PiRhDy: Learning Pitch-, Rhythm-, and Dynamics-aware Embeddings for Symbolic Music

Hongru Liang
Institute of Big Data
College of Computer Science
Nankai University
lianghr@mail.nankai.edu.cn

Wenqiang Lei*
National University of Singapore
wenqianglei@gmail.com

Paul Yaozhu Chan
Institute for Infocomm Research,
A*STAR, Singapore
ychan@i2r.a-star.edu.sg

Zhenglu Yang
Institute of Big Data
College of Computer Science
Nankai University
yangzl@nankai.edu.cn

Maosong Sun
Tsinghua University
sms@tsinghua.edu.cn

Tat-Seng Chua
National University of Singapore
chuats@comp.nus.edu.sg

Abstract

Definitive embeddings remain a fundamental challenge of computational musicology for symbolic music in deep learning today. Analogous to natural language, music can be modeled as a sequence of tokens. This motivates the majority of existing solutions to explore the utilization of word embedding models to build music embeddings. However, music differs in two key aspects from natural languages: (1) musical token is multi-faceted – it comprises of pitch, rhythm and dynamics information simultaneously; and (2) musical context is two-dimensional – each musical token is dependent on the surrounding tokens from both melodic and harmonic contexts. In this work, we attempt to provide a comprehensive solution by proposing a novel framework named PiRhDy that integrates pitch, rhythm, and dynamics information seamlessly. PiRhDy adopts a hierarchical strategy which can be decomposed into two steps: (1) token (i.e., note event) modeling, which separately represents pitch, rhythm, and dynamics and integrates them into a single token embedding; and (2) context modeling, which utilizes melodic and harmonic knowledge to train the token embedding. To examine our method, we make a thorough study by decomposing PiRhDy on components and strategies. We further validate our embeddings in three downstream tasks – melody completion, accompaniment suggestion, and genre classification. We demonstrate that our PiRhDy embeddings significantly outperform the baseline methods on both sequence- and song-level tasks.

CCS Concepts

• **Information systems** → *Music retrieval*; • **Computing methodologies** → **Learning latent representations**.

*Corresponding author

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3414032

Keywords

Symbolic Music, Representation Learning, Embeddings

ACM Reference Format:
Hongru Liang, Wenqiang Lei, Paul Yaozhu Chan, Zhenglu Yang, Maosong Sun, and Tat-Seng Chua. 2020. PiRhDy: Learning Pitch-, Rhythm-, and Dynamics-aware Embeddings for Symbolic Music. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3414032

1 Introduction

With the evolution of digital devices, learning symbolic music becomes an emerging topic in computational musicology with numerous real-world applications like audio-score alignment [20], music generation [7], song recommendation [6], etc. It is critical to mine useful information from symbolic music, because the quality of data representations directly influences the success of machine learning algorithms [2]. In this work, we are interested in an efficient way to represent the nature of symbolic music, that is, embed key music information into a shared low-dimensional space. In this way, the understanding of complex music can be formulated as the processing of these computational representations.

Symbolic music is the most direct and reliable intermediary from the obscure music sheets to dulcet mechanized sounds [21]. It is similar to natural language in many aspects [13, 17]. For example, both contain sequential tokens and are context-dependent. Hence, several recent efforts of symbolic music embeddings focus on investigating the potential application of word embeddings techniques (e.g., CBOW and skip-gram [18, 19]) towards music. The surface form (i.e., musical token) towards the concept of "word" in symbolic music involves fixed-length slice (i.e., note event) [4, 10], a set of notes (i.e., chord) [12, 15], and a sequence of notes (i.e., motif) [1, 11]. Amongst these, the training paradigm predicts either the center musical token from context (CBOW) or the context from the center token (skip-gram).

Although these works have produced seemingly promising music embeddings, they are far from satisfactory due to the inability to capture the special characteristics of music. In particular, music differs in two key aspects from natural languages: (1) musical tokens

Table 1: Existing embeddings w.r.t. musical information (the pitch, rhythm and dynamics columns) and context (the melody and harmony columns) they model.

music embeddings	pitch	rhythm	dynamics	melody	harmony
chordripple [12]	✓			✓	
chord2vec [15]	✓			✓	
Herremans and Chuan [10]	✓			✓	✓
Chuan et al. [4]	✓			✓	✓
melody2vec [11]	✓	✓		✓	
Alvarez and Gómez-Martin [1]	✓			✓	

is a combination of multi-faceted features including pitch, rhythm, and dynamics; and (2) music is multi-dimensional constitutionally with its melodic context progressed in the horizontal axis and harmonic context organized in the vertical axis. Thus, we argue that two fundamental problems need to be solved: 1) *how to leverage pitch, rhythm, and dynamic features simultaneously*, and 2) *how to encode both melodic and harmonic contexts comprehensively*.

To the best of our knowledge, there is still no unified framework that addresses these problems comprehensively. A brief summary current music embeddings are listed in Table 1. For information utilization, with the exception of melody2vec [11], existing methods focus on the pitch information. Few consider the dynamics, which carries the variations in the loudness of the music and is one of the most expressive elements of music [22]. This makes these methods unable to distill enough features for general tasks. Another observation is that aside from the note-event based approaches [4, 10], most embeddings do not model harmonic context, which contains the vertical knowledge of music. This this leaves the vertical dimension of music unaccounted for, leaving them incapable to learn complete knowledge from musical contexts.

Considering the limitations of existing solutions, we believe that it is critical to develop a framework that not only integrates multi-faceted features of musical tokens but also transfers knowledge from both melody and harmony into embeddings. Therefore, we propose a hierarchical framework consisting of two-stage modeling aligned with the fundamental problems. First, we design a token (note event) modeling network to fuse pitch, rhythm, and dynamic features seamlessly. This network is built on our delicately designed musical vocabulary and consists of several efficient strategies to extract vital information. For example, pitch modeling is developed to fuse chroma and octave features into pitch information. Secondly, we build a context modeling network that can predict the probability distribution of music thoroughly. In this network, music embeddings are pre-trained at the token-level (i.e., note event-level) context and fine-tuned at the sequence-level (i.e., period-level and track-level) context. In this way, both short-term and long-term relations are encoded into the embeddings.

Our study contributes to a comprehensive understanding of symbolic music by learning computational embeddings. As far as we know, this is the first music embedding approach that leverages key musical features as well as knowledge from both the melodic and harmonic contexts. The main contributions of this work are:

- An integrated framework (PiRhDy) that can learn pitch-, rhythm-, and dynamics-aware embeddings for symbolic music from both melodic and harmonic contexts.
- An extensive study of PiRhDy and demonstration of the necessity of integrating key features, the effectiveness of utilizing comprehensive contexts, and the robustness of our embeddings.
- An examination of the ability of PiRhDy embeddings to capture musical knowledge on tasks at different levels, that is, the sequence-level (melody completion, accompaniment assignment) and song-level (genre classification) tasks.

2 Related work

The low-dimensional embeddings in symbolic music can be separated into approaches based on chord, note event and motif, which correspond to the concept of "word" applied to music.

Chord-based approaches [12, 15] aim to learn chord (a set of simultaneous notes) representations in the word2vec models. However, chords, from the accompaniment track, are purely aiding components of notes from the melody track. In other words, they only contribute to harmony. Hence, these works cannot generate universal embeddings for symbolic music. Besides, the study of chords requires all the chord attributes to be annotated and thus needs the help of experts.

Note event-based approaches treat music as an organized sequence of note events, which are the smallest unit of naturalistic music [4, 10]. However, these works only train embeddings on the most frequent "words" to overcome the long-tail issues caused by huge vocabularies. For example, only 500 out of 4075 notes events are considered in [4], leading to incomplete learning of prior distribution and knowledge from a corpus.

Motif-based approaches [1, 11] keep tracks of the sequences of notes that may be referred to as motifs. Although motifs are the most similar in concept to words in natural language, there are no established dictionaries for motifs in music. Towards the study of motifs, [1] redefines motif as fixed-length pitch intervals, which only cover melodic information. Alternatively, [11] extracts motifs from melody tracks using the generative theory of tonal music [14], which is a rule-based approach. Either way, the learning procedure is limited in the melody track neglecting rich information in the accompaniment tracks. Moreover, similar to note-event based approaches, the long-tail distribution of motifs remains a problem. For example, more than half of the motifs cannot be sufficiently trained in melody2vec [11].

Besides contextual embeddings, there is also a trend to represent symbolic music as a sparse two-dimensional matrix. Conventional **matrix-based approaches** transform MIDI files into piano rolls [8, 25] (i.e. binary-valued time-pitch matrices). Specifically, if a bar is sliced into 96 time steps horizontally and 128 pitch values vertically, the size of piano-roll matrix of this bar is 96×128 [27]. Notably, instead of manipulating data sequentially, [5] integrates knowledge from music theory (i.e., tonnetz network) by formatting a slice of music into a binary-valued $4 \times 24 \times 12$ matrix. Thus, such representation naturally contains tonal relationships (e.g., ordered according to the circle-of-fifths), yet at the cost of losing the temporal information among the notes in the slice. Both pianoroll and tonnetz representations are more likely to meet the challenge of limited computational and storage resources than low-dimensional dense embeddings.

In addition to Table 1, we review the utilization of various information in matrix-based approaches. Both the piano roll [8, 25]

Table 2: Overview of useful terms

term	notation	definition
note event	n	the playing of a note at a certain time span (duration).
pitch	P	a set of human-defined numbers describing the frequency degree of music sound, e.g., A4.
chroma	c	a.k.a, pitch class, the octave-invariant value of a pitch. For example, both C3 and C4 refer to chroma C (*do*), and the pitch interval from C3 to C4 indicates an octave (notated as o).
chord	–	a set of two or more simultaneous notes.
duration	–	the temporal length of a musical note, e.g., 1/4.
onset	–	the beginning of a musical note.
note state	s	the current state of a note event, involving *on*, *hold*, and *off* indicating the beginning, playing, and ending of a note, respectively.
inter-onset-interval	i	abbreviated to IOI, the duration between the onsets of two consecutive notes.
rhythm	R	the temporal pattern of notes, related to note duration, note onset, and IOI.
dynamics	D	the variation in loudness between notes or phrases.
velocity	v	the dynamics markings (e.g., "*mf*" and "*p*").
motif	–	a.k.a, motivate, a group of note events forming the smallest identifiable musical idea, normally one-bar long.
phrase	*ph.*	a group of motifs forming a complete musical sense, being normally four-bar long.
period	*per.*	a pair of consecutive phrases.
melody	–	a sequence of single notes from the melody track[1], which plays the most impressive sounds among the entire song.
harmony	–	the simultaneous or overlapping notes on accompaniment tracks to produce an overall effect along with the melody.

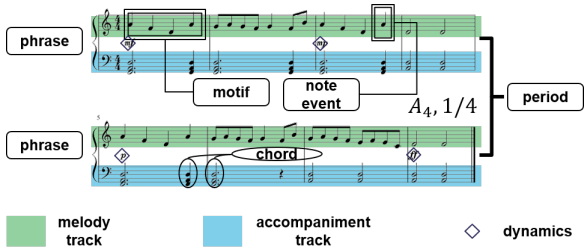

Figure 1: Example of music scores of a song, which consists of one period, a melody track and an accompaniment track.

and tonnetz [5] can model melody and harmony simultaneously. However, the tonnetz only leverages pitch features. While the piano roll utilizes both pitch and rhythm information apart from melody2vec [11]. None of these methods untilize adequate information to encode music.

Altogether, there are four issues in existing approaches: cost of expert annotation, cost of computational and storage resources, incomplete prior distribution learning, and inadequate information learning. As a result, it remains challenging to 1) *leverage pitch, rhythm, and dynamic features simultaneously* and 2) *encode both melodic and harmonic contexts comprehensively*. In contrast to the above-mentioned embeddings, we plan to learn distributed embeddings for symbolic music from scratch. We carefully construct a very concise vocabulary based on pitch, rhythm, and dynamics without regards to expert knowledge in order to learn robust embeddings from a limited corpus. We optimize the embeddings on both melodic and harmonic contexts in order to transfer both horizontal and vertical knowledge from music into the embeddings.

3 Preliminary

Table 2 presents an overview of useful terms in this paper. Besides, we illustrate part of these terms in Figure 1. In favour of [4, 10], we study music as note events but from a more delicate perspective (see section 4). In practice, we need to integrate the pitch, rhythm, and dynamic information of note events.

First of all, pitch information is translates to chroma and the octave. For example, $a4$ can be expressed as the chroma a on the 1/4 octave group. As such, we define the pitch information as $P = \mathcal{F}_P(c, o)$, where $\mathcal{F}_P$ is the fusing function.

Secondly, the rhythm information is related to note duration, inter-onset-interval (IOI), and note state. We only use IOI and note state values when modeling rhythm, because the note duration values are may derived competely from note events. Thus, we define the rhythm information as $R = \mathcal{F}_R(i, s)$, where $\mathcal{F}_R$ is the fusing function.

Third, the dynamics information is expressed as velocity values. We formulate it as $D = \mathcal{F}_D(v)$, where $\mathcal{F}_D$ is the processing function.

Therefore, the study of note events is summarised in the modeling of chroma, octave, IOI, note state, and velocity features. This may be expressed as $n = \mathcal{F}(\mathcal{F}_P(c, o), \mathcal{F}_R(i, s), \mathcal{F}_D(v))$, where $\mathcal{F}$ denotes the integrating function. Note that, for convenience, we reuse some of these notations accented by a right arrow to represent the distributed representation of responding information or values. For example, we use $\vec{c}$ to indicate the distributed representation of c. Besides, we use d to represent the dimension of vectors, and W, b, U to represent trainable parameters.

In this paper, we consider symbolic music formatted as musical instrument digital interface (MIDI) files, which consists of multiple chunks of information, directing the computing platforms to play the music properly. To accommodate the subsequent procedures, we conduct a series of modifications of the original MIDI files: 1) Inspired by [5, 11], we normalize the keys of the song by transposing them to C Major. This enables us to learn stable information about notes and music in the same tonal space. 2) In theory, the melody track is made up of a sequence of single notes. As such, we define the track with the highest average pitch values as the melody track. We also split the accompaniment tracks into a series of sub-tracks by the octave to confine the chords to the same octave space. 3) MIDI files encode music using three kinds of time units (seconds, ticks, and beats). To avoid ambiguity, we normalize all kinds of time values into the music notation used in the scores, that is, the temporal value is represented in the form of semibreves (whole notes), minims (1/2 notes), crotchets (1/4 notes), etc.

4 Methodology

We realize the proposed PiRhDy framework in two stages, as illustrated in Figure 2. The first is a token modeling network to simultaneously fuse the pitch, rhythm, and dynamics information. This network consists of an input module extracting key features from note events based on a concise vocabulary, an embedding module encoding each feature into dense embeddings and a fusion module integrating all embeddings into a unified representation.

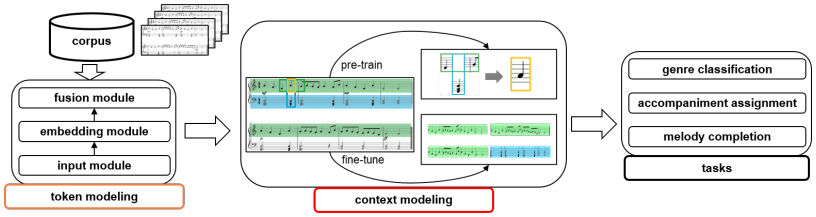

Figure 2: Architecture of PiRhDy

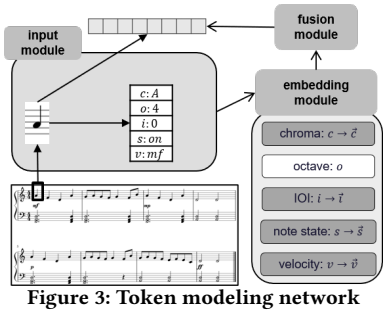

Figure 3: Token modeling network

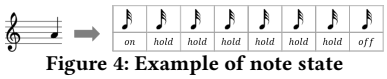

Figure 4: Example of note state

The second is a context modeling network to capture the hidden relationship and distribution property of music at the sequence-level. In this network, the pre-trained embeddings are smoothed over both horizontal (melodic) and vertical (harmonic) contexts. Apart from this, we introduce the fine-tune strategy on global contexts to inject long-term contextual information into the embeddings.

4.1 Token Modeling Network

In this work, we treat the musical token as a single note event, which is the smallest meaningful variable of music. A note event is a combination of pitch, rhythm, and dynamics information. The basic idea of the token modeling network is to extract key features from a note event and encode such information into a unified embedding of this musical token, as displayed in Figure 3.

Input module

The input module is to extract pitch, rhythm, and dynamic features from a music token. To this end, we define a comprehensive vocabulary consisting of all related information. Specifically, the entire vocabulary consists of three fundamental vocabularies:

- **Chroma vocabulary** is defined as $\mathbb{C} = \{c, c\#, d, d\#, e, f, g, g\#, a, a\#, b, R\}$, where R denotes the chroma value of the empty note event. We combine chords into the chroma vocabulary because the simultaneous notes presented in a chord may all be represented in the croma. For example, the C major chord $\{c, e, g\}$ is represented as the symbol C in the vocabulary.
- **Velocity vocabulary** is defined as $\mathbb{V} = \{pppp, ppp, pp, p, mp, mf, f, ff, fff, ffff, r\}$, where the number of p and f indicate the level of softness and loudness respectively, m indicates *moderation* and r represents the velocity of an empty note event.
- **Note state vocabulary** is defined as $\mathbb{S} = \{on, hold, off, r\}$, where *on*, *hold* and *off* denote the start, sustain, and end actions of a note respectively; and r represents the actions of an empty note event. As shown in Figure 4, if a note event is defined as a 1/32 note long, the note duration of $a4$ is computed as $t_{off} - t_{on}$, note onset is t_{on}, and IOI is $t'_{on} - t_{on}$, where t'_{on} represents the onset of the next note. As such, we can get all rhythm-related information from the note state alphabet within only four unique symbols.

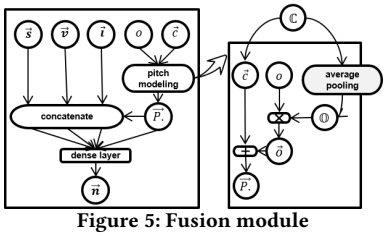

Figure 5: Fusion module

As mentioned in section 3, the features of a single note event contain five dimensions including its chroma, octave, IOI, note state, and velocity. In practice, the IOI is computed from the current and a selected note's events (onsets). Take the first 1/4 note event in Figure 3 as an example, the IOI value against itself is 0, against the second 1/4 note event is −1, and so on. The chroma and octave values make up the pitch information, while the IOI, and note state features form the rhythm information. The velocity features constitute the dynamic information.

Embedding module

The embedding module encodes all properties extracted from the input module into distributed vectors. For chroma, note state, and velocity, which are defined in the vocabulary, we use an embedding layer to encode the information into dense vectors (i.e., $\vec{c}$, $\vec{s}$, and $\vec{v}$). The embedding layer outputs the product of the high-dimensional one-hot representations of chroma, velocity, and note state with an embedding matrix $W_E \in \mathbb{R}^{V \times d}$, where V indicates the vocabulary size and d is the dimension of vectors. Considering how the concept of IOI reflects the relative positions between node events, we use the trigonometric position encoding [28] to obtain the IOI embeddings. Octave value (o) is not computed in this module and fed into the next module (i.e., note module) directly.

Fusion module

The fusion module (Figure 5) integrates vectors produced from the embedding module. Considering how pitch is composed of chroma and octave, we introduce a pitch modeling technique to obtain the dense representation of pitch ($\vec{P}$) from o and $\vec{c}$. We exploit the fact that musical notes that are linearly spaced on the MIDI scale are in fact logarithmically spaced on the frequency scale [3]. Assuming octave information is shared on all chromas, and map the average pooling on all chroma embeddings from $\mathbb{C}$ to get the octave representation $\mathbb{O}$. The octave representation ($\vec{o}$) of a note event is defined as $o \times \mathbb{O}$, where $\times$ is the scalar multiplication. Accordingly, the pitch representation is defined as $\vec{c} + \vec{o}$. We deploy a dense layer on the concatenation of $\vec{P}$, $\vec{i}$, $\vec{v}$, and $\vec{s}$ to generate the unified representation of a note event.

4.2 Context Modeling Network

The context modeling network smooths the embeddings on both melodic and harmonic contexts. To this end, we build a local context model and a global context model to capture short-term and long-term knowledge, respectively. In practice, we employ a hierarchical strategy which pre-trains the embeddings on the local context model (token-level), and fine-tunes these embeddings on the melodic and harmonic global contexts(sequence-level), separately. In this way, we can obtain the melody-preferred and harmony-preferred embeddings accordingly.

Local context module

Inspired by word2vec [18, 19], we develop a token-level module to handle local contextual information through sliding windows. The local context module (Figure 6) aims to predict the center note event (yellow part) from both melodic (green part) and harmonic(blue part) contexts.

We define the state value for center note event from the melody track to be *on*. Melodic context is the set of note events with state values *on* that surrounds the center note event from the melody track. Harmonic context comprise of note events with the same IOI value as the center note event from the accompaniment tracks. As such, the value of IOI is computed between the current note event and the center note event. Unlike the position value in word embeddings, the IOI in this module may not be continuous. Besides, it is not necessary to consider the constant IOI information when encoding the harmonic context. As illustrated in Figure 6, if we set a note event to be a 1/32 note long, and take $\{0, a, 4, mf, on\}$ as the center note event. The melodic context is $\{(-8, d, 4, mf, on), (8, g, 4, mf, on)\}$ and the harmonic context is $\{(d-f-b, 4, mf, on)\}$, both with window size set as 1. In practice, the harmonic window is always much larger than the melodic window.

The core of the module is a recurrent neural network (RNN)- and attention-based layers. The RNN-based layer is used to capture the temporal information over the context and the attention layer is used to balance the importance of different note events. The RNN-based layer is defined as $RnnL : \mathcal{H} := [h_k]_{k=1}^m \in \mathbb{R}^{d\times m} \rightarrow [t_k]_{k=1}^m \in \mathbb{R}^{d\times m}$, expressed as $t_k = \sigma(Wh_k + Uh_{k-1} + b)$, where $\mathcal{H}$ is the input matrix and built by a stack of m vectors h_k, t_k is the target vector learned by the layer, *sigma* is an activate function, and U is a trainable parameter like W and b. Similarly, we define the attention-based layer as $AttL : \mathcal{H} := [h_k]_{k=1}^m \in \mathbb{R}^{d\times m} \rightarrow t \in \mathbb{R}^d$, expressed as $t = \sum_{k=1}^m \alpha_k h_k$, $\alpha_k = \frac{exp\left(\hat{h}_k{}^\top \hat{h}_c\right)}{\sum_{j=1}^m exp\left(\hat{h}_j{}^\top \hat{h}_c\right)}$, $\hat{h}_k = tanh(Wh_k + b)$, where α_k is the attention weight scalar of h_k and evaluated by the latent vector $\hat{h}_k$ of h_k along with the randomly initialized context vector $\hat{h}_c$, and t is the target vector.

We encode melodic context in a network (denoted as *PiRhDy-melody*), where vectors from the token module are fed into *RnnL* layers to get the temporal information of the melodic context. The output of *RnnL* layers is followed by an *AttL* layer to generate the hidden representation of the melodic context. Consider that the harmonic note events share the same IOIs, the network for harmonic context (denoted as *PiRhDy-harmony*) slightly modifies the melodic network by removing the *RnnL* layers.

Global context module

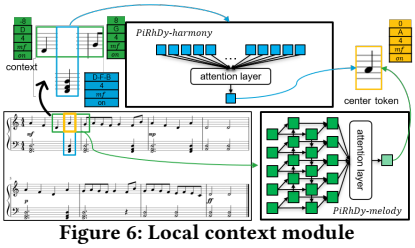

Figure 6: Local context module

The global context module is a sequence-level network smoothing embeddings on long-term contexts with the fine-tune strategy. Similar to the local context module, we define the center phrase as a phrase from the melody track, the melodic context as the following phrase of the center phrase, and the harmonic context as the set of phrases from the accompaniment tracks corresponding to the center phrase. Either context is encoded into phrase representations through the *PiRhDy-melody* network. Specifically, we model global melodic and harmonic contexts separately, so that we can learn the melody-preferred and harmony-preferred embeddings.

To encode global contexts, we design the period encoder and track encoder for melodic context and harmonic context, respectively. The period encoder starts from *PiRhDy-melody*, followed by phrase-level *RnnL*s and *AttL* sequentially, to encode the melodic context into a dense representation of the period. This encoder is written as $\vec{per}. = AttL\{RnnL[PiRhDy\text{-}melody(ph.{}_k)_{k=1}^n]\}$, where $ph.{}_k$ is a sequence of note event vectors for phrase k, n is the number of input phrases, $\vec{per}.$ is the latent representation of the period. Following the local modeling network, the track encoder is a modified period encoder without the *RnnL*s. As such, the track encoder is formulated as $\vec{TRK} = AttL[PiRhDy\text{-}melody(ph.{}_k)_{k=1}^n]$, where $\vec{TRK}$ is the representation of vertical phrase-level context.

Optimization

We aim to estimate the probability distribution of symbolic music from both horizontal and harmonic contexts locally and globally. To this end, we formulate the loss function of local modeling as

$$\mathcal{L} = f\Big(\sum \mathcal{L}_h, \sum \mathcal{L}_v\Big), \tag{1}$$

$$\mathcal{L}_h = -\sum_{-w_h \le k \le w_h, k\neq 0} \mathcal{P}\Big[(p_k, c_k, o_k, v_k, s_k)\,|\,(p_0, c_0, o_0, v_0, s_0)\Big], \tag{2}$$

$$\mathcal{L}_v = -\sum_{-w_v \le k \le w_v, k\neq 0} \mathcal{P}\Big[(c_k, o_k, v_k, s_k)\,|\,(c_0, o_0, v_0, s_0)\Big], \tag{3}$$

where $\mathcal{L}_h$ denotes the loss calculated from the melodic contexts, w_h is the sliding window size on melodic contexts, and $\mathcal{L}_v$ and w_v serve the same functions as $\mathcal{L}_h$ and w_h in the harmonic context, respectively. Additionally, we introduce $f(*)$ as a weighting function to balance the importance of melodic and harmonic costs. As for global modeling, we aim to maximize the probability of $\sum \mathcal{P}(ph. * |ph.)$, where $ph.*$ is either the following phrase or an accompaniment phrase of a given phrase $ph.$, depending on the fine-tuning method.

For fast and stable optimization, inspired by word2vec [18, 19], we adopt the negative sampling strategy for both pre-training and fine-tuning. Notably, towards local modeling, we consider a novel four-level method instead of sampling a negative sample for each positive sample. Practically, for every positive sample, we randomly replace one, two, three, and all features except for IOIs. In other words, there are four negative samples for a positive sample labeled

Table 3: Overview of the model names. LN indicates the line number.

LN	name	description
1	*chroma*	the base model for the study of features only leverages the chroma features. Contexts involve both melody and harmony. The fusion operation follows the weighted function EQ. 1.
2	*chroma + octave*	leverages both chroma and octave features
3	*chroma + IOI*	leverages both chroma and IOI features
4	*chroma + note state*	leverage both chroma and note state features
5	*chroma + velocity*	leverages both chroma and velocity features
6	*PiRhDy-melody*	leverages all features on the melodic context
7	*PiRhDy-harmony*	leverages all features on the harmonic context
8	*PiRhDy-AVG*	leverages all features except for IOI on both melodic and harmonic contexts following the average operation, that is, reformulating EQ 1 as $\mathcal{L} = \frac{1}{2}(\sum \mathcal{L}_h + \sum \mathcal{L}_v)$
9	*PiRhDy-WT* or PiRhDy	leverages all features on both melodic and harmonic contexts following the weighted operation, that is, EQ 1
10	*PiRhDy-CP*	leverages cartesian product-based vocabulary

Table 4: The effects of *features* (LN 1-5), *contexts* (LN 6-7), and *fusion operations* (LN 8-9) demonstrated by binary cross-entropy values under different proportions of training data size (e.g., 20% in the table means using 20% of the training corpus).

LN	model	training data size				
		20%	40%	60%	80%	100%
1	*chroma*	0.5079	0.5078	0.5076	0.5076	0.5076
2	*chroma + octave*	0.3639	0.3637	0.3636	0.3635	0.3636
3	*chroma + IOI*	0.5078	0.5074	0.5073	0.5069	0.5071
4	*chroma + note state*	0.3445	0.344	0.3439	0.3436	0.3426
5	*chroma + velocity*	0.4049	0.4045	0.4049	0.4048	0.4048
6	*PiRhDy-melody*	**0.0645**	**0.0627**	**0.0627**	**0.0619**	0.0619
7	*PiRhDy-harmony*	0.3693	0.3534	0.3617	0.3613	0.3548
8	*PiRhDy-AVG*	0.1384	0.1364	0.1359	0.1357	0.1362
9	*PiRhDy-WT*	0.0668	0.0633	**0.0627**	**0.0619**	**0.0617**

as $[1, 1, 1, 1]$. The optimization turns to a multi-label classification task rather than a binary classification task of word2vec. In this way, the model can be sensitive to all the features during the training procedure. As such, $\mathcal{L}$ can be estimated through the binary cross-entropy as

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{4}\left[y_k^j \log(\hat{y}_k^j) + (1 - y_k^j)\log(1 - \hat{y}_k^j)\right], \quad (4)$$

where N is the number of samples, y is the ground truth value, and $\hat{y}$ is the predicted value.

5 Settings

Following melody2vec [11], we use the Lakh MIDI dataset [24], which is a collection of 178,561 unique MIDI files. For efficient learning, we omitted melodic phrases with less than 75% of valid note events, harmonic phrases with less than 50% of valid note events, and songs with less than two periods. After preprocessing, we obtained 643 unique chords. Thus, the sizes of chroma, velocity and note state vocabularies are 656 (13+643), 11 and 4, respectively, resulting in an entire vocabulary containing 671 symbols, which is much lesser than melody2vec (286,003 symbols).

6 Experiment I: Study of PiRhDy

To comprehensively examine PiRhDy embeddings, we designed two sets of empirical evaluations. In this section, we conduct a detailed study of each strategy set in the local context module by decomposing the proposed framework. In this next section (i.e., Section 7), we further examine the robustness and effectiveness of our embeddings fine-tuned on global contexts by designing both sequence-level and song-level downstream tasks. To analyze the impact of strategies proposed in this work, we randomly split the corpus into training (90%) and testing (10%), and design a detailed study about features, contexts, fusion operations, training data size, and vocabularies. All these studies are built on the local context module (section 4.2). Table 3 describes the model names used in this section, we also call the "*PiRhDy-WT*" model as "PiRhDy". Following [15], we report the binary cross-entropy values of EQ. (4) on the test data in Tables 4 and 5.

Study of features

As explained in section 3, a single note event consists of chroma, octave, IOI, note state, and velocity features. We deploy a base model (denoted as *chroma*), which only leverages the chroma feature input. We choose chroma as the base feature because it is the only feature used by all related works. The performance of other features are studied by adding them to base model. For instance, we use the model (*chroma + octave*) to denote the adding of octave values to chroma as input. Note that this note representation is equal to the pitch representation. The remaining features are studied in the same way as the octave features. Apart from this, *PiRhDy-WT* is the corresponding model that leverages all features (i.e., chroma, octave, IOI, note state, and velocity).

From the top part (LN 1-5) of Table 4, we can see that the addition of note state to the chroma model achieves the best performance, even though it contains only four symbols. The addition of octave gives the next best results, indicating the significance of pitch information (the combination of chroma and octave) in music. The addition of velocity, expresses the loudness of each note provides the next best improvement. The least improvement is seen in the addition of IOI, which expresses the relative position of note events from the melodic context. Both velocity and IOI slightly contribute to predicting the center note event. We conjecture that this is because the dataset contains little variation in velocity changes, limiting our learning of its variation. IOI values were insufficient to capture a significant pattern. Further to this, from the last line (*PiRhDy-WT*) of Table 4, we can see that even better results, indicating that it is best to use all features together.

Observation 1: It is beneficial to integrate pitch, rhythm, and dynamics information, namely, fusing chroma, octave, IOI, note state, and velocity features.

Study of contexts

There are two types of contexts, namely, melodic context from the melody track and harmonic context from the accompaniment track. Towards the study of these contexts, we leverage all features in the melodic context. Note that, however, only the chroma, octave, note state, and velocity features are leveraged in the harmonic context. This is because the IOI feature is the same for all note events of the harmonic context.

From the middle part (LN 6-7) of Table 4, we can see that *PiRhDy-melody* always works better than *PiRhDy-harmony*. This indicates that the center note event is more related to melodically surrounding note events than harmonically surrounding ones.

Observation 2: Melodic context contributes more to the center note event than the harmonic context. Besides, these contexts may possess different weights for the local distribution of music.

Study of fusion operations

Based on *Observation 2*, we defined a weighted fusion operation in EQ. (1). To study the impact of EQ. (1), we conduct the average operation on the outputs of *PiRhDy-melody* with *PiRhDy-harmony* as a baseline method, denoted as *PiRhDy-AVG*. Similarly, we refer to

the model following the weighting function EQ. (1) as *PiRhDy-WT*. The results are presented in the bottom part (LN 8-9) of Table 4.

PiRhDy-WT works better than the average operation to fuse the results encoded from melody and harmony. This implies that melodic and harmonic contexts have different importances when predicting the center note event, and confirms *Observations 2* again. The performances of *PiRhDy-melody* is inferior to *PiRhDy-WT*, but superior to *PiRhDy-AVG*. This suggests it is necessary to find an efficient way to integrate melodic and harmonic knowledge.

Observation 3: Melodic and harmonic contexts contribute differently to predicting the center note event. It is necessary to balance the importance of melodic and harmonic knowledge appropriately.

Study of training data size

We conduct the study of training data size (Table 4) on all feature- (e.g., *chroma*), context- (e.g., *PiRhDy-melody*) and fusion operation-related (e.g., *PiRhDy-AVG*) models. This study is conducted by gradually increasing the percentage of training corpus from 20% to 100%, while the testing data size remains the same.

Generally, a larger training data size produces better performances, and feature-related models are converge more readily than others. For example, *chroma* converges with 60% of the training corpus, while *PiRhDy-melody* needs more than 80% of the corpus to converge. We conjecture that this is because the learned information of feature-related models is less than that of others. Note that, when no more than 40% of the training data is used, *PiRhDy-melody* is the best approach; otherwise, PiRhDy is the best. We argue this is because significant amounts of harmonic information is needed for sufficient coverage in training. Consequentially, inefficient harmonic information decreases the performances of approaches based on both melodic and harmonic contexts.

Observation 4: Larger corpus can produce more efficient embeddings, and the improvements shrink with larger data size.

Study of vocabularies

As explained in section 4.1, we build our feature-based vocabulary as a combination of chroma, velocity, and note state vocabularies. In this way, a music token (note event) is represented as a five-dimensional variable, involving chroma, octave, IOI, note state, and velocity features. We learn the embeddings of features and obtain the token representation by combining such embeddings accordngly. A natural alternative is to build the vocabulary on the cartesian products of chroma, octave, note state and velocity features, and embed such cartesian products into vectors. Similar approaches have been explored in [4, 10], yet only using the pitch feature. We name the model using cartesian product-based vocabulary *PiRhDy-CP*, and re-conduct the PiRhDy model on 1%, 5%, and 10% of the training corpus. We compare the effectiveness of embeddings derived from the feature-based vocabulary and that derived from the cartesian product-based vocabulary in Table 5. Note that, in theory, the vocabulary size of *PiRhDy-CP* is 346,368 (656 chromas, 12 octaves, 11 velocities, and 4 note states), of which we find 216,031 unique occurrences in the entire corpus.

PiRhDy works better than *PiRhDy-CP* in all conditions, especially when training data size is small. In detail, PiRhDy can produce remarkable and almost converging results when the percentage is more than 20%. However, the seemingly competitive results made by *PiRhDy-CP* need more than 80% of training data. In addition, the parameters of *PiRhDy-CP* (56,723,500) is more than 30 times larger than PiRhDy (1,854,252). This means that the former is more difficult to train than the latter.

Table 5: Study of vocabularies

datasize	1%	5%	10%	20%	40%	60%	80%	100%
PiRhDy-CP	0.4536	0.4106	0.2805	0.2004	0.1353	0.1172	0.0959	0.0804
PiRhDy	0.3395	0.1512	0.1017	0.0668	0.0633	0.0627	0.0619	0.0617

Observation 5: The feature-based vocabulary is better than the cartesian product-based one on both robustness and effectiveness.

7 Experiment II: Downstream Tasks

In section 6, we studied the strategies set in the local context module. In this section, we deploy sequence- and song-level downstream tasks to evaluate the PiRhDy embeddings fine-tuned in the global context. We use *PiRhDy-WT*, the best performing local modeling strategy as discussed in section 6. To clarify, we denote the embeddings fine-tuned on global melodic context as *PiRhDy_GM* (i.e., melody-preferred embeddings), and embeddings fine-tuned on harmonic context as *PiRhDy_GH* (i.e., harmony-preferred embeddings). We evaluate these embeddings on the melody completion, accompaniment assignment and genre classification tasks.

7.1 Baseline Models

According to our survey, all existing contextual embeddings [1, 4, 10–12, 15] for symbolic music are based on word2vec [18, 19], and vary in the surface formats. For an efficient comparison, we choose melody2vec [11] as our baseline model, because it is the state-of-the-art technique, leveraging both pitch and duration information, while also utilizing the Lakh dataset as its corpus. We deploy both forward and backward maximum matching algorithms to conduct motif segmentation over symbolic music. Together with the matrix-based approaches, we summarize the baseline models as

- **melody2vec_F** [11], melody2vec embeddings with the forward maximum matching algorithm.
- **melody2vec_B** [11], melody2vec embeddings with the backward maximum matching algorithm.
- **tonnetz** [5], the music bar is represented as a tonnetz matrix.
- **pianoroll** [8], the music bar is represented as a time-pitch matrix.

For melody2vec, we leverage the embeddings released by [11]. For tonnetz and pianoroll, we apply convolutional neural networks following [5] and [8], respectively.

7.2 Melody Completion

Inspired by [16], we design a melody completion task defined as follows: "*Given a melodic phrase ($ph._q$) and a set of candidate melodic phrases ($[ph.]_{melody}$), find its most related consecutive phrase from $[ph.]_{melody}$.*" This is a sequence-level (period-level) evaluation task, with a concrete application of *PiRhDy-melody*. There are 1,784,844 pairs in the training dataset, and 199,270 pairs in the testing dataset. For each $ph._q$ in the training dataset, we randomly select a phrase to form a negative sample. For the testing dataset, we generate 49 negative samples for each positive sample. We use mean average precision (MAP) and hits@k (k=1, 5, 10, 25, indicating the rate of the candidates containing the correct next phrase) as evaluation metrics, and report the results in Figure 7.

Our embeddings (*PiRhDy_GH* and *PiRhDy_GM*) perform better than melody2vec and tonnetz, suggesting that PiRhDy can capture

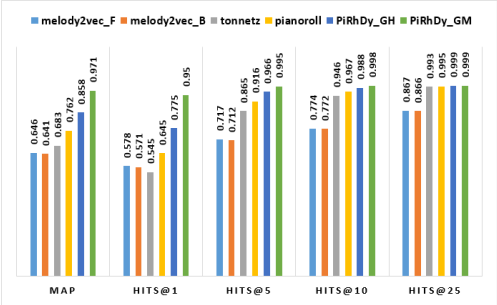

Figure 7: Results of melody completion task

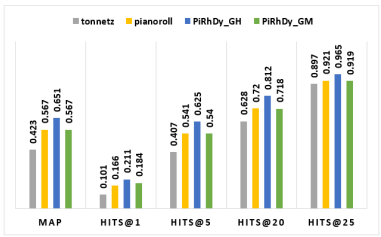

Figure 8: Results of accompaniment assignment task

melodic continuity. It is worth mentioning that *PiRhDy_GH* is outperformed by *PiRhDy_GM*, indicating that the vertical distribution is not as significant as the horizontal distribution of music. Thus, it is necessary to model globally vertical and harmonic contexts separately in specific tasks.

7.3 Accompaniment Assignment

Similar to the melody completion task, we formulate the accompaniment assignment task as follows: "*Given a melodic phrase ($ph._q$) and a set of candidate harmonic phrases ($[ph.]_{harmony}$), find its most related accompaniment phrase from $[ph.]_{harmony}$.*" This is also a sequence-level (track-level) task, with a concrete application of *PiRhDy-harmony*. There are 7,890,554 phrase pairs in the training set, and 200,300 samples in the testing set. For each $ph._q$, there are more than one correct candidates from $[ph.]_{harmony}$. Thus, in the training dataset, we randomly generate a negative sample for each positive sample. In the testing set, for each $ph._q$ with N correct accompaniment phrases, we randomly generate $(50 - N)$ phrases as negative samples. We use MAP and hits@k as metrics, and report the results in Figure 8. We omit the melody2vec approach [11] because its embeddings only work for melodic phrases.

We observe that both *PiRhDy_GH* and *PiRhDy_GM*, perform better than baseline models. This demonstrates that *PiRhDy* is best at capturing vertical continuity of music. We also observe that even *PiRhDy_GM*, which only learns vertical information from the local context, gains remarkable results as compared to the results of *PiRhDy_GH* in the next phrase prediction task. This implies that the relationship between locally and globally vertical context is stronger than that of the horizontal context.

7.4 Genre Classification

Genre classification is a song-level task. Following Ferraro and Lemström [9], we conduct evaluations on TOP-MAGD and MASD datasets [26]. TOP-MAGD contains 22,535 files of 13 genres while MASD contains 17,785 files of 25 styles. We use AUC-ROC and F1 scores as metrics, and 5-fold cross-validation for evaluation in line with [9, 23]. Table 6 presents the results of different methods. SIA and P2 are baseline models proposed in [9].

Table 6: Results of genre classification task

Model	Top-MAGD		MASD	
	AUC-ROC	F1	AUC-ROC	F1
SIA	0.753	0.637	0.761	0.455
P2	0.816	0.649	0.815	0.431
melody2vec_F	0.885	0.649	0.777	0.299
melody2vec_B	0.883	0.647	0.776	0.293
tonnetz	0.871	0.627	0.794	0.253
pianoroll	0.876	0.64	0.774	0.365
PiRhDy_GH	0.891	0.663	0.782	0.448
PiRhDy_GM	**0.895**	**0.668**	**0.832**	**0.471**

As can be seen from Table 6, our *PiRhDy_GM* embeddings outperform all methods on both metrics and both datasets, while *PiRhDy_GH* come in second. These performances indicate that our proposed embeddings are also suitable for song-level tasks in addition to period- and track-level tasks. The more significant performances of *PiRhDy_GM* against *PiRhDy_GH* suggests that melody outweighs harmony in music genre. The results on MASD are less remarkable than on TOP-MAGD, because the styles are sub-genres of genres in TOP-MAGD. In other words, the genre classification on MASD is more challenging. Both *PiRhDy_GH* and *PiRhDy_GM* generate competitive results, showing the scalability and robustness of PiRhDy embeddings.

8 Conclusion

In this paper, we proposed a comprehensive framework (PiRhDy) that embeds music from scratch. The framework is built on a hierarchical strategy that handles music from token-level to sequence-level. We first designed a token modeling network to fuse key features into unified token representations. Secondly, we develop a context modeling network (sequence-level) to smooth embeddings on global contexts . The experimental results validate the robustness and effectiveness of our PiRhDy embeddings.

We hope insights from PiRhDy will inspire future developments of computational musicology. PiRhDy embeddings can be applied to numerous other areas related to deep symbolic music learning. We plan to further explore embeddings smoothed on the rhythm and tonal patterns, which is analogous to the syntax of natural languages. This study can help understand and interpret the structural characteristics of music. Another future direction would be to inject the audio content into the learning process, further enabling embeddings smoothed on multimodal content. Additionally, we would also like to further investigate the potential of our embeddings in tasks like music similarity matching and music generation.

9 Acknowledgments

This work was supported in part by the Ministry of Education of Humanities and Social Science project under grant 16YJC790123 and National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Aitor Arronte Alvarez and Francisco Gómez-Martin. 2019. Distributed Vector Representations of Folksong Motifs. In *Proceedings of the International Conference on Mathematics and Computation in Music*. Springer, 325–332.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[3] Sebastian Böck and Markus Schedl. 2012. Polyphonic piano note transcription with recurrent neural networks. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 121–124.

[4] Ching-Hua Chuan, Kat Agres, and Dorien Herremans. 2020. From context to concept: exploring semantic relationships in music with word2vec. *Neural Computing and Applications* 32, 4 (2020), 1023–1036.

[5] Ching-Hua Chuan and Dorien Herremans. 2018. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Proceedings of the 32nd AAAI conference on artificial intelligence*.

[6] Khalil Damak and Olfa Nasraoui. 2019. SeER: An Explainable Deep Learning MIDI-based Hybrid Song Recommender System. *arXiv preprint arXiv:1907.01640* (2019).

[7] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[8] Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang. 2018. Pypianoroll: Open source Python package for handling multitrack pianoroll. In *Late-Breaking Demos of the 18th International Society for Music Information Retrieval Conference*.

[9] Andres Ferraro and Kjell Lemström. 2018. On Large-Scale Genre Classification in Symbolically Encoded Music by Automatic Identification of Repeating Patterns. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology* (Paris, France) *(DLfM '18)*. Association for Computing Machinery, New York, NY, USA, 34–37. https://doi.org/10.1145/3273024.3273035

[10] Dorien Herremans and Ching-Hua Chuan. 2017. Modeling Musical Context With Word2Vec. In *Proceedings of the First International Conference on Deep Learning and Music*. 11–18.

[11] Tatsunori Hirai and Shun Sawada. 2019. Melody2Vec: Distributed Representations of Melodic Phrases based on Melody Segmentation. *Journal of Information Processing* 27 (2019), 278–286.

[12] Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z Gajos. 2016. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. 241–250.

[13] Ray Jackendoff. 2009. Parallels and nonparallels between language and music. *Music Perception: An Interdisciplinary Journal* 26, 3 (2009), 195–204.

[14] Fred Lerdahl and Ray S Jackendoff. 1996. *A generative theory of tonal music*. MIT press.

[15] Sephora Madjiheurem, Lizhen Qu, and Christian Walder. 2016. Chord2vec: Learning musical chord embeddings. In *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems*.

[16] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 195–204.

[17] Erin McMullen and Jenny R Saffran. 2004. Music and language: A developmental comparison. *Music Perception: An Interdisciplinary Journal* 21, 3 (2004), 289–311.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the advances in neural information processing systems*. 3111–3119.

[20] Nicola Montecchio and Arshia Cont. 2011. A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 193–196.

[21] Kia Ng, Kia Ng, and Paolo Nesi. 2007. *Interactive Multimedia Music Technologies*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA.

[22] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. 2018. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications* (2018), 1–13.

[23] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. 2017. Multi-Label Music Genre Classification from Audio, Text and Images Using Deep Features. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull (Eds.). 23–30.

[24] Colin Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Ph.D. Dissertation. Columbia University.

[25] Colin Raffel and Daniel PW Ellis. 2014. Intuitive analysis, creation and manipulation of midi data with pretty midi. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*. 84–93.

[26] Alexander Schindler, Rudolf Mayer, and Andreas Rauber. 2012. Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*. 469–474.

[27] Joan Serra, Meinard Müller, Peter Grosche, and Josep Lluis Arcos. 2012. Unsupervised detection of music boundaries by time series structure features. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.