MUSEBERT: PRE-TRAINING OF MUSIC REPRESENTATION FOR MUSIC UNDERSTANDING AND CONTROLLABLE GENERATION

Ziyu Wang
Music X Lab, NYU Shanghai
ziyu.wang@nyu.edu

Gus Xia
Music X Lab, NYU Shanghai
gxia@nyu.edu

ABSTRACT

BERT has proven to be a powerful language model in natural language processing and established an effective pre-training & fine-tuning methodology. We see that music, as a special form of language, can benefit from such methodology if we carefully handle its highly-structured and polyphonic properties. To this end, we propose MuseBERT and show that: 1) MuseBERT has detailed specification of note attributes and explicit encoding of music relations, without presuming any pre-defined sequential event order, 2) the pre-trained MuseBERT is not merely a language model, but also a controllable music generator, and 3) MuseBERT gives birth to various downstream music generation and analysis tasks with practical value. Experiment shows that the pre-trained model outperforms the baselines in terms of reconstruction likelihood and generation quality. We also demonstrate downstream applications including chord analysis, chord-conditioned texture generation, and accompaniment refinement.

1. INTRODUCTION

BERT [1] has proven to be one of the leading language models in natural language processing, which learns natural language representation in an unsupervised manner and achieved state-of-the-art results in many downstream language understanding tasks. The methods involved in BERT are not unfamiliar in the music domain. For example, the "masked language model" (MLM) objective in BERT is similar to the Bach chorale inpainting studies [2–4], where the grids in a four-part piano-roll are randomly masked and the model is trained to reconstruct them from context. The Transformer architecture has also been applied to different styles of music generation [5, 6].

We aim to develop a pre-trained music-domain BERT for better music understanding and generation. A straightforward approach is to use existing music representations used in existing Transformer-based models, such as MIDI-like representations [5, 7], REMI [6], or CP [8], and simply train the original BERT model. However, we argue that such approach overlooks major distinctions between music and natural language. Firstly, unlike natural language, music (especially polyphony) does not follow a unique sequential order; abruptly flatten music into a 1-d sequence imposes extra protocol and often undermines the local structure of music, adding extra burden to the model. Secondly, music involves rich relations and contexts. A single music event is barely meaningful, and common music concepts such as rhythmic patterns and harmonies are established upon relative positions of notes. However, current sequential models, in particular Transformer-based models, still struggle to capture these relations [9].

To overcome the limits above, we propose MuseBERT, a Transformer-based pre-trained model with tailored handling of music positional information and music relations. [1] Since the innate music positions are expressed in the time-frequency space, we use onset and pitch information as the absolute positional encoding, which can be masked at the input and then reconstructed at the output. Moreover, multiple musical relations are represented via the design of *generalized* relative positional encoding (RPE) modified from the original RPE design [10]. In our implementation, we find the traditional sequential positional encoding unnecessary, and thus our model does not rely on any pre-defined sequential assumptions.

Under such tailored design, we show the pre-trained MuseBERT is not merely a language model but also a controllable music generator. In the fine-tuning stage, the model naturally empowers a wide spectrum of fine-tuning tasks, involving both music analysis and generation. Specifically, we showcase that MuseBERT can perform polyphonic music generation controlled by chord and texture, chord extraction, and accompaniment refinement. In summary, the contributions of this paper are:

- We demonstrate the possibility and importance of training a Transformer-based music language model in an *unordered* approach.
- We develop generalized RPE for BERT-like models, which not only reveals music relations from multiple perspectives but also has a potential to be adopted to other domains.
- We show MuseBERT, as a controllable music generator, giving the fine-tuning procedure practical music meanings.

[1] Code and demos can be accessed via https://github.com/ZZWaang/musebert.

2. DATA REPRESENTATION

Unlike natural language processing, there is not a common way to tokenize symbolic music. A music piece is generally considered as a combination of note events, where each note event is a tuple of attributes. Note-based tokenization is commonly seen in music-tailored packages [11, 12], softwares [13–15], and researches [8, 16, 17]. On other occasions, mainly for the ease of neural network modeling, music is also simplified as a sequence of controls with pre-defined order, such as MIDI-like event representation [5,7] (including its follow-up methods [6]), and frame-wise piano-roll representations [2, 18–20].

In MuseBERT, we consider note-based tokenization without presumption of any sequential order. Specifically, we invent two note-based music representations for MuseBERT: *basic data representation* and *factorized data representation*, denoted by $\mathcal{R}_{\text{base}}$ and $\mathcal{R}_{\text{fac}}$ respectively. A music input is first encoded as $\mathcal{R}_{\text{base}}$, and then converted into $\mathcal{R}_{\text{fac}}$, serving as the input format to MuseBERT.

In this paper, we primarily consider 2-bar music segments in 4/4 time signature. Longer music and richer time signatures can be generalized and we leave it for future study.

2.1 $\mathcal{R}_{\text{base}}$ and $\mathcal{R}_{\text{fac}}$

$\mathcal{R}_{\text{base}}$ represents a music segment X as an *unordered set* of note events $\{x_i\}_{i=1}^{N}$, and a note event $x_i \in X$ consists of three attributes: onset ($\texttt{o}$), pitch ($\texttt{p}$), and duration ($\texttt{d}$), where $\texttt{o} \in [0..31]$ and $\texttt{d} \in [0..32]$ are counted by semiquavers, and $\texttt{p} \in [0..127]$ is the MIDI note number. [2] We denote the *attribute set* of $\mathcal{R}_{\text{base}}$ by $\mathcal{A}_{\text{base}} := \{\texttt{o}, \texttt{p}, \texttt{d}\}$ and use $x_i.a, a \in \mathcal{A}_{\text{base}}$ to indicate the attribute a of a note event x_i.

$\mathcal{R}_{\text{base}}$ is not the MuseBERT input data format because the representation is *ambiguous* for BERT-like models to reconstruct well. Consider, for example, two simultaneous quarter notes in a music segment that are unordered (by definition of $\mathcal{R}_{\text{base}}$) and whose attributes except $\texttt{p}$ are the same. When their pitch attributes are masked at the input, the two tokens have completely the same encoding and are unable to be distinguished by BERT, in which case BERT will yield the same output distributions.

We therefore invent $\mathcal{R}_{\text{fac}}$ which is unambiguous for MuseBERT (proved in section 4). $\mathcal{R}_{\text{fac}}$ also represents a music segment as an unordered set of note events, but it uses a more detailed attribute set $\mathcal{A}_{\text{fac}}$ (discussed in section 2.2), and contains a stack of relation matrices storing pairwise note relations (discussed in section 2.3).

2.2 Factorized Attributes

In $\mathcal{R}_{\text{fac}}$, onset, pitch, and duration, due to their hierarchical nature [21,22], are regarded as meta-attributes to be factorized. The factorization operation we will introduce is analogous to storing a number by its digits (e.g., $49 \mapsto [4, 9]$), so that we can mask partial information at certain hierarchy (e.g., $[4, \texttt{MASK}]$) and the remaining information is still retained. In music, musicians sometimes prefer to interpret 49 as $50 - 1$ rather than $40 + 9$, depending on the context. Such subtlety is welcomed by our design.

2.2.1 Onset and Duration Factorization

We factorize onset ($\texttt{o}$) into *beat position* ($\texttt{o\_bt} \in [0..8]$) and *subdivision* ($\texttt{o\_sub} \in [0..4]$), and duration $\texttt{d}$ into *half note counts* ($\texttt{d\_hlf} \in [0..4]$) and *the remainder semiquaver counts* ($\texttt{d\_sqv} \in [0..7]$), satisfying

$$\texttt{o} = (4 \times \texttt{o\_bt} + \texttt{o\_sub}), \tag{1}$$

$$\texttt{d} = (8 \times \texttt{d\_hlf} + \texttt{d\_sqv}). \tag{2}$$

We allow $\texttt{o\_sub}$ to take both positive and negative values, since a subdivision can be interpreted as an "off-beat" of the current downbeat, or an "upbeat" preceding the next downbeat. Consequently, the $\texttt{o}$ attribute can be mapped to multiple $\{\texttt{o\_bt}, \texttt{o\_sub}\}$ pairs, e.g., $\{\texttt{o}:1\}$ maps to $\{\texttt{o\_bt}:0, \texttt{o\_sub}:1\}$ or $\{\texttt{o\_bt}:1, \texttt{o\_sub}:-3\}$.

In our implementation, we *non-deterministically* sample one of the factorizations. In the future, a posterior distribution of $p(\texttt{o\_bt}, \texttt{o\_sub} | \texttt{o}, X)$ can be explored in finer detail.

2.2.2 Pitch Factorization

We factorize pitch in a similar fashion into three attributes: 1) *pitch highness* ($\texttt{p\_hig} \in [0..6]$): voice types (e.g., SATB), 2) *pitch register* ($\texttt{p\_reg} \in [0..2]$): the relative octave given $\texttt{p\_hig}$, and 3) *pitch degree* ($\texttt{p\_deg} \in [0..11]$), satisfying:

$$\texttt{p} = \begin{cases} 24 + 12 \times (\texttt{p\_hig} + \texttt{p\_reg}) \\ \qquad\qquad + \texttt{p\_deg}, & 0 \le \texttt{p\_hig} \le 4 \\ 12 \times \texttt{p\_reg} + \texttt{p\_deg}, & \texttt{p\_hig} = 5 \\ 108 + 12 \times \texttt{p\_reg} + \texttt{p\_deg}, & \texttt{p\_hig} = 6. \end{cases} \tag{3}$$

Here, $0 \le \texttt{p\_hig} \le 4$ corresponds to the pitch range of a standard piano (MIDI pitch 24-107), and $\texttt{p\_hig} = 5$ and $\texttt{p\_hig} = 6$ handles the extra-low or extra-high regions, respectively. Similar to onset, pitch factorization is also handled non-deterministically, since a pitch can be interpreted as a lower voice type having a high register, or a higher voice type having a low register.

To summarize, $\mathcal{R}_{\text{fac}}$ uses the attribute set:

$$\mathcal{A}_{\text{fac}} := \{\texttt{o\_bt}, \texttt{o\_sub}, \texttt{p\_hig}, \texttt{p\_reg}, \texttt{p\_deg}, \texttt{d\_hlf}, \texttt{d\_sqv}\}. \tag{4}$$

2.3 Note Event Relations

Music is rich in relations and we explicitly represent the most fundamental relations in $\mathcal{R}_{\text{fac}}$. We consider less-than, equal-to, and greater-than relations on a subset of attributes $\mathcal{S} := \{\texttt{o}, \texttt{o\_bt}, \texttt{p}, \texttt{p\_hig}\}$. Specifically, $\forall a \in \mathcal{S}$, we define a mapping γ_a from an input note event pair (x_i, x_j) to their relation symbol: [3]

[2] We use $[a..b]$ to denote the integer interval $\{x | a \le x \le b, x \in \mathbb{Z}\}$ including both endpoints.

[3] When $a = \texttt{o}$, $x_i.\texttt{o}$ is a shorthand notation interpreted as the return value of Eqn (1). A similar case holds when $a = \texttt{p}$.

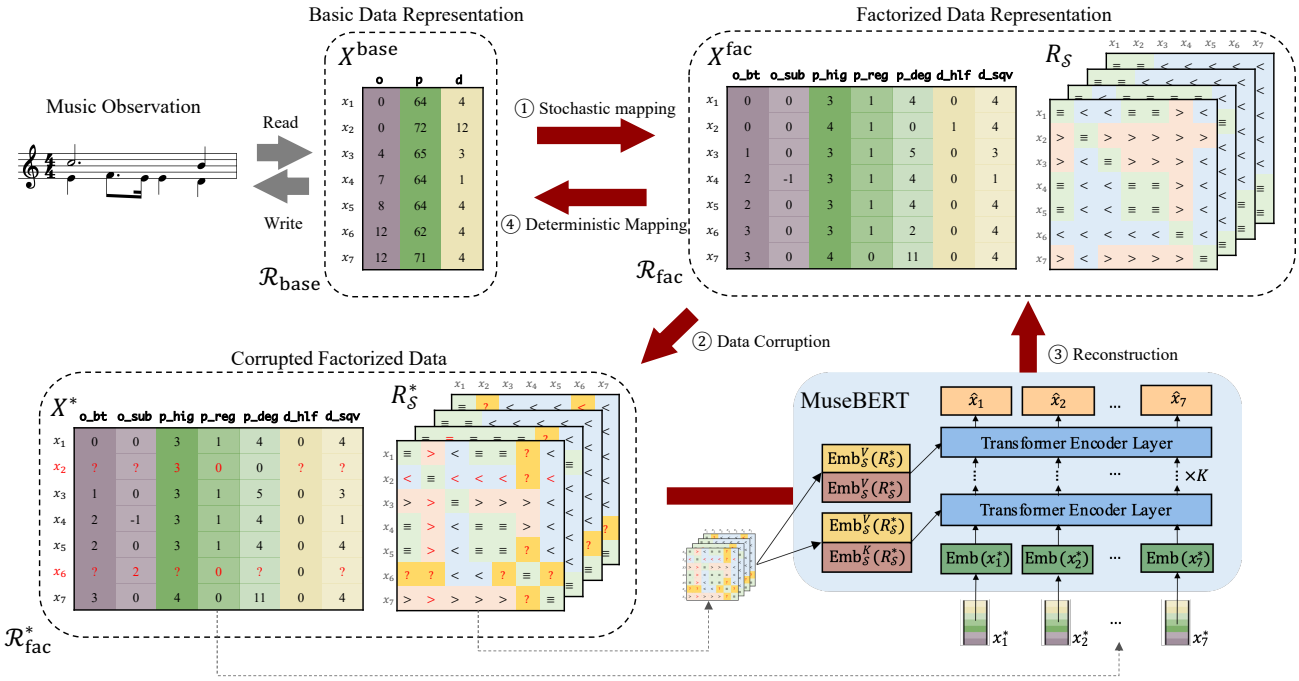


Figure 1. The workflow of MuseBERT pre-training.

$$\gamma_a(x_i, x_j) = \begin{cases} <^a, & x_i.a < x_j.a \\ \equiv^a, & x_i.a \equiv x_j.a \\ >^a, & x_i.a > x_j.a. \end{cases} \tag{5}$$

The relation symbols are stored using a stack of $N \times N$ *relation matrices*, denoted by $R_{\mathcal{S}} := \{R_a | a \in \mathcal{S}\}$, where $R_a = \{r_{ij}^a\}$ and $r_{ij}^a = \gamma_a(x_i, x_j)$.

3. PRE-TRAINING MUSEBERT

Figure 1 shows the overall pre-training stage. A music data is first read as $\mathcal{R}_{\text{base}}$ format as X^{base}, and then stochastically converted to $\mathcal{R}_{\text{fac}}$ consisting of unordered note events X^{fac} and relation matrices $R_{\mathcal{S}}$. Similar to BERT, the pre-training stage uses "masked language model" (MLM) training objective, where X^{fac} and $R_{\mathcal{S}}$ will be corrupted before fed into MuseBERT, which is trained to reconstruct the original music segment.

3.1 Data Corruption

The original BERT randomly selects a part of the input tokens to be corrupted. In MuseBERT, data corruption has finer controls: 1) it operates on individual attributes of a note event rather than an entire token, and 2) it corrupts relation matrices as well.

For note events X^{fac}, the corrupter randomly selects 15% of the note events $C \subset X$, and corrupts their attributes $\{x_i.a | x_i \in C, a \in \mathcal{A}_{\text{fac}}\}$ in one of the three ways: 1) masked with $[\texttt{MASK}]_a$ 80% of the time, 2) replaced with a random token 10% of the time, and 3) kept unchanged 10% of the time. Each attribute is corrupted independently of the choice of the other attributes. For relation matrices $R_{\mathcal{S}}$, each matrix entry will be masked 30% of the time independently. Before corruption, a re-computation of the matrix may be required so as to maintain matrix symmetry and align with the attributes that are replaced with other values.

We denote the representation after corruption as $\mathcal{R}_{\text{fac}}^*$, and the resulting note events and relation matrices as X^* and $R_{\mathcal{S}}^*$, respectively.

3.2 Overall Model Architecture

The model adopts the bi-directional Transformer encoder architecture based on the original Transformer implementation described in [23]. The input to the model is the embedding of a corrupted music segment $X^* = \{x_i^*\}_{i=1}^N$. The embedding is computed as the sum of all attribute embeddings:

$$\text{Emb}(x_i^*) = \sum_{a \in \mathcal{A}_{\text{fac}}} \text{Emb}_a(x_i^*.a). \tag{6}$$

Here $\text{Emb}_a(\cdot)$ are linear embedding layers and the default absolute positional encoding is *not* applied due to unorderedness. $R_{\mathcal{S}}^*$ is fed into the model as *generalized relative positional encoding* to be discussed in detail in section 3.3.

The output is a distribution of reconstructed note events in $\mathcal{R}_{\text{fac}}$, denoted by $\hat{X} = \{\hat{x}_i\}_{i=1}^N$. Specifically, let $h_{1:N}$ be the last hidden vector of the Transformer encoder, each note attribute is reconstructed by a separate linear transformation normalized with a softmax layer:

$$p_{\text{model}}(\hat{x}_i.a | X^*, R_{\mathcal{S}}^*) = \text{softmax}(\text{FFN}_a(h_i)). \tag{7}$$

The training loss is the mean of the negative log-likelihood on corrupted attributes $\{x_i.a | x_i \in C, a \in \mathcal{A}_{\text{fac}}\}$ only:

$$\mathcal{L}(\hat{X}|X^*, R^*_{\mathcal{S}}) = \\ -\frac{1}{|C|}\sum_{\{i|x^*_i\in C\}}\Big(\sum_{a\in\mathcal{A}_{\text{fac}}}\log p_{\text{model}}(x^{\text{fac}}_i.a|X^*, R^*_{\mathcal{S}})\Big), \quad (8)$$

From the denoising autoencoder perspective [24], we define the reconstruction distribution $p_{\text{recon}}(\hat{X}|X^*, R^*_{\mathcal{S}})$ induced from Eqn (7) for later analysis purposes:

$$p_{\text{recon}}(\hat{X}|X^*, R^*_{\mathcal{S}}) := \prod_{i=1}^{N} p_{\text{recon}}(\hat{x}_i|X^*, R^*_{\mathcal{S}}), \text{ where}$$
$$p_{\text{recon}}(\hat{x}_i|X^*, R^*_{\mathcal{S}}) := \prod_{a\in\mathcal{A}_{\text{fac}}} p_{\text{recon}}(\hat{x}_i.a|X^*, R^*_{\mathcal{S}}) \text{ and}$$
$$p_{\text{recon}}(\hat{x}_i.a|X^*, R^*_{\mathcal{S}}) := \begin{cases} p_{\text{model}}(\hat{x}_i.a|X^*, R^*_{\mathcal{S}}), x_i \in C \\ \mathbb{1}_{\{\hat{x}_i.a\equiv x^*_i.a\}}, \text{otherwise.} \end{cases} \quad (9)$$

In other words, during inference, we sample from p_{model} for corrupted tokens while simply keeping the uncorrupted tokens.

3.3 Generalized Relative Positional Encoding

Let $h_{1:N}$ be the hidden vectors at any Transformer layer. We add additional *key-relation embeddings* $\text{Emb}^K_{\mathcal{S}}(\cdot)$ of the relations in the query-value product term by

$$e_{ij} = \frac{h_i W^Q \Big(h_j W^K + \sum_{a\in\mathcal{S}} \text{Emb}^K_a(r^a_{ij})\Big)^T}{\sqrt{d_z}}, \quad (10)$$

where d_z is the hidden vector size of attention heads, W^Q, W^K are query and key transformations, and e_{ij} is used to compute attention weights.

Similarly, we also add *value-relation embeddings* $\text{Emb}^V_{\mathcal{S}}(\cdot)$ in the attention weight application:

$$z_i = \sum_{j=1}^{N} \alpha_{ij}\Big(h_j W^V + \sum_{a\in\mathcal{S}} \text{Emb}^V_a(r^a_{ij})\Big), \quad (11)$$

where $\alpha_{ij} = \text{softmax}_j(e_{ij})$, W^V is the value transformation, and z_i is fed into the position-wise feed-forward sublayer to compute the next layer's h_i.

In our implementation, both key-relation embeddings and value-relation embeddings are linear transforms. In addition, for different attention layers and attention heads, we train different embeddings.

We call the above operation *generalized* relative positional encoding (RPE) because the summation of key-relation and value-relation embeddings allows us to hint the MuseBERT from multiple musical perspectives (attribute relations), which is not considered in the original RPE implementation [10].

4. THEORETICAL ANALYSIS

In this section, we first show that $\mathcal{R}_{\text{fac}}$ and generalized relative positional encoding are theoretical necessities for a BERT model to handle unordered data. We further show that MuseBERT is a powerful controllable music generator, giving birth to various fine-tuning tasks with musical merits.

4.1 Well-definedness of MuseBERT

The fundamental assumption made in MuseBERT is the refusal of traditional *sequential* positional encoding. In return, the mathematical property of MuseBERT we gain is *permutation invariance*, i.e., the input order will not change the model output. On the other hand, a natural downside of a permutation invariant model is *ambiguity*: without extra efforts, masked tokens will have the same output distributions and hence not distinguishable.

The problem is solved by factorized data representation and generalized RPE because the former allows us to specify music in more detail. We summarize the discussion with the well-definedness theorem of MuseBERT. Part one of the theorem is maintained by the property of Transformer [25], and part two is self-evident from the discussion so far.

Theorem 1 (Well-definedness) *Pre-training MuseBERT is well-defined: Let $(X^*, R^*_{\mathcal{S}})$ be a corrupted music segment in $\mathcal{R}_{fac}$:*
1) (Permutation invariance) Let $\sigma(\cdot)$ be an arbitrary permutation. The reconstruction distribution is unchanged after permutation:

$$p_{recon}(\hat{X}|X^*, R^*_{\mathcal{S}}) \equiv p_{recon}(\sigma(\hat{X})|\sigma(X^*),\sigma(R^*_{\mathcal{S}}))$$

2) (Unambiguity) For arbitrary pair of $x_i, x_j \in X^*$, whose attributes are corrupted but their note relations are not, the reconstruction distributions of the two tokens are not always the same, i.e., there exists a set of model parameters such that

$$p_{recon}(\hat{x}_i|X^*, R^*_{\mathcal{S}}) \neq p_{recon}(\hat{x}_j|X^*, R^*_{\mathcal{S}}).$$

4.2 MuseBERT as a Music Generator

The operations of MuseBERT can be regarded as a single round of a Markov chain transition which alternatively adds noise and denoises (as shown in Figure 1), consisting of four steps: 1) stochastic data conversion to $\mathcal{R}_{\text{fac}}$, 2) data corruption, 3) MuseBERT reconstruction, and 4) deterministic data conversion to $\mathcal{R}_{\text{base}}$.

Benjio et al. [26] showed such transition induced from denoising autoencoder is a special type of Generative Stochastic Network and proved that under mild condition, the transition operations form a Markov Chain whose stationary distribution is the true data distribution. Hence, we conclude the pre-training MuseBERT can be used as the MCMC transition operator, and generates music via iterative sampling.

4.3 Controllability of MuseBERT

Moreover, we show that MuseBERT is a Constraint Optimization Problem (COP) solver [27] by regarding uncorrupted attributes or relations as unary or binary constraints,

respectively. The COP has the form:

$$\begin{aligned} \max_{\hat{X}} \quad & p_{\text{recon}}(\hat{X}|X^*) \qquad (12) \\ \text{s.t.} \quad & \hat{x}_i.a = x_i^*.a, \quad \forall x_i^* \notin C, a \in \mathcal{A}_{\text{fac}} \\ & \gamma_a(x_i, x_j) = \gamma_a(x_i^*, x_j^*), \\ & \forall a \in \mathcal{S} \quad \text{if } \gamma_a(x_i^*, x_j^*) \text{ uncorrupted.} \end{aligned}$$

Apparently, music continuation and in-painting can be formalized as a special COP problem, and monophony, melodic directions, rhythmic patterns, texture, etc. are all expressible concepts by MuseBERT constraints. Hence, MuseBERT, as a music generator, is also controllable, which can be fine-tuned to solve problem-specific generation tasks and controls.

5. FINE-TUNING MUSEBERT

By defining problem-specific data corrupters, we can fine-tune the model for a wide spectrum of conditioned music generation tasks, such as music continuation, inpainting, rhythm or texture-conditioned music generation. In this section, we in turn show two less obvious applications by fine-tuning, in which we extend the vocabulary of $\mathcal{R}_{\text{fac}}$ to represent chord and vocal tracks, and apply MuseBERT for both generation and analysis.

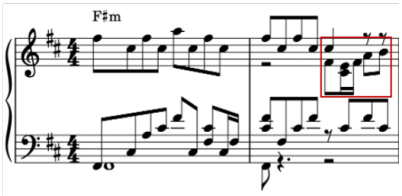
5.1 Case 1: Texture Generation and Chord Analysis

MuseBERT can be fine-tuned for chord and texture controls, which are commonly-studied controlling factors in automatic music generation [28, 29]. The fine-tuning task is set by slight modification from the pre-training stage in two steps: 1) We represent chord progression as $\mathcal{R}_{\text{fac}}$ events and prepend it to the music segment. Specifically, we define `p_hig` $= 7$ for chord event, with `p_reg` $= 0$, `p_reg` $= 1$, `p_reg` $= 2$ for chord root, chroma, and bass, respectively. 2) We apply an additional "in-chord" inter-relation to indicate whether a note event is a chord-tone or not. The conditioned generation can be trained by masking all the attributes from the music segment while keeping the chord attributes and all relations. Figure 2 shows an example of music generation conditioned on the texture in Figure 2(a) and the chord F major. The generated segment keeps the textural property and deals with inner voices smoothly (shown by the red boxes).

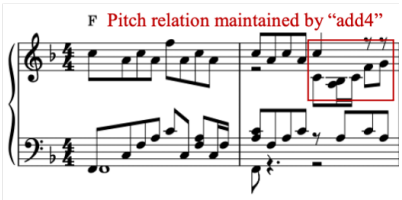
Similarly, by corrupting the chord events and "in-chord" relation, while maintaining the music segment and the other relations, the model can learn to extract chord progressions from a given piece. The fine-tuning result on POP909 [30] shows the model has a 99.35% accuracy on token-wise (MIDI) chord extraction (99.37% for root, 99.05% for chroma and 99.35% for bass).

5.2 Case 2: Accompaniment Refinement

In the accompaniment refinement task, we want to refine an imperfect accompaniment according to a given lead melody. To achieve this, we represent the melody and the accompaniment both as $\mathcal{R}_{\text{fac}}$ and concatenate them together. We then randomly corrupt some accompaniment attributes while keeping the melody and relations unchanged during training. Figure 3 shows an example of accompaniment refinement from a corrupted sample Figure 3(a). In (a), the red note heads mark the note being replaced and the masked notes are replaced by rests. The result shows that the generated piece is more consistent with the melody while keeps the original musical content.

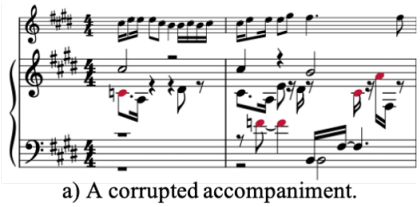


a) An original piece in F#m chord.

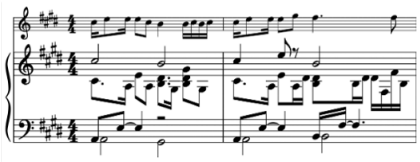


b) The generated piece in F chord using the texture of a).

Figure 2. An example of controlled generation by specifying chord and texture.



a) A corrupted accompaniment.



b) The generated accompaniment.

Figure 3. An example of accompaniment refinement conditioned on melody.

6. EXPERIMENTAL RESULT

6.1 Dataset and Training

We collect around 5K classical and popular piano pieces from Musicalion [4] and the POP909 dataset [30]. We only keep the pieces with 2/4 and 4/4 meters and cut them into 8-beat music segments. In all, we have 19.8K samples. We randomly split the dataset (at song-level) into training set (90%) and test set (10%). All training samples are further augmented by transposing to all 12 keys.

We pre-train our MuseBERT using 12 Transformer layers with 8 attention heads, 0.1 dropout rate, and GELU ac-

[4] Musicalion: `https://www.musicalion.com`.

tivation [31]. The hidden dimensions of self-attention and position-wise feed-forward layers are $d_{\text{model}} = 256$ and $d_{\text{ff}} = 512$. We train with a batch size of 128 and Adam optimizer with learning rate of 5e-4 and linear learning rate decay to 5e-6 after 15K warmup steps. The pre-training takes 20 epochs to converge.

In fine-tuning tasks, we kept the same hyperparameters except the initial learning rate is 2e-4 without warmup. The epochs to fine-tune the model are task-specific, approximately 4 epochs on average.

6.2 Objective Evaluation

Given that $\mathcal{R}_{\text{fac}}$ and data corruption are stochastic, we evaluate the expectation of reconstruction likelihood in terms of token-wise onset, pitch and duration, similar to the algorithm in [4]. We compare among 1) the pre-trained MuseBERT, 2) the fine-tuned models using attribute-specific data corrupters, and three baseline models ablating the use of factorized attributes $\mathcal{A}_{\text{fac}}$ and binary relations. Table 1 summarizes the results where we see our pre-trained model outperforms the baselines on all three criteria and the fine-tuned model can perform even better. The use of $\mathcal{A}_{\text{fac}}$ and binary relations not only makes the model well-defined, but also are crucial for high-quality reconstruction.

Models	Onset	Pitch	Duration
$\mathcal{A}_{\text{fac}}$ w/ rel (ours)	**0.902 ± 0.002**	**0.813 ± 0.003**	**0.815±0.003**
$\mathcal{A}_{\text{fac}}$ w/o rel*	0.472 ± 0.002	0.740 ± 0.004	0.743 ± 0.003
$\mathcal{A}_{\text{base}}$ w/ rel	0.856 ± 0.001	0.785 ± 0.004	0.785 ± 0.003
$\mathcal{A}_{\text{base}}$ w/o rel*	0.488 ± 0.003	0.733 ± 0.003	0.733 ± 0.002
$\mathcal{A}_{\text{fac}}$ w/ rel (fine-tuned)	**0.958 ± 0.001**	**0.924 ± 0.002**	**0.927 ± 0.002**

Table 1. Objective evaluation of the expectation of reconstruction likelihood. Here, "w/ rel" and "w/o rel" indicates whether generalized RPE is applied, and "$*$" indicates the ill-defined models.

6.3 Self-attention Visualization

We observe that different attention heads of multiple layers in the pre-trained MuseBERT captures high-level music knowledge. As shown in Figure 4, metrical structure is revealed in (a) & (b), and voice-leading can be seen in (c) & (d). We show the attention matrices on a piano-roll for better interpretability, where an arrow $x_i \rightarrow x_j$ shows the (j, i) entry and the brightness indicates the attention strength.

6.4 Subjective Evaluation of Music Generation

We invite people to subjectively rate the generation quality of the pre-trained model through a double-blind online survey. During the survey, the subjects listen to 8 groups of samples, each containing 4 generated segments together with a ground-truth human composition. The generated segments come from the four models (the same as in Table 1 excluding the fine-tuned model) with the same initial corruption state (60% of the total tokens) from the original piece. Both the order of groups and the sample order within each group are randomized. After listening

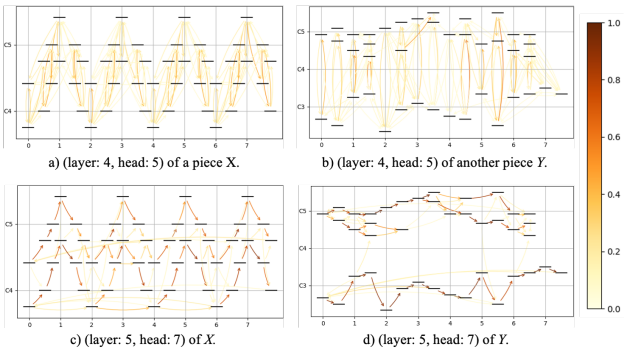


Figure 4. Visualization of attention matrices on selected self-attention layers in pre-trained MuseBERT.

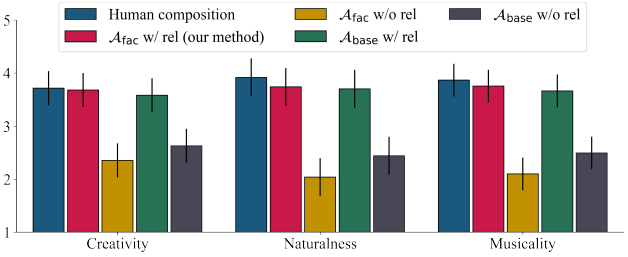


Figure 5. Subjective evaluation results on pre-trained MuseBERT.

to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria: *creativity*, *naturalness* and *musicality*.

A total of 37 subjects (15 females and 22 males) with different musical backgrounds have completed the survey. The aggregated result (as shown in Figure 5) shows that the pre-trained models considering music relations (the well-defined models) are significantly better than the models without relations (the ill-defined models) (with p-value < 0.005), and are marginal to the human composition (with p-value > 0.05).

7. CONCLUSION

In conclusion, we contributed MuseBERT, a full treatment of BERT in the music domain as a powerful music understanding and generation model. The main novelty lies in the proposed *generalized* relative positional encoding, which successfully reveals the non-sequential, polyphonic structure of music and at the same time turns the masked language model objective into a well-defined controllable generative framework. Also, with the music-tailored factorized data representation, the controls are fine-grained. Furthermore, by fine-tuning MuseBERT, we demonstrate that the model is *general purpose*, capable of various downstream tasks such as chord analysis, accompaniment generation and refinement etc. as long as the constraints can be expressed via binary relations of different music attributes.

8. REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1362–1371.

[3] Y. Yan, E. Lustig, J. VanderStel, and Z. Duan, "Part-invariant model for music generation and harmonization." in *19th International Society for Music Information Retrieval*, 2018, pp. 204–210.

[4] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," *arXiv preprint arXiv:1903.07227*, 2019.

[5] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[6] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Generating music with rhythm and harmony," *arXiv preprint arXiv:2002.00212*, 2020.

[7] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," https://magenta.tensorflow.org/performance-rnn, 2017.

[8] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," *arXiv preprint arXiv:2101.02402*, 2021.

[9] J. Jiang, G. Xia, and T. Berg-Kirkpatrick, "Discovering music relations with sequential attention," in *Proceedings of the 1st workshop on NLP for music and audio (NLP4MusA)*, 2020, pp. 1–5.

[10] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[11] C. Raffel and D. P. Ellis, "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*, 2014, pp. 84–93.

[12] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," 2010.

[13] "Garageband for mac." [Online]. Available: https://www.apple.com/mac/garageband/

[14] "Logic pro." [Online]. Available: https://www.apple.com/logic-pro/

[15] "Cubase guides you on your music production journey." [Online]. Available: https://new.steinberg.net/cubase/

[16] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "Pianotree vae: Structured representation learning for polyphonic music," *arXiv preprint arXiv:2008.07118*, 2020.

[17] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3060–3070.

[18] H. H. Mao, T. Shin, and G. Cottrell, "Deepj: Style-specific music generation," in *IEEE 12th International Conference on Semantic Computing*. IEEE, 2018, pp. 377–382.

[19] E. S. Koh, S. Dubnov, and D. Wright, "Rethinking recurrent latent variable model for music composition," in *IEEE 20th International Workshop on Multimedia Signal Processing*. IEEE, 2018, pp. 1–6.

[20] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *32nd AAAI Conference on Artificial Intelligence*, 2018.

[21] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.

[22] F. Lerdahl, *Tonal pitch space*. Oxford University Press, 2004.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference on Machine Learning*, 2008, pp. 1096–1103.

[25] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3744–3753.

[26] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *International Conference on Machine Learning*. PMLR, 2014, pp. 226–234.

[27] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[28] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," *arXiv preprint arXiv:2008.07122*, 2020.

[29] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," *arXiv preprint arXiv:1806.00195*, 2018.

[30] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," *arXiv preprint arXiv:2008.07142*, 2020.

[31] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.