MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training

Mingliang Zeng, Xu Tan,[*] Rui Wang, Zeqian Ju, Tao Qin, Tie-Yan Liu
Microsoft Research Asia
{v-minzeng,xuta,ruiwa,v-zeju,taoqin,tyliu}@microsoft.com

Abstract

Symbolic music understanding, which refers to the understanding of music from the symbolic data (e.g., MIDI format, but not audio), covers many music applications such as genre classification, emotion classification, and music pieces matching. While good music representations are beneficial for these applications, the lack of training data hinders representation learning. Inspired by the success of pre-training models in natural language processing, in this paper, we develop MusicBERT, a large-scale pre-trained model for music understanding. To this end, we construct a large-scale symbolic music corpus that contains more than 1 million music songs. Since symbolic music contains more structural (e.g., bar, position) and diverse information (e.g., tempo, instrument, and pitch), simply adopting the pre-training techniques from NLP to symbolic music only brings marginal gains. Therefore, we design several mechanisms, including OctupleMIDI encoding and bar-level masking strategy, to enhance pre-training with symbolic music data. Experiments demonstrate the advantages of MusicBERT on four music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification. Ablation studies also verify the effectiveness of our designs of OctupleMIDI encoding and bar-level masking strategy in MusicBERT.

1 Introduction

Music understanding, including tasks like genre classification, emotion classification, music pieces matching, has attracted lots of attention in both academia and industry. A better understanding of melody, rhythm, and music structure is not only beneficial for music information retrieval (Casey et al., 2008) but also helpful for music generation (Huang et al., 2018; Sheng et al., 2020). Similar to natural language, music is usually represented in symbolic data format (e.g., MIDI) (Jackendoff, 2009; McMullen and Saffran, 2004) with sequential tokens, and some methods (Mikolov et al., 2013a,b) from NLP can be adopted for symbolic music understanding. Since the labeled training data for each music understanding task is usually scarce, previous works (Liang et al., 2020; Chuan et al., 2020) leverage unlabeled music data to learn music token embeddings, similar to word embeddings in natural language tasks. Unfortunately, due to their shallow structures and limited unlabeled data, such embedding-based approaches have limited capability to learn powerful music representations.

In recent years, pre-trained language models (e.g., BERT) have been verified to be powerful for representation learning from large-scale unlabeled text corpora (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; Song et al., 2019; Brown et al., 2020; Song et al., 2020). However, it is challenging to directly apply the pre-training techniques from NLP to symbolic music because of the difference between natural text data and symbolic music data. First, since music songs are more structural (e.g., bar, position) and diverse (e.g., tempo, instrument, and pitch), encoding symbolic music is more complicated than natural language. The existing pianoroll-like (Ji et al., 2020) and MIDI-like (Huang and Yang, 2020; Ren et al., 2020) representations of a song are too long to be processed by pre-trained models. For example, the length of a music song encoded by REMI (Huang and Yang, 2020) has an average length of 15,679, as shown in Table 1. Due to the limits of computational resources, the length of sequences processed by a Transformer model is usually cropped to below 1,000. Thus such representations cannot capture sufficient information for song-level tasks. Accord-

[*] Corresponding author: Xu Tan, xuta@microsoft.com

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 791–800
August 1–6, 2021. ©2021 Association for Computational Linguistics

ingly, an effective, efficient, and universal symbolic music encoding method is needed for music representation learning. Second, due to the complicated encoding of symbolic music, the pre-training mechanism (e.g., the masking strategy like the masked language model in BERT) should be carefully designed to avoid information leakage in pre-training. Third, as pre-training relies on large-scale corpora, the lack of large-scale symbolic music corpora limits the potential of pre-training for music understanding.

Encoding	OctupleMIDI	CP-like	REMI-like
Tokens	**3607**	6906	15679

Table 1: The average number of tokens per song on LMD dataset with different encoding methods.

In this paper, we develop MusicBERT, a large-scale pre-trained model with carefully designed music encoding and masking strategy for music understanding.

- We design a novel music encoding method called OctupleMIDI, which encodes each note into a tuple with 8 elements. These 8 elements represent the different aspects of the characteristics of a musical note, including time signature, tempo, bar, position, instrument, pitch, duration, and velocity. OctupleMIDI has several advantages: 1) It largely reduces the length of a music sequence (4x shorter than REMI (Huang and Yang, 2020) and 2x shorter than CP (Hsiao et al., 2021)), thus easing the modeling of music sequences by Transformer considering that music sequences themselves are very long. 2) It is note centric. Since each note contains the same 8-tuple structure and covers adequate information to express various music genres, such as changing time signature and long note duration, OctupleMIDI is much simpler and more universal than previous encoding methods.

- We carefully analyze the masking strategies for symbolic music understanding and propose a bar-level masking strategy for MusicBERT. The masking strategy in original BERT for NLP tasks randomly masks some tokens, which will cause information leakage in music pre-training. For example, some attributes are usually the same in a segment of consecutive tokens, such as time signature, tempo, instrument, bar, and position. Therefore, the masked tokens can be easily predicted by directly copying from the adjacent tokens since they are probably the same. Meanwhile, adjacent pitches usually follow the same chord so that a masked pitch token can be easily inferred from the adjacent tokens in the same chord. Therefore, we propose a bar-level masking strategy, which masks all the tokens of the same type (e.g., time signature, bar, instrument, or pitch) in a bar to avoid information leakage and encourage effective representation learning.

- Last but not least, we collect a large-scale and diverse symbolic music dataset, denoted as Million MIDI Dataset (MMD), that contains more than 1 million music songs, with different genres, including Rock, Electronic, Rap, Jazz, Latin, Classical, etc. To our knowledge, it is the largest in current literature, which is 10 times larger than the previous largest dataset LMD (Raffel, 2016) in terms of the number of songs as shown in Table 2. Thus, this dataset greatly benefits representation learning for music understanding.

We fine-tune the pre-trained MusicBERT on four downstream music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification, and achieve state-of-the-art results on all the tasks. Furthermore, ablation studies show the effectiveness of the individual components in MusicBERT, including the OctupleMIDI encoding, the bar-level masking strategy, and the large-scale corpus.

The main contributions of this paper are summarized as follows:

- We pre-train MusicBERT on a large-scale symbolic music corpus that contains more than 1 million music songs and fine-tune MusicBERT on some music understanding tasks, achieving state-of-the-art results.

- We propose OctupleMIDI, an efficient and universal music encoding for music understanding, which leads to much shorter encoding sequences and is universal for various kinds of music.

- We design a bar-level masking strategy as the pre-training mechanism for MusicBERT, which significantly outperforms the naive token-level masking strategy used in natural language pre-training.

2 Related Works

2.1 Symbolic Music Understanding

Inspired by word2vec (Mikolov et al., 2013a,b) in NLP, previous works on symbolic music understanding learn music embedding by predicting a music symbol based on its neighborhood symbols. Huang et al. (2016); Madjiheurem et al. (2016) regard chords as words in NLP and learn chords representations using the word2vec model. Herremans and Chuan (2017); Chuan et al. (2020); Liang et al. (2020) divide music pieces into non-overlapping music slices with a fixed duration and train the embeddings for each slice. Hirai and Sawada (2019) cluster musical notes into groups and regard such groups as words for representation learning. However, the word2vec-based approaches mentioned above only use relatively small neural network models and take only a few (usually 4-5) surrounding music tokens as inputs, which have limited capability compared with recently developed deep and big pre-trained models like BERT (Devlin et al., 2018), which takes a long sentence (e.g., with 512 words/tokens) as input. In this paper, we pre-train big/deep models over a large-scale music corpus and use more context as input to improve symbolic music understanding.

2.2 Symbolic Music Encoding

There are two main approaches to encode symbolic music: pianoroll-based and MIDI-based.

In pianoroll-based methods (Ji et al., 2020; Brunner et al., 2018), music is usually encoded into a 2-dimensional binary matrix, where one dimension represents pitches, and the other represents time steps. Each element in the matrix indicates whether the pitch is played at that time step. As a result, a note is always divided into multiple fixed intervals, which is inefficient, especially for long notes.

MIDI is a technical standard for transferring digital instrument data. Many works in symbolic music (Oore et al., 2020; Huang et al., 2018) encode music pieces based on MIDI events, including note-on, note-off, time-shift, etc. REMI (Huang and Yang, 2020) improves the basic MIDI-like encoding using note-duration, bar, position, chord, and tempo. Inspired by REMI (Huang and Yang, 2020), PopMAG (Ren et al., 2020) and Compound Word (CP) (Hsiao et al., 2021) compress the attributes of a note, including pitch, duration, and velocity, into one symbol and reduces duplicated position events. Although such MIDI-like approaches avoid redundancy for long notes, they still need multiple tokens to represent the attributes, position, and metadata of a single note, which can be further compressed. This paper proposes OctupleMIDI, a MIDI-based encoding method, which is efficient due to the reduced sequence length and universal to support various music genres.

2.3 Masking Strategies in Pre-training

Masking strategies play a key role in NLP pre-training. For example, BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) randomly mask some tokens in an input sequence and learn to predict the masked tokens. Furthermore, since adjacent tokens may form a word or a phrase, some works consider masking consecutive tokens. For example, MASS (Song et al., 2019) randomly masks a fragment of several consecutive tokens in the input, and SpanBERT (Joshi et al., 2020) randomly masks contiguous spans instead of tokens. However, symbolic music is different from language. First, symbolic music contains structural (e.g., bar, position) and diverse information (e.g., tempo, instrument, and pitch), while natural language can be regarded as homogeneous data, which only contains text. Second, music and language follow different rules. Specifically, the language rules include grammar and spelling, while the music rules include beat, chord, etc. Accordingly, the masking strategies for symbolic music need to be specifically designed; otherwise, it may limit the potential of pre-training because of information leakage, as we analyzed before. In this paper, we carefully design a bar-level masking strategy for symbolic music pre-training.

3 Methodology

In this section, we introduce MusicBERT, a large-scale Transformer model for symbolic music understanding. We first overview the model structure and then describe the OctupleMIDI encoding and masking strategy for pre-training. At last, we describe the large-scale music corpus with over 1 million songs used in MusicBERT pre-training.

3.1 Model Overview

As shown in Figure 1, MusicBERT pre-trains a Transformer encoder (Vaswani et al., 2017; Devlin et al., 2018), with masked language modeling where some tokens in the input music sequence are masked and are predicted in the model output. To

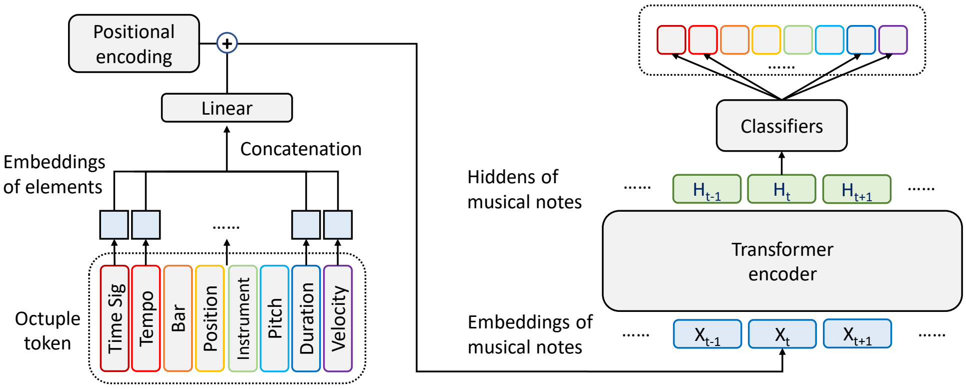


Figure 1: Model structure of MusicBERT.

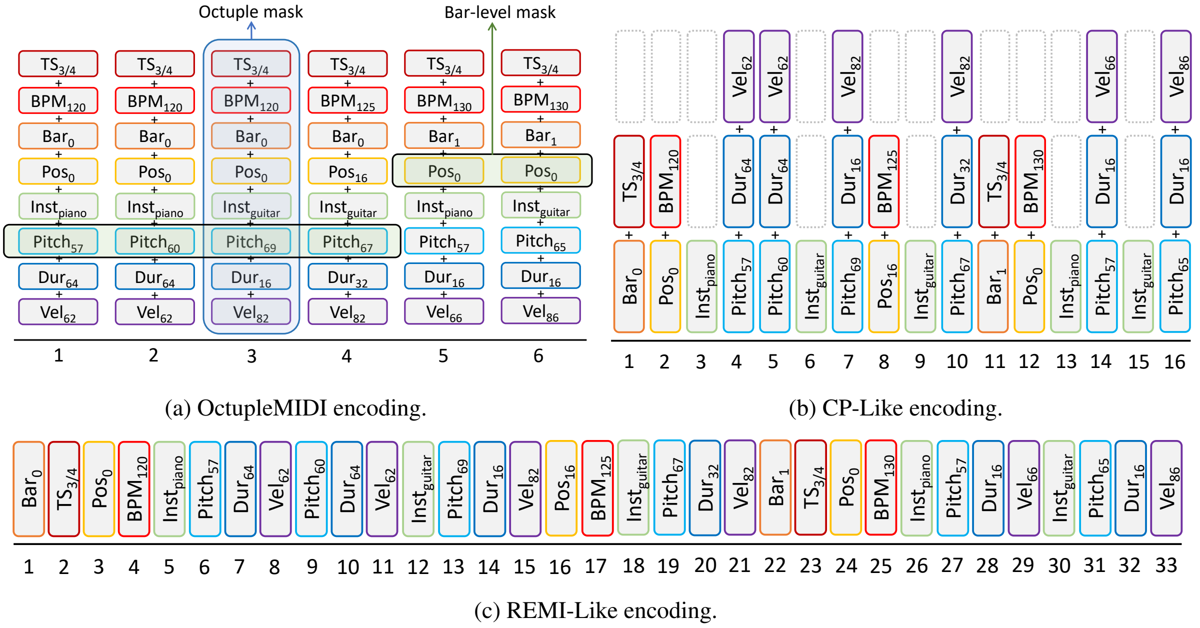
encode the music sequence more efficiently, we propose a novel encoding method called OctupleMIDI, which encodes a symbolic music piece into a sequence of octuple tokens (an 8-tuple) that contains 8 basic elements related to a music note (we introduce OctupleMIDI in detail in Sec. 3.2). To convert the octuple tokens in each sequence step into the input of the Transformer encoder, we concatenate the embeddings of the 8 elements and use a linear layer to convert them into a single vector. Then, the converted vector is added with the corresponding position embeddings and taken as the input of the Transformer encoder. To predict each of the 8 tokens in the 8-tuple from the Transformer encoder, we add 8 different softmax layers to map the hidden of the Transformer encoder to the vocabulary sizes of 8 different element types, respectively.

3.2 OctupleMIDI Encoding

Previous works (Liang et al., 2020) encode the symbolic music in a pianoroll-like way, which is not efficient since a note is always divided into multiple fixed small intervals (e.g., a quarter note is represented with 8 consecutive tokens). MIDI-like approaches (Huang and Yang, 2020; Ren et al., 2020; Hsiao et al., 2021) encode a note into several tokens based on MIDI events, making encoding much shorter, and has been widely used in music generation tasks. However, previous MIDI-like representations are still long for the Transformer structure due to computation complexity and learning efficiency. Accordingly, we propose a compact symbolic music encoding method called OctupleMIDI for music understanding tasks. As shown in Fig. 2, OctupleMIDI encodes 6 notes into 6 tokens, which is much shorter than 33 tokens with REMI (Huang and Yang, 2020) and 16 tokens with CP (Hsiao et al., 2021). Meanwhile, OctupleMIDI is general for various kinds of music. For example, OctupleMIDI supports changeable time signature and tempo.

In OctupleMIDI, we use sequences of octuple tokens to represent symbolic music. Each octuple token corresponds to a note and contains 8 elements, including time signature, tempo, bar, position, instrument, pitch, duration, and velocity. We introduce the details of each element as follows:

- Time signature. A time signature is denoted as a fraction (e.g., 2/4), where the denominator is a power of two in range [1, 64], representing the length of a beat (measured by note duration, e.g., a quarter note in 2/4), and the numerator is an integer in range [1, 128], representing the number of beats in a bar (e.g., 2 beats in 2/4). The value of the fraction measures the duration of a bar normalized by a whole note (e.g., 2/4 means that the duration of a bar is a half note). We consider the duration of a bar is no more than two whole notes. Otherwise, we divide a long bar into several equal-duration bars no longer than two whole notes. Therefore, there are 254 different valid time signatures in OctupleMIDI.
- Tempo. Tempo is measured in beats per minute (BPM), which describes the pace of music. In most music samples, tempo values are in range [24, 200]. For OctupleMIDI encoding, we quantize tempo values to 49 different values from 16 to 256, forming a geometric sequence.
- Bar and position. We use bar and position to indicate the on-set time of a note hierarchically. In the coarse level, we use 256 tokens ranging from 0 to 255 to represent the bar, supporting up to 256 bars in a music piece, which is sufficient in most cases. In the fine-grained level (inside each bar),


Octuple mask
Bar-level mask
$TS_{3/4}$ BPM_{120} Bar_0 Pos_0 $Inst_{piano}$ $Pitch_{57}$ Dur_{64} Vel_{62}
$TS_{3/4}$ BPM_{120} Bar_0 Pos_0 $Inst_{piano}$ $Pitch_{60}$ Dur_{64} Vel_{62}
$TS_{3/4}$ BPM_{120} Bar_0 Pos_0 $Inst_{guitar}$ $Pitch_{69}$ Dur_{16} Vel_{82}
$TS_{3/4}$ BPM_{125} Bar_0 Pos_{16} $Inst_{guitar}$ $Pitch_{67}$ Dur_{32} Vel_{82}
$TS_{3/4}$ BPM_{130} Bar_1 Pos_0 $Inst_{piano}$ $Pitch_{57}$ Dur_{16} Vel_{66}
$TS_{3/4}$ BPM_{130} Bar_1 Pos_0 $Inst_{guitar}$ $Pitch_{65}$ Dur_{16} Vel_{86}
1 2 3 4 5 6
Bar_0 $TS_{3/4}$
Pos_0 BPM_{120}
$Inst_{piano}$
$Pitch_{57}$ Dur_{64} Vel_{62}
$Pitch_{60}$ Dur_{64} Vel_{62}
$Inst_{guitar}$
$Pitch_{69}$ Dur_{16} Vel_{82}
Pos_{16} BPM_{125}
$Inst_{guitar}$
$Pitch_{67}$ Dur_{32} Vel_{82}
Bar_1 $TS_{3/4}$
Pos_0 BPM_{130}
$Inst_{piano}$
$Pitch_{57}$ Dur_{16} Vel_{66}
$Inst_{guitar}$
$Pitch_{65}$ Dur_{16} Vel_{86}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Bar_0 $TS_{3/4}$ Pos_0 BPM_{120} $Inst_{piano}$ $Pitch_{57}$ Dur_{64} Vel_{62} $Pitch_{60}$ Dur_{64} Vel_{62} $Inst_{guitar}$ $Pitch_{69}$ Dur_{16} Vel_{82} Pos_{16} BPM_{125} $Inst_{guitar}$ $Pitch_{67}$ Dur_{32} Vel_{82} Bar_1 $TS_{3/4}$ Pos_0 BPM_{130} $Inst_{piano}$ $Pitch_{57}$ Dur_{16} Vel_{66} $Inst_{guitar}$ $Pitch_{65}$ Dur_{16} Vel_{86}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

(a) OctupleMIDI encoding.

(b) CP-Like encoding.

(c) REMI-Like encoding.

Figure 2: Different encoding methods for symbolic music.

we use position with a granularity of 1/64 note to represent the on-set time of a note, starting from 0 in each bar. Therefore, we need 128 tokens to represent position since the duration of a bar is no more than two whole notes, as described above. For example, in a bar with a time signature of 3/4, the possible value of position is from 0 to 47.

- Instrument. According to the MIDI format, we use 129 tokens to represent instruments, where 0 to 127 stands for different general instruments such as piano and bass, and 128 stands for the special percussion instrument such as drum.

- Pitch. For notes of general instruments, we use 128 tokens to represent pitch values following the MIDI format. However, for notes of percussion instruments, there are no pitches but percussion types (e.g., bass drum, hand clap). Therefore, we use another 128 "pitch" tokens to represent percussion type for percussion instruments.

- Duration. To support the long note duration (up to 60 whole notes in the common music genre) with a fixed set of duration tokens, we propose a mixed resolution method: using high resolution (e.g., sixty-fourth note) when the note duration is small and using a low resolution (e.g., thirty-second note or larger) when the note duration is large. Specifically, we use 128 tokens to represent duration, starting from 0, with an increment of sixty-fourth note for the first 16 tokens, and double the increment (i.e., thirty-second note) every time for next 16 tokens. The duration for percussion instruments is meaningless, so we always set them to 0.

- Velocity. We quantize the velocity of a note in the MIDI format into 32 different values with an interval of 4 (i.e., 2, 6, 10, 14, . . . , 122, 126).

An example of a music sequence in OctupleMIDI encoding is shown in Fig. 2a.

3.3 Masking Strategy

Inspired by the masked language model in BERT (Devlin et al., 2018), we randomly mask some elements in the input sequence of octuple tokens and predict the masked ones. A naive masking strategy is to randomly mask some octuple tokens (mask all the elements in an octuple token), which is denoted as octuple masking as shown in Fig. 2a. However, considering the specific rules of music, such naive strategy will cause information leakage, and thus cannot well learn the contextual representation.

A music song consists of multiple bars, which can be regarded as highly internally related units. An octuple token can be easily inferred from the adjacent tokens in the same bar. Specifically, time signature, tempo, and bar usually remain the same in the same bar. Instrument and position values in the same bar follow regular patterns, where the instrument is limited to a small-scale fixed set of values, and the position values are non decreasing. Moreover, a chord is a fixed combination of pitches, which always appear in adjacent positions. Accordingly, we propose a novel bar-level masking strategy, where the elements with the same type in the same bar are regarded as a unit and are masked simultaneously. In this way, information leakage can be avoided, and better contextual representation can be learned through pre-training. An example of bar-level masking is shown in Fig. 2a.

For the masked elements, 80% of them are replaced with $[MASK]$, 10% of them are replaced with a random element, and 10% remain unchanged, following the common practice (Devlin et al., 2018; Joshi et al., 2020; Liu et al., 2019). Inspired by RoBERTa (Liu et al., 2019), we remove the next sentence prediction task in pre-training and adopt a dynamic masking strategy, where the masked sequence is generated every time when feeding a sequence to the model.

3.4 Pre-training Corpus

A large-scale music dataset is necessary to learn good music representations from pre-training. However, previous symbolic music datasets are usually of small scale: 1) the MAESTRO dataset (Hawthorne et al., 2019) contains only one thousand piano performances; 2) the GiantMIDI-Piano dataset (Kong et al., 2020a,b) contains slightly larger but still only ten thousands of piano performances; 3) the largest open-sourced symbolic music dataset by now is the Lakh-MIDI Dataset (LMD) (Raffel, 2016), which contains about 100K songs.

To pre-train a powerful model with good music representations, we build a large-scale symbolic music corpus with over 1.5 million songs. Specifically, we first crawled a large amount of music files, cleaned files that are malformed or blank, and then converted those files into our symbolic music encoding. Since OctupleMIDI encoding is universal, most MIDI files can be converted to our encoding without noticeable loss of musical information. We found that these music files may have almost identical music content even if their hash values are different. Therefore, we developed an efficient way to deduplicate them: we first omitted all elements except instrument and pitch in the encoding, then got hash values of the remaining sequence and use it as the fingerprint of this music file, which is further used for deduplication. After cleaning and deduplication, we obtained 1.5 million songs with 2 billion octuple tokens (musical notes). We denote our dataset as Million-MIDI Dataset (MMD). We compare the sizes of different music datasets in Table 2.

Dataset	Songs	Notes (Millions)
MAESTRO	1,184	6
GiantMIDI-Piano	10,854	39
LMD	148,403	535
MMD	**1,524,557**	**2,075**

Table 2: The sizes of different music datasets. Since LMD also consists of MIDI files from various websites, we perform the same cleaning and deduplication process as used in MMD and get 148,403 songs in LMD.

4 Experiments and Results

In this section, we first introduce the pre-training setup for MusicBERT, and then fine-tune MusicBERT on several downstream music understanding tasks to compare it with previous approaches. Finally, more method analyses are conducted to verify the effectiveness of the designs in MusicBERT.

4.1 Pre-training Setup

Model Configuration We pre-train two versions of MusicBERT: 1) MusicBERT$_{\text{small}}$ on the small-scale LMD dataset, which is mainly for a fair comparison with previous works on music understanding such as PiRhDy (Liang et al., 2020) and melody2vec (Hirai and Sawada, 2019), which are also pre-trained on LMD; 2) MusicBERT$_{\text{base}}$ on the large-scale MMD dataset, for pushing the SOTA results and showing the scalability of MusicBERT. The details of the two MusicBERT models are shown in Table 4. We use our proposed bar-level masking strategy with a masking probability of 15%. In addition, we use tokens of 8 duplicated elements to represent the class token and the end of sequence token. They are also masked with a 15% probability.

Pre-training Details The average sequence length of OctupleMIDI representation of a song is 3607 tokens as shown in Table 1, which is too long to model in Transformer. Therefore, we randomly sample segments with a length of 1024 tokens for pre-training. Following Liu et al. (2019), we pre-train MusicBERT on 8 NVIDIA V100 GPUs for 4 days, and there are 125,000 steps in total, with a

Model	Melody Completion					Accompaniment Suggestion					Classification	
	MAP	HITS @1	HITS @5	HITS @10	HITS @25	MAP	HITS @1	HITS @5	HITS @20	HITS @25	Genre F1	Style F1
melody2vec$_{F}$	0.646	0.578	0.717	0.774	0.867	-	-	-	-	-	0.649	0.299
melody2vec$_{B}$	0.641	0.571	0.712	0.772	0.866	-	-	-	-	-	0.647	0.293
tonnetz	0.683	0.545	0.865	0.946	0.993	0.423	0.101	0.407	0.628	0.897	0.627	0.253
pianoroll	0.762	0.645	0.916	0.967	0.995	0.567	0.166	0.541	0.720	0.921	0.640	0.365
PiRhDy$_{GH}$	0.858	0.775	0.966	0.988	0.999	0.651	0.211	0.625	0.812	0.965	0.663	0.448
PiRhDy$_{GM}$	0.971	0.950	0.995	0.998	0.999	0.567	0.184	0.540	0.718	0.919	0.668	0.471
MusicBERT$_{small}$	0.982	0.971	0.996	0.999	1.000	0.930	0.329	0.843	0.993	0.997	0.761	0.626
MusicBERT$_{base}$	**0.985**	**0.975**	**0.997**	**0.999**	**1.000**	**0.946**	**0.333**	**0.857**	**0.996**	**0.998**	**0.784**	**0.645**

Table 3: Results of different models on the four downstream tasks: melody completion, accompaniment suggestion, genre classification, and style classification. We choose four baseline models: Melody2vec (Hirai and Sawada, 2019) is widely used in music understanding tasks, tonnetz (Chuan and Herremans, 2018) and pianoroll (Dong et al., 2018) are classical methods for music representation, PiRhDy (Liang et al., 2020) is a new model that significantly outperforms previous models in all four downstream tasks. Results of melody2vec on accompaniment suggestion task are emitted since it only encodes melody part of music.

MusicBERT	small	base
Number of layers	4	12
Element embedding size	512	768
Hidden size	512	768
FFN inner hidden size	2048	3072
#Attention heads	8	12
Pre-training dataset	LMD	MMD

Table 4: The model configurations of MusicBERT.

batch size of 256 sequences, each has a maximum length of 1024 tokens. We use Adam (Kingma and Ba, 2014) optimizer with β_1=0.9, β_2=0.98, ϵ=1e-6 and L_2 weight decay of 0.01. The learning rate is warmed up over the first 25,000 steps to a peak value of 5e-4 and then linearly decayed. Dropout value on all layers and attention weights are set to 0.1.

4.2 Fine-tuning MusicBERT

We fine-tune MusicBERT on four downstream tasks: two phrase-level tasks (i.e., melody completion and accompaniment suggestion) and two song-level tasks (i.e., genre and style classification). For the two phrase-level tasks, the learning rate is warmed up over the first 50,000 steps to a peak value of 5e-5 and then linearly decayed until reaching 250,000 total updates. For the two song-level tasks, the learning rate is warmed up over the first 4,000 steps to a peak value of 5e-5 and then linearly decayed until reaching 20,000 total updates. The batch size is set to 64 sequences for both tasks. Other settings are the same as pre-training. We compare MusicBERT with previous works on symbolic music understanding, including PiRhDy (Liang et al., 2020) and melody2vec (Hirai and Sawada, 2019).

4.2.1 Melody Completion

Melody completion (Liang et al., 2020) is to find the most matched consecutive phrase in a given set of candidates for a given melodic phrase. There are 1,793,760 data pairs in the training set and 198,665 data groups in the test set in this task (Liang et al., 2020). Each training data pair consists of one positive sample and one negative sample, while each test data group consists of 1 positive sample and 49 negative samples. We use mean average precision (MAP) and HITS@k (k=1, 5, 10, 25, indicating the rate of correctly chosen phrase in the top k candidates) as evaluation metrics, making comparisons with PiRhDy (Liang et al., 2020), melody2vec (Hirai and Sawada, 2019), pianoroll (Dong et al., 2018), tonnetz (Chuan and Herremans, 2018). As shown in Table 3, MusicBERT$_{small}$ outperforms all previous works on the same pre-training dataset LMD, indicating the advantage of MusicBERT on learning representations from melodic context. MusicBERT$_{base}$ with a larger model and pre-training corpus can further achieve better results, showing the effectiveness of large-scale pre-training.

4.2.2 Accompaniment Suggestion

Accompaniment suggestion (Liang et al., 2020) is to find the most related accompaniment phrase in a given set of harmonic phrase candidates for a given melodic phrase. There are 7,900,585 data pairs in the training set, each consisting of one positive sample and one negative sample, and 202,816 data groups in the test set (Liang et al., 2020). Each group in the test set consists of N positive samples and (50-N) negative samples when there are N accompaniment tracks in the MIDI file of that sample. We use mean average precision (MAP) and HITS@k as metrics, making comparisons with PiRhDy (Liang et al., 2020), pianoroll (Dong et al., 2018), tonnetz (Chuan and Herremans, 2018). As shown in Table 3, MusicBERT models perform much better than previous works, indicating the advantages of MusicBERT in understanding harmonic context.

4.2.3 Genre and Style Classification

Genre classification and style classification (Ferraro and Lemström, 2018) are multi-label classification tasks. Following Ferraro and Lemström (2018), we use the TOP-MAGD dataset for genre classification and the MASD dataset for style classification. TOP-MAGD contains 22,535 annotated files of 13 genres, and MASD contains 17,785 files of 25 styles. We evaluate MusicBERT on TOP-MAGD and MASD using 5-fold cross-validation and use the F1-micro score as the metric (Liang et al., 2020; Ferraro and Lemström, 2018; Oramas et al., 2017). Due to the limitation of computational resources, for songs with more than 1,000 octuple tokens, we randomly crop segments with 1,000 tokens. On average, the selected segment covers more than 1/4 of a music song according to Table 1, which is enough to capture sufficient information for identifying genres and styles. As shown in Table 3, MusicBERT models significantly outperform previous works, indicating that MusicBERT can perform well on song-level tasks.

4.3 Method Analysis

In this subsection, we analyze the effectiveness of each design in MusicBERT, including OctupleMIDI encoding, bar-level masking strategy, and the pre-training itself. We conduct experiments on MusicBERT$_{\text{small}}$ with a maximum sequence length of 250 due to the huge training cost of MusicBERT$_{\text{base}}$. For simplicity, we treat melody completion and accompaniment suggestion as binary classification tasks: classifying matched and not matched melody or accompaniment pairs, instead of ranking a group of pairs by predicted match score. We use the accuracy percentage score as the metric for these two binary classification tasks.

Effectiveness of OctupleMIDI We compare our proposed OctupleMIDI encoding with REMI (Huang and Yang, 2020) and CP (Hsiao et al., 2021) by training MusicBERT$_{\text{small}}$ models with each encoding respectively and evaluate on downstream tasks. As shown in Table 5, for song-level tasks (i.e., genre and style classification), OctupleMIDI significantly outperforms REMI and CP based encoding, since the model can learn from a larger proportion of a music song with the compact OctupleMIDI encoding, given all encoding methods use the same length of sequence for pre-training. For phrase-level tasks (melody completion and accompaniment suggestion), the input sequence length is usually less than the truncate threshold. Thus, benefiting from the short representation, OctupleMIDI significantly reduces the computational complexity of the Transformer encoder, which is only 1/16 of that with REMI-like encoding and 1/4 of that with CP-like encoding. Moreover, according to Table 5, OctupleMIDI performs better than the other two encoding methods on phrase-level tasks.

Encoding	Melody	Accom.	Genre	Style
CP-like	95.7	87.2	0.719	0.510
REMI-like	92.0	86.5	0.689	0.487
OctupleMIDI	**96.7**	**87.9**	**0.730**	**0.534**

Table 5: Results of different encoding methods. "Accom." represents accompaniment suggestion task.

Effectiveness of Bar-Level Masking We compare our proposed bar-level masking strategy with two other strategies: 1) Octuple masking, as mentioned in Sec. 3.3 and Fig. 2a; 2) Random masking, which randomly masks the elements in the octuple token similar to the masked language model in BERT. We use the same masking ratio for all these strategies. As shown in Table 6, our proposed bar-level masking can effectively boost results on downstream tasks.

Mask	Melody	Accom.	Genre	Style
Random	96.3	87.8	0.708	0.533
Octuple	96.0	87.3	0.722	0.530
Bar	**96.7**	**87.9**	**0.730**	**0.534**

Table 6: Results of different masking strategies.

Effectiveness of Pre-training To show the advantage of pre-training in MusicBERT, we compare the performance of MusicBERT$_{\text{small}}$ with and without pre-training. As shown in Table 7, pre-training achieves much better scores on the four downstream tasks, demonstrating the critical role of pre-training on symbolic music understanding.

Model	Melody	Accom.	Genre	Style
No pre-train	92.4	76.9	0.662	0.395
MusicBERT	**96.7**	**87.9**	**0.730**	**0.534**

Table 7: Results with and without pre-training.

5 Conclusion

In this paper, we developed MusicBERT, a large-scale pre-trained model for symbolic music understanding. Instead of simply adopting the pre-training methods from NLP to symbolic music, we handle the distinctive challenges in music pre-training with several careful designs in MusicBERT, including the efficient and universal OctupleMIDI encoding, the effective bar-level masking strategy, and the large-scale symbolic music corpus with more than 1 million music songs. MusicBERT achieves state-of-the-art performance on all of the four evaluated symbolic music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification. Method analyses also verify the effectiveness of each design in MusicBERT. For future work, we will apply MusicBERT on other music understanding tasks such as chord recognition and structure analysis to boost the performance.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao. 2018. Symbolic music genre transfer with cyclegan. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 786–793. IEEE.

Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696.

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):1023–1036.

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 AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang. 2018. Pypianoroll: Open source python package for handling multitrack pianoroll. *Proc. ISMIR. Late-breaking paper;[Online] https://github.com/salu133445/pypianoroll*.

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*, pages 34–37.

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

Dorien Herremans and Ching-Hua Chuan. 2017. Modeling musical context with word2vec. *arXiv preprint arXiv:1706.09088*.

Tatsunori Hirai and Shun Sawada. 2019. Melody2vec: Distributed representations of melodic phrases based on melody segmentation. *Journal of Information Processing*, 27:278–286.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. *arXiv preprint arXiv:2101.02402*.

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*, pages 241–250.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, AM Dai, MD Hoffman, and D Eck. 2018. Music transformer: Generating music with long-term structure (2018). *arXiv preprint arXiv:1809.04281*.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188.

Ray Jackendoff. 2009. Parallels and nonparallels between language and music. *Music perception*, 26(3):195–204.

Shulei Ji, Jing Luo, and Xinyu Yang. 2020. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *arXiv preprint arXiv:2011.06801*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Qiuqiang Kong, Bochen Li, Jitong Chen, and Yuxuan Wang. 2020a. Giantmidi-piano: A large-scale midi dataset for classical piano music. *arXiv preprint arXiv:2010.07061*.

Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. 2020b. High-resolution piano transcription with pedals by regressing onsets and offsets times. *arXiv preprint arXiv:2010.01815*.

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*, pages 574–582.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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 (NIPS2016), Barcelona, Spain*.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. 2020. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967.

Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. 2017. Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1198–1206.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2020. Songmass: Automatic song writing with pre-training and alignment constraint. *arXiv preprint arXiv:2012.05168*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.