perturbation,

which originally distorted heavily the model’s outputs. The results of this investigation are presented in the experimental section.

In Chapter 6, we extend the study by conducting ablation studies that reveal successful strategies to improve the model’s robustness. Beyond that, we conducted experiments comparing the base model with two other promising architectures: *Transformers* and *Temporal Convolutional Networks* (TCN). We show that the basic Transformer architecture is not on par with the base model, but small TCNs achieve comparable results, while being more computationally efficient than the original convolutional recurrent networks.

Moreover, we extended the base model by incorporating an additional component for handling dynamics. Additionally, we proposed a method to evaluate the quality of dynamics transcription as an extension of the MV2H metric while maintaining its foundational principles.

At the end of the thesis, in Chapter 7, we summarize the entire work. We describe potential improvements paths.

This thesis aims to be accessible without requiring prior knowledge of advanced music theory. While we have strived to clarify musical concepts as much as possible, it is beyond the scope of this work to explain all of used musical terms. Readers interested in a deeper understanding of musical terminology are encouraged to refer to [Rea69], which offers a detailed explanation of the musical elements discussed.

All models were trained on the *Entropy* computing cluster, provided by the *Faculty of Mathematics, Informatics, and Mechanics* at the *University of Warsaw*.

I would like to thank the authors of the paper [LKMB22], who kindly facilitated reproducing the results by providing tensor-to-MIDI converters. I also acknowledge the assistance of *ChatGPT* in helping me translate some of my clumsy sentences into a more human-readable format.

Chapter 1

Representation of Music Information

The representation of music, a multifaceted and intricate domain, plays a central role in the comprehension and analysis of musical works. Music, inherently abstract and temporal, can be encapsulated in two primary forms of representation: *audio* and *symbolic*.

Audio representation captures the live or recorded sound of music, preserving the performance's temporal and dynamic qualities. This form of representation is essential in fields such as musicology, audio engineering, and digital music processing, where the actual sound and its nuances are of prime interest. The shift from a symbolic to an audio-based representation marks a transition from a visual and interpretative medium to a direct auditory experience of music.

Symbolic music representation translates musical ideas into structured symbols. This encompasses the abstract musical qualities such as pitch, rhythm, and dynamics through established notational systems such as Western sheet music and digital formats like MIDI (Musical Instrument Digital Interface). Western sheet music, with its historical roots and standardized notation, provides a visual framework for interpreting and performing music. These notations, evolved over centuries, cover various elements like notes on staves, clefs, key signatures, and time signatures, which collectively guide the performer in realizing the composer's intent. While the internal structure of a music score is not a focus of this work, a curious reader may find useful information in [Rea69].

Different music representations offer distinct insights and tools for understanding, analyzing, and transforming music. This chapter delves into these two primary ways of representing music, defining basic notions around these.

1.1. Audio Signals

Audio-based music representation refers to a form of signal representation. The most prevalent of these is the *waveform*, a real-valued function in the time domain that represents the temporal amplitude of a sound. Another common signal representation is the *spectrogram*, which provides information about the local¹ frequency distribution within an audio sample.

Although this form of representation is not the primary focus of this work, understanding certain notions is beneficial for a more comprehensive grasp of the general *Automatic Music Transcription* framework, of which *Performance MIDI to Score Transcription* is a part.

1.1.1. Waveform

Digital audio signal is not continuous but discrete. The transformation from a continuous-time signal to a discrete-time signal is called *sampling*. During this process, a single value of the signal at a specific point in time is referred to as a *sample* [Smi99, p. 34–35]. Additionally, an entire sample recording is sometimes called an *audio sample*.

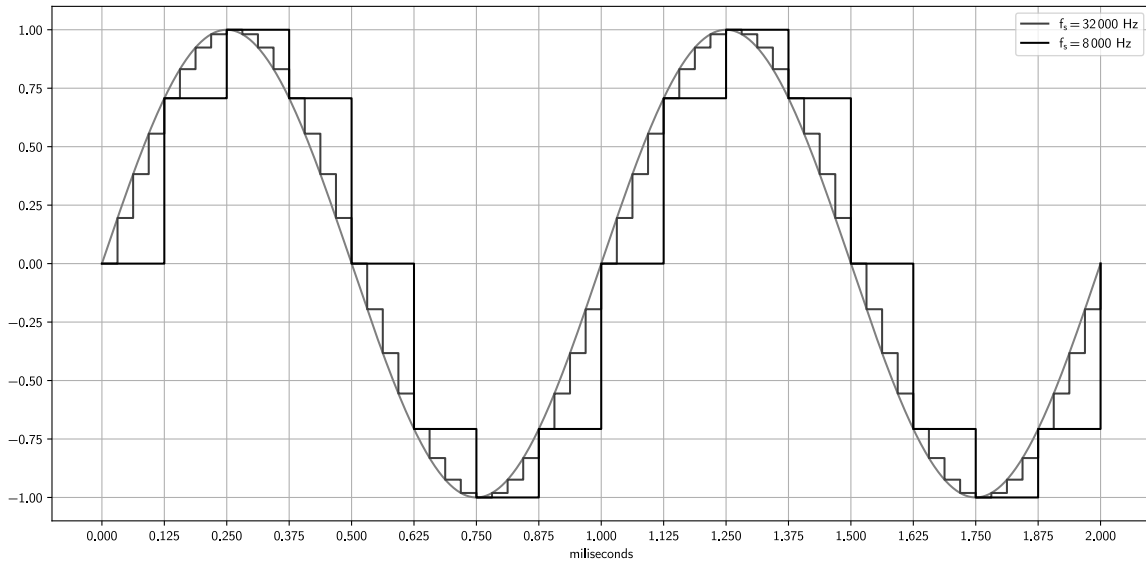


Figure 1.1: An example of a sampling a sine wave of the frequency 500 Hz with different sampling rates: $f_s = 8000$ Hz and $f_s = 32000$ Hz. The quality of the audio is degraded but the sampling fidelity is better for the higher sampling rate.

There are two main parameters of sampling process which directly affect the overall quality of the obtained sound:

- *Sampling rate* — the average number of samples in one second.
- *Bit depth* — the number of bits that one sample encodes.

¹With respect to the time domain.

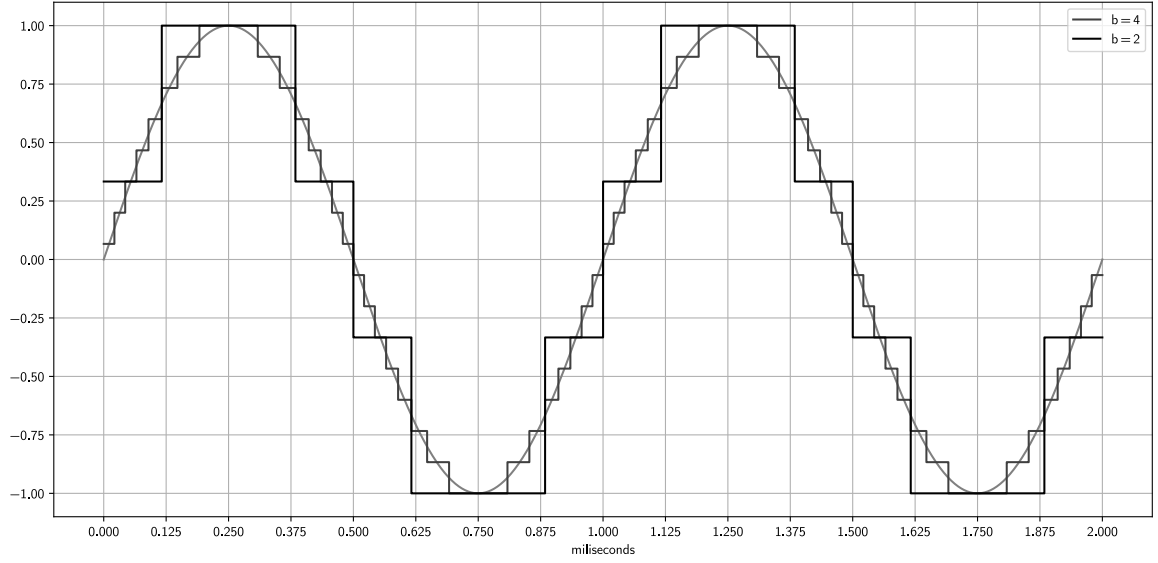


Figure 1.2: An illustration of bit depth affecting the quality of an audio: a sine wave of the frequency 500 Hz sampled with two different vertical resolutions: 2-bit and 4-bit. The 2-bit sample barely reflects the original signal. Sampling rate is assumed to be infinite.

The sampling rate sets an upper limit on the frequency that can be accurately represented in a discrete signal. This principle is conveyed by the *Nyquist–Shannon sampling theorem* [Smi99, p. 40]. On the other hand, the bit depth refers to the vertical resolution of a sound, determining the dynamic range of the signal [Smi99, p. 36].

Waveform representation easily allows the analysis of amplitude peaks and the audio envelope, which includes aspects of audio dynamics: attack, sustain, decay, and release of a sound.

1.1.2. Spectrogram

The waveform representation is not the only one possible way of encoding an audio signal. Each signal can be decomposed into frequencies, meaning that each (regular enough) function can be viewed as a sum of trigonometric functions. More precisely, a regular enough (i.e. continuous) real-valued function $f: \left[-\frac{P}{2}, \frac{P}{2}\right] \rightarrow \mathbb{R}$ can be expressed as an infinite series:

$$f(x) = \sum_{n=-\infty}^{+\infty} c_n e^{2\pi i \frac{n}{P} x}$$

where c_n is defined as:

$$c_n = \frac{1}{P} \int_{-\frac{P}{2}}^{\frac{P}{2}} f(x) e^{-2\pi i \frac{n}{P} x} dx$$

for $n \in \mathbb{Z}$. See [Rud76, p. 185–186] for more details.

When a Fourier transform is applied to short time windows of a discrete signal using *Discrete Short-time Fourier Transform* (DSTFT), a series of signal frequency decompositions is obtained. The process results in a *spectrogram*, a visual representation of an audio signal. A spectrogram displays how the intensity of various frequency components changes over time, offering a detailed view of the signal’s frequency spectrum [Smi99, p. 365–366].

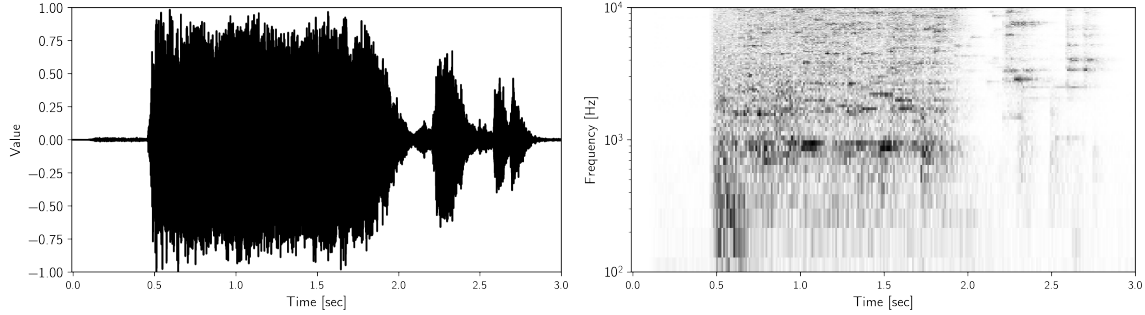


Figure 1.3: A comparison of two different representations of the same sound: a glass break: *waveform* (left) and *spectrogram* (right). The audio is rich in various frequency bands.

The spectrogram representation is vital for retrieving frequency information, aiding in the extraction of features such as pitch (the fundamental frequency) and timbre. However, short-time Fourier transforms are constrained by a trade-off between time and frequency resolutions. Achieving high resolution in one inherently limits the resolution in the other [Smi99, p. 365–366].

1.2. Symbolic Representation

Symbolic representation encodes music through structured, discrete set of symbols, which abstracts certain musical elements such as pitch, duration or dynamics. The ISO/IEC 14496-23:2008 defines *Symbolic Music Representation* (SMR) as *a logical structure based on: symbolic elements representing audiovisual events, the relationship between those events, and aspects related to how those events can be rendered (visually as music notation or audibly) and synchronized with other media types* [ISO08].

The definition encompasses various forms of a musical notation, with traditional sheet music as the most established form.

1.2.1. Sheet Music













Sheet music, also known as a *score*, is a form of musical notation that can be handwritten or printed. It utilizes musical symbols to represent various elements such as the pitches, rhythms, and chords of a song or an instrumental piece [Wik24c]. Sheet music provides

There are many forms of sheet music notation, originated in different cultures. The music notation of our interest is the notation of Western Classical Music, originated in Europe.

Frédéric Chopin



Figure 1.4: First bars of Frédéric Chopin’s *Prelude Opus 28, No. 4* [Cho39].

- *Staff*: Consisting of five horizontal lines and four spaces, each representing a different musical pitch [Rea69, p. 27].
- *Clef*: Positioned at the beginning of the staff, it specifies the pitch of the notes on the staff (e.g., treble clef , bass clef : [Rea69, p. 51–55]. For instruments of a wide pitch range (such as piano), it may change during the course of a piece to accommodate the varying pitch ranges.
- *Notes and Rests*: Symbols that represent the pitch and duration of musical sounds and silences, such as whole note , half note , quarter note , etc., and corresponding rests like whole rest , half rest , quarter rest , etc. Dotted notes, like the dotted half note , are extended by half their length, equating to three quarter notes in this example. For further information, refer to [Rea69, p. 64–65, 96, 103, 113–114].
- *Key Signature*: Indicates the music’s key by specifying consistently altered *flats* () or *sharps* () [Rea69, p. 135–136]. For example, a sharp symbol  next to the clef:



as seen in Figure 1.4, signifies that each note on the indicated line should be played a semitone higher (in this case, $F\sharp$ instead of F), corresponding to the E minor scale (enharmonically equivalent² to G major).

- *Tempo Marking*: Indicates a song’s tempo, specifying the duration of a note. This can be an exact *beats per minute* (BPM) value, such as $\text{♩} = 90$, or described conventionally using Italian terms, ranging from *larghissimo* (extremely slow, under 40 BPM) to *prestissimo* (extremely fast, over 208 BPM) [Rea69, p. 276–277].
- *Time Signature*: Dictates the rhythmic structure by defining the number of beats per measure and the note value for one beat [Rea69, p. 148–150]. Represented as a fraction $\frac{n}{m}$, with the denominator m being a power of 2. For example, $\frac{3}{4}$ indicates three quarter notes per measure. Common time signatures like C and C are equivalent to $\frac{4}{4}$ and $\frac{2}{2}$, respectively.
- *Dynamics*: Symbols and terms indicating the volume of the music, marked from *ppp* (pianississimo), through *mp* (mezzo piano) and *mf* (mezzo forte), to *fff* (fortississimo) [Rea69, p. 250]. These markings are interpretative and not absolute.
- *Articulation Marks*: Instructions on the execution of notes, such as staccato or legato, visually represented by accents and slurs [Rea69, p. 260–268].

A score of a music piece is comprehensive enough to encapsulate a variety of musical elements, serving as a detailed blueprint for performers to interpret and bring to life the composer’s intentions. However, the sheet notation as it is cannot be used by computers directly. Thus we need to limit ourselves only to digital representations of musical scores.

1.2.2. Musical Instrument Digital Interface

The standard for music communication is *Musical Instrument Digital Interface* (MIDI). However the term MIDI is overloaded and means usually several things:

- A protocol enabling musical information (notes, velocities, control signals) communication between electronic devices.
- A technical standard detailing the specifications for digital interfaces and hardware connectors.
- An abstraction representing musical information encompassed within the MIDI protocol.
- A file format that stores a sequence of recorded MIDI messages.

²Two notes are *enharmonically equivalent* if they have the same pitch but are notated differently [KP94, p. 10]. This is a property of the standard twelve-tone equal temperament tuning (12-ET). Other tuning systems may not have this property.

- A MIDI stream, or a content of a MIDI file.

The specific interpretation of MIDI depends on the context. In this work, *performance MIDI* refers to the last point, where the stream is captured through a digital piano or other MIDI-compatible digital keyboards³.

A MIDI stream comprises events, each consisting of two components: a MIDI time and a MIDI message. The time value indicates the duration to wait before executing the next message in the MIDI data sequence [dOO17, p. 3]. There are two principal message types:

- *Note on*: Sent when a performer presses a key on an instrument.
- *Note off*: Triggered when a player releases a key, turning off the corresponding note.

Each message specifies two values: *pitch* and *velocity*⁴. All values are encoded by a 7-bit integer, from 0 to 127.

MIDI streams do not encode sound nor they specify it. The sound rendering is not a responsibility of the MIDI protocol.

The MIDI standard is built upon the 12-tone equal temperament (12-ET) system, which divides an octave into 12 logarithmically equal parts⁵. This system is represented in the conventional 88-key keyboard layout, ranging from A0 to C8, corresponding to MIDI pitch values within the range of [21, 108]. The relationship between frequency and pitch in standard tuning⁶ is expressed by the formula:

$$m = 12 \log_2 \left(\frac{f_m}{440 \text{ Hz}} \right) + 69$$

where m is the pitch for a given frequency f_m [Ass96, p. 47–48].

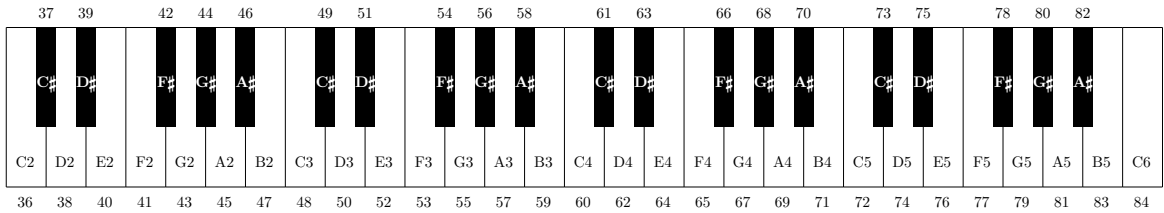


Figure 1.5: A piano keyboard layout with 49 keys. The integers stand for MIDI note values.

A velocity value for a played note should be non-zero, typically controlling the volume of the note. However, there is no universal agreement on the exact gain corresponding to a given velocity value.

³Although MIDI is not exclusively used for keyboards, the primary focus of the deep learning methods discussed here is on piano compositions.

⁴A note intensity, usually correlated with the volume, but not necessarily. This term derives from how some digital instruments measure the speed at which a key is pressed. Some instruments may not respond to this attribute at all [Hub07, p. 22].

⁵The frequency of a note is perceived logarithmically.

⁶The widely accepted standard tuning, A440, established by ISO [ISO75], sets the frequency of the A4 note at 440 Hz. This A4 note corresponds to the MIDI pitch value of 69.

The smallest MIDI time unit is *tick*. The length of a single tick is determined by two MIDI parameters:

1. *tempo*, which describes microseconds per quarter note.
2. *ticks-per-quarter-note*, with a common value of 480.

For example, in the tempo of 120 beats per minute (BPM) the quarter note lasts 500,000 microseconds. In that case, assuming 480 ticks per quarter note, one tick lasts:

$$\text{tick length} = \frac{500,000 \text{ microseconds}}{480 \text{ ticks}} = 1041.67 \text{ microseconds}$$

This feature allows MIDI to handle both physical lengths, as well as musical ones.

While this work primarily focuses on MIDI representation, other notable symbolic representations include:

- *MusicXML*: An XML-based file format for representing Western musical notation, designed for both human and machine readability [Goo01].
- *ABC*: A simplified musical notation system encoding essential musical elements [Wal13].
- *LilyPond*: As authors claim, *LilyPond* is *designed to solve the problems we found in existing software and to create beautiful music that mimics the finest hand-engraved scores* [Tea02]. *LilyPond* has been used to generate all scores and musical glyphs present in this dissertation.

Symbolic notation has its limitations: it can only describe predefined musical aspects, ignoring a multitude of other factors⁷. For instance, the sheet notation does not capture the timbre of sound and is very limited in expressing dynamic sound ranges. While the format was not primarily designed to represent symbolic music but to allow communication between musical devices, it may be used for that purpose as it can handle musical as well as physical onset times and note durations [GCM14].

Most symbolic representations are based on the standard division of an octave into 12 tones, which makes them less suitable for music from non-Western European traditions, such as the Indonesian pentatonic *slendro* or seven-note *pelog* systems, for gamelan instruments [Set05, p. 73]. Certain music genres, like *musique concrète*, are not expressible through traditional notation at all [SND12, p. 69–77].

In the context of this work, we limit ourselves to the heritage of Western European Music, as we are focused on classical European piano works, composed by Bach, Mozart, Beethoven, Schubert, Chopin, Liszt and others.

⁷This is, as any kind of abstraction, rather a desired property.

Chapter 2

Automatic Music Transcription

Automatic Music Transcription (AMT) is the design of computational algorithm converting an acoustic musical signal into some form of musical notation, such as sheet music or MIDI files [BDDE19]. AMT is a subset of the broader field of *Music Information Retrieval*.

2.1. Music Information Retrieval

Music Information Retrieval (MIR) is an interdisciplinary research field concerning extraction and inference of meaningful features from music, indexing of music using these features, and the development of different search and retrieval schemes [SGU14].

It involves several different scientific backgrounds such as: musicology, psychoacoustics, signal processing, informatics, and most recently, machine learning [Wik24b].

The acronym MIR (*Musical Information Retrieval*) was first used by Kassler in 1966, referring to a special-purpose programming language named MIR [Kas66]. However, the term has significantly evolved since then.

2.1.1. Examples of Problems in Music Information Retrieval

The scope of Music Information Retrieval is very extensive. It includes but is not limited to:

- *Classification and Indexing.* The amount of available digital music is constantly increasing [Ori06] and there is a need to organize music data into categories in order to facilitate navigation through rich audio libraries, link related audio items or managing music archives.
- *Audio Identification.* This task aims to identify a fragment of a given music recording under conditions of recording noise or other distortions. A well-known system for this purpose is *Shazam*. A variant of this task is *query by humming*, where a user hums or sings a melody, and the system recognizes the corresponding song [SGU14].

- *Measuring Music Similarity.* This involves comparing different music pieces to determine their similarity. Analysis may include melody, harmony, rhythm, and lyrics, aiding in genre classification, mood-based recommendations, and identifying cover songs or similar musical styles [BLEW03].
- *Instrument/vocal recognition and separation.* The most general expression of the problem is to identify and separate each instrument from a given audio sample. A specific application is separating a vocal track from a full song.
- *Recommendation Systems.* Certain music systems propose a list of music pieces (usually *playlists*) based on user musical preferences with some parameters involved: *accuracy*, *diversity* or *serendipity* (a measure how surprising a recommendation is) [SGU14].
- *Audio Transcription.* In this task, audio recordings are converted into a written or symbolic form, such as sheet music or MIDI files. This involves detecting and notating various elements like pitches, rhythms, and possibly lyrics from the audio signal.

Our primary interest lies in the last area, audio transcription, which will be the focus of the next section.

2.1.2. Sources of Music Information

While the general scope of tasks and problem in Music Information Retrieval is very broad, the information can primarily be sourced from two categories of music representation: *audio-based* and *symbolic*. These two primary modes of representing music information have been discussed in detail in the preceding chapter.

Besides the two previously mentioned ways of music representation, there is also a bibliographic source of information. That includes: title, composer/arranger/performer, publisher, publication date, and sometimes additional information such as the date of a recording, information about the label publishing, catalog numbers, or music categorization by genre or style. Such bibliographic details can be integral to certain MIR systems, offering a different dimension of information beyond the audio and symbolic representations.

2.1.3. Music Facets

There are many aspects of musics one wish to extract from. Downie lists a few aspects, though not mutually exclusive, that can be subject to Music Information Retrieval [Dow03]:

- *Pitch Facet.*
- *Temporal Facet.*
- *Harmonic Facet.*
- *Timbral Facet.*

- *Editorial Facet.*
- *Bibliographic Facet.*

Extracting each of these facets involves distinct methodologies as they operate on different layers of musical information.

For instance, generating a complete score from a performance MIDI requires the extraction of pitch, temporal, and editorial facets. While pitch and temporal aspects are present in the MIDI stream, the underlying harmonic and temporal structures, such as key and time signature, are not explicitly present and must be inferred.

In case of pitch, even for a MIDI file, ambiguity is still possible. For example, a MIDI note represented by the number 73 could, depending on the musical context, be interpreted as either C \sharp or its enharmonic equivalent, D \flat . Similarly, note lengths are usually not preserved. For instance, a pianist’s hand movement between notes (unless playing *legato*) typically shortens the actual length of a note compared to its notated duration in the score.

The list above mentioned by Downie is not exhaustive: music is a very multimodal phenomenon and it comes not only with an audio, but also text (lyrics), images (album covers and photographs of musicians) or gestures of performers [SGU14].

2.2. Automatic Music Transcription

As defined by Benetos et al. [BDDE19], *Automatic Music Transcription* (AMT) is *the design of computational algorithms to convert acoustic music signals into some form of music notation*. Typically, the input of an AMT system is a raw audio signal, encoded in one of common audio format, whether lossless (like WAVE, AIFF, FLAC) or compressed (such as MP3, AAC, or Ogg). The output of AMT systems varies and depends on specific use cases.

Regardless of the final output form, the general AMT task is a multi-staged process involving several different subtasks:

1. Instrument recognition and separation
2. (Multi-)pitch estimation
3. Note onset and offset detection
4. Beat and rhythm tracking
5. Interpretation of dynamics
6. Interpretation of instrument articulations
7. Score typesetting

AMT systems may focus on addressing only a subset of these problems, meaning not all stages are necessarily involved in every AMT task. Therefore, AMT should be viewed not as

a singular task but as a class of various tasks. Often, the audio sample may be pre-processed into an intermediate form, such as a MIDI stream, before being subjected to further AMT processing.

2.2.1. Application of Music Transcription

AMT systems can be applied to facilitate a variety of routine tasks that musicians would do manually otherwise.

Benetos et al. (2013) highlights that one of *the most immediate application of automatic music transcription is for allowing musicians to record the notes of an improvised performance in order to be able to reproduce it* [BDG⁺13].

These systems can significantly aid in the learning process for individuals wishing to practice songs for which published scores are either unavailable or hard to access. These systems also may speed up the process of composing, when the tedious task of transcribing recordings is no longer necessary. The latter is the promise of performance MIDI to score transcription systems.

2.2.2. Levels of Music Transcription

Music transcription can be classified into four distinct levels: *frame-level*, *note-level*, *stream-level*, and *notation-level*, each with its unique purposes and challenges [BDDE19].

2.2.2.1. Frame-level Transcription

Frame-level transcription, also known as *multi-pitch estimation* (MPE), involves estimating the pitch of notes at the level of individual samples (or frames), treated independently for each frame [BL23]. This form of transcription may be conducted under the assumption of either monophonic or polyphonic instrumentation, with the latter being more complex.

The outcome is a partial function (or a set of such functions) $f: I \rightarrow \mathbb{R}_{>0}$ where for each point of time $x \in I$ a frequency is assigned. Additionally, frame-level transcription often includes sound energy estimation.

This level of transcription is vital for pitch-correction tools such as *Melodyne*. However, this level of transcription fails to capture a broader context of music structure.

2.2.2.2. Note-level Transcription

Note-level transcription is one abstraction layer higher than frame-level transcription. It aims to estimate an entire note entity, recognize a note’s pitch within a note onsets and offset. It often includes determining note velocities.

One possible format of the output of note-level transcription is a MIDI stream.

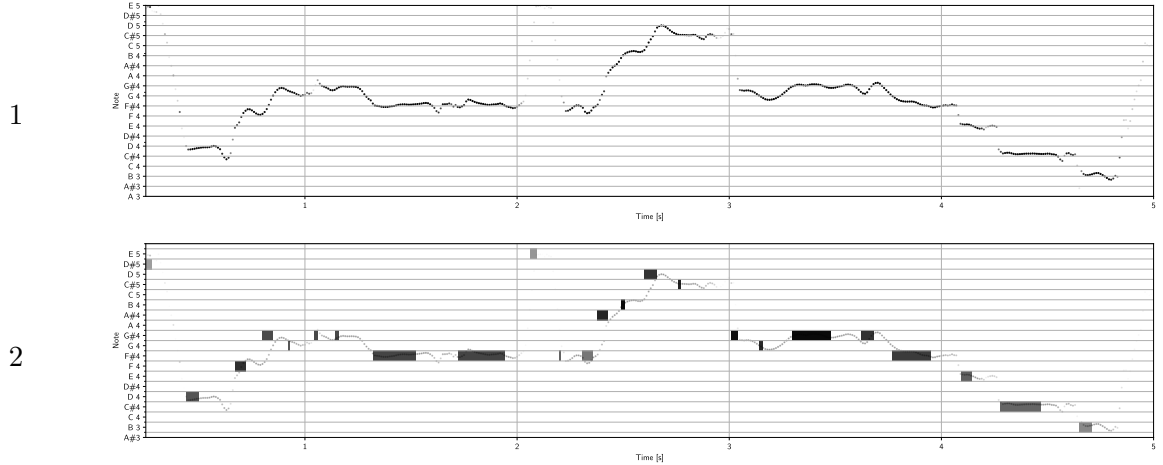


Figure 2.1: A frame-level (1) and note-level (2) transcription of a vocal audio sample. The *CREPE* algorithm has been used for a monophonic pitch estimation [KSLB18], and *CREPE notes* for note quantization [RD23]. Frequency instability leads to misaligned pitch attribution on a note-level transcription.

2.2.2.3. Stream-level Transcription

Stream-level transcription, also known as *multi-pitch streaming* (MPS), involves grouping estimated pitches or notes into separate streams, typically corresponding to individual instruments [BL23]. On that level of transcription, multi-pitch estimation is not sufficient as distinguishing different instruments is needed. The process of instrument differentiation is in fact a clustering problem.

Instruments differ in their timbres, or tone colors, which is the perceived quality of sound that makes two instruments sound distinct even when they play the same pitch at the same energy level. The timbre largely depends on an instrument’s harmonic series and its variation over time (the envelope). This is merely a simplification: not all sounds conform to harmonic series representations; for example, noise or most percussion instruments are exceptions, though tonal instruments are generally well-described by harmonic series [Set05, p. 27–28].

Duan et al. (2014) defines the streaming tasks as: *the clustering objective is to maintain timbre consistency, based on the assumption that sound objects coming from the same source have similar timbre* [DHP14]. Notes coming from the same instrument are expected to have similar timbre characteristic. Depending on the method, the number of instrument clusters may or may not be specified beforehand.

The timbre of an instrument is not static and can vary based on several factors, including:

- The method and force of inducing vibration. For instance, a violin emits different sounds based on the bowing technique, and a piano’s sound quality changes with the force applied to the keys.
- The fundamental frequency. The harmonic series of an instrument varies with the

played note, often resulting in different overtone distributions in lower and higher registers.

- Playing technique. The way a musician produces sound can significantly affect timbre. On wind instruments, for example, variations in embouchure, tonguing, and breath control can create diverse timbral effects.

The timbre clustering becomes particularly challenging when similar but distinct instrument are played simultaneously, and their harmonic series are overlapping.

2.2.2.4. Notation-level Transcription

Notation-level transcription is the process of converting audio recordings into a traditional music notation or a score [BL23]. This process transforms the musical content into a structured format as outlined in the previous section on *Sheet Music*.

Besides the musical content, that is notes and rests, notation-level transcription involves assigning other various musical elements:

- Instruments are organized by separate staves. Each instrument needs to be recognized and appropriately organized within the score.
- Tempo markings are usually indicated at the beginning of a piece but may vary throughout the composition.
- Clefs are used to suggest the pitch range of the voice or instrument. For instruments with a wide pitch range, such as the piano, the clef may change during the piece.
- Time signatures are essential to determine the meter grid of a piece. While the time signature generally remains consistent across instruments, it can vary within a piece.

Some notation-level transcription methods might come with certain limitations and not provide all these elements. For instance, they might default to using the C major treble clef and the common $\frac{4}{4}$ time signature for all staves, regardless of the song's actual key or time signature. Some methods may also assume that a single instrument is being played.

2.2.3. Stages of Music Transcriptions

As previously discussed, the general transcription is a complex task consisted of several steps.

Assume we have a raw audio signal containing a recording of music. For the sake of simplicity, let us consider a raw audio signal containing a recording of music from a single polyphonic instrument such as piano or guitar.

In order to retrieve notes from the recording, usually the audio signal is converted to a spectrogram. This representation allows both pitch and energy estimation.

The transcription involves three challenges:

- *Detecting Pitch Frequencies.* Notes usually produce a series of harmonics – multiples of a fundamental frequency – each with varying energy levels that change over time. Additionally, the fundamental frequency itself may fluctuate due to vibration and glide. This task is further complicated by the limited frequency resolution of the short-time Fourier transform.
- *Recognizing Time Onsets and Offsets:* Identifying when a note begins and ends is rather a straightforward task in monophonic music. It becomes significantly more complex in polyphony, where multiple notes are often played with slightly different timings, and their envelopes overlap with each other. As in the previous task, time resolution of the short-time Fourier transform is limited as well.
- *Measuring Note Velocity.* Notes can be played with different energy, resulting a sound with different amplitude. Accurately assigning velocity to each note is difficult, especially when notes have overlapping harmonic series and energy attribution becomes ambiguous. Limited resolution of a Fourier transform also contributes to this effect.

There are different strategies to overcome these obstacles. Assuming that there is only one instrument playing, these are covered by note-level transcription.

The outcome of this process is a *performance MIDI* or a *MIDI stream*, comprising a sequence of notes characterized by four features: pitch, note onset, duration, and velocity.

A MIDI stream lacks information about the underlying music structure. More specifically, certain score elements are yet to be determined: time signatures, key signatures, voicing (e.g. hand part assigning for piano recordings). Moreover, notes and rests need to be quantized into standardized lengths (e.g., whole notes ♩ , half notes ♪ , quarter notes ♫ , etc.). The problem of imposing the metric structure of a piece, and placing abstracted note values onto the measures, is called *beat quantization*.

This is the final, notation-level stage of transcription: transforming the performance MIDI into a scored format. This involves imposing the structure mentioned above and then rendering the score in a specific music data format.

This stage of audio transcription is the main focus of this work, concerning mostly on digital piano recordings. From now on we will assume that the performance MIDI is given, either retrieved from the audio signal, or recorded directly via some MIDI instrument. However, we do not care about the visual aspects of the score, only its content. The more precise definition of *content* will be formalized in the next fragment.

2.2.4. Performance MIDI to Score Transcription

Let us review basic assumptions of the performance MIDI to score process. In this section, we follow the name convention of Liu et al. (2022), the authors of the main performance MIDI to score model discussed [LKMB22]. However, a few adjustments have been made.

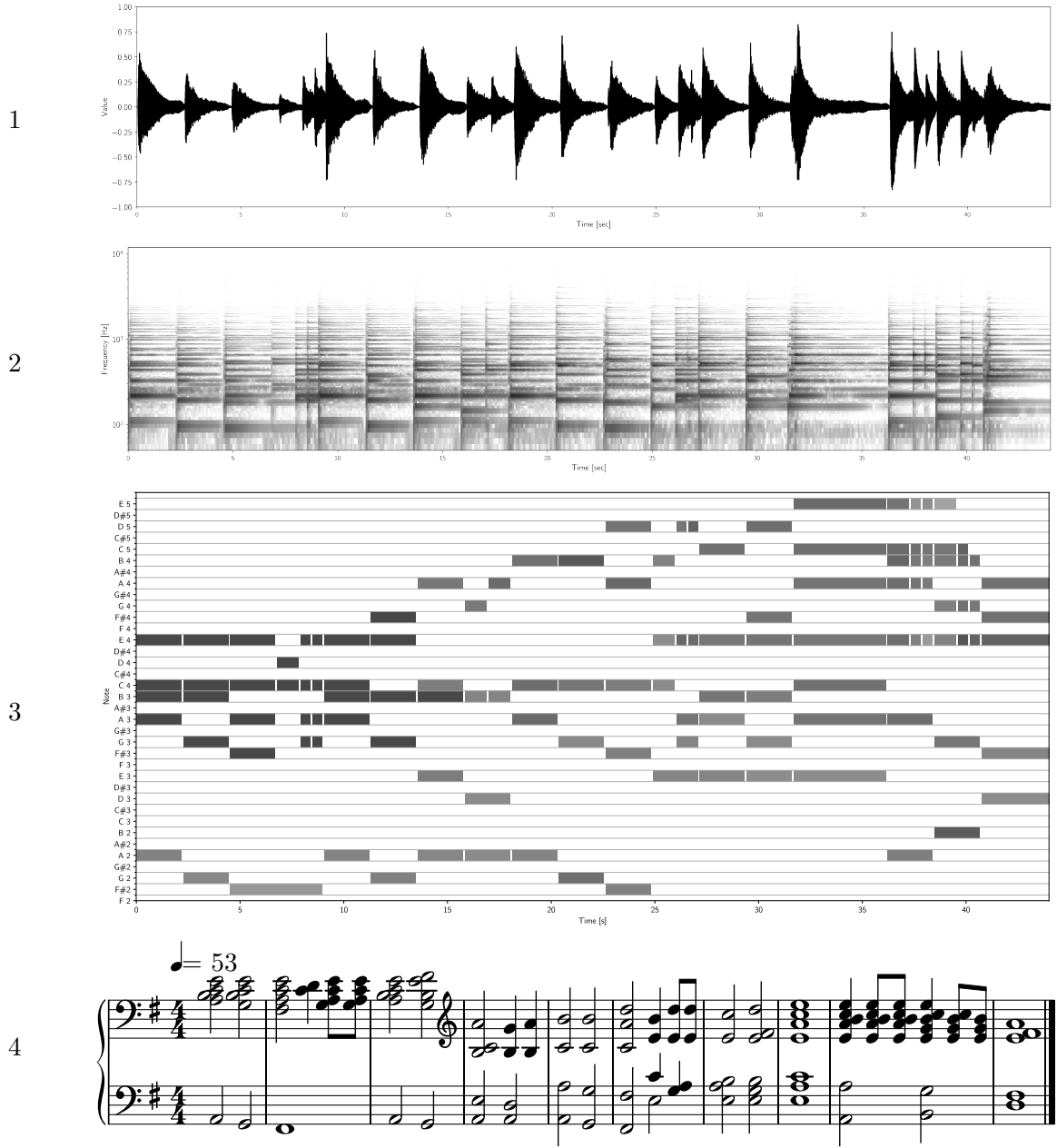


Figure 2.2: An example of an Automatic Music Transcription task on a concrete piano recording. First, the audio sample (1) is transformed to a spectrogram (2) using the short-time Fourier transform, from which notes are retrieved. The result of that process is a sequence of a MIDI stream (3) for which the other aspects of music score are determined (here: clefs, key signature, time signature, tempo marking) and transcribed to a final score (4).

2.2.4.1. Performance MIDI Encoding

The input performance MIDI may be encoded as a sequence $\mathbf{X} = \{p_i, o_i, d_i, v_i\}_{i=1}^N$ of a variable length N , of four features: *pitch* p_i (an integer from the range 0 to 127), *onset* o_i being a positive real number, *duration* d_i , also a positive real number, and *velocity* v_i , an integer from

range 0 to 127. The velocity may be further normalized to a rational number from the unit interval. We assume also that the sequence is sorted firstly by the onset, and secondly by the pitch in the ascending order.

The encoding of a performance MIDI is in fact a two-dimensional tensor. We will call such encoding also a performance MIDI, when no confusion arises.

2.2.4.2. Music Score Encoding

In this section, we define a music score in terms of the essential elements it needs to encode. The output of our automatic transcription method focuses solely on the abstract elements of a score, entirely disregarding the visual layout.

These elements include:

- *Tempo*, quantized as an integer. Practically, only values above 30 are expected.
- *Downbeats* and *beats*, marking sections of each measure¹. This segmentation helps convert a rhythmically unstructured MIDI into structured measures and identify the initial beats of these measures. Both elements are binary.
- *Musical onsets*, indicating the start of notes in terms of quantized beat divisions. Whole integers correspond to notes beginning on a downbeat.
- *Note values*, representing durations as quantized lengths in the form of a fraction (e.g., 1 for a whole note, $\frac{1}{2}$ for a half note). However, note values should be treated as categorical.
- *Key signature*, chosen from one of 12 possible scales: C, C \sharp , D, ..., A \sharp , B, each treated as a separate category.
- *Time signature*, consisting of a numerator and denominator. Since there are only several common time signatures, one can narrow down the choice of possible values: 0, 2, 3, 4 and 6 for the numerator, and 0, 2, 4 and 8 for the denominator. The value 0 stands for other values.

Musical onsets, note values, and even key/time signatures may be assigned for each note. Similarly, downbeats and beats prediction can be done on the note level, although some of these event may not occur without any note being played precisely on these beats.

Since we expect only piano recordings, we also aim to differentiate between left and right hand parts, represented as two staves with treble and bass clefs, respectively. This assignment is binary.

Note that pitch values are derived directly from the input. Articulation and dynamic markings are not considered within the scope of the model proposed by Liu et al. (2022).

¹The downbeat is the first beat of a measure.

To summarize, the output score \mathbf{Y} can be represented as a pair of two tensors ($\mathbf{Y}_b, \mathbf{Y}_n$):

- The beat tensor $\mathbf{Y}_b = \{t_j, db_j\}_{j=1}^B$ encapsulates the real times t_i of beats, along with a binary downbeat classification db_i .
- The note-level tensor $\mathbf{Y}_n = \{mo_i, nv_i, h_i, tn_i, td_i, k_i, tem_i\}_{i=1}^N$ where mo_i is quantized music onset, nv_i is the note value, h_i indicates the hand part of a note (left or right), tn_i and td_i are time signature numerators and denominators, k_i is the key signature, and tem_i is the BPM represented as an integer. All values are categorical.

The objective of performance MIDI to score automatic transcription is to find a good enough function $\mathbf{X} \rightarrow \mathbf{Y}$. Further sections will be devoted to measuring the quality of the resulting score in order to define the *good enough* part.

2.3. Challenges in the Transcription Task

Before providing an overview of methods for MIDI to score conversion, it is essential to first understand the challenges of the transcription task. We need to examine how a performance captured in a MIDI file deviates from traditional sheet music and identify the types of information that are lost in the process of translating between these two forms of musical representation.

Some challenges are directly related to the way musical data is encoded in the MIDI format, such as the issue of *note spelling*². This leads to ambiguities that require further disambiguation through additional algorithms or human intervention.

Other challenges arise from the nature of musical performance itself and the variations that occur when a piece is interpreted and played. A live performance rarely, if ever, adheres strictly to the abstract representations of rhythm and timing found in traditional notation. Some vital aspects of the human performance are described in more detail in the next section.

2.3.1. Common Transcription Problems

Traditional music notation represents an idealized version of how a piece should be played. While it provides a structured framework for pitch, rhythm, dynamics, and articulation, it abstracts away from many performance nuances that are either too subtle, too variable or simply irrelevant to be captured in symbolic notation.

These deviations can make the direct conversion of performance MIDI into readable sheet music difficult, as exemplified in Figure 2.3. Below, we will explore some of the most common transcription problems.

²See the Section 2.3.1.4 for the explanation.

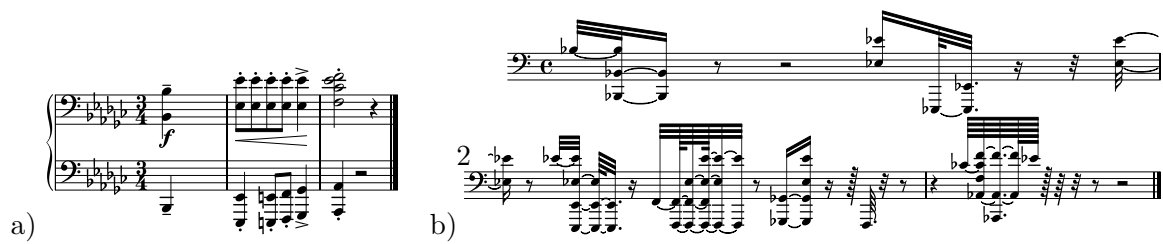


Figure 2.3: An example of the imported performance MIDI of Chopin’s *Sonata No. 2, Op. 35, 2nd movement* „as it is”: a) the original sheet, b) the imported performance MIDI without time/key signature assignment. Not only the actual durations render the sheet unreadable and impractical, but also introduce error resulting from a note quantization attempt to an incorrect meter grid. Notice that there is also no hand separation.

2.3.1.1. Tempo

MIDI files operate with a fixed tempo, or more precisely, a discrete set of tempo changes. A human performer, however, never aligns to the tempo perfectly, even in a course of a single measure.

In many cases, these deviations are deliberate. For example, a performer might slow down (*rubato*) or sustain a note longer than notated (*fermata*) for expressive purposes. While these nuances may sometimes be annotated in the score, it is not always the case, and even when marked, the precise nature of the tempo fluctuation is left to the performer’s interpretation. Performers have considerable freedom in determining the extent and suddenness of tempo changes, which MIDI struggles to represent accurately.

More importantly, performance MIDI files often do not reflect internal tempo structures, rendering the ticks-per-quarter-note parameter ineffective. This issue is further addressed in Section 2.3.2.

2.3.1.2. Note Duration

In live performances, note durations often deviate from those specified in the score. Pianists, for example, tend to slightly shorten note lengths to prepare for the next note or chord.

In *legato* playing, some notes may be overlapping and their actual length is a bit longer than indicated by the notation. This elongation is especially prominent in rapid, fluid passages. Conversely, *staccato* notes, which are played with almost immediate release, are never denoted with actual expected time of playing. Traditional notation primarily indicates note boundaries, leaving the actual performed lengths to be interpreted by the pianist.

2.3.1.3. Dynamics

The traditional notation system offers a relatively limited palette of dynamic markings, ranging from *pianississimo* (**ppp**) to *fortississimo* (**fff**). Changes in volume over a phrase are

marked by *crescendo* (<) and *diminuendo* (>) symbols³. Thorough centuries, the notation has been expanded and introduced new markings such as *sforzando* (sfz) or *subito forte* (sf).

The dynamics are not absolute, rather contextual and depending on the surrounding musical elements. Since these markings are guidelines, there is some room for interpretation by the performer.

In contrast, MIDI format stores a concrete note velocity with 7-bit precision. As a rule of thumb, the following arbitrary classification is adopted:

Dynamic	Marking	Velocity
<i>pianississimo</i>	<i>ppp</i>	16
<i>pianissimo</i>	<i>pp</i>	33
<i>piano</i>	<i>p</i>	49
<i>mezzo-piano</i>	<i>mp</i>	64
<i>mezzo-forte</i>	<i>mf</i>	80
<i>forte</i>	<i>f</i>	96
<i>fortissimo</i>	<i>ff</i>	112
<i>fortississimo</i>	<i>fff</i>	127

Table 2.1: The dynamics table [Wik24a].

While the MIDI format represents the dynamics numerically instead of symbolically, there is no direct translation of the actual note velocity to dynamics markings: these describe broader phrases rather than individual notes. An analysis is required to translate raw velocity data into meaningful dynamics guidelines.

The standard transcription model does not handle dynamics explicitly. Experimental methods addressing this issue are discussed in Chapter 6.

2.3.1.4. Harmonic Equivalence

As the MIDI file format does not distinguish enharmonically equivalent pitches, we generally cannot solve the problem of pitch spelling, that is assigning correct *accidentals*: flats (\flat) or sharps (\sharp) [Cam00]. For example, in keys like D major or G major, a MIDI note 66 is likely to be spelled as F \sharp , while the same pitch may function as G \flat in keys such as E \flat or B \flat minor.

The ambiguity can be resolved through assigning the key signature, that is determining the key of the scale. This approach has been successfully used in [LKMB22].

2.3.1.5. Voicing

A *voice* (or *stream*) is defined as a *monophonic sequence of successive, non-overlapping musical tones* [KNP⁺07]. This notion can be broadened to any sequence of notes, whether monophonic or polyphonic, that is perceived as a single musical entity [Cam08].

³ *Crescendo* is a gradual increase of the volume, and *diminuendo* is a gradual decrease.

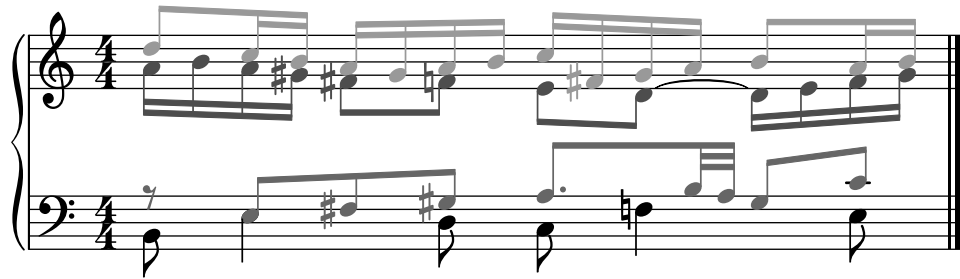


Figure 2.4: Four independent voices in the Fugue No.1 in C Major, BWV846 by J.S. Bach.

A single MIDI track can be viewed as a collection of notes, and the MIDI format does not offer any levels of further subdivisions. Consequently, exporting a score to a MIDI file loses all information regarding the original voicing. For that reason in the case of piano music, it is common (but not guaranteed) to separate left-hand and right-hand notes into two tracks, corresponding to the two hands. Since there can be more voices than only indicated by the left and hand right (as the example above shows), the hand separation is a subset of a broader challenge of voice separation. For a related work in that matter, refer to [KNP⁺07].

2.3.2. Score-Informed MIDI files

As noted previously, performance MIDI files are generally not aligned to an internal MIDI tempo. One can correct a MIDI file by assigning the tempo so that every beat in the piece corresponds to the same number of ticks. This process effectively regularizes the performance by aligning it with the underlying metric structure of the music.

Grohgan et al. [GCM14] identify two types of MIDI files based on this alignment:

1. *S-MIDI*: *Score-informed* MIDI files, which have been aligned to a specific tempo, ensuring that the musical events correspond accurately to the beat structure.
2. *P-MIDI*: Unaligned *performance* MIDI files, where the real tempo of the performance is unknown or inconsistent with the internal timing of the MIDI format.

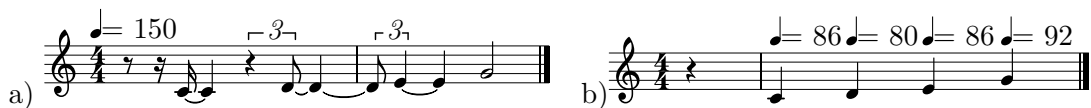


Figure 2.5: An actual recording of a simple melody: a) the score-agnostic recording with incorrectly quantized musical onsets not aligned with the MIDI tempo, b) the score-informed representation of the performance. The metrical structure and the rhythmic intent is clear in the second excerpt. Notice that the short silence at the beginning is correctly identified as a separate (shortened) measure.

The beat quantization problem may be thought as a process of converting a P-MIDI into a S-MIDI file.

Most musical software programs struggle to correctly import P-MIDI as they rely on the internal tick-based structure to interpret note onsets. Without accurate alignment to a tempo grid, the resulting transcription can appear rhythmically distorted, with notes placed in incorrect positions relative to the beat (see: Figure 2.5). This makes beat quantization an essential step in the transcription of human-performed music.

2.3.3. Summary of Steps

In summary, the procedure of translating the raw performance MIDI to a score involves at least:

- note quantization, which consists of:
 - recognizing the beat grid within the downbeats
 - interpreting the musical onsets according to the grid
 - assigning proper musical duration to notes
- assigning time signatures
- assigning key signatures
- separating independent voices (here: separating hands)

Each of these subtasks will be implemented as a separate module within the main model architecture.

2.3.4. Limitations

While AMT systems aim to convert performance data into a readable musical score, only a subset of musical elements is typically included in the generated output. Many nuances remain beyond the scope of current AMT systems, including the main model described here.

Key limitations include:

- **Accents and articulation marks:** Elements such as *staccato*, *tenuto*, or accents are vital for conveying how each note should be played in terms of attack, duration, and emphasis. None of them are handled by presented AMT models.
- **Pedalization:** While MIDI format is able to capture pedal events, the considered transcription process does not translate these events into notated pedal marks. A human intervention is needed to transform raw MIDI signals into a clean notation.
- **Ornaments:** Grace notes, trills, mordents, and other ornaments add complexity to both performance and notation. As the MIDI does not distinguish these special notes⁴, all of such ornaments are shown as ordinary notes. To the author’s knowledge, there is no model that attempts to transcribe performance recording recognizing ornaments.

⁴On the other hand, MusicXML does.

- **Dynamics and expression marks:** While note velocity can give a rough indication of dynamics, AMT systems do not capture more complex dynamic markings like crescendo, diminuendo, or sudden dynamic shifts. A partial work in that matter can be found in the Chapter 6.
- **Clefs:** The changes in the clefs are not registered in a MIDI stream. Thus, it is not a task of the model to assign clefs⁵. For simplicity, we may assume that the treble clef constantly corresponds to the right hand, while the left hand is governed by the bass clef, although this is only a crude simplification.
- **Phrasing:** The subtle shaping of musical phrases, which performers use to convey musical direction and intention. Common group indications like slurs are absent in the generated scores.

This is, of course, by no means an exhaustive list.

⁵Musical software often interprets MIDI files and handles clef assignments for the user.

Chapter 3

MIDI-to-Score Transcription Evaluation

When transcribing a performance MIDI into a musical score, one needs to evaluate how faithful an output is to the original sheet. A transcription quality metric should take a transcription of a performance-MIDI and a ground-truth score as input, and produce a non-negative real number as an output, ideally within the range $[0, 1]$.

As the musical transcription is a multifaceted process, measuring transcription fidelity is far from a trivial task. Music notation is complex, consisting of a variety of interdependent elements. Solely for that reason, it is difficult to achieve metric that simultaneously:

- assigns a single value to an entire transcription
- encompasses all aspects mentioned in the Section 2.2.4.2.

On top of that, there are additional layers of complications:

- Even for a single aspect it may be not clear how to measure quality of a transcription. For example, assessing voice separation is a hard task on its own¹.
- Performers tend to deviate from the written score, whether intentional through artistic interpretation or not due to errors. For example, a pianist can introduce own ornaments or make mistakes by omitting certain notes or playing wrong ones. While the basic structure, such as time and key signature, is usually preserved in a performance, we may expect differences in a note stream. The goal of the evaluation is not to judge the performance's fidelity to the notation but rather the accuracy of a transcription.
- Musical elements are interconnected. A transcription error in one area can cascade into another. For instance, inaccurate beat recognition may make difficult to assign a

¹The term *voice* itself is ambiguous and there is no single definition that tells exactly what constitutes a voice, let alone how to separate voices. An extensive discussion on that topic can be found in [Cam08].

proper time signature. It is not clear how to penalize a single transcription error once, and only once. It is also not easy to fairly attribute the faithfulness of a transcription of an interpretation that substantially diverges from the sheet.

- In general, there can be multiple valid notations of the same musical idea, and in some case neither of them are more correct than other ones. A well-designed metric should allow some degree of freedom when it does not hurt the quality of the transcription.

As a result, more often than not, single aspects of a model are being analyzed and evaluated in isolation to other parts of a transcription. The overall result may be an aggregation of submetrics, though this approach is not always justifiable.

Earlier approaches to assessing transcription quality were often lacking in rigor. Some common pitfalls in the literature include:

- **Lack of Thorough Analysis.** A rigorous evaluation requires an assessment of all relevant aspects of a proposed solution. Failing to attribute vital parts of a model or algorithm renders the analysis incomplete.
- **Limited Case Demonstration.** The results of an algorithm were often demonstrated on a very limited set of musical pieces. It is not obvious how a model performs outside showcased examples, and it endangers the research to cherry-picking. This is the case of both works [TSO⁺02] or [YCV05], which are described in more detail in Chapter 4.
- **No Public Dataset or Reproducibility.** For both training and evaluation, the dataset used should be clearly described and, ideally, made publicly available. Without this, results are difficult or impossible to reproduce, as seen in [TSO⁺02].
- **Absence of Human Expert Evaluation.** While statistical benchmarks are necessary for quantitative assessment, there may overlook nuances that are essential for performers relying on transcriptions.

In the following sections we will explore various attempts from the literature to provide transcription quality metric.

3.1. Manual Attribution

Manual attribution refers to any form of quality assessment that involves human intervention. It almost exclusively requires experts, as the task of evaluating music transcription demands understanding of both musical theory and performance practice. Human evaluators are typically trained musicians, musicologists, or composers who can assess various dimensions of a transcription beyond what automated systems can measure. These experts are called upon

to determine not only the technical accuracy of a transcription but also its musicality, readability, and adherence to stylistic norms. The output of human evaluation may be either qualitative or quantitative.

One of the key benefits of manual attribution is its ability to handle complex, nuanced and multi-layered aspects of transcription. Trained musicians are well-equipped to judge how well a transcription serves musicians.

Some research has investigated the relationship between computational metrics and human evaluation. A fairly recent study shows that, surprisingly enough, there is no significant correlation between certain transcription metrics² and human quality ratings [HBKW21]. However, combined metrics correlated more strongly (0.682) with human expert evaluations, although this correlation was still lower than the inter-rater correlation between two human experts (0.754).

Despite a large value of human evaluation, it comes with limitations. Manual attribution is time-consuming and labor-intensive. Carefully reviewing a transcription requires domain expertise, time and effort. Access to qualified experts can be costly, and the process is not easily scalable. Another drawback coming from manual attribution that evaluators may be subjective and biased towards certain notation practices. This of course not to say, that quantitative benchmarks are free of biases.

3.2. Binary and Multiclass Classifier Metrics

The transcription problem can be viewed as a set of interdependent classification tasks. In particular:

- Beat quantization can be thought as the classification of beats and downbeats.
- The hand part model is a binary classifier (left or right hand).
- Time and key signatures assignment is a multiclass classification problem³.

Hence, all loss functions and metrics that apply to classifiers can be used both for both training and evaluation. Since these are standard measures, they won't be discussed in much detail, and notions of true/false positive and negatives are assumed to be known⁴.

3.2.1. Cross-entropy Loss

The *cross-entropy loss* is a commonly used loss function for training classifiers in machine learning, derived from the *cross-entropy* between two probability distributions over a shared

²Two metrics *missing note rate* and *onset time error* had no significant correlation with human score. These metrics are going to be explained in further sections.

³The time signature model has been further reduced to a binary classifier.

⁴A discussion on these metrics can be found in [MRS08, p. 142–144].

set of events:

$$H(X, Y) = -\mathbb{E}_{t \sim X} \log Y(t) = -\sum_t P_X(t) \log P_Y(t)$$

For a dataset with true labels y_n and model predictions \hat{y}_n , where $n = 1, \dots, N$, the *cross-entropy loss function* is defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^N [y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n)]$$

Further discussion with justification of cross-entropy can be found in [GBC16, p. 73–75, 178].

3.2.2. Accuracy, Precision and Recall

In binary classification, one class is labeled as *positive* (or *true*) and the other as *negative* (or *false*). Items belonging to a class are *relevant*, while those predicted by the model as belonging to the class are considered *retrieved* [Roe22, p. 76].

The *accuracy* of a predictor is defined as the ratio of correctly identified elements to the total number of elements:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{false positives} + \text{true negatives} + \text{false negatives}}$$

Precision represents the proportion of correctly retrieved items to the total number of items retrieved:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall measures the proportion of correctly retrieved items to the total number of relevant items:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

These definitions naturally extend to multiclass classification.

3.2.3. F_1 score

F_1 score (*F-measure*) is a measure that equally balances precision and recall. It is defined as a harmonic mean of the precision and recall:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

For multiclass classification, two common methods can be used to compute the *F*-measure: *macro* and *micro* averaging [MRS08, p. 259–261]. These approaches handle cases of class imbalance differently:

- **Macro F_1** : The unweighted average of F_1 scores across all classes, treating each class equally.
- **Micro F_1** : A weighted average of F_1 scores, where weights correspond to the number of instances in each class.

In the context of hand part assignment, applying the F_1 score is dubious. It stems from the fact that the metric is sensitive to label switching; the score is affected by which hand is labeled as “true” [SL09], whether the labeling of hands should not matter at all. Nevertheless, we adopt this measure for consistency with the considered work.

3.3. MUSTER: Music Score Transcription Error Rates

Nakamura et al. (2018) proposed a set of metrics to evaluate their extension of a Hidden Markov Model [NBYD18] for automatic transcription. These metrics build upon earlier work [NYD17] and include:

- **Pitch error rate E_p** . Measures the proportion of notes detected with incorrect pitch.
- **Missing note rate E_m** . Measures the proportion of notes that are present in the ground-truth score but missing from the transcription.
- **Extra note rate E_e** . Measures the proportion of notes that appear in the transcription but are not present in the original score.
- **Onset-time error rate E_{on}** . Quantifies errors in note onset times relative to the expected values.
- **Offset-time error rate E_{off}** . Measures errors in note offset times after normalizing to the correct onset times.
- **Overall Error Rate E_{all}** . The arithmetic mean of all other error rates.

The metrics rely on two components: *note alignment* and *rhythm correction cost*.

The overall set of all metrics is called *MUSTER: MUsic Score Transcription Error Rates* [HNY21].

3.3.1. Note Alignment

The transcription metrics used to evaluate the performance of a model depend on determining how many notes are “correct” or “incorrect” based on several criteria: whether the notes are missing, superfluous, or played with incorrect pitch. Classifying notes in these categories is a non-trivial task. In particular, due to timing variations, missing or extra notes⁵ a direct note-by-note comparison is not a viable approach.

⁵Note that intended ornament notes are classified as superfluous too, henceforth incorrect.

The note alignment process is designed by Nakamura et al. in [NYK17] to robustly align the performed MIDI notes with the corresponding score notes. The algorithm’s objective is to identify and categorize notes into three main groups:

- **Matched notes.** These include both notes that are correctly matched in terms of pitch and onset time, and notes that have correct onset timing but pitch errors (wrong pitch).
- **Extra notes.** These are notes that appear in the performance but do not exist in the original score (false positives). They may occur due to performance errors or expressive choices like embellishments.
- **Missing notes.** Notes that are present in the score but were omitted in the performance or failed to be detected by the transcription system (false negatives) are considered missing.

For more information the exact problem statement, the procedure and algorithm evaluation, refer to [NYK17].

3.3.2. Rhythm Correction Cost

In music, note values are not absolute and depend on the tempo choice. A quarter note played at a tempo 60 beats per minute lasts the same duration as a half note played at a tempo 120 BPM. The choice of the unit is to somewhat arbitrary. Yet, doubling or halving the tempo may disrupt transcription metrics, as some naive metrics might inaccurately treat an entire passage as incorrect when the tempo is altered. This remark points out that the metric values should be analyzed in relationship to other notes, not as absolute values [NYS17, p. 7].

To address this issue, Nakamura et al. introduced *Rhythm Correction Cost* (RCC), which is defined as: *the minimum number of scale and shift operations for onset score times* [NBYD18]. This metric takes into account tempo variations and adjusts for structural differences in note onsets between the transcription and the ground truth via two operations: scales and shifts.

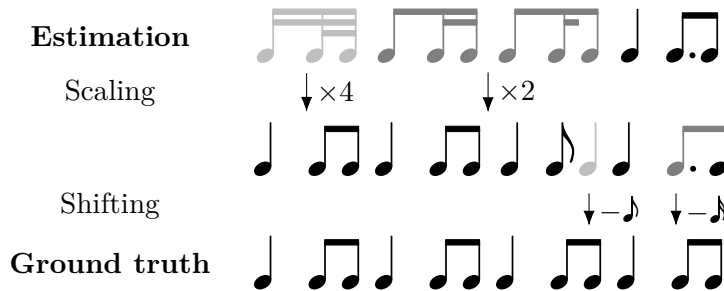


Figure 3.1: Scaling and shift operations that recover the rhythm structure [NYK17].

In the Figure 3.1, two scaling and two shift operations are needed to recover the ground-truth rhythm structure, and that number is minimal. Hence, the RCC is 4.

3.3.3. Metrics

Let us review the definitions of metrics from the paper [NBYD18].

The *pitch error rate* metric quantifies how many notes are detected with incorrect pitch. The formula is rather straightforward:

$$E_p = \frac{\text{number of notes with incorrect pitch}}{\text{total number of ground-truth notes}}$$

The *missing note rate* and *extra note rate* values are derived similarly:

$$E_m = \frac{\text{number of missing notes}}{\text{total number of ground-truth notes}}$$

$$E_e = \frac{\text{number of extra notes}}{\text{total number of ground-truth notes}}$$

Onset/offset-time error rates rely on RCC. The *onset-time error rate* is defined as:

$$E_{\text{on}} = \frac{\text{RCC}}{\text{number of matched notes}}$$

while *offset-time error rate* is derived from the number of notes with an incorrect offset score time after normalization using the closest onset score time:

$$E_{\text{off}} = \frac{\text{number of notes with an incorrect offset score time after normalization}}{\text{number of matched notes}}$$

The exact procedure of how to calculate the numerator is present in [NYD17].

Finally, E_{all} is calculated as the arithmetic mean of all metrics above. It should reflect the overall error rate of a score.

3.3.4. Limitations

The metrics proposed by Nakamura et al. primarily evaluate specific aspects of musical transcription accuracy, focusing on note-related parameters such as pitch, onset, and offset times. While effective for assessing the correctness of a note stream, these metrics do not encompass broader transcription dimensions, including voice leading, harmonic context, or overall musical structure. Consequently, they are limited in their ability to fully represent a broader musical context of a transcription.

The overall metric allows penalty accumulation, meaning that a single transcription error may impact several ratings. For example, a misplaced note may be marked as “missing”

from its expected position while also counted as “extra” in its actual position, leading to compounded penalties.

Additionally, empirical studies, such as Holzapfel et al. (2021) [HBKW21], suggest that several of these metrics may not align closely with human expert evaluations.

3.4. MV2H Metric

The MV2H (*M*ulti-pitch detection, *V*oice separation, *M*etrical alignment, note *V*alue detection and *H*armonic analysis) metric, introduced by McLeod and Steedman [MS18], is a comprehensive evaluation framework designed to assess AMT systems. The MV2H metric integrates several dimensions of transcription quality into a single evaluative measure. This section elaborates on the components of MV2H, the method of calculating the metric, and the advantages it offers over traditional evaluation metrics.

The goal of the MV2H metric is to evaluate AMT systems comprehensively across five key aspects: *multi-pitch detection*, *voice separation*, *metrical alignment*, *note value detection* and *harmonic analysis*. These components represent the essential features of music transcription, including pitch accuracy, rhythm, note durations, and harmony.

One of key assumptions of the MV2H metric is its *disjoint penalty principle*, which prevents cascading penalties across multiple transcription aspects due to a single mistake.

The metric is the main evaluation tool used for the main model quality assessment.

3.4.1. Components of the MV2H Metric

The MV2H metric is composed of five submetrics, each targeting a distinct transcription task. These metrics are calculated independently and combined to form the overall MV2H score, which, as previously, is the arithmetic mean of all submetrics.

Below, we describe each component and how it is calculated.

3.4.1.1. Multi-pitch Detection

The multi-pitch detection submetric evaluates how well the AMT system detects the pitch and onset time of notes. A transcribed note is considered correctly detected if it satisfies two criteria:

1. The onset time must be within a small threshold (50 ms by default) of the ground truth onset time.
2. The pitch of the transcribed note must match the corresponding pitch in the ground truth.

Other detected notes are false positives, and unmatched ground-truth notes are false negatives. Note offsets are ignored. Then, the multi-pitch detection metric is defined as the F -measure.

Notes that are not correctly detected in the multi-pitch detection stage are excluded from subsequent evaluations, such as voice separation or metrical alignment. This ensures that the multi-pitch detection errors are taken into account only once.

A serious drawback of this approach is inability to compare a performance MIDI with a sheet that contains idealized note onsets. In the subsequent version of the metric [McL19], the authors enhanced the metric to handle non-aligned musical scores, using a variant of *Dynamic Time Warping* (DTW), a dynamic programming algorithm introduced in 1978 by Sakoe et al. [SC78] to compare two time-dependent sequences that may vary in speed or length.

3.4.1.2. Voice Separation

Voice separation evaluates how well an AMT system groups notes into coherent musical voices. This aspect is crucial for polyphonic music, where multiple voices may play simultaneously, such as in fugues or piano pieces.

Voice separation is evaluated by comparing the system’s transcribed voices to the ground-truth voices in the score. A note is considered correctly placed if it is assigned to the correct voice in addition to its onset time and pitch being correct. The same F -measure used for multi-pitch detection is employed here, but only for notes that have already been correctly matched in terms of pitch and onset. This prevents over-penalization due to errors in the multi-pitch detection stage, aligning with the disjoint penalty principle.

3.4.1.3. Metrical Alignment

Metrical alignment measures how well the transcribed notes align rhythmically with the ground truth. Here, beat groupings (sub-beat, beat, and bar level) are evaluated to determine whether the system correctly identifies the temporal structure of the piece.

To calculate the metrical alignment score, the system compares the onset and offset times of transcribed notes against those in the ground truth. A note is considered correctly aligned if its onset and offset times fall within a small threshold (50 ms by default) of the expected beat location. The metrical alignment is thus based on comparing the actual onset positions between a transcription and the corresponding ground-truth notation.

3.4.1.4. Note Value Detection

Note value detection evaluates the accuracy with which an AMT system transcribes the duration of each note. Note durations, referred to as note values in musical terminology,

are normalized to account for tempo variations. This ensures that a note played in a faster tempo is still transcribed correctly relative to the piece’s overall structure. For example, a quarter note at 60 BPM should be equivalent in duration to a half note at 120 BPM⁶.

Only a subset of correctly identified notes is subject to the metric. A note is included in the analysis, if corresponds with a true positive ground-truth:

1. multi-pitch detection
2. voice segment

For each correctly matched note (in terms of pitch and onset), its duration is compared to the ground truth. The error in note duration is computed, with a score of 1 indicating a perfect match and a score of 0 for complete mismatch. For a single transcription note and its length d_{det} and the corresponding ground-truth length d_{gt} , the individual note score is:

$$\text{score} = \max \left(0, 1 - \frac{|d_{\text{gt}} - d_{\text{det}}|}{d_{\text{gt}}} \right)$$

The overall note value detection score for a transcription is computed as the arithmetic mean of individual note value scores, evaluated for all correctly matched notes.

3.4.1.5. Harmonic Analysis

Harmonic analysis assesses the system’s ability to correctly transcribe the harmonic structure of the music, which includes key and chord detection.

The metric is based on standard key detection evaluation [RMH⁺14], which measures how closely the transcribed key signature matches the ground truth, with partial credit given for related keys (e.g., relative or parallel major/minor of the correct key or key a fifth too high).

The standard chord tracking evaluation is *chord symbol recall* (CSR), described in detail in [Har10]. It can be defined as *proportion of the input for which the annotated chord matches the ground truth chord* [MS18].

The Harmonic Analysis metric is the arithmetic mean of key detection metric and CSR.

3.4.2. Disjoint Penalty Principle

The disjoint penalty principle is a core feature of MV2H, designed to prevent a single mistake from being penalized across multiple components of the metric. For example, if a note is missed during multi-pitch detection, it is excluded from the voice separation, metrical alignment, and note value detection evaluations. This prevents errors from compounding and ensures that the system is evaluated fairly across all aspects.

⁶Unlike the previous set of metrics and the RCC component, the tempo misalignment is not considered here as an error.

The principle is applied by first evaluating multi-pitch detection and removing any unmatched notes from further evaluations. The remaining correctly transcribed notes are then used for evaluating voice separation, metrical alignment, note value detection, and harmonic analysis. Subsequently, note value detection is performed only on correctly metrically aligned notes.

3.4.3. Limitations

The metric assumes linear, measure-by-measure alignment with limited ability to handle structural variations such as repeats, cuts, or large-scale rearrangements of musical sections.

It is also highly dependent of the multi-pitch detection as the subsequent components rely on the result of the alignment. The design of the metric prevents a cascade of a single error, however since the later submetrics operate on filtered data, the result may be biased.

Additionally, the harmony submetric implicitly assumes tonal framework. This is a non-issue in the case of classical music datasets, however this limitation should be acknowledged.

Finally, to the author’s knowledge, the MV2H metric has not yet been rigorously evaluated through human expert assessment, and the relevance and weight of each submetric remain to be validated against expert judgment.

3.5. Score Similarity

Most methods of score comparison operate at the level of MIDI files. As previously discussed, several limitations in the MIDI format complicate its use as a reliable representation of scores. These limitations include the absence of key and time signatures, pitch spelling, clefs, dynamics, and articulation markings. Such issues are sometimes addressed by adding supplementary information as annotations, either embedded as meta messages within the MIDI file or in external files.

In 2017, Cogliati et al. [CD17] proposed a metric that directly compares scores in the MusicXML format, allowing for the evaluation of musical elements not typically encoded in MIDI. The *Score Similarity* metric is calculated in two stages: note alignment and musical element comparison.

3.5.1. Note Alignment

To prepare for comparison, the transcription is aligned to a ground-truth reference based solely on pitch content, disregarding all other musical elements at this stage.

The alignment process employs *Dynamic Time Warping*, similarly to the revised MV2H metric for non-aligned transcriptions. In this application, DTW minimizes mismatched pitches, irrespective of duration, pitch spelling, or staff position [CTD16, p. 409]. This

ensures that transcription musical elements are being properly compared to the ground-truth counterparts, even if the transcription meter alignment is incorrect.

3.5.2. Comparison

Once the transcription is aligned with the reference, musical objects inside each portion of the two scores are grouped into sets. These sets are then compared according to several categories:

- barlines
- clefs
- key signatures
- time signatures
- rests
- notes (with grouping)

For barlines, clefs, key signature and time signatures binary matching is used. Rests are evaluated not only for duration, but also staff assignment. Missing or extra rests are considered as errors. Notes are matched for the following qualities: spelling, duration, stem direction, staff assignment, and grouping into chords. Incorrect note spelling (of a correct pitch) counts as an error.

Each category is analyzed for discrepancies, and errors are recorded based on the alignment. Some errors, such as clefs and key signatures, are counted globally, while others, like notes and durations, are normalized by the total number of instances.

Not all musical errors are of equal importance. For that reason, each category is weighted according to its perceptual importance, based on a linear regression model trained to align with human ratings. Additional details on the weighting procedure are available in [CD17, p. 410–411].

3.5.3. Limitations

The metric is designed specifically for MusicXML⁷. This renders it incompatible with MIDI files, as MIDI lacks many of these musical components required for a complete comparison.

As in the case of the MV2H metric, the score similarity measure use of DTW, while effective for basic alignment, assumes linear relationships between elements in both sequences. This makes it less effective in cases of large-scale reordering or insertion/deletion of structural elements, such as repeats or cuts in the transcription.

⁷The metric code uses *music21* [CA10] as the backend, so it is possible to measure scores faithfully represented within this framework.

Finally, the human-rating alignment is fixed for all evaluations, which may be inadequate for certain musical styles or genres. The weighting parameters were calibrated using musical examples from a single source [KP94], making the metric particularly suited to classical piano compositions but potentially less applicable to other genres, such as jazz.

Chapter 4

Overview of Existing Methods for MIDI to Score Conversion

4.1. A Brief History

While the modern formulation of the problem appears to be present in the academic discourse since in 2016 [CTD16], the challenges with interpreting various elements of the performance MIDI have been recognized much earlier. Cambouropoulos [Cam00], for instance, explored necessary elements to process in order to translate a performance MIDI into a score.

Cogliati et al. present a brief but succinct history of the development of AMT systems focused on transcribing performance data into either musical scores or some other intermediate forms [CTD16]. Several key contributions include:

- Aforementioned Cambouropoulos [Cam00] identifies musical elements and tasks needed to transcribe a performance recording into a score.
- Takeda et al. [TSO⁺02] introduce the use of Hidden Markov Models (HMMs) for the automatic transcription of monophonic recordings.
- Karydis et al. [KNP⁺07] address the problem of assigning note clusters to separate voices, of which the hand separation is a subproblem.
- In 2005, Yang et al. [YCV05] propose a dynamic programming approach to solve beat quantization problem.
- Temperley provides a unified approach to solve three problems of a music transcription: metrical analysis, harmonic analysis, and stream segregation [Tem09].
- Grohganz et al. [GCM14] define the concepts of score-performed and score-informed MIDI files and propose a method for transforming the former to the latter.

After the formulation, the problem has continued to evolve, incorporating the advancements from machine learning. Some of the more recent advances include:

1. Nakamura et al. [NYD17] apply Markov Random Fields for piano transcription.
2. Liu et al. [LKMB22] use a deep-learning method involving convolutional neural networks and gated recurrent units, which is the main focus of the thesis.

4.2. Modern Formulation of the Problem

While complete systems for performance MIDI to score systems have been existed for a while¹, the particular implementations were not the subject of scientific publications until 2016, when Cogliati et al. defined a comprehensive formulation of the process of transcribing an unannotated and unquantized MIDI file into a score.

Earlier AMT systems focused on different aspects of the transforming a recording — either from an audio or a MIDI file — into some form of musical notation. Many of these systems recognized only low level of musical information [CTD16], usually concentrating on only one aspect of the transcription such as pitch detection or rhythm quantization. Some authors use the term *complete audio-to-score AMT* [FMR⁺20] or *complete music transcription* (CMT) [YB18] to distinguish more advanced systems from low-level ones.

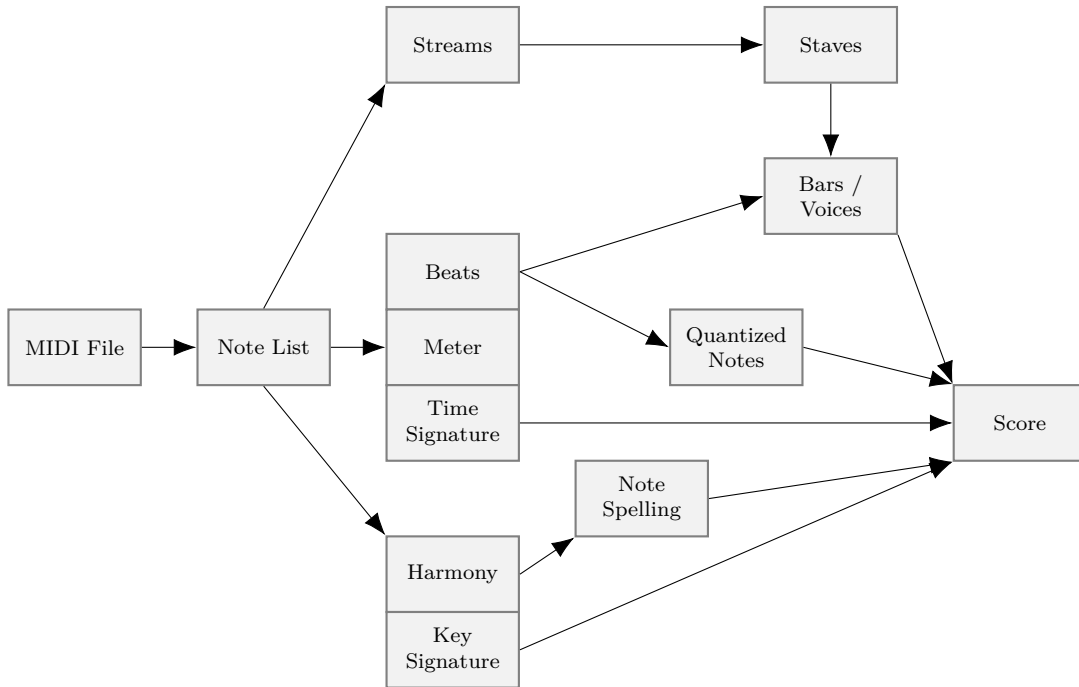


Figure 4.1: Performance MIDI to score system proposed by Cogliati et al. [CTD16].

The system proposed by Cogliati et al. (Figure 4.1) handles multiple aspects of the transcription process, taking into account not only beat quantization, but also imposes both key and time metric signatures. The former solves the note spelling problem (refer to Section

¹*MuseScore 3* has a special module for importing performance MIDI.

2.3.1.4) through key signature alignment.

In the following sections, we will describe some of the methods introduced in previous works that contributed to the development of this system.

4.3. Hidden Markov Model

The *Hidden Markov Model* (HMM) is a statistical framework that allows to estimate variables that are not directly observable (referred to as *hidden* variables) which are inferred through other dependent variables, known as *observations* [JM09, p. 210–213].

An HMM is specified by the following components:

- $Q = \{q_1, q_2, \dots, q_S\}$: A set of (hidden) *states*.
- $A = (a_{ij})_{i,j=1}^S$: A *transition probability matrix*, where each a_{ij} represents the probability of transitioning from state i to state j . Each row is a probability distribution, meaning $\sum_{j=1}^S a_{ij} = 1$ for each i .
- $X = (x_t)_{t=1}^T$: A sequence of *observations*, drawn from some random variable,
- $B = \{b_i(x_t)\}_{i=1}^S$: A sequence of *emission probabilities*, expressing the probability of an observation x_t being generated by state i .
- $\{\pi_i\}_{i=1}^S$: An *initial distribution over states*.

The HMM framework operates on two fundamental assumptions for the model:

- **Markov Assumption:** The probability of being in a particular state depends only on the previous state:

$$P(q_i | q_1, \dots, q_{i-1}) = P(q_i | q_{i-1})$$

- **Output Independence:** The probability of an output observation o_i depends only on the state that generated the observation q_i :

$$P(x_i | q_1, \dots, q_i, \dots, q_T, x_1, \dots, x_i, \dots, x_T) = P(x_i | q_i)$$

HMMs are said to be characterized by *three fundamental problems* [JM09, p. 213]:

1. **Likelihood:** Given an HMM $\lambda = (A, B)$ and observation sequence X , determine the likelihood $P(X | \lambda)$
2. **Decoding:** Given an observation sequence X and an HMM $\lambda = (A, B)$ determine the optimal state sequence X
3. **Learning:** Given an observation sequence X and the set of states in the HMM, learn the model parameters A and B .

These problems can be solved algorithmically. For further details, see [JM09, p. 213–226].

4.3.1. Hidden Markov Model for Beat Quantization

Takeda et al. [TSO⁺02] employed HMMs to solve the beat quantization problem for performance MIDI streams in a tempo-free scenario.

The core idea behind the approach is to treat performance note durations as *observations* of some unknown, “ideal” (nominal, intended time value). The fluctuations in actual note lengths can be modeled by some Gaussian distribution with the mean of the length of the nominal note duration. The performed note duration x_t at time t is represented by a probability density function $b_i(x_t)$ where i denotes the *intention*, i.e. nominal time value of the note. The observations are measured via *inter-onset intervals* (IOI), which are the time intervals between consecutive note onsets.

Let $Q = \{q_1, q_2, \dots, q_N\}$ be the time sequence of identified intentions, and let $X = \{x_1, x_2, \dots, x_N\}$ be the observations: the sequence of actual durations (observations). The probability of observing the entire sequence is given by:

$$P(X|Q) = \prod_{t=1}^N b_{q_t}(x_t)$$

The authors introduce two models for predicting note lengths:

- *Note n-gram Model.* Note length is predicted based on preceding $n - 1$ notes. For example, in a bigram model, the length of note is predicted base on the previous note’s length.
- *Rhythm Vocabulary Model.* The model consists of predefined rhythmic patterns for a unit time. The collection of all patterns makes a *vocabulary*.

Both models are represented as probabilistic state transition networks. Each state is associated with an intended note length. For example, in a 3-gram model we may represent a state as $a_{ij,k}$, where two preceding notes are of length i and j , and the succeeding note length is k . In general, instead of using tuples, all possible stats are labeled and enumerated, yielding transition matrix $(a_{ij})_{i,j}$. Thus, the probability that state number changes along a time sequence $Q = \{q_1, q_2, \dots, q_N\}$ is given by the product:

$$P(Q) = \pi_{q_0} \prod_{t=1}^N a_{q_{t-1}, q_t}$$

for some initial probability π_i starting with a state i . This the foundation of Hidden Markov Model:

- The matrix $A = (a_{ij})_{i,j}$ constitutes a *transition probability matrix*.
- The sequence $B = \{b_i(x_t)\}_i$ is a sequence of *emission probabilities*.

4.3.2. Optimal Sequence of States

From the Bayes theorem, one have:

$$P(Q|X) = \frac{P(X|Q)P(Q)}{P(X)}$$

To maximize a posteriori probability $P(Q|X)$, one needs to maximize $P(Q|X)$ as for given sequence X , the probability $P(X)$ is constant. Although it is theoretically possible to examine all possible state sequences and compute the likelihood for each, the number of possible sequences grows exponentially, rendering this approach infeasible.

Instead, the optimal sequence of states is efficiently found using the *Viterbi algorithm*, which is an established dynamic programming algorithm [JM09, p.210–220].

Given the HMM $\lambda = (A, B)$, the procedure is as follows:

Algorithm 4.1 Viterbi algorithm.

Require: the HMM model, the sequence of observations $X = (x_t)_{t=1}^N$

Ensure: the optimal sequence of hidden states $Q = (q_1, \dots, q_N)$

Initialization

- 1: initialize the matrices v and bp of the size $S \times N$
- 2: **for each** state i **from** 1 **to** S **do**
- 3: $v_1(i) \leftarrow \pi_i \cdot b_i(x_1)$
- 4: $bp_1(j) \leftarrow 0$
- 5: **end for**

Recursion

- 6: **for each** time step **from** $t = 2$ **to** N **do**
- 7: **for each** state **from** $j = 1$ **to** S **do**
- 8: $v_i(t) \leftarrow \max_{j=1}^S v_j(t-1) \cdot a_{ji} \cdot b_i(x_t)$
- 9: $bp_i(t) \leftarrow \arg \max_{j=1}^S v_j(t-1) \cdot a_{ji} \cdot b_i(x_t)$
- 10: **end for**
- 11: **end for**

Termination

- 12: $p \leftarrow \max_{j=1}^S v_j(N)$ \triangleright best path probability
 - 13: $q \leftarrow \arg \max_{j=1}^S v_j(N)$ \triangleright best path pointer
 - 14: best path \leftarrow the path starting at the state q , that follows the backpointer bp to states back in time
 - 15: **return** best path, p
-

The procedure returns the optimal sequence of intended notes:

$$\hat{Q} = \arg \max_Q P(X|Q)P(Q)$$

4.3.3. Training

Hidden Markov Models are subject to training in order to obtain transition matrix A and the emission probabilities B . More specifically, given an observation sequence X and the corresponding sequence of hidden states Q , the model parameters A and B are inferred from training data. The *Baum-Welch algorithm* is commonly used for this purpose [JM09, p. 220–226].

Takeda et al. used approximately 50 000 notes in MIDI data of classical and jazz music to train the n -gram model. For the rhythm vocabulary model, they extracted 267 one-bar-long rhythm patterns from 88 various pieces. The exact details of the dataset, however, were not provided.

4.3.4. Limitations

The model focuses solely on beat quantization problem, it gives no insight about other music aspects as key signature. It however, partially deals with time signature: for each key signature there is a separate model.

As the Gaussian distribution is unimodal, the model assumes that the tempo is constant. To account for fluctuating tempi, multiple models are run in parallel, and the one that maximizes the likelihood for the observed note durations is selected. The model typically uses a limited, discrete set of predefined tempi².

The algorithm was designed for single-voice music transcriptions. The authors suggest that with enriched rhythm vocabulary containing overlapping notes (as in fugues or canons) it is possible to use the framework on polyphonic music, however it would require chord identification to group notes played at almost the same time into a single entity. The paper does not provide more details on such procedure.

No public implementation of the algorithm is available, making it difficult to compare its performance with modern solutions.

4.3.5. Polyphonic Extension

A polyphonic extension of the method can be found in [NYS17]. The method extends the HMM framework for polyphonic music.

4.4. Dynamic Programming Approach

In 2005, Yang et al. proposed another method for solving beat quantization problem [YCV05]. The authors used dynamical programming approach to segment a piece into sections with

²The authors used a set of six equally spaced tempi, ranging from 60 to 120 beats per minute.

different *tatums*. A *tatum* is often defined as “the smallest cognitively meaningful subdivision of the main beat”³ [IBWW97]. The resulted tatums form a grid to which notes are spanned for quantization.

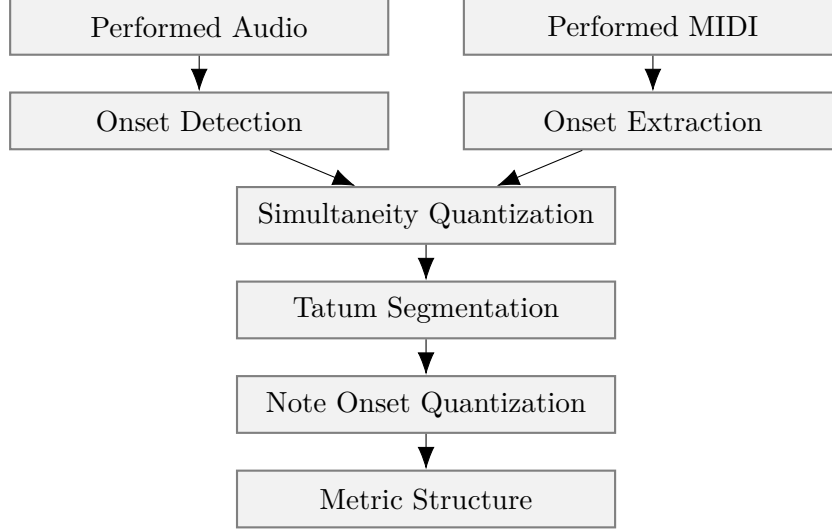


Figure 4.2: System Diagram [YCV05]. The algorithm from the paper handles in fact three steps: *Simultaneity Quantization*, *Tatum Segmentation* and *Note Onset Quantization*. Imposing the metric structure of the transcribed piece is not a goal of the algorithm.

4.4.1. Simultaneity Quantization

As the name suggests, *Simultaneity Quantization* is a preprocessing step that identifies events played almost simultaneously, such as chords. The procedure consolidates events within a specified time window by averaging their onset times into a single group.

The size of the window needs to be carefully adjusted: too small window may not correctly recognize all simultaneous entities but too large window may collapse musical ornaments such as thrills or grace notes. For some pieces, there is no viable fixed threshold that allows discerning fast arpeggiated chords from thrills: some chords that should be treated as one entity may be played in fact slower than some ornaments that needs to be separated.

4.4.2. Tatum Segmentation

The central component of the algorithm is *Tatum Segmentation* for finding an optimal metric grid that minimizes the tatum-to-note error. The algorithm takes only onsets as inputs $\mathcal{O} = \mathcal{O}_1^n = \{o_i\}_{i=1}^n$ and disregards note-off events⁴. The output of the algorithm is a sequence of segmentation points $S = \{S_j\}_{j=1}^m$ within the optimal tatums $\{T_j\}_{j=1}^m$.

³Usually coincides with sixteenth-, twenty-fourth- or thirty-second notes.

⁴This means that inter-onset intervals (IOI) are used for beat quantization.

The authors define recursively the *optimal segmentation function* as:

$$\text{OPT}(k) = \min_{1 \leq i \leq k} \left[\text{OPT}(i) + \text{ERR} \left(\mathcal{O}_{i+1}^k \right) \right]$$

where $\mathcal{O}_{i+1}^k = \{o_{i+1}, \dots, o_k\}$. The function ERR describes the error obtained by the best tatum assignment for the sequence of onsets \mathcal{O}_{i+1}^k :

$$\text{ERR} \left(\mathcal{O}_{i+1}^k \right) = \min_p e(p, i+1, k) = \min_p \sum_{j=1}^{k-i-1} d_{\mathbb{Z}} \left(\frac{o_{j+1} - o_j}{p} \right)^2$$

where $d_{\mathbb{Z}}(x)$ is the distance of x to the nearest integer, defined as $\left| x - \left\lfloor x + \frac{1}{2} \right\rfloor \right|$. The error function represents the remainder squared error (RSE), and the expression $o_{j+1} - o_j$ is the IOI between onsets j and $j+1$. In other words, the error quantifies the total error of snapping onsets to the grid. An observant reader may have noticed that the optimal tatum segmentation minimizes the RSE of the entire piece, that is $\text{ERR}(\mathcal{O})$.

Let $S(k)$ be the *segmentation boundary*: an argument that minimizes the cost function:

$$S(k) = \arg \min_{1 \leq i \leq k} \left[\text{OPT}(i) + \text{ERR} \left(\mathcal{O}_{i+1}^k \right) \right]$$

Similarly, let the best *tatum level* minimizing $\text{ERR} \left(\mathcal{O}_{i+1}^k \right)$ be:

$$T(k) = \arg \min_p e(p, i+1, k)$$

With the notation settled down, we can outline the main algorithm consisted of two parts: the main procedure, which determines the optimal tatums, and the trace-back that finds the segmentation boundaries. The entire algorithm has time complexity of $O(n^3)$.

4.4.3. Note Onset Quantization

Once the segmentation is complete, the onsets are snapped to the obtained grid for each piece segment. The notes are snapped from left to right, and the first note in a segment determines the first grid point.

4.4.4. Limitations

There are several limitations of the proposed solution. Similarly to the previous algorithm, it addresses only the beat quantization. Moreover, it does not recognize metric structure of a performed piece.

The presented dynamic programming approach does not handle tempo variations inside a single tatum segment. This means that it is not able to capture *rubato* or other expres-

Algorithm 4.2 Dynamic programming tatum segmentation.

Require: list of onsets \mathcal{O}_1^n

Ensure: the segmentation boundaries $\{S_j\}_{j=1}^m$ and optimal tatum levels $\{T_j\}_{j=1}^m$

Main procedure

- 1: initialize tables $S[\cdot]$ and $T[\cdot]$ from 1 to n
- 2: **for** $k = 1$ **to** n **do**
- 3: $S[k] \leftarrow \arg \min_{1 \leq i \leq k} \text{OPT}(i) + \text{ERR}(\mathcal{O}_{i+1}^k)$
- 4: $\text{OPT}(k) \leftarrow \text{OPT}(S_k) + \text{ERR}(\mathcal{O}_{S_k+1}^k)$
- 5: $T[S_k] \leftarrow \arg \min_p e(p, S_k + 1, k)$
- 6: **end for**

Trace-back

- 7: initialize sequences S and T
 - 8: $k \leftarrow n$
 - 9: $j \leftarrow 0$
 - 10: **while** $k > 0$ **do**
 - 11: $j \leftarrow j + 1$
 - 12: $S_j \leftarrow S[k]$
 - 13: $T_j \leftarrow T[k]$
 - 14: $k \leftarrow S_k$ ▷ move to the previous segmentation point
 - 15: **end while**
 - 16: **return** reversed $\{S_j\}_{j=1}^m$
-

sive timing changes. This may result in uniform, inflexible sections or distorted time grids. Furthermore, some notes may be misaligned due to snapping to the grid.

This comes closely with the next assumption of the algorithm: it minimizes RSE for each segment, but the underlying assumption is that the notes are supposed to be played uniformly in each segment, which is not always the case. Henceforth, the algorithm is not able to handle other expressive techniques as gradual tempo increase (*accelerando*) or decrease (*ritardando*).

Finally, the algorithm is computationally expensive and not suitable for real-time performance.

Chapter 5

Performance MIDI-to-Score Conversion by Neural Beat Tracking

The main model of interest [LKMB22], developed by Liu et al. (2022), is a state-of-the-art machine learning model for complete music transcription. It has been awarded as the best paper of the *International Society for Music Information Retrieval Conference* (ISMIR) in December 2022.

The model is composed of two parts:

- Beat Tracking.
- Score Element Assignment.

The first part is responsible for quantizing the raw material into beats and downbeats. Each measure is consisted of several beats, and the number of beats in a measure is governed by the time signature. The beats mark the tempo, and they are the basis for the musical onsets detection.

The second part assigns every other aspect of the musical score to the MIDI stream. Let us recall these elements: musical onsets, note values, hand parts, key signature and time signature.

Let us review both components.

5.1. Beat Tracking

A single measure consists of several beats, and the rhythmic structure of beats is determined by time signature (or signatures). A performance MIDI does not contain information about beats and one of the objectives of the model is to predict locations of these.

After these predictions, note beginnings (onsets) can be placed in a subdivision of a single beat. The smallest unit of time distance is thus dictated by the size of the beat subdivision. More precisely, a musical onset mo_i of the i -th note is defined as

$$mo_i = \frac{s_n}{S}$$

where S is the number of subdivisions per beat in rhythm quantization, and s_n is the musical onset time expressed in the number of subdivisions.

The authors provide a novel approach to detecting beats, splitting them into two categories, for which there are two separate methods.

Beats can fall into two distinct groups:

- *In-note*: beats concurrent with at least one note onset. This means that some notes play within the start of the beat.
- *Out-of-note*: a beat is not concurrent with any note onset. No notes are beginning with such beats.

5.1.1. In-note Beat Prediction

For predicting in-note beats, one can reduce the problem to a binary classification task on the note sequence \mathbf{X} . This sequence is assumed to have a length N .

The probability P_n that n -th note is concurrent with a beat is defined as:

$$P_n = \mathbb{P}(B_n|\mathbf{X})$$

where $B_n \in \{0, 1\}$ is the ground-truth beat label for the n -th note from the note sequence. The model is trained using the standard cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^N B_n \ln P_n + (1 - B_n) \ln (1 - P_n)$$

The authors used CRNN with 3 convolutional layers and 2 bidirectional gated recurrent unit (GRU) layers. The probability threshold for positive classification has been set dynamically, depending on the maximum probability in a fixed segment length.

5.1.2. Out-of-note Beat Prediction

The approach to predicting out-of-note beats, which do not align with note onsets, requires distinct approach. Liu et al. (2022) proposed a dynamic programming strategy to solve this problem [LKMB22].

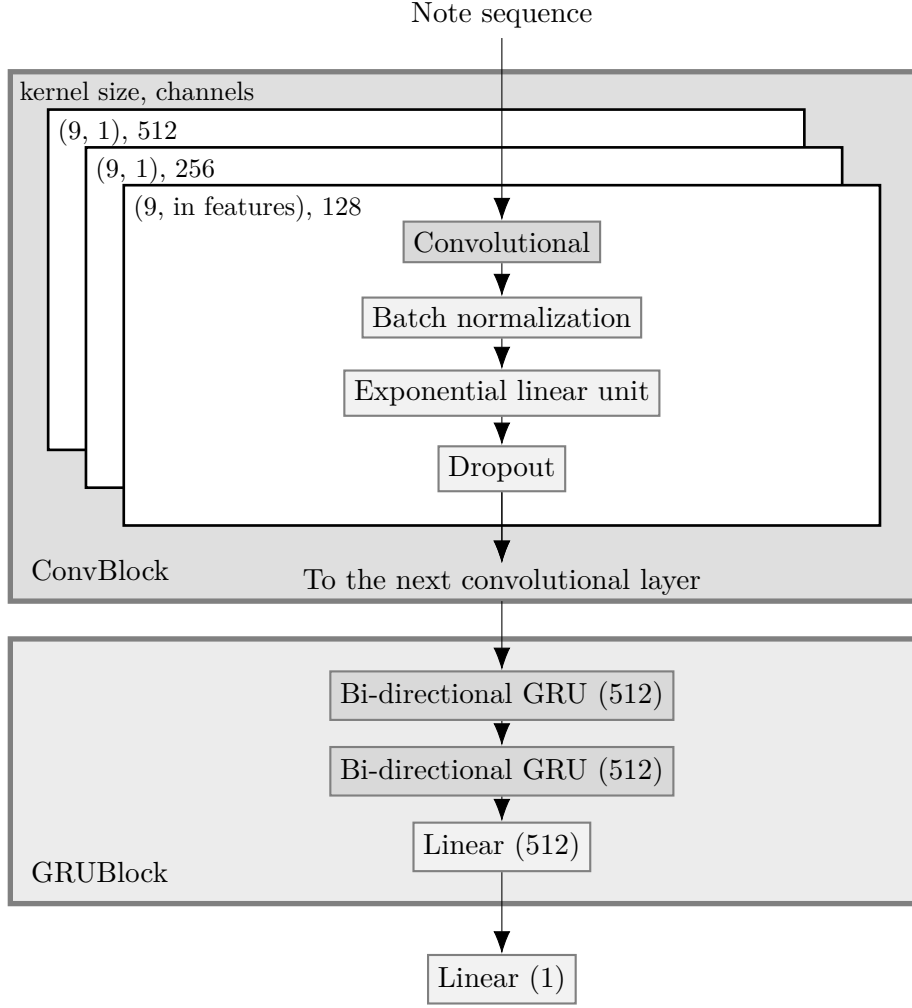


Figure 5.1: In-note beat prediction model with a CRNN with 3 convolutional layers and 2 bi-directional gated recurrent unit (GRU) layers.

Let us assume that there are B^i in-note beats $\{b_n^i\}_{n=1}^{B^i}$ in total, and out-of-note beats are at subdivisions of the neighboring in-note beats¹ b_n^i and b_{n+1}^i .

The goal of the procedure is to find out-of-note beats b^o from a set of candidates:

$$b_{n,K}^o = \left\{ b_n^i + \frac{k}{K+1} (b_{n+1}^i - b_n^i) \right\}_{k=1}^K \quad (5.1)$$

where $K \in \{0, 1, 2, 3\}$ is the number of out-of-note beats to insert inside the (b_{n+1}^i, b_n^i) interval.

The number of candidates is selected in order to minimize the tempo change after adding out-of-note beats. The function may be represented as the sum:

$$\mathcal{O}_1 = \sum_{n=1}^{B-2} \left| \ln \left(\frac{b_{n+2} - b_{n+1}}{b_{n+1} - b_n} \right) \right|$$

¹We may select only one note per in-note beat, if there are more.

where $\{b_n\}_{n=1}^B$ is the sequence of all B beats (both in-note and out-of-note, sorted chronologically).

To discourage the procedure from adding too many out-of-note beats, which leads to an unnecessarily subdivided output, an additional penalty is associated with the objective function:

$$\mathcal{O} = \mathcal{O}_1 + \lambda B^o \quad (5.2)$$

where B^o is the number of added out-of-note beats, and λ is the penalty coefficient. The coefficient is to be found experimentally, however the default value set by the authors is 1.

Algorithm 5.1 Out-of-note beat prediction.

Require: list of in-note notes \mathcal{B}^i

Ensure: list of all beats \mathcal{B} including added out-of-note beats

```

1:  $n \leftarrow 1$ 
2: for  $K = 0, 1, 2, 3$  do
3:   initialize objective function  $\mathcal{O}_K \leftarrow 0$ 
4:   initialize beat sequence  $\mathcal{B}_K \leftarrow \{b_1\}$ 
5: end for
6: for  $n = 1, 2, \dots, B^i - 2$  do
7:   for  $K_{\text{cur}} = 0, 1, 2, 3$  do
8:     get out-of-note beats for current step by (5.1)
9:     if tempo is beyond tempo range limits then
10:      continue
11:    end if
12:    for  $K_{\text{prev}} = 0, 1, 2, 3$  do
13:      update objective function by (5.2)
14:    end for
15:    select the minimum objective among all  $K_{\text{prev}}$  values
16:    add out-of-beats for the current step to the beat sequence mapped to the selected
     $K_{\text{prev}}$ 
17:  end for
18:  for  $K_{\text{cur}} = 0, 1, 2, 3$  do
19:    update  $\mathcal{O}_K, \mathcal{B}_K$  mapped with  $K_{\text{cur}}$ 
20:  end for
21: end for
22: return the beat sequence  $\mathcal{B}_K$  with the minimum objective function  $\mathcal{O}_K$ 

```

5.2. Input Data Encoding

The entire score needs to be transformed to a suitable data format before passing to any of the model components. Given a note sequence tensor as described earlier, the data is encoded into a variety of features:

- 128-dimensional one-hot encoding of MIDI pitches.

- One-hot onset time-shift ($o_i - o_{i-1}$) quantised by 10 ms resolution, with maximum value of 4 s, 401 features in total.
- Raw duration values in seconds.
- Velocities normalized to the unit interval $[0, 1]$.

There are 531 features in total.

The authors tried different encoding schemes, for pitches, onset times and durations, including raw float values in seconds. The full study is available in the paper [LKMB22].

5.3. Score Elements Assignment

As discussed in the Section 2.2.4.2, score elements assignment may be viewed as a function from a note sequence \mathbf{X} into a score encoded by a tensor \mathbf{Y}_n representing musical onsets, note values, key signature, time signature, and hand parts, for each note separately.

Besides the quantization model, which relies on the beat tracking module, key signature, time signature and hand parts modules can be treated independently. **The quantization model is not used for the final MIDI creation, and we won't focus on that part in much detail.**

The initial version of time signature model handled a finite set of time signature numerators and denominators, as described in Section 2.2.4.2, changing over time. Later, after the paper release, the model has been reduced to recognize only two the most common time signatures: $\frac{4}{4}$ and $\frac{3}{4}$, and only for the entire song. The architecture has been simplified to only convolutional layers, with the GRUBlock removed. With an additional support of beat quantizer, additional two time signatures $\frac{2}{4}$ and $\frac{6}{8}$ could be recognized. For performance analysis, we use the latter approach. Though, the role of the time signature model is rather supplementary: in principle, the time signature could be inferred from the beats and downbeats arrangement, obtained by the beat model.

Instead of direct tempo estimation, for each note an *inter-beat-interval* time is estimated using classification, capturing time intervals between successive beats with 10 ms resolution, up to 4 seconds.

5.4. Score Generation

The final score generation process synthesizes the source MIDI stream, incorporating pitch sequence, with the model's output. The resultant annotated MIDI encompasses essential information for visual score generation, including key/time signatures, which are not typically present in standard MIDI files.

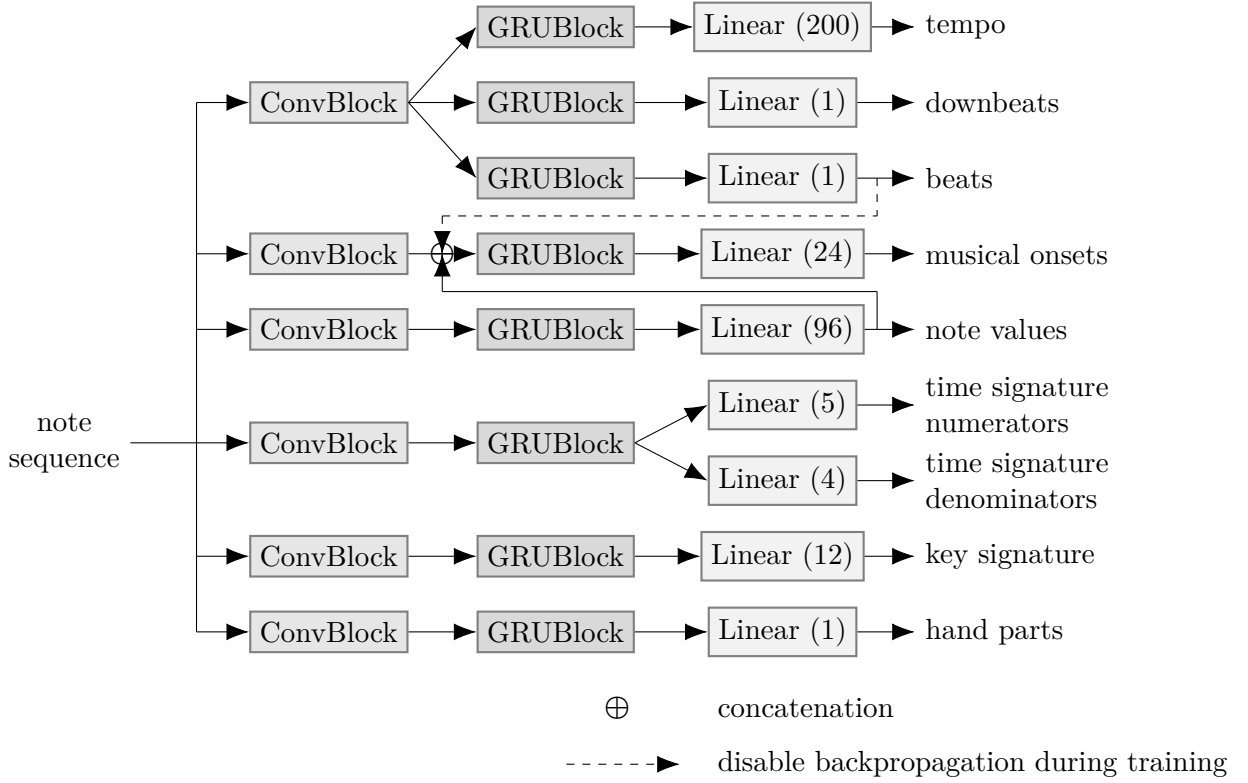


Figure 5.2: The initial architecture of the model proposed in [LKMB22]. There are five separate modules of the entire model in total: *beat*, *quantization*, *time signature*, *key signature* and *hand parts*.

5.5. Training and Evaluation

The training and evaluation procedures were implemented to replicate those described by the original authors. In the following sections, we provide a detailed overview of the datasets, augmentation techniques, and specific configurations for training and evaluation setups.

5.5.1. Datasets

The dataset used by the authors of the paper [LKMB22] integrates three sources of MIDI files:

- The *Classical Piano MIDI* (CPM) database [Kru96].
- The *Augmented MIDI Aligned Piano Sounds* (A-MAPS) [YB18].
- The *Aligned Scores and Performances* (ASAP) dataset [FMR⁺20].

The datasets consists of a variety of classical piano pieces by composers including as Bach, Mozart, Beethoven, Schubert, Chopin, Liszt, and others from the Western European classical repertoire.

Certain musical features, such as time and key signatures, are not always encoded in MIDI files. Consequently, the MIDI files have been annotated using different strategies.

Statistics	Train	Valid	Test	Total
Distinct pieces	426	49	29	504
Performances	1324	157	29	1510
Duration (hours)	108.3	12.7	2.2	123.2
Notes (10^3)	3984.0	517.6	73.2	4574.7

Table 5.1: Statistics of the dataset used for training [LKMB22]. Performances of the same piece are counted only once.

The datasets partially overlap; for instance, A-MAPS is derived in part from CPM. Authors avoid using the same pieces in train, validation and test sets by using distinct music piece labels coming from different datasets.

Dataset	Items	Annotation structure
A-MAPS	266	external TSV files
ASAP	1586	embedded in MIDI
CPM	337	embedded in MIDI

Table 5.2: Annotation structure for the subdatasets.

All dataset are jointly combined into one dataset, *A Dataset of Aligned Classical Piano Audio and Scores* (ACPAS) [LMB21].

Below is a brief overview of each data source.

5.5.1.1. Classical Piano MIDI Database

The *Classical Piano MIDI* (CPM) database was created by Bernd Krüger, who produced hundreds of MIDI files containing interpretations of classical piano works. Krüger describes his motivation as follows [Kru96]:

The page serves to describe and make available my interpretations of classical piano works. Although I am a layman in terms of music, I have set myself the goal of painstakingly interpreting difficult works. I would like to make these works accessible to as many musically interested people as possible.

The dataset consists 337 pieces with a cumulative duration of approximately 23 hours. All MIDI files are score-informed, with separate tracks for the left and right hands. and time signatures are encoded in the MIDI files as meta messages.

However, despite the fact that the MIDI files are tempo-varied, they were manually crafted, not performed. For instance, note onsets which lie on the same beat, occur si-

multaneously. In musical performance, even chords are played with certain time variation. Some authors disregard the source as not reliable. For example, Beyer and Dai claim [BD24]:

The underlying data is score-derived and has been manually adjusted to represent aspects of performance and score at the same time. This leads to information leakage as highly regular onset/offset alignment remains in the *performance* data.

5.5.1.2. Augmented MIDI Aligned Piano Sounds

The *Augmented MIDI Aligned Piano Sounds* (A-MAPS) dataset builds on the original *MIDI Aligned Piano Sounds* (MAPS) dataset, introduced by Emiya et al. [VBDB10]. MAPS contains approximately 65 hours of data, including both MIDI and corresponding audio recordings, and is divided into four main categories:

- **ISOL**: Isolated notes and monophonic excerpts.
- **RAND**: Chords with random pitch notes.
- **UCHO**: Usual chords from Western music.
- **MUS**: Piano music pieces, sourced from the *Classical Piano MIDI* database.

MAPS has been widely used as a benchmark for AMT systems [YB18]. However, the information provided by the MIDI files is limited to basic attributes: pitch, onsets and offsets in seconds, and velocity. The A-MAPS dataset enriches the original with additional annotations, including meter, note values, key signatures and hand separation. Moreover, A-MAPS provides also tempo curves and sustain pedal activations, although this information is not used by the considered model.

The A-MAPS dataset contains 269 files of 159 unique pieces.

5.5.1.3. Aligned Scores and Performances

The *Aligned Scores and Performances* (ASAP) dataset consists of 222 musical score aligned with 1068 performances, of the total duration of 92 hours [FMR⁺20].

The ground truth is provided twofold: as musical scores in MusicXML format, and quantized MIDI files.

For each performance, including the quantized MIDI file, annotations are provided in tab-separated values (TSV) format. These annotation specify timestamps for:

- Beats (b) and downbeats (db).
- Time signature changes.
- Key signature changes.

Timestamp	Timestamp	Annotation
2.845313	2.845313	db, 6/8, 2
3.446876	3.446876	b
3.858854	3.858854	db
4.218229	4.218229	b

Table 5.3: An example of a performance MIDI annotation in TSV format. The first row indicates an initial time signature of $\frac{6}{8}$. The piece begins in the key of D major, encoded as 2.

The MIDI are actual human performances recorded via digital instruments. The annotations for performance provide ground-truth labels for all considered models. Otherwise, one would need to compare all different interpretations to a single score, risking judging the performance’s fidelity to the original instead of transcription quality.

5.5.2. Data Augmentation

To increase model’s versatility, the authors considered four data augmentation techniques, that should not affect the transcription. The note sequence tensors have been transformed using the following methods:

- **Pitch Shift.** Transpose all notes by a constant pitch from the range of $\{-12, -11, \dots, 12\}$.
- **Tempo Change:** Change the tempo by multiplication by a constant from the range $[0.8, 1.2]$.
- **Note Removal.** For each group of m concurrent notes, a random number from 0 to $m - 1$ of notes is being removed.
- **Note Introduction.** For some notes a concurrent ones are being added, with the same velocity and duration as

Section 6.1.1 shows ablation studies for data augmentation, for the beat-tracking model.

5.5.3. Training and Evaluation

To reproduce the original results, we followed the authors’ training and evaluation setup, adhering to the same training, validation, and test dataset divisions. In the following sections, we provide a detailed examination of the training and evaluation pipeline, including feature preparation and the training and evaluation scheme.

5.5.3.1. Feature Preparation

As the data comes from different sources, they need to be unified before training.

Initially, all MIDI files were gathered from the datasets outlined in Section 5.5.1. Following the authors’ methodology, all pieces in the **ENSTDkC1** subset of the MAPS dataset were

designated as the test set. The remaining pieces were randomly allocated to the validation set, with 90% reserved for training². Refer to Table 5.1 for dataset division details.

The next step involved extracting note sequence tensors and additional features from MIDI files and, where relevant, from external annotations, as in the case of the ASAP dataset. The features encompassed all essential elements for transcription, including beats, downbeats, time signatures, key signatures, onsets, note durations, and hand part assignments.

Once processed, the datasets were ready for model training.

5.5.3.2. Training

Each model was trained with a maximum of 50 epochs. Training examples were augmented according to the procedures discussed in Section 5.5.2.

5.5.3.3. Evaluation

During training, validation metrics were calculated after each epoch. The metrics used varied by model: for binary classifiers (such as the time signature and hand part models), standard metrics (accuracy, precision, recall, and F_1 score) were employed. For the key signature model, which is a multiclass classifier, both micro and macro metrics were used (see Section 3.2).

Upon completion of training, the model with the highest F_1 score on the validation set was selected for further evaluation on the test set. A joint performance score across all models was then calculated using the MV2H metric on the test set.

5.6. Model Performance

We reproduced the results using the training and evaluation setup provided by the authors, described in the previous sections. We achieved a comparable MV2H metric scores on the test set as the paper’s authors.

5.6.1. Validation Metrics

Below, we present the validation metrics, including F_1 scores, for each submodel.

Metric	Validation	Test
Accuracy	0.955	0.934
Precision	0.849	0.917
Recall	0.851	0.900
F_1 score	0.847	0.908

Table 5.4: Validation metrics of the hand part model.

²Specifically, pieces with an ID divisible by 10 were assigned to the validation set.

Type	Metric	Validation	Test
Macro	Precision	0.741	0.806
	Recall	0.697	0.796
	F_1 score	0.701	0.792
Weighted	Precision	0.891	0.915
	Recall	0.785	0.915
	F_1 score	0.809	0.905

Table 5.5: Validation metrics of the time signature model.

Metric	Validation	Test
Accuracy	0.560	0.552
Precision	0.315	0.444
Recall	0.230	0.333
F_1 score	0.258	0.381

Table 5.6: Validation metrics of the key signature model.

Part	Metric	Validation	Test
Beat	Accuracy	0.880	0.869
	Precision	0.879	0.859
	Recall	0.873	0.884
	F_1 score	0.864	0.859
Downbeat	Accuracy	0.853	0.860
	Precision	0.622	0.665
	Recall	0.467	0.486
	F_1 score	0.508	0.545

Table 5.7: Validation metrics of the beat quantization model.

5.6.2. MV2H Metric

The MV2H metric results on the test set are presented below, alongside the original authors’ evaluation results. It should be noted that minor changes have been introduced since the release of the model, so the comparison is not exact.

We use the following notation for the MV2H components:

- Multi-pitch detection F_p
- Voice separation F_{vs}
- Metrical alignment F_{ma}
- Note value detection F_{nv}
- Harmonic analysis F_{ha}

F stands for the final MV2H metric, the average of all submetrics.

Model	F_p	F_{vs}	F_{ma}	F_{nv}	F_{ha}	F
Original	0.998	0.870	0.617	0.999	0.911	0.879
Reproduced	0.986	0.826	0.598	1.000	1.000	0.882

Table 5.8: MV2H metric evaluation on the test set, with results compared to the original model evaluation.

5.7. Case Studies

To evaluate the model’s performance, we conducted an analysis of specific outputs from the validation and test sets. This investigation aimed to assess how effectively the model handled the transcription task. Specifically, we sought to answer the following questions:

- What are the most common types of errors?
- How disruptive are these errors to the resulting score?
- What level of effort is required to repair a flawed transcription?

The model performed quite well on human-crafted MIDI songs. Transcriptions of live performances exhibited a greater prevalence of note metric alignment errors and, in some cases, were rendered unreadable due to an overabundance of inaccuracies.

For key or time signature misalignments, the corrections required were generally straightforward and demanded a single change in the transcription. Errors in key signature assignments were relatively rare, but the time signature model’s classifier constraints made it impossible to handle non-standard odd signatures.

The model occasionally struggled with establishing the correct metric grid, particularly for pieces with unusual or complex time signatures. These instances, while infrequent, significantly impacted the overall transcription quality. Section 5.7.3 provides an example of such a case.

Below, we outline three common errors observed in the transcriptions: a metrical shift, hand part misalignments and the most severe of them: meter misalignments.

5.7.1. Unrecognized Pickup Measures

One prevalent error type involves a systematic shift of the entire piece by a fraction of a measure. A special case of this error is the failure to recognize a pickup measure (an incomplete measure at the beginning of a piece).

This type of error usually requires some manual adjustments: shifting the entire piece and, probably, correcting note ties that extend across measure boundaries.



Figure 5.3: *Sevilla* by Isaac Albéniz: a) the piece with a distinguished pickup measure (first three notes for each hand), b) the model transcription. Since the model assumed the time signature of $\frac{3}{4}$, the resulting content is shifted by the half of a measure. Other musical elements have been correctly assigned.

5.7.2. Hand Part Misalignments

Hand part assignment errors, although not so common, do occur. These errors can be particularly problematic when a single note is misattributed to the wrong hand.

Some errors of this type are easily noticeable, while others are subtle and harder to detect. Even a single hand part misalignment can render a piece unplayable, requiring manual verification and correction. In some cases, the piece remains playable despite the error, but the transcription quality is slightly degraded.

5.7.3. Meter Misalignment

One illustrative example of meter misalignment is found in the transcription of Claude Debussy's *Clair de Lune*, a piece from the composer's *Suite bergamasque*. The original score employs a non-standard $\frac{9}{8}$ time signature in the key of $D\flat$.

In this case, the model incorrectly imposed a $\frac{4}{4}$ time signature, resulting in a mixture of quarter notes and triplets that do not align with the original meter. This error not only disrupts the metric structure but also breaks the temporal relationships between consecutive notes. Consequently, simply changing the time signature to $\frac{9}{8}$ is insufficient to resolve the issue. Correcting such a transcription requires substantial effort.

Errors of this type are particularly common in pieces with unusual time signatures. While the severity of the error varies depending on the extent of the misalignment, such issues often require significant manual intervention to repair. In many cases, these errors are confined to specific phrases or sections, but they are more prevalent in transcriptions of fast-paced live



Figure 5.4: The beginning of Claude Debussy’s *Clair de Lune*: a) the original score from the MAPS test dataset, b) the model transcription. The model misaligned the metric structure by mixing quarter notes with triplets, disrupting the inner relationships between notes within a measure. Some right-hand notes (e.g., C in the second measure) are misattributed. The key signature was assigned correctly.

performances.

5.8. Robustness Analysis

The quality of the model can be considered in many dimensions. Evaluating model quality extends beyond the accuracy of outputs relative to ground truth; it also includes assessing how robust the model is to transformations that should theoretically leave certain outputs unchanged.

For this analysis, we make the following assumptions:

- Note velocity should not influence time signature or hand part assignment.
- Note pitch should be irrelevant to time signature prediction.
- Note pitch is essential for determining key signature and hand part assignment, while other features should have minimal impact on these aspects.
- Note duration should not affect key signature assignment.

These are intended as general guidelines rather than strict rules. There may be situations where slight deviations occur. For instance, a chord played by one hand tends to be of uniform intensity but the second hand may play a note with markedly different articulation than the other. These nuances explain why certain features cannot be entirely disregarded across all cases. For a more in-depth analysis of feature importance, refer to Section 6.1.

A robust model should obey to these rules to some reasonable extent. For example, it is natural to expect the model to assign the same time signatures to a transposed version of a

piece, as a pitch shift should not affect the time signature.

We conducted an analysis of how the model is robust to transformation, for each of three components:

- The **hand part** f_H .
- The **key signature** model f_K .
- The **time signature** model f_T .

In the subsequent sections, we use models pretrained by the authors unless stated otherwise.

5.8.1. *Ceteris Paribus* Analysis

We conducted a *ceteris paribus* (Latin for *all other things being equal*) type of analysis by augmenting original note sequence tensors. This approach allowed us to compare model outputs for transformed note sequences, isolating and measuring the influence of specific features.

A natural way of augmenting data in this context is to randomly distort a single feature (e.g., pitch, duration, or velocity), while keeping all others intact.

5.8.1.1. Perturbations

For a sample $M = 50$ musical pieces from the dataset, we introduced perturbations to test the assumptions given in the Section 5.8:

- Changing velocity with standard deviations σ_v of 8, 16, 32, and 64, but with clipping to the range $[1, 127]$.
- Scaling note lengths by a factor from an interval $\alpha \in (\alpha_l, 1)$, where $\alpha_l \in \{0.9, 0.75, 0.5, 0.2\}$ (note extension may lead to overlapping).
- Changing pitch with standard deviations σ_p of 12 and 24 (whole octave, double octave). However, we ensure, that the entire piece stays in the playable range of pitches $[21, 108]$, corresponding to the range of a typical 88-key piano.

Each transformation was applied independently to each feature, note by note. Importantly, note onsets were not altered, as this would fundamentally change the structure of the music.

For each feature model f : *hand part* f_H , *key signature* f_K , *time signature* f_T , we calculated the average error between the output sequence and the output of a perturbed sequence.

For each element \mathbf{x}_i in the dataset sample $i \in \{1, 2, \dots, 50\}$ and for each change, we sampled $m = 10$ perturbations $\mathbf{x}^{(j)}$ of the original item and measured the error:

$$\text{error}(f) = \frac{1}{M} \sum_{i=1}^M \frac{1}{m} \sum_{j=1}^m \text{error}\left(f(\mathbf{x}_i), f(\mathbf{x}_i^{(j)})\right)$$

Here, error between two sequences $\mathbf{x} = (x_i)_{i=1}^N$ and $\mathbf{y} = (y_i)_{i=1}^N$ is defined as the mean number of indices for which these sequences differ:

$$\text{error}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N [x_i \neq y_i]$$

Higher error values indicate a greater sensitivity of the model to the applied transformation. Note that this analysis does not assess whether a change improves or worsens the model’s performance; it solely measures the impact of the transformation.

5.8.1.2. Results

The results of the perturbation experiments are summarized below.

The key signature model f_K demonstrated high robustness across all perturbations, with an average error of less than 1.5% in all categories.

The time signature model f_T was robust to pitch and velocity manipulations, with average errors below 13%. However, it exhibited sensitivity to changes in note durations, consistent with the role of duration in determining meter. See Table 5.9 for details.

Model	Velocity change σ_v				Duration change α_l				Pitch shift σ_p	
	8	16	32	64	0.9	0.75	0.50	0.20	12	24
f_H	7.61	15.65	25.69	34.80	0.29	0.76	1.67	3.08		
f_K	0.07	0.15	0.32	0.46	0.12	0.32	0.75	1.13		
f_T	1.05	2.10	3.78	9.77	3.42	10.10	25.21	39.67	6.50	12.73

Table 5.9: The average errors of certain perturbations (in percent).

The hand part model f_H was robust to note duration changes (maximum average error of 3%), but it exhibited significant inconsistency when note velocities were modified. This issue was mitigated when velocity perturbations were applied uniformly to notes played simultaneously, reflecting the common practice of maintaining similar velocities for chords played by one hand (see Table 5.10 for detailed results).

5.8.1.3. Hand Part Mitigation Strategies

To address the hand part model’s sensitivity to velocity perturbations, we explored two strategies:

1. **Data Augmentation:** Introducing random noise to the velocity feature during training, with a probability of application set to 50% and a standard deviation $\sigma_v = 12$.
2. **Feature Removal:** Excluding the velocity feature entirely.

The first approach effectively resolved the issue without compromising model performance, reducing the error rate from 34.80% to 3.24% under the most extreme perturbation scheme. Figure 5.5 illustrates the results of this approach.

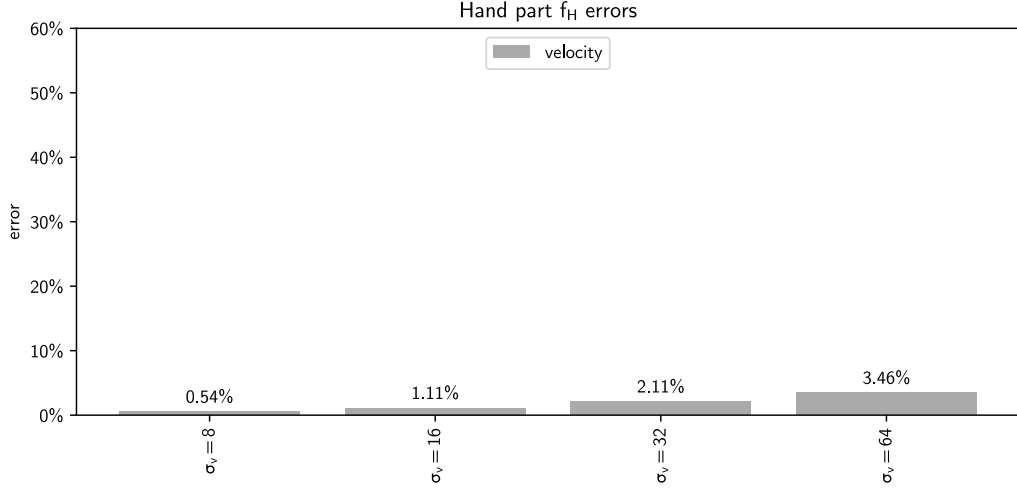


Figure 5.5: Impact of data augmentation on hand part model robustness. Even under extreme velocity perturbations ($\sigma_v = 64$), the model’s decision was only marginally affected.

The second approach, while more radical, produced similar results. Ablation studies (Section 6.1) indicated that the absence of the velocity feature did not noticeably affect model performance.

Variant	Velocity change σ_v			
	8	16	32	64
Standard	7.61	15.65	25.69	34.80
Uniform within groups	2.70	6.44	11.50	15.14

Table 5.10: The average errors for the hand part model f_H for 1. standard perturbation, 2. uniform random change for notes played in the same time. The second transformation introduces less inconsistencies.

Figure 5.8 shows the result of hand part assignment for velocity perturbation.

5.9. Local Feature Importance

To better understand the models’ behavior, we investigated the local influences of individual note features on model’s decisions. In particular, we posed several questions:

- Is it possible to determine which note features influenced the model’s decisions?
- Can we identify the specific notes that contribute most significantly to a model’s predictions, such as key signature attribution?

- Can we quantify the strength and direction of a particular note feature’s influence? For example, can we measure whether a note feature supports or opposes a right-hand assignment?

This type of questions, along with previous analysis, may help understanding the decision process of the model, facilitate recognizing weak spots of AMT systems and its limitations, and, finally, guide towards developing better AMT system.

We encountered challenges with common explainable machine learning approaches:

- The input data is a very high-dimensional space, computationally infeasible for certain methods (e.g. Shapley values).
- Pitch space is a discrete space that lacks a meaningful measure of distance.
- The note sequence tensors consists of interdependent features. In particular, they are not statistically independent, both horizontally and vertically.

To overcome these obstacles, we developed a custom solution inspired by the *Local Interpretable Model-agnostic Explanations* (LIME) framework [RSG16].

5.9.1. LIME-Based Analysis for Hand Part Assignment

We designed a locally interpretable approach to assess the influence of individual note features on model outputs. The approach involves generating a locally modified version of the input MIDI tensor by selectively randomizing the pitch or velocity of individual notes. This approach aids in explaining the findings from the previous section (see Figure 5.8 for an example).

The procedure follows these steps:

1. For each note in the sequence, create $n = 50$ perturbations by replacing a specific feature (e.g., pitch or velocity) with a randomized value.
2. Measure the proximity of the perturbed value to the original value. For velocity, we assign a weight $w = \exp\left(-\frac{|d|}{20}\right)$, where d is the absolute difference. For pitch, the weight is binary, with $w = 1$ if the perturbed value differs from the original.
3. Train a linear regression model to predict the difference between the model’s output for the original sequence and the perturbed sequence, using the proximity weights.
4. Record the regression coefficients as feature influence scores.

The resulting influence scores form a tensor of size $2 \times N \times N$, where N is the number of notes in the sequence, and the two dimensions correspond to pitch and velocity influences. A single vector contains influences of a particular feature onto a decision on the level of a single note.

5.9.1.1. Observations and Limitations

The proposed approach ignores temporal aspect of the sequence, as not taking onsets and durations into account. Moreover, the approach assumes feature independence, which is clearly not justified. However, it may give a basic insight into two influence of two basic MIDI elements: pitch and velocity.

Despite these limitations, the approach offers a basic understanding of the influence of two key MIDI features (pitch and velocity) on model decisions. For example, Figure 5.8 illustrates how changing the velocity of right-hand notes impacts hand part assignments for neighboring notes.

As one-note replacement usually won't lead to key signature change, this practice is useful mostly for the hand-part assignment. For the key signature model we propose another technique.

5.9.2. Key Signature Attribution via Note Omission

For the key signature model, where the pitch space is discrete and lacks a meaningful distance metric, we adopted a different strategy based on note omission. This method estimates the contribution of individual notes to key signature attribution.

The procedure is as follows:

1. Compute the key signature prediction values (before the last activation function) for the entire sequence.
2. For each note, remove it from the sequence and calculate the difference between the original prediction and the prediction for the modified sequence.
3. Aggregate the differences for each note. Only predictions on the current key signature are taken into account.

This approach enables the measurement of each note's contribution to the attribution of a specific scale. Refer to Figure 5.9 for an illustrative example.

Using this technique we were able to estimate the influence of certain note feature to a model's decision.

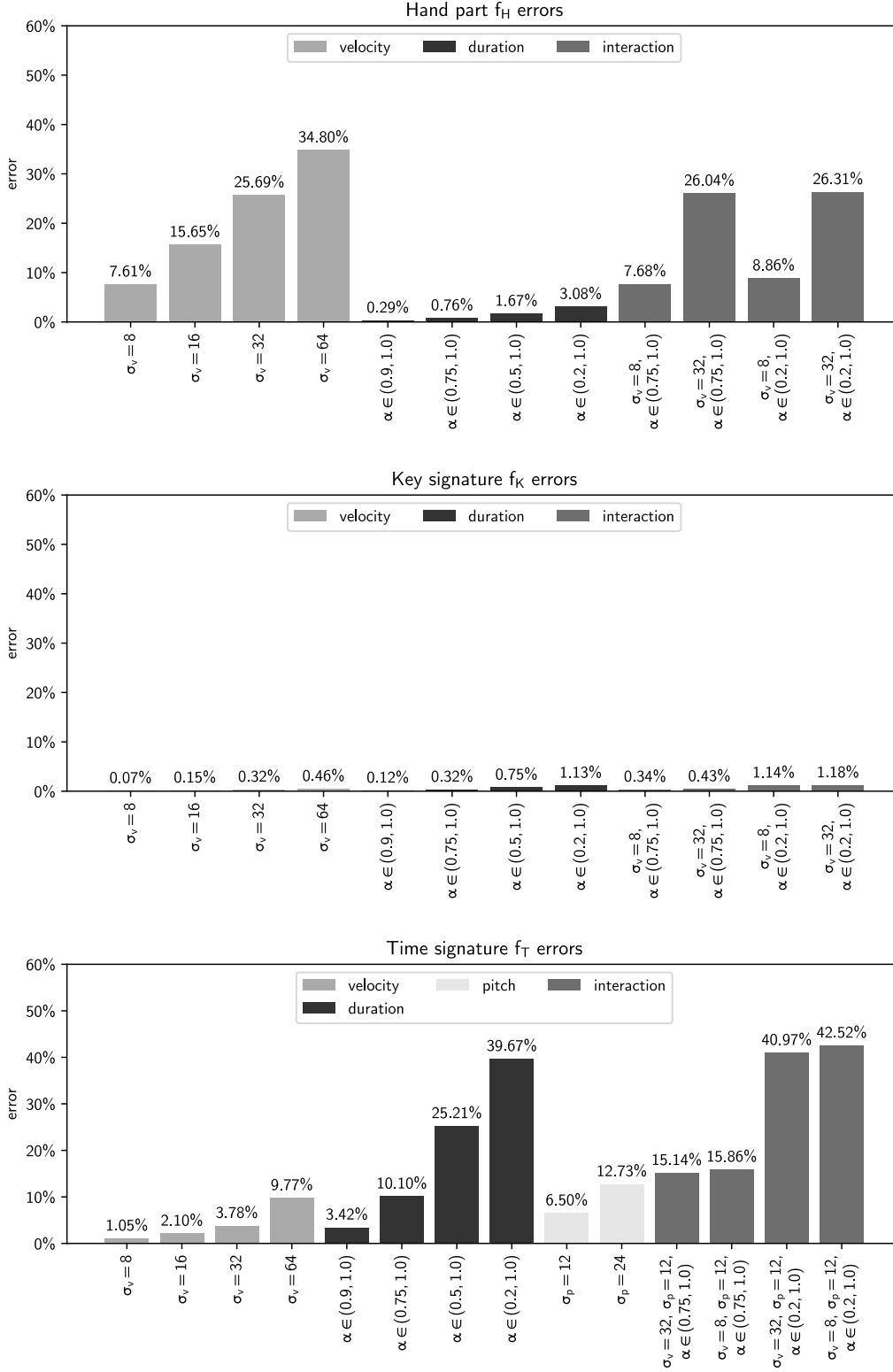


Figure 5.6: Results of the *ceteris paribus* experiment for the hand part model f_H , the key signature model f_K and the time signature model f_T . The hand part model is robust to note duration changes while it is susceptible to velocity manipulation. On the other hand, the time signature model is sensitive to note duration changes, which is expected to some extent, and relatively robust to other transformations. The key signature model is robust to all perturbations.

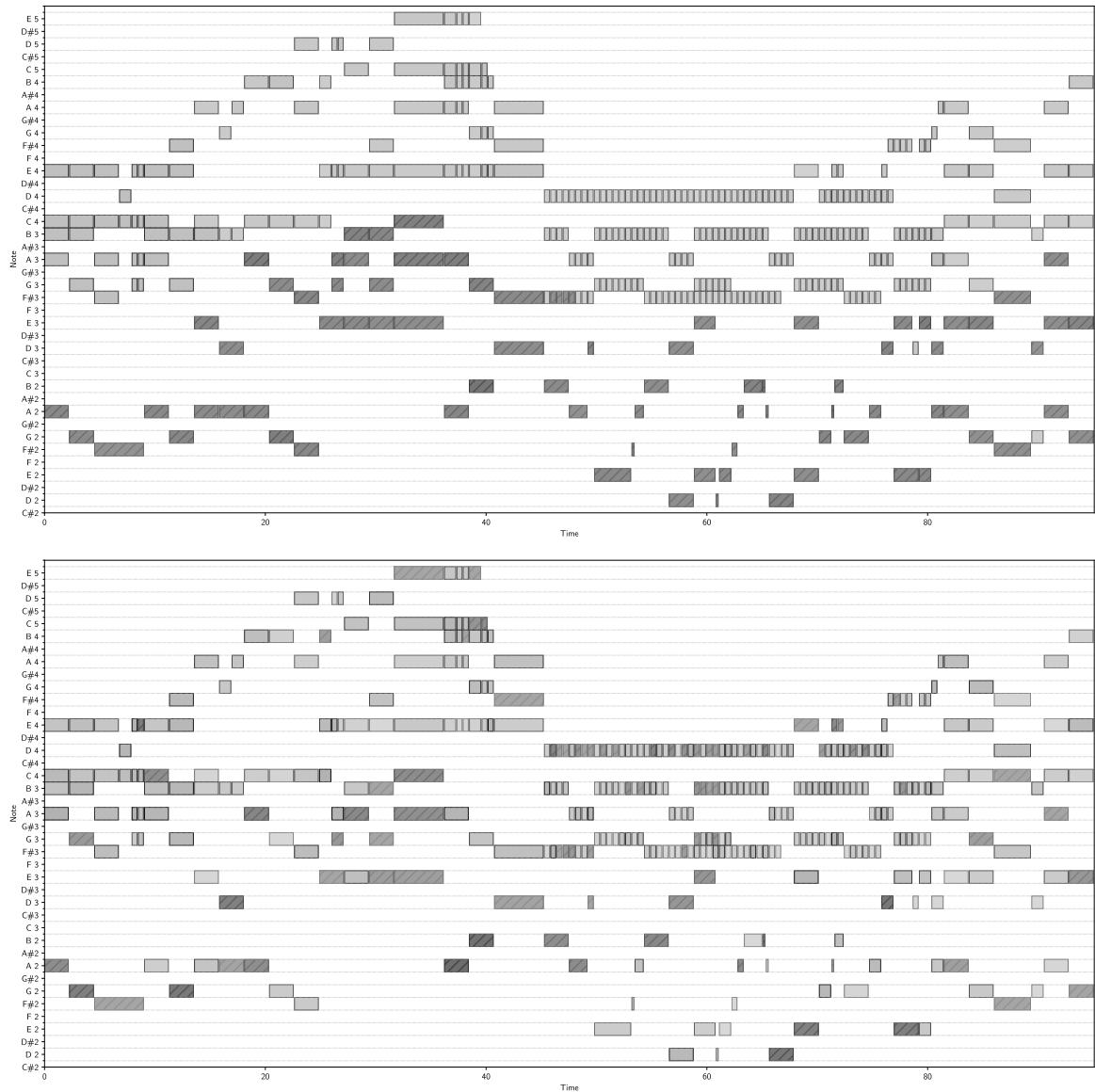


Figure 5.7: Hand part assignment errors after velocity perturbations. The first figure shows the original sequence, while the second shows the sequence after perturbations. Left-hand notes are diagonally patterned. Many hand part misalignments can be observed in the second figure. As a rule of thumb, left-hand notes should be at the bottom, while right-hand notes should be at the top.

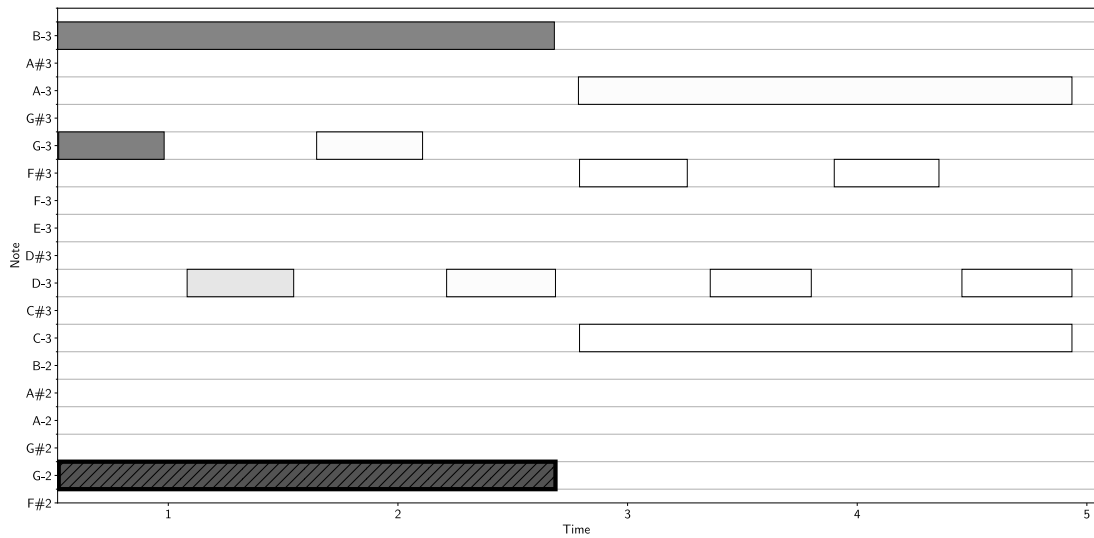


Figure 5.8: Velocity influence on hand part assignment. The G2 note (thick outline) is influenced by the velocity of neighboring notes. Negative influence is represented by the diagonal pattern. Increasing the velocity of the low note causes misalignment of G3 and B3 notes to the left hand, while reducing the G2 velocity shifts it to the right hand. This behavior is undesirable.

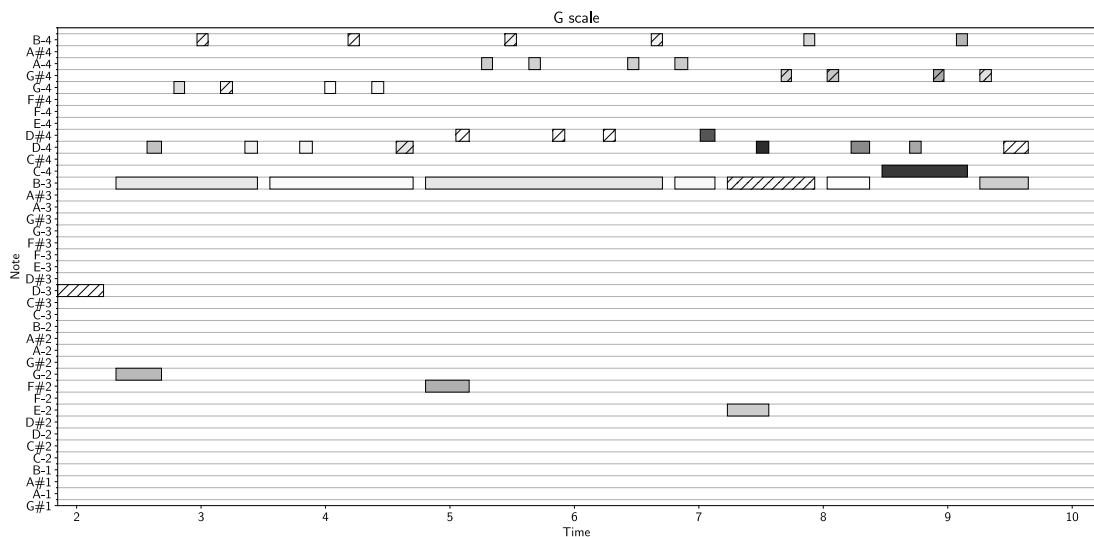


Figure 5.9: Key signature alignment for G major scale. The influence of each note on the key signature attribution is represented by grayscale shading, with a diagonal pattern indicating negative influence. Notes D# and G#, which are outside the scale, have the strongest negative impact. Sporadically, some other notes in the scale, especially B, voted slightly against the current key signature. The strongest negative vote changes the assignment probability by 6 percentage points.

Chapter 6

Experiments and Improvements

In order to enhance the proposed model, we employed different strategies for finding improvements:

- Removing some input features to improve the robustness of the model.
- Testing smaller, more data-efficient networks to optimize the current architecture.
- Experimenting with more powerful architectures, such as Transformers, which have demonstrated remarkable success in natural language processing (NLP).
- Extending the model by introducing an additional module for handling other musical elements: dynamics.

In subsequent experiments we stepped out from the full Transformer for attention mechanism only. We wanted to check whether access to the entire tensor sequence is beneficial for the transcription task. However, in the preliminary experiments combinations of self-attention mechanism with other components didn't perform as well as the base model nor Transformer-based ones, and the self-attention-based models have been excluded from the study.

Let us present results of these experiments.

6.1. Ablation Studies

To validate the assumptions presented in Section 5.8, we conducted ablation studies by training models that utilized only a restricted set of input features. Specifically, the following configurations were employed:

- The **hand part** model was trained only on note pitches and onsets.
- The **key signature** model was trained only on note pitches and onsets.
- The **time signature** model was trained using only note onsets.

As a result, all models became invulnerable to transformations described in the Section 5.9. This section evaluates the impact of such feature reduction on model performance.

Each model was evaluated using two metrics:

- Model-specific F_1 scores.
- The MV2H metric, with restricted models replacing their corresponding original components.

All evaluations were conducted on the test subset of the dataset, with both base and restricted models trained exclusively on the training set. For a fair comparison, the model with the best validation F_1 score across 50 epochs was selected for each ablation study.

6.1.1. Beat Tracking

The authors of the original paper provided ablation studies for the beat-tracking model, analyzing how the omission of features and data augmentation influenced its performance. The results of these analyses are summarized in Tables 6.1 and 6.2.

Feature omitted	Precision	Recall	F_1 score
Pitch	0.904	0.937	0.906
Onset	0.846	0.728	0.764
Duration	0.895	0.931	0.901
Velocity	0.895	0.939	0.906
Use all features	0.912	0.943	0.913

Table 6.1: Feature omission study [LKMB22].

Not surprisingly, note onsets are the most important feature used by the beat-tracking model. While neglecting other features doesn’t hardly impact the model, still feature omission results in a slight quality degradation.

Velocity may carry some information about beats as they tend to occur along with heavier notes [LKMB22]. Harmonic structure inferred by pitch may also transfer such information, for example it is quite common to play chords at the beginning of a measure. Note durations may also offer hints about the underlying meter.

Augmentation method omitted	Precision	Recall	F_1 score
Pitch shift	0.920	0.920	0.906
Tempo change	0.918	0.895	0.897
Note removal	0.915	0.909	0.904
Extra note insertion	0.912	0.943	0.913
Use all methods	0.929	0.937	0.922
No data augmentation	0.897	0.952	0.909

Table 6.2: Ablation study for data augmentation [LKMB22].

6.1.2. Hand Part Model

A variant of the hand part model was trained without using note durations or velocities. As the hand part assignment influences only the voice separation subscale of the MV2H metric, other submetrics were excluded from the evaluation.

Metric	Standard	Restricted
Accuracy	0.956	0.949
Precision	0.851	0.843
Recall	0.848	0.845
F_1 score	0.842	0.840
Voice separation	0.820	0.818

Table 6.3: Ablation study for the hand part model.

The results (Table 6.3) show comparable performance between the restricted and original models across all metrics. This indicates that velocity provides marginal, if any, additional information for hand part assignment. These findings align with the observations in Section 5.8.1.2, where velocity had a limited influence on model predictions.

6.1.3. Key Signature Model

The restricted key signature model, trained using only note pitches and onsets, achieved performance similar to the original model.

Type	Metric	Standard	Restricted
Macro	Precision	0.812	0.807
	Recall	0.789	0.788
	F_1 score	0.787	0.786
Weighted	Precision	0.924	0.924
	Recall	0.898	0.904
	F_1 score	0.896	0.900
Harmony		0.972	0.972

Table 6.4: Ablation study for the key signature model.

Table 6.4 shows the results with included the harmony MV2H subscale. Let us remind that weighted metric results are calculated independently, and weighted by the number of true labels of each class, while macro metrics are unweighted.

The observed differences fall within the expected range of model stochasticity. The result agrees also with the negligible influence of other factors to the final model decision on key signature, described in Section 5.8.1.2.

6.1.4. Time Signature Model

A restricted time signature model, trained solely on note onsets, outperformed the original convolutional network classifier in terms of F_1 scores.

Metric	Standard	Restricted
Accuracy	0.552	0.656
Precision	0.444	0.571
Recall	0.333	0.667
F_1 score	0.381	0.615

Table 6.5: Ablation study for the time signature model.

Interestingly, as the MV2H metric does not directly depend on time signature predictions, substituting the original model with the restricted version had no impact on the overall evaluation. This shows that not only introducing full information to the model is not beneficial, but actually hurts the performance of the model.

6.2. Transformers

Transformers are a class of neural network architectures introduced by Vaswani et al. in the landmark paper *Attention Is All You Need* [VSP⁺17]. Originally developed for natural language processing (NLP), have since become versatile and powerful framework for modeling sequential data across numerous other fields, including image processing [DBK⁺21], audio generation [BMV⁺23] and even symbolic music analysis [ZNFW21].

The key innovation in Transformers is the *self-attention* mechanism, which enables the model to weigh the relevance of each part of an input sequence relative to other parts, regardless of their distance from each other in the sequence. Unlike recurrent neural networks, which process data sequentially and can struggle with long-range dependencies, Transformers use that mechanism to directly model these dependencies in parallel.

As the authors state at the end of the paper [LKMB22]:

Possible next steps include investigating more powerful model architectures such as the Transformer.

We analyzed various setups involving Transformer and attention mechanisms, preserving the training and evaluation setup.

6.2.1. Vanilla Transformer

We evaluated a standard Transformer encoder architecture, limited to the encoder block since the transcription task does not involve decoding. As a rule of thumb, we decided to keep the

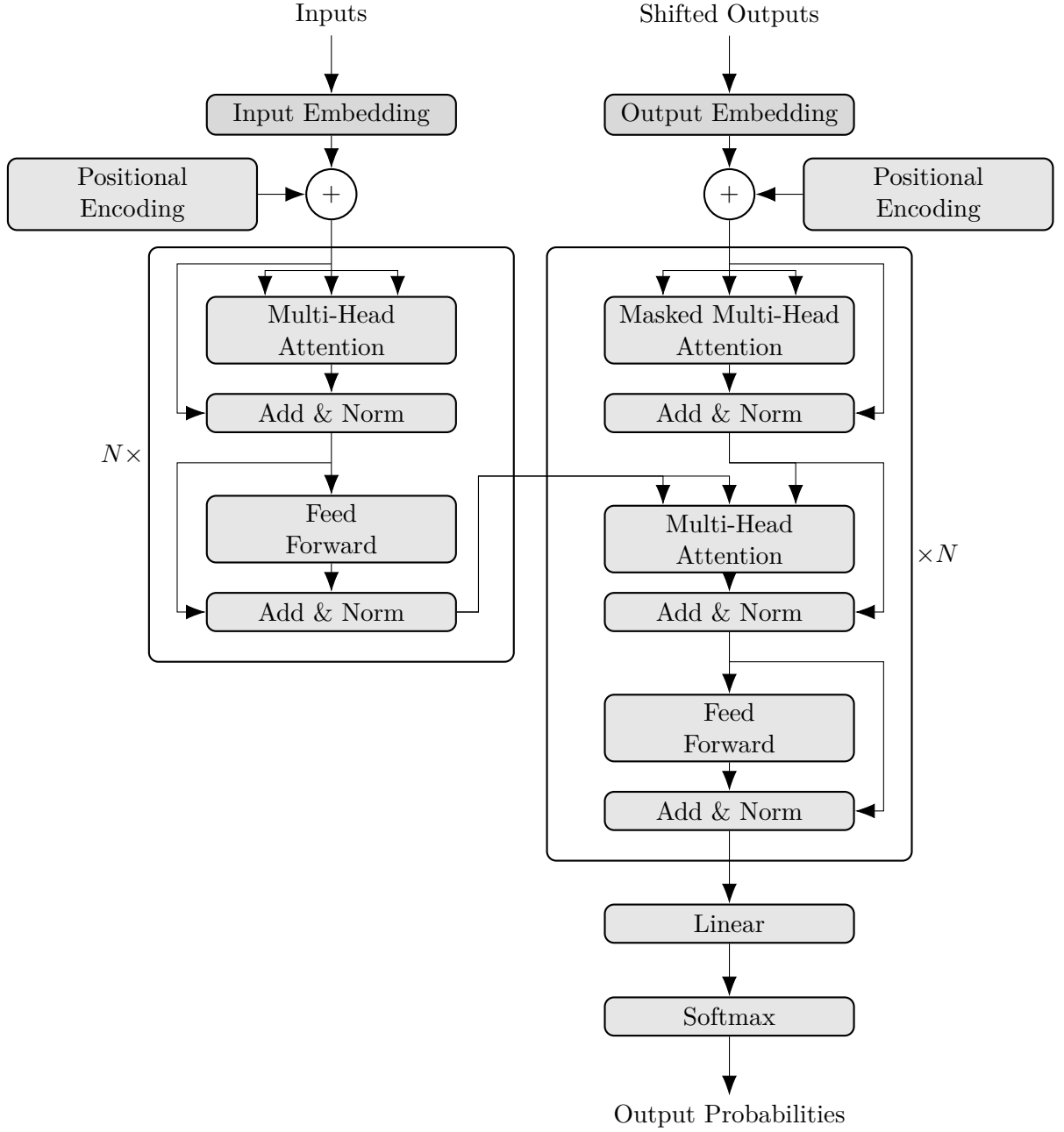


Figure 6.1: The Transformer architecture [VSP⁺17].

model size comparable to the original network. Different number of embedding sizes have been employed: 128, 256 and 512 dimensions.

Given that Transformers are designed for sequential discrete data, we applied an additional encoding scheme to discretize note durations and velocities. Features were concatenated prior to positional encoding. Preliminary experiments demonstrated that this approach outperformed the use of continuous floating-point encodings for durations and velocities. The note durations were encoded with the same number of discrete features as onsets, while ve-

locities were reduced to eight discrete categories. The total feature space thus consisted of:

$$\underbrace{128}_{\text{pitch}} + \underbrace{401}_{\text{onset}} + \underbrace{401}_{\text{duration}} + \underbrace{8}_{\text{velocity}} = 938$$

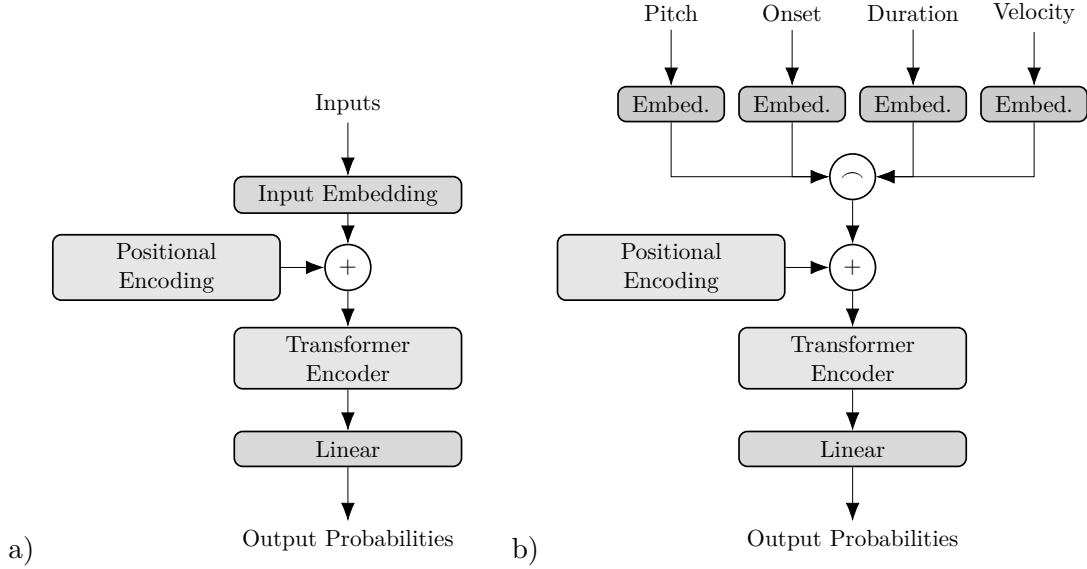


Figure 6.2: The Transformer Encoder block: a) default encoding, b) feature embedding. The \curvearrowright symbols stands for concatenation. The second architecture was selected for subsequent experiments. The positional encoding is optional.

For training, the learning rate was reduced to 1×10^{-4} , with a warmup period of 2500 steps. We also tested configurations without positional encoding¹. All experiments utilized the default `pytorch` Transformer encoder implementation.

6.2.2. Results

Overall, Transformer-based models did not match the performance of the base architecture. Positional encoding, in particular, often degraded performance. Even when achieving comparable validation results on specific subtasks, Transformer-based models struggled to generalize as effectively as the original network. The sole exception was the time signature model, where Transformers outperformed the original convolutional neural network.

We present various setups for all submodels: the beat model, the hand part model, the key and time signature models.

6.2.3. Beat Model

The vanilla Transformer architecture failed to learn accurate beat predictions, particularly for downbeat classification. This was consistent across all model variants.

¹Certain language models have been shown to perform well without explicit positional encodings [HRP⁺22].

Model Parameters		Base 27.5 M	T128 4.2 M	T128 PE 4.2 M	T256 10.1 M	T256 PE 10.1 M	T512 27.0 M	T512 PE 27.0 M
Val. <i>b</i>	Acc	0.880	0.713	0.700	0.711	0.703	0.702	0.717
	Prec	0.879	0.710	0.695	0.697	0.698	0.682	0.743
	Rec	0.873	0.625	0.665	0.706	0.635	0.696	0.629
	<i>F</i>	0.864	0.642	0.647	0.670	0.644	0.659	0.644
Val. <i>db</i>	Acc	0.853	0.818	0.817	0.815	0.815	0.821	0.822
	Prec	0.622	0.481	0.534	0.507	0.500	0.557	0.548
	Rec	0.467	0.098	0.193	0.219	0.117	0.186	0.221
	<i>F</i>	0.508	0.148	0.245	0.258	0.158	0.246	0.266
Test <i>b</i>	Acc	0.869	0.705	0.696	0.711	0.680	0.714	0.724
	Prec	0.859	0.693	0.684	0.705	0.702	0.691	0.718
	Rec	0.884	0.656	0.660	0.649	0.549	0.698	0.652
	<i>F</i>	0.859	0.651	0.641	0.649	0.576	0.677	0.667
Test <i>db</i>	Acc	0.860	0.821	0.810	0.824	0.817	0.829	0.822
	Prec	0.665	0.523	0.511	0.584	0.545	0.582	0.513
	Rec	0.486	0.178	0.244	0.187	0.114	0.186	0.182
	<i>F</i>	0.545	0.246	0.303	0.258	0.173	0.264	0.254
Test	<i>F_{ma}</i>	0.579	0.391	0.390	0.402	0.375	0.406	0.433

Table 6.6: Transformer results for the beat model. The symbol *b* stands for beat prediction, and *db* stands for downbeat classification. Models with the positional encoding are labeled as PE.

Sinusoidal position encoding turned out to be not helpful, and most of the time worsened the results a bit. This suggests that for Transformers in order to properly predict the metric structure of a piece, a more adequate positional encoding is needed. The *Rotary Positional Embedding* (RoPE) [SAL⁺24] is a fairly recent technique that may warrant further exploration in this context.

Additionally, a note’s position within a song does not map cleanly onto tensor indices, as tensor positions lack a direct correspondence to the actual temporal context of note onsets. This indicates a need for an alternative positional encoding feature that can represent a note’s position relative to its onset.

6.2.4. Hand Part Model

While the original hand part model consistently outperformed Transformer-based alternatives, the latter achieved results close to the base model in many cases. Positional encoding slightly improved performance, but the overall difference was rather small.

Notably, despite comparable validation performance, Transformer-based hand part models performed worse on the MV2H metric, suggesting issues with generalization.

Model Parameters		Base 10.5 M	T128 1.5 M	T128 PE 1.5 M	T256 3.7 M	T256 PE 3.7 M	T512 9.9 M	T512 PE 9.9 M
Val.	Acc	0.955	0.897	0.934	0.894	0.931	0.897	0.929
	Prec	0.849	0.803	0.834	0.783	0.830	0.783	0.833
	Rec	0.851	0.764	0.825	0.800	0.815	0.800	0.825
	F	0.846	0.775	0.824	0.783	0.817	0.783	0.821
Test	Acc	0.940	0.863	0.910	0.864	0.922	0.871	0.921
	Prec	0.923	0.865	0.917	0.849	0.913	0.828	0.912
	Rec	0.899	0.766	0.834	0.790	0.864	0.839	0.866
	F	0.909	0.804	0.869	0.809	0.885	0.826	0.884
Test	F_{vs}	0.821	0.726	0.744	0.742	0.752	0.751	0.756

Table 6.7: Transformer results for the hand part model.

6.2.5. Key Signature Model

For the key signature model, some Transformer configurations achieved superior performance in individual evaluation categories. However, none of these models matched the high harmony component score achieved by the original model in the MV2H metric.

Model Parameters		Base 10.5 M	T128 1.5 M	T128 PE 1.5 M	T256 3.7 M	T256 PE 3.7 M	T512 9.9 M	T512 PE 9.9 M
Val. Macro	Prec	0.741	0.782	0.758	0.782	0.771	0.767	0.763
	Rec	0.697	0.788	0.745	0.758	0.737	0.744	0.745
	F	0.701	0.779	0.738	0.753	0.733	0.738	0.741
Val. Micro	Prec	0.891	0.839	0.879	0.902	0.886	0.869	0.875
	Rec	0.785	0.833	0.831	0.836	0.791	0.796	0.816
	F	0.809	0.833	0.835	0.843	0.802	0.806	0.828
Test Macro	Prec	0.812	0.855	0.726	0.809	0.747	0.768	0.780
	Rec	0.789	0.848	0.749	0.802	0.739	0.779	0.759
	F	0.787	0.830	0.726	0.785	0.731	0.761	0.758
Test Micro	Prec	0.924	0.916	0.775	0.860	0.825	0.851	0.924
	Rec	0.898	0.873	0.776	0.812	0.781	0.837	0.865
	F	0.894	0.868	0.768	0.810	0.791	0.835	0.873
Test	F_{ha}	0.972	0.944	0.917	0.942	0.880	0.942	0.940

Table 6.8: Transformer results for the key signature.

6.2.6. Time Signature Model

Transformer-based architectures outperformed the original convolutional network for the binary time signature classification task. This is rather unsurprising as the base network was not so expressive.

The time signature model does not affect the MV2H metric.

Model Parameters		Base 0.7 M	T128 1.5 M	T128 PE 1.5 M	T256 3.7 M	T256 PE 3.7 M	T512 9.9 M	T512 PE 9.9 M
Validation	Acc	0.560	0.506	0.677	0.606	0.636	0.663	0.663
	Prec	0.315	0.305	0.277	0.245	0.291	0.297	0.297
	Rec	0.230	0.218	0.229	0.244	0.292	0.296	0.296
	<i>F</i>	0.258	0.236	0.238	0.245	0.291	0.296	0.296
Test	Acc	0.552	0.621	0.621	0.621	0.724	0.586	0.586
	Prec	0.444	0.526	0.529	0.546	0.833	0.500	0.500
	Rec	0.333	0.833	0.750	0.500	0.417	0.167	0.167
	<i>F</i>	0.381	0.645	0.621	0.522	0.556	0.250	0.250

Table 6.9: Transformer results for the time signature.

6.2.7. Future Work

While Transformers hold significant promise for sequence modeling tasks, the results suggest that their application to musical transcription may require more specialized adaptations. In particular:

- Transformers struggled to outperform the base model for the beat prediction. This suggests that a different quantization scheme may be needed, potentially involving modifications to the target variables.
- Classical positional encodings were often detrimental, indicating a need for more sophisticated positional representations. Moreover, note positions in a sequence do not directly map to tensor indices, and the order of notes alone provides insufficient information about temporal relationships within the sequence.
- The dataset used in these experiments is not fully representative of performance MIDI, as it includes a significant number of human-crafted MIDI files. Transformers likely require larger and more diverse datasets to achieve superior results. Addressing that issue demands refining the dataset and acquiring more data of high quality.
- The time signature model could benefit from two enhancements: expanding its scope to include a wider variety of time signature classes as initially, and incorporating beat and downbeat predictions as auxiliary inputs to refine its classification accuracy.

6.3. Temporal Convolutional Network

Classical convolutional networks usually have a rather small receptive field², and the canonical way to enlarge the size is to stack several layers of the network. As the input sequence can

²*Receptive field* can be understood as the size of the region in the input data that produces a feature [ANS19]. For a point on a grid, these and only these parts in that region are accessible by the network.

be very large, covering greater portions of data requires more model parameters, which at some point may yield too complex architecture for a given data scenario.

6.3.1. Dilated Convolution

To keep the size of the model small, one could use a *dilated convolution*, which expands the receptive field by spacing out the kernel element with gaps, called *dilation rates*.

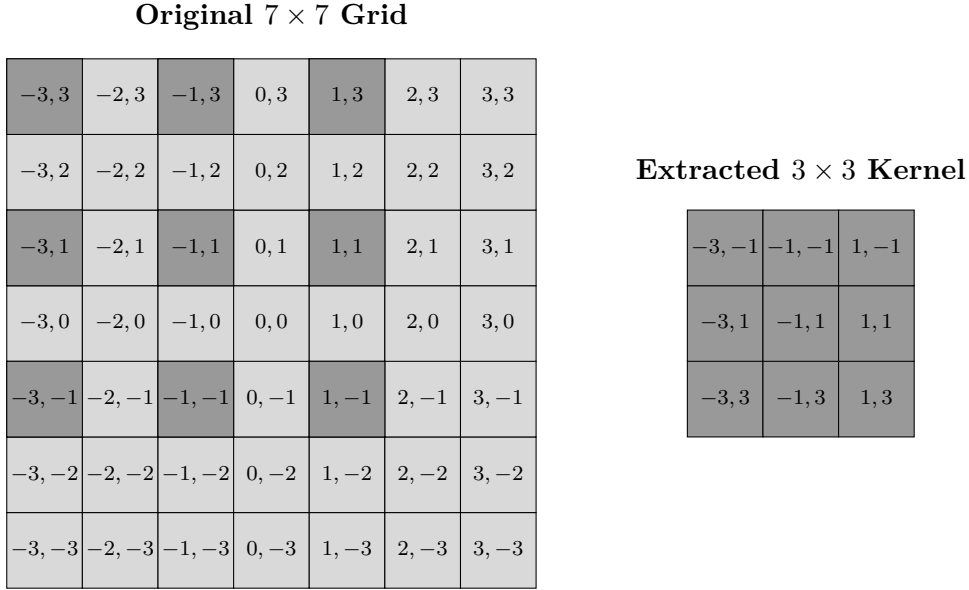


Figure 6.3: Dilated convolution with kernel size $k = 3$ and dilation rate $d = 2$, illustrated on the $(-1, 1)$ point on a 7×7 grid.

Dilated convolutional layers increase the receptive field of the network exponentially with linear parameter accretion.

While the idea can be traced back to wavelet decompositions from 1989 [HKMMT89], the rediscovery of the method in the machine learning field can be attributed to Chen et al. [CPK⁺15].

6.3.2. Temporal Convolutional Network

Temporal Convolutional Network (TCN), introduced by [LVRH16], leverage the idea of dilated convolutions, to cover varied regions of data in a hierarchical way. The network consists of layers of dilated convolutions, where the dilation rate increases exponentially (e.g., $d = 1, 2, 4, \dots$) across layers.

For sequential or time-series data, this structure efficiently captures both short- and long-term dependencies. Unlike recurrent networks, TCNs process data in parallel, making them computationally efficient.

Compared to recurrent neural networks, TCNs offer several advantages:

- **Parallel Computation:** TCNs allow parallel processing of sequences, unlike recurrent networks where computation is sequential and each state depends on the previous one. This significantly reduces training time.
- **Long-Term Dependency Modeling:** TCNs achieve flexible receptive fields through dilated convolutions, enabling the direct control of dependency range without architectural adjustments.
- **Stability:** Convolutional networks, including TCNs, generally exhibit stable training dynamics. Recurrent networks, on the other hand, are prone to issues such as vanishing or exploding gradients, although gated mechanisms like GRUs mitigate some of these challenges.

In many cases, these advantages come without sacrificing performance, as we demonstrate in subsequent sections. We evaluate TCNs on various transcription subtasks and provide details about the model structure.

6.3.3. Model Architecture

Since musical features are not spatial, the convolutions are one-dimensional. For our experiments, we used a TCN variant with a kernel size of 9 and dilation rates of 1, 2, and 4. The receptive field r for L layers can be calculated as follows:

$$r = 1 + \sum_{i=1} (k_i - 1) \cdot d_i$$

where k_i and d_i denote the kernel size and dilation rate, respectively, at the i -th layer. For three layers, the receptive field is $r_3 = 57$. See Figure 6.4 for an illustration of the architecture.

We tested two model variants, with 128 and 256 filters per convolutional layer. The training and evaluation procedures followed the same setup as the baseline models.

6.3.4. Results

Experiments with TCNs showed that these networks achieved comparable or superior performance for certain transcription subtasks, while requiring fewer parameters and training faster compared to recurrent networks.

We use standard abbreviations for the metrics: *acc*, *prec*, *rec* and F for accuracy, precision, recall and F_1 score, respectively.

6.3.4.1. Beat Model

The beat quantization task was less suited to the TCN architecture. While the larger TCN variant performed close to the baseline, the original model consistently outperformed both

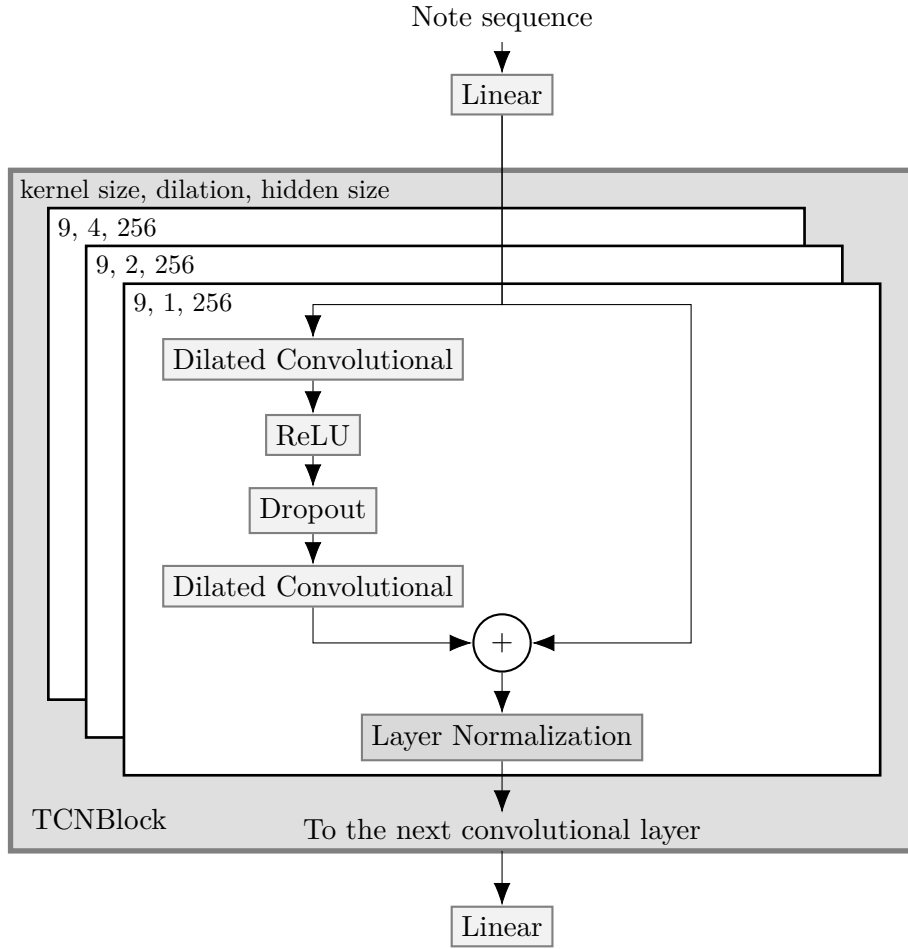


Figure 6.4: The Temporal Convolutional Block.

TCN variants.

Adding more layers or increasing the receptive field worsened model performance, indicating that the architecture may require further tuning for this task.

The dynamic programming algorithm for out-of-note beat prediction has been left intact.

6.3.4.2. Hand Part Model

The TCN-based hand part model achieved results comparable to the baseline while benefiting from a significantly smaller and faster architecture. This suggests that recurrent networks are not strictly necessary for the hand part assignment task.

6.3.4.3. Key Signature Model

The TCN-based key signature model struggled with rarer key signatures, resulting in a noticeable gap between macro and weighted metrics. Variants with more layers didn't improve the quality of the model.

Model Parameters		Original 27.5 M	TCN 128 7.6 M	TCN 256 10.9 M
Beats	Accuracy	0.880	0.812	0.848
	Precision	0.879	0.814	0.832
	Recall	0.873	0.770	0.831
	F_1 score	0.864	0.777	0.819
Downbeats	Accuracy	0.853	0.847	0.867
	Precision	0.622	0.628	0.660
	Recall	0.467	0.386	0.568
	F_1 score	0.508	0.439	0.577
Beats	Accuracy	0.869	0.823	0.851
	Precision	0.859	0.795	0.842
	Recall	0.884	0.848	0.847
	F_1 score	0.859	0.802	0.831
Downbeats	Accuracy	0.860	0.854	0.860
	Precision	0.665	0.640	0.704
	Recall	0.486	0.453	0.406
	F_1 score	0.545	0.513	0.494
Meter		0.579	0.537	0.576

Table 6.10: Temporal Convolutional Network results for the beat model.

Model Parameters		Original 10.5 M	TCN 128 954 K	TCN 256 3.7 M
	Accuracy	0.955	0.948	0.958
	Precision	0.849	0.844	0.852
	Recall	0.851	0.834	0.838
	F_1 score	0.846	0.834	0.842
	Accuracy	0.956	0.921	0.950
	Precision	0.923	0.908	0.919
	Recall	0.899	0.865	0.931
	F_1 score	0.909	0.885	0.925
Voice separation		0.820	0.805	0.806

Table 6.11: Temporal Convolutional Network results for the hand part model.

6.3.4.4. Time Signature Model

The TCN-based time signature model significantly outperformed the baseline convolutional network. As observed in earlier experiments, time signature estimation does not impact the MV2H metric.

6.4. Dynamics

In a musical score, dynamics indicate the intensity or volume with which notes should be played. To enhance the transcription capabilities of the model, we introduced a new sub-

Model Parameters		Original 10.5 M	TCN 128 955 K	TCN 256 3.7 M
Macro	Precision	0.741	0.447	0.425
	Recall	0.697	0.345	0.306
	F_1 score	0.701	0.376	0.344
Weighted	Precision	0.891	0.989	0.988
	Recall	0.785	0.657	0.626
	F_1 score	0.809	0.758	0.736
Macro	Precision	0.812	0.307	0.351
	Recall	0.789	0.239	0.284
	F_1 score	0.787	0.260	0.307
Weighted	Precision	0.924	0.954	0.951
	Recall	0.898	0.673	0.682
	F_1 score	0.896	0.771	0.777
Harmony		0.972	0.878	0.834

Table 6.12: Temporal Convolutional Network results for the key signature model.

Model Parameters		Original 676 K	TCN 128 954 K	TCN 256 3.7 M
Accuracy	Accuracy	0.560	0.612	0.575
	Precision	0.315	0.280	0.261
	Recall	0.230	0.236	0.249
	F_1 score	0.258	0.248	0.253
Accuracy	Accuracy	0.552	0.655	0.793
	Precision	0.444	0.625	0.750
	Recall	0.333	0.417	0.750
	F_1 score	0.381	0.500	0.750

Table 6.13: Temporal Convolutional Network results for the time signature model.

module for transcribing dynamics. This submodule classifies notes into one of ten dynamic levels, ranging from *pppp* (very soft) to *ffff* (very loud). Several simplifications were made to address common challenges in interpreting dynamics:

- Gradual dynamic changes such as *crescendo* and *diminuendo* were excluded.
- Accents on individual notes (marked as >) were not considered.
- Dynamics were assumed to apply uniformly to both hands, disallowing independent dynamics for left and right hands.

As in standard musical practice, dynamic markings are displayed only when there is a change. If no marking is explicitly given, *mezzo-forte* is assumed as the default level. The dynamics model operates independently of the other transcription submodules.

6.4.1. Architecture

The architecture of the dynamics model follows the modular design of other submodules. It mirrors the structure of the key signature model, comprising convolutional layers, a GRU block, and a final linear layer with 10 output categories corresponding to the dynamics levels.

The extended architecture of the model, including the dynamics module, is shown in Figure 6.5. The modularity of the model enables the dynamics block to be replaced with alternative architectures, such as Transformer or TCN-based blocks.

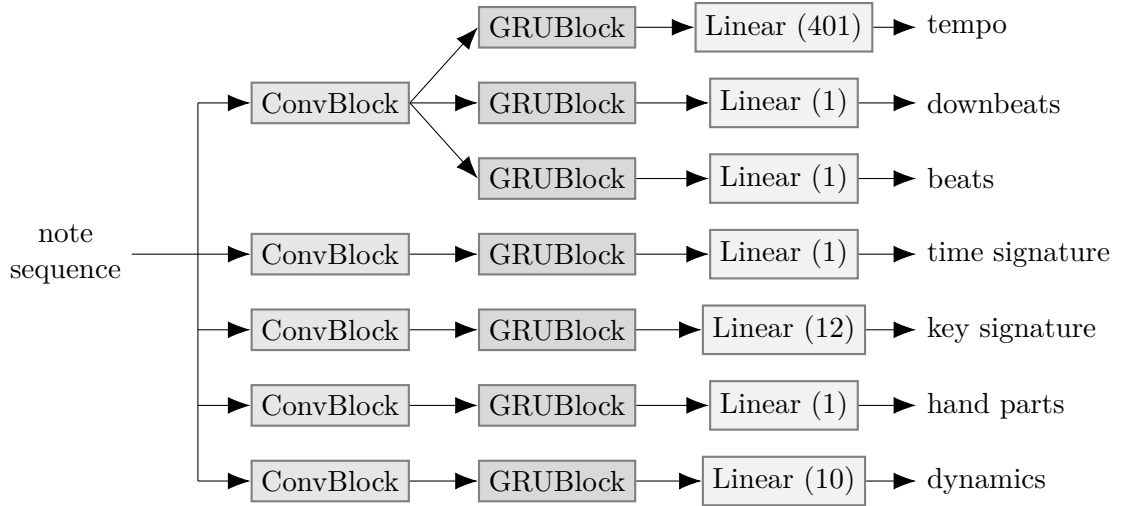


Figure 6.5: The extended architecture of the model, incorporating the *dynamics* module. Note that the quantization model is not shown, as it does not directly contribute to the final MIDI score generation.

6.4.2. Challenges

Two major challenges arose during the development of the dynamics model:

- No direct dynamics annotations exist in any of the datasets considered.
- The interpretation of dynamics in musical performances is subjective and thus varies considerably.

6.4.2.1. Dynamics Data

The ACPAS dataset does not contain any direct information about dynamics. Neither the MIDI files nor the annotations provide this data. Among the three datasets used, only the ASAP dataset includes dynamics annotations, but only for the ground truth score in the MusicXML format.

For the performance MIDI files in the ASAP dataset, dynamics annotations were recovered through an imperfect automation process. Table 6.14 presents an example of the augmented

annotation structure.

Timestamp	Timestamp	Annotation
2.845313	2.845313	db, 6/8, 2, <i>f</i>
3.446876	3.446876	b
3.858854	3.858854	db
4.218229	4.218229	b

Table 6.14: An example of augmented dynamics annotations in the ASAP dataset, showing a starting *forte* dynamics marking encoded as *f*.

Additionally, certain musical pieces, such as those written for the harpsichord or organ (e.g., some of Bach’s fugues), lack dynamic markings, limiting the dataset even more.

6.4.2.2. Matching Algorithm

The matching algorithm parses the original ground truth MusicXML score to extract dynamics annotation positions aligned to corresponding note blocks.

MIDI performance files use absolute time in seconds, while scores are in beats, so direct comparison is not possible. To overcome this, we employed a basic matching algorithm that relies on the ratio of already played notes to the total number of notes. Deviations are allowed to account for missing or extra notes.

Chords with a dynamics annotation from the score are matched to chords with identical notes and corresponding positions in the original performance. Any unmatched or misaligned chords are discarded. We achieved the total coverage of 58.6% of all dynamics markings.

While more sophisticated algorithms could be used, we opted for simplicity. We believe that high-quality dynamics annotations are essential for building a reliable dynamics model, and even more advanced matching algorithms are unlikely to resolve the core issue of data quality. The current solution is a preliminary step toward incorporating a richer vocabulary for musical scores.

6.4.3. Subjectivity of Dynamics Markings

A significant challenge in modeling dynamics is the inherent subjectivity of dynamics annotations. The interpretation of dynamics depends on various factors, including the instrument on which the music is performed, the era and genre of the music, and the performing style of the musician.

This variability leads to considerable differences in expected outcomes, as the relationship between the score and the performance is far from exact.

6.4.4. Evaluation

The model was trained using the standard negative log-likelihood loss, similar to the approach used for the key signature model. We evaluated the model using the standard set of multiclass classification metrics, with the macro F_1 score as the final evaluation metric, consistent with the evaluation scheme for the key signature model.

To avoid penalizing misclassifications between similar dynamic classes (e.g., distinguishing *f* from *ff*), one could apply a standard distance metric between labels, where 0 represents the softest dynamics, *pppp*, and 1 represents the loudest, *ffff*. However, we followed the same setup as the key signature model for simplicity.

Since the intersection of the original test set and the ASAP dataset is empty, the test set is not available.

6.4.4.1. MV2HD

Since dynamics can be evaluated independently of other features, assuming they are correctly aligned with the note stream, we propose an enhancement to the MV2H metric, called **MV2HD**, where D stands for dynamics. The dynamics submetric is simply the F_1 score for correctly aligned notes, and the total metric is calculated as:

$$\text{MV2HD} = \frac{5}{6}\text{MV2H} + \frac{1}{6}F_1^{(D)}$$

Here, $F_1^{(D)}$ is the dynamics submetric score. Since the F_1 score lies in the interval $[0, 1]$, the MV2HD metric is bounded between 0 and 1. The weight of $\frac{1}{6}$ for dynamics is debatable, as dynamics do not play the same role as, for example, meter alignment. However, this weight was chosen for symmetry in the overall metric.

Because dynamics are independent of other features, the MV2HD metric still adheres to the principle of disjoint penalties.

Currently, the MV2HD metric is a theoretical proposal and would require implementation in the Java code provided by McLeod et al. [McL19].

6.4.4.2. Results

The base architecture (convolutional layers followed by a GRU block) achieved the best result among all considered architectures. However, the final results is unsatisfactory.

These results are understandable, given the challenges discussed earlier. We believe that refining the dataset, particularly with respect to dynamics annotations, could significantly enhance the performance of all models. The availability of high-quality data appears to be a more important factor than the specific architectural choices for the model.

Model Parameters		Base 10.5 M	T256 3.7 M	T256 PE 3.7 M	T512 9.9 M	T512 PE 9.9 M	T128 PE 1.0 M	T256 PE 3.7 M
Val. Macro	Prec	0.522	0.414	0.418	0.431	0.371	0.386	0.394
	Rec	0.522	0.447	0.498	0.490	0.432	0.312	0.337
	F	0.493	0.402	0.428	0.431	0.362	0.302	0.332
Val. Micro	Prec	0.649	0.497	0.453	0.490	0.429	0.703	0.640
	Rec	0.589	0.464	0.494	0.506	0.428	0.469	0.449
	F	0.584	0.441	0.459	0.476	0.393	0.495	0.477

Table 6.15: Results for the dynamics model.

Chapter 7

Conclusions

We have explored the automatic transcription task from performance MIDI to symbolic scores in detail, discussing its components, challenges, and earlier methods. The state-of-the-art deep learning model has been analyzed thoroughly, with an in-depth examination of its performance and behavior.

Our analysis included the following:

- A comprehensive evaluation of the primary model’s performance and behavior.
- Ablation studies addressing model anomalies, especially the undesired influence of the velocity feature.
- Development of local feature importance methods to interpret the model’s decisions.
- Comparison with alternative architectures, such as Transformers and Temporal Convolutional Networks (TCNs).

Additionally, we proposed an enhancement to the model to handle an additional musical feature: dynamics.

Below is a summary of the results.

7.1. Robustness Analysis

Our analysis uncovered artifacts in the transcription models, particularly related to velocity’s influence on hand part assignment. The time signature model showed sensitivity to transformations that should have been inconsequential, while the key signature model remained robust across all perturbations.

By employing data augmentation and selective feature removal, we mitigated these undesired behaviors without degrading model performance.

7.2. Feature Importance Methods

Symbolic music data presents unique challenges for explainable artificial intelligence (XAI). Existing XAI methods often do not adapt well to this domain. Developing more targeted and specialized XAI tools could improve model interpretability.

We implemented two methods to analyze submodel behavior:

- A semi-LIME approach to visualize how velocity features influenced hand part assignment.
- A note omission method to quantify how individual notes contributed to key signature assignment decisions.

These methods provided insights into the models’ decision but have certain limitations. The semi-LIME approach, for example, assumes feature independence. On the other hand, both methods disregard temporal relationships.

A broader challenge in developing XAI tools for symbolic music lies in the absence of a meaningful pitch space representation that encodes musical relationships comprehensively. The current treatment of pitch as a discrete space hinders the analysis of key signature assignment and limits the interpretability of related tasks.

7.3. Experiments

A series of experiments compared the state-of-the-art performance with alternative architectures. Below are the findings:

7.3.1. Transformers

Vanilla Transformers didn’t perform as good as other networks. We hypothesize that Transformer architecture could outperform the proposed network but in a data-abundant scenario. In the considered scenario the entire dataset is though too small (or not diverse enough) and needs more data-efficient models. For a limited amount of data, Transformers may be prone to overfitting.

Another suspected reason is that some features like hand part assignment are local, and long-term dependencies are not that useful as they could be, or they are not encoded correctly. This indicates that the traditional trigonometric positional encoding does not serve the purpose well. Some other form of positional encodings, that represent an actual onsets more than the position in a tensor sequence, may significantly improve the quality of the model.

Recently, there were a successful use of Transformers in the field [BD24] on a subset of the considered dataset. The authors provide a RoFormer encoder-decoder model with a custom

token embedding. They also provided additional data augmentations as duration and onset jitter, introducing noise to the note beginning and ends. The authors modified the initial ACPAS dataset as they claim that a hand-crafted part of the is not representative for the performance.

The aforementioned work operates on different score format (MusicXML) and uses a different set of metrics than [LKMB22], making hard to compare both models without further investigation.

7.3.2. Temporal Convolutional Networks

TCNs provided a compelling alternative to recurrent architectures for certain transcription tasks. These networks are computationally efficient and maintain stability during training.

While TCNs achieved comparable results for hand part assignment and time signature models, their performance lagged in key signature prediction.

7.3.3. Dynamics

We enhanced the base model with a dedicated dynamics submodel to transcribe score dynamics, ranging from *pppp* to *ffff*. While this addition captures a significant aspect of musical expression, two major challenges arose:

- **Data scarcity:** None of the datasets provided direct dynamics annotations, requiring heuristic and automated processes to align performance data with score annotations.
- **Subjectivity:** Dynamics markings are context-dependent and subjective, varying by instrument, genre, and performer interpretation. As a result, the relationship between a musical sheet and its realization is far from strict.

The dynamics model demonstrated an average, but reasonable under above circumstances, performance.

We proposed an extension of the MV2H metric, denoted MV2HD, which incorporates an F_1 score for dynamics. This enhancement maintains disjoint penalty principles and provides a more comprehensive evaluation of transcription quality. However, the metric is yet to be implemented.

Future work should focus on refining dynamics annotations in the dataset, implementation of the MV2HD metric, and experimenting with alternative architectures, such as Transformers and TCNs, may yield further improvements.

Bibliography

- [ANS19] André Araujo, Wade Norris, and Jack Sim. Computing receptive fields of convolutional neural networks. 4(11), 11 2019.
- [Ass96] MIDI Manufacturers Association. *Complete MIDI 1.0 Detailed Specification*, 1996.
- [BD24] Tim Beyer and Angela Dai. End-to-end piano performance-midi to score conversion with transformers. 09 2024.
- [BDDE19] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Process. Mag.*, 36(1):20–30, 2019.
- [BDG⁺13] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, December 2013.
- [BL23] Bhuwan Bhattarai and Joonwhoan Lee. A comprehensive review on music transcription. *Applied Sciences*, 13(21), 2023.
- [BLEW03] Adam Berenzweig, Beth Logan, Daniel P.W. Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. *Computer Music Journal*, 28, November 2003.
- [BMV⁺23] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audioldm: A language modeling approach to audio generation. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 31:2523–2533, June 2023.
- [CA10] Michael Scott Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In J. Stephen Downie and Remco C. Veltkamp, editors, *ISMIR*, pages 637–642. International Society for Music Information Retrieval, 2010.

- [Cam00] Emílios Cambouropoulos. From midi to traditional musical notation. 2000.
- [Cam08] Emílios Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception - MUSIC PERCEPT*, 26:75–94, 09 2008.
- [CD17] Andrea Cogliati and Zhiyao Duan. A metric for music notation transcription accuracy. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 407–413, 2017.
- [Cho39] Frédéric Chopin. Prélude Opus 28 No. 4 in E minor. <https://musescore.com/classicman/chopin-opus-28-no-4>, 1839. [Online; accessed 01-March-2024].
- [CPK⁺15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2015.
- [CTD16] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing human piano performances into music notation. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 758–764, 2016.
- [DBK⁺21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
- [DHdR89] Peter Desain, Henkjan Honing, and Klaus de Rijk. A connectionist quantizer. In *Proceedings of the 1989 International Computer Music Conference, ICMC 1989, Columbus, Ohio, USA, November 2-5, 1989*. Michigan Publishing, 1989.
- [DHP14] Zhiyao Duan, Jinyu Han, and Bryan Pardo. Multi-pitch streaming of harmonic sound mixtures. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22:138–150, January 2014.
- [dOO17] Hélio de Oliveira and Raimundo Oliveira. Understanding midi: A painless tutorial on midi format. May 2017.

- [Dow03] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37(1):295–340, 2003.
- [FMR⁺20] Francesco Foscarin, Andrew McLeod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. ASAP: a dataset of aligned scores and performances for piano transcription. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 534–541, 2020.
- [GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. <http://www.deeplearningbook.org>.
- [GCM14] Harald Grohganz, Michael Clausen, and Meinard Müller. Estimating musical time information from performed midi files. In *International Society for Music Information Retrieval Conference*, 2014.
- [Goo01] Michael Good. Musicxml: An internet-friendly format for sheet music. January 2001.
- [Har10] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.
- [HBKW21] Andre Holzapfel, Emmanouil Benetos, Andrew Killick, and Richard Widdess. Humanities and engineering perspectives on music transcription. *Digital Scholarship in the Humanities*, 37(3):747–764, 10 2021.
- [HKMMT89] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Philippe Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. *Wavelets, Time-Frequency Methods and Phase Space*, -1:286, 01 1989.
- [HNY21] Yuki Hiramatsu, Eita Nakamura, and Kazuyoshi Yoshii. Joint estimation of note values and voices for audio-to-score piano transcription. In Jin Ha Lee, Alexander Lerch, Zhiyao Duan, Juhan Nam, Preeti Rao, Peter van Kranenburg, and Ajay Srinivasamurthy, editors, *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, pages 278–284, 2021.
- [HRP⁺22] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

- [Hub07] D.M. Huber. *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*. Audio Engineering Society Presents Series. Focal Press, 2007.
- [IBWW97] Vijay Iyer, Jeff Bilmes, Matt Wright, and David Wessel. A novel representation for rhythmic structure. In *Proceedings of the 23rd International Computer Music Conference*, pages 97–100. Citeseer, 1997.
- [ISO75] ISO/IEC. *Acoustics. Standard tuning frequency (Standard musical pitch)*. January 1975.
- [ISO08] ISO/IEC. *Coding of audio-visual objects*. February 2008.
- [JM09] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J., 2009.
- [Kas66] Michael Kessler. Toward musical information retrieval. *Perspectives of New Music*, 4(2):59–67, 1966.
- [KNP⁺07] Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Proceedings 4th Sound and Music Computing Conference (SMC’2007)*, 2007.
- [KP94] Stefan Kostka and Dorothy Payne. *Tonal Harmony with an Introduction to Twentieth-Century Music*. McGraw-Hill Education, New York, 8 edition, 1994.
- [Kru96] Bernd Krueger. Classical piano midi, 1996.
- [KSLB18] Jong Kim, Justing Salamon, Peter Li, and Juan Bello. Crepe: A convolutional representation for pitch estimation. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, April 1988.
- [LKMB22] Lele Liu, Qiuqiang Kong, Veronica Morfi, and Emmanouil Benetos. Performance midi-to-score conversion by neural beat tracking. In Preeti Rao, Hema A. Murthy, Ajay Srinivasamurthy, Rachel M. Bittner, Rafael Caro Repetto, Masataka Goto, Xavier Serra, and Marius Miron, editors, *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022*, pages 395–402, 2022.
- [LMB21] Lele Liu, Veronica Morfi, and Emmanouil Benetos. Acpas: a dataset of aligned classical piano audio and scores for audio-to-score transcription. 2021.

- [LVRH16] Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 47–54, Cham, 2016. Springer International Publishing.
- [McL19] Andrew McLeod. Evaluating non-aligned musical score transcriptions with MV2H. *CoRR*, abs/1906.00566, 2019.
- [MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [MS18] Andrew McLeod and Mark Steedman. Evaluating automatic polyphonic music transcription. In Emilia Gómez, Xiao Hu, and Eric Humphrey, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pages 42–49, November 2018. 19th International Society for Music Information Retrieval Conference, ISMIR 2019 ; Conference date: 23-09-2018 Through 27-09-2018.
- [NBYD18] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. pages 101–105, 04 2018.
- [NYD17] Eita Nakamura, Kazuyoshi Yoshii, and Simon Dixon. Note value recognition for piano transcription using markov random fields. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(9):1846–1858, September 2017.
- [NYK17] Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, pages 347–353. International Society for Music Information Retrieval, 2017. Publisher Copyright: © 2019 Eita Nakamura, Kazuyoshi Yoshii, Haruhiro Katayose.; 18th International Society for Music Information Retrieval Conference, ISMIR 2017 ; Conference date: 23-10-2017 Through 27-10-2017.
- [NYS17] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. Rhythm transcription of polyphonic piano music based on merged-output hmm for multiple voices. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(4):794–806, April 2017.

- [Ori06] Nicola Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1:1–, November 2006.
- [RD23] Xavier Riley and Simon Dixon. Crepe notes: A new method for segmenting pitch contours into discrete notes. *arXiv preprint arXiv:2311.08884*, 2023.
- [Rea69] Gardner Read. *Music Notation: A Manual of Modern Practice*. Crescendo book. Allyn and Bacon, 1969.
- [RMH⁺14] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P.W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, pages 367–372, 2014.
- [Roe22] Thomas Roelleke. *Information Retrieval Models: Foundations and Relationships*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Springer International Publishing, 2022.
- [RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.
- [Rud76] Walter Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976.
- [SAL⁺24] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [SC78] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:159–165, 1978.
- [Set05] William Sethares. *Tuning, Timbre, Spectrum, Scale*. January 2005.
- [SGU14] Markus Schedl, Emilia Gómez, and Julián Urbano. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval*, 8(2-3):127–261, 2014.
- [SL09] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45:427–437, 07 2009.

- [Smi99] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1999.
- [SND12] Pierre Schaeffer, Christine North, and John Dack. *In Search of a Concrete Music*. California Studies in 20th-Century Music. University of California Press, 2012.
- [Suz21] Masahiro Suzuki. Score transformer: Generating musical score from note-level representation. In Changwen Chen, Helen Huang, Jun Zhou, Tatsuya Harada, Jianfei Cai, Wu Liu, and Dong Xu, editors, *MMAasia '21: ACM Multimedia Asia, Gold Coast, Australia, December 1 - 3, 2021*, pages 31:1–31:7. ACM, 2021.
- [Tea02] The LilyPond Development Team. *LilyPond — Essay on automated music engraving*. LilyPond Software, 2002.
- [Tem09] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.
- [TSO⁺02] Haruto Takeda, Naoki Saito, Tomoshi Otsuki, Mitsuro Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. Hidden markov model for automatic transcription of midi signals. In *2002 IEEE Workshop on Multimedia Signal Processing.*, pages 428–431, 2002.
- [VBDB10] Valentin, Nancy Bertin, Bertrand David, and Roland Badeau. MAPS - A piano database for multipitch estimation and automatic transcription of music. Research report, July 2010.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [Wal13] Chris Walshaw. *Abc musical notation — standard version 2.2*. ABC Community, 2013.
- [Wik24a] Wikipedia. Dynamics (music) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Dynamics_\(music\)](https://en.wikipedia.org/wiki/Dynamics_(music)), 2024. [Online; accessed 01-December-2024].

- [Wik24b] Wikipedia. Music information retrieval — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Music%20information%20retrieval&oldid=1189049395>, 2024. [Online; accessed 01-December-2024].
- [Wik24c] Wikipedia. Sheet music — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Sheet%20music&oldid=1210471824>, 2024. [Online; accessed 01-December-2024].
- [YB18] Adrien Ycart and Emmanouil Benetos. A-maps: Augmented maps dataset with rhythm and key annotations. In *19th International Society for Music Information Retrieval Conference, ISMIR, Late Breaking and Demos Papers.*, 2018.
- [YCV05] Aaron Yang, Elaine Chew, and Anja Volk. A dynamic programming approach to adaptive tatum assignment for rhythm transcription. In *Seventh IEEE International Symposium on Multimedia (ISM 2005), 12-14 December 2005, Irvine, CA, USA*, pages 577–584. IEEE Computer Society, 2005.
- [ZNFW21] Hongyuan Zhu, Ye Niu, Di Fu, and Hao Wang. Musicbert: A self-supervised learning of music representation. In *Proceedings of the 29th ACM International Conference on Multimedia, MM '21*, page 3955–3963, New York, NY, USA, 2021. Association for Computing Machinery.