Pop Music Generation: From Melody to Multi-style Arrangement

HONGYUAN ZHU and QI LIU, University of Science and Technology of China
NICHOLAS JING YUAN, Huawei Cloud&AI
KUN ZHANG, University of Science and Technology of China
GUANG ZHOU, Microsoft
ENHONG CHEN, University of Science and Technology of China

Music plays an important role in our daily life. With the development of deep learning and modern generation techniques, researchers have done plenty of works on automatic music generation. However, due to the special requirements of both melody and arrangement, most of these methods have limitations when applying to multi-track music generation. Some critical factors related to the quality of music are not well addressed, such as chord progression, rhythm pattern, and musical style. In order to tackle the problems and ensure the harmony of multi-track music, in this article, we propose an end-to-end melody and arrangement generation framework to generate a melody track with several accompany tracks played by some different instruments. To be specific, we first develop a novel *Chord based Rhythm and Melody Cross-Generation Model* to generate melody with a chord progression. Then, we propose a *Multi-Instrument Co-Arrangement Model* based on multi-task learning for multi-track music arrangement. Furthermore, to control the musical style of arrangement, we design a *Multi-Style Multi-Instrument Co-Arrangement Model* to learn the musical style with adversarial training. Therefore, we can not only maintain the harmony of the generated music but also control the musical style for better utilization. Extensive experiments on a real-world dataset demonstrate the superiority and effectiveness of our proposed models.

CCS Concepts: • **Information systems** → **Data mining**; • **Computing methodologies** → **Artificial intelligence**; **Sequential decision making**; **Multi-task learning**;

Additional Key Words and Phrases: Music generation, melody and arrangement generation, musical style, multi-task joint learning, Harmony evaluation

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2016YFB1000904) and the National Natural Science Foundation of China (No.s 61672483, 61922073, and 61727809). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299) and the MOE-Microsoft Key Laboratory of USTC.

Authors' addresses: H. Zhu, Q. Liu (corresponding author), K. Zhang, and E. Chen, School of Computer Science and Technology, University of Science and Technology of China, Huangshan Road, Shushan District, Hefei, Anhui, 230027, China; emails: zhy19933628@gmail.com, qiliuql@ustc.edu.cn, zhkun@mail.ustc.edu.cn, cheneh@ustc.edu.cn; N. J. Yuan (corresponding author), 410 Jianghong Road, Binjiang District, Huawei Cloud&AI, Hangzhou, Zhejiang, 310007, China; email: nicholas.yuan@huawei.com; G. Zhou, Microsoft, Suzhou International Science Park, Wu Phase Creative Industrial Zone, Suzhou, China; email: guazho@microsoft.com.

1556-4681/2020/07-ART54 $15.00
https://doi.org/10.1145/3374915

ACM Reference format:
Hongyuan Zhu, Qi Liu, Nicholas Jing Yuan, Kun Zhang, Guang Zhou, and Enhong Chen. 2020. Pop Music Generation: From Melody to Multi-style Arrangement. *ACM Trans. Knowl. Discov. Data* 14, 5, Article 54 (July 2020), 31 pages.
https://doi.org/10.1145/3374915

1 INTRODUCTION

As one of the greatest invention in human history, music can help people to release pressure, comfort emotion, and connect with others [4], which has a vital influence on people's daily life. However, composing music needs plenty of professional knowledge and skills. How to generate music automatically has become a hot topic in recent years. Many companies and research institutes have done interesting works in this area.

For instance, Conklin et al. [15] proposed a statistical model for the problem of music generation. They employed a sampling method to generate music from extant music pieces. In order to generate creative music which is not in extant music pieces, N-gram and Markov models [12, 46] were applied in music generation. These methods could generate novel music, but require manual inspection of the features. Recently, Google Magenta[1] [10] created piano music with Deep Recurrent Neural Network [24] by learning MIDI (a digital score format) data. However, most of these works only took single track music into consideration, which is insufficient for the requirement in real world, since music pieces in the real word are usually more complicated with the collaboration of different musical instruments.

Indeed, generating pop music has plenty of challenges. As shown in Figure 1, pop music consists of melody and arrangement. Whether the music is pleasant to listen depends on several critical characteristics. Specifically,

- —Chord progression generally exists in pop music, which could guide melody procession. Thus, it is beneficial to capture chord progression as input for music generation. Besides, pop music has several fixed rhythm patterns, which make the music more structural and pause suitably. However, existing studies [29, 32] usually generate music note-by-note and without considering the rhythm pattern. On the other hand, though several works [25, 45] utilize chord for music generation, they only use a single chord as a feature of input and without considering the progression of chords when generating melody.
- —Complete music typically has multi-track arrangement[2] considering chord, beats, rhythm patterns, and the like, with accompanying background music played with other instruments, such as drum, bass, string, and guitar. Recent works [23, 45, 48] could generate melody of music. However, they failed to take into account the multi-track arrangement. More importantly, different tracks and instruments have their own characteristics, while they should be in harmony with each other. A few existing works tried to tackle the generation of multi-track music [13, 18], but none of them considered the harmony between multiple tracks.
- —Music usually has multiple styles, such as classic, jazz, rock, and pop. Recent generated music without considering the musical style [13, 24, 45] sounds boring, inflexible, and unreal. Therefore, how to control the musical style is a new problem to be solved urgently. There are some challenges for controlling musical style. On one hand, it is difficult to control the musical style in a supervised method without massive parallel data of musical style. On the

[1] https://magenta.tensorflow.org/.
[2] http://mnsongwriters.org/accelsite/media/1051/Elements%20of%20a%20Song.pdf.

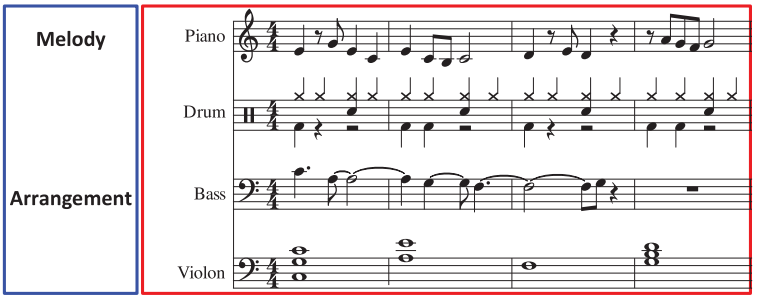


Fig. 1. The example of our generated music.

other hand, musical style is closely related to arrangement and is influenced by multiple instruments [54], but previous works [9, 16, 40] did not deal with it.

In our preliminary work [63], we proposed an end-to-end melody and arrangement generation framework for pop music generation, called XiaoIce Band [63]. To be specific, we proposed a *Chord based Rhythm and Melody Cross-Generation Model (CRMCG)* to generate melody conditioned on the given chord progression for single track music. Then, we introduced a *Multi-Instrument Co-Arrangement Model (MICA)* for multi-track music. Two information-sharing strategies (i.e., *Attention Cell* and *MLP Cell*) were designed to capture the information interactions among these tasks. The former model utilized chord progression to guide the note relationships between periods based on music knowledge. The latter shared the information among different tracks to ensure the harmony of arrangement and improve the quality of music generation. Extensive experiments on real-world dataset demonstrated the superiority of our proposed models over baselines on single-track and multi-track music generation. In practical, our model [63] had created many pop music and passed the Turing test in CCTV1[3].

However, real music usually has own style. Recent music generation models cannot satisfy the realistic requirements without considering musical style. How to control the style of generated music still remains an open problem. In this article, we further propose a *Multi-Style Multi-Instrument Co-Arrangement Model (MSMICA)*, a novel architecture of generating multi-style music. Inspired by *SeqGAN* [57], we treat the *MICA* as the generator and develop two different discriminators (i.e., *multi-style discriminator and harmony discriminator*) for multi-style music generation. The *multi-style discriminator* is helpful for *MSMICA* to control the musical style in an unsupervised manner. The *harmony discriminator* is utilized to ensure the overall harmony of generated music. Finally, we conduct systematic experiments on real-world dataset. Experimental results demonstrate that our proposed model can not only generate music with different styles but also ensure the harmony of generated music. The contributions can be summarized as follows:

—We propose an end-to-end multi-track music generation system, including both the melody and arrangement.
—For melody generation, we utilize chord progression to guide melody procession and rhythm pattern to learn the structure of a song. Then, we use rhythm and melody cross-generation method for song generation. For arrangement generation, we develop a multi-task joint generation network using other task states at every step in the decoder layer, which improves the quality of generation and ensures the harmony of multi-track music.

[3]http://tv.cctv.com/2017/11/24/VIDEo7JWp0u0oWRmPbM4uCBt171124.shtml.

— To control the musical style, we further propose a multi-style arrangement model based on reinforcement learning with adversarial training, which includes multiple discriminators to control style and harmony synchronously. Specifically, we utilize a multi-task arrangement model as the generator and share reward among different tracks to improve generation performance of musical style as well as harmony.
— Extensive experiments over real-world dataset demonstrate the superiority of effectiveness of our proposed models compared with state-of-the-art methods.

2 BACKGROUND AND RELATED WORK

In general, the related work can be grouped into the following three categories: (1) *Sequence learning*; (2) *Multi-task learning*; and (3) *Music generation.*

2.1 Sequence Learning

Sequence data is widespread in real data, such as text, signal, stock trend, and so on. Over the past century, there has been a dramatic increase in modeling the sequence problem. Traditional method as Hidden Markov model (HMM) [19] is widely used in language model, text-to-speech, biological sequences and so on. For example, Baldi et al. used HMM to model families of biological with a smooth and convergent algorithm to adapt the parameters of the model. Additionally, Tokuda et al. derived an algorithm for speech parameter generation from HMM with unobservable vector. However, traditional methods need to design and extract massive manual features, which consumes lots of manpower and time. Recently, deep learning shows excellent performance in sequence modeling [55, 58], which could model sequence in an end-to-end manner [51] to avoid massive manual features [21, 62], such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs). However, deep neural networks need more data to train massive parameters and sometimes there are not enough data for training. How to model sequence problem with few data has become a new challenge and attracts plenty researches to deal with it. For instance, Lample et al. proposed an unsupervised machine translation method using monolingual corpora only. Liu et al. [35] utilized SeqGAN [57] to generate text with few unparalleled image-text data. Previous research studies mainly focus on single-sequence model, but there are many multi-sequence problems in real world. For example, multi-track music has many tracks, and each track could be defined as a sequence and the whole is a multi-sequence problem. How to handle and define this multi-sequence problem still needs to explore.

2.2 Multi-task Learning

Multi-task learning is often used to share features within related tasks, since the features learned from one task may be useful for others. In previous works, multi-task learning has been used successfully across all applications of machine learning, from natural language processing [14, 36, 59] to computer vision [22, 60]. For example, Zhang et al. [61] proposed to improve generalization performance by leveraging the information of related tasks. Hashimoto et al. [27] pre-defined a hierarchical architecture consisting of several Natural Language Processing (NLP) tasks and designed a simple regularization term to improve the performances of all tasks. Kendall et al. [31] adjusted each task's relative weight by deriving a multi-task loss function based on maximizing the Gaussian likelihood. There still remains plenty of works on multi-task learning [38, 39, 42, 47].

2.3 Music Generation

Music generation has been a challenging task over the last decades. A variety of approaches have been proposed [5, 8]. Typical data-driven statistical methods usually employed N-gram or Markov models [12, 46, 53]. Besides, a unit selection methodology for music generation was used in [7]

Table 1. Comparing Music Generation Models (**G**: Generation, **Mt**: Multi-track, **M**: Melody, **Cp**: Chord progression, **Ar**: Arrangement, **Sa**: Singability, and **Ms**: Musical style)

Methods	G	Mt	M	Cp	Ar	Sa	Ms
Markov music [53]	√		√				
Music unit selection [7]			√				
Magenta [10]	√		√				
Song from PI [13]	√	√	√			√	
DeepBach [25]	√		√	√			
GANMidi [56]	√		√				
Sampling music sequences [45]			√	√			
XiaoIce Band [63]	√	√	√	√	√	√	
MSMICA	√	√	√	√	√	√	√

which spliced music units with ranking methods. Moreover, a similar idea was also proposed by [45], which used chords to choose melody. However, traditional methods require massive manpower and domain knowledge.

Recently, deep neural networks have been applied in music generation by the end-to-end method, which solved above problems. For example, Johnson [29] combined one recurrent neural network and one non-recurrent neural network to represent the possibility of more than one note at the same time. An RNN-based Bach generation model was proposed in [25], which was capable producing four-part chorales by using a Gibbs-like sampling procedure. Contrary to models based on RNNs, Sabathé et al. [48] used VAEs [32] to learn the distribution of musical pieces. Furthermore, Yang et al. [56] and Mogren [44] adopted Generative Adversarial Networks (GANs) [23] to generate music, which treated random noises as inputs to generate melodies from scratch. Different from single track music, Chu et al. [13] utilized hierarchical Recurrent Neural Network to generate both the melody as well as accompanying effects such as chords and drums.

Although extensive research studies have been carried out on music generation, no single study exists that considers the specificity of music, such as chord, rhythm, and instrument. For the pop music generation, previous works do not consider the chord progression and rhythm pattern. Specially, chord progression usually guides the melody procession and the rhythm pattern decides whether the song is suitable for singing. Besides, instrument characteristics should also be preserved in pop music. Lastly, harmony plays a significant role in multi-track music, but it has not been addressed very well in previous studies.

In addition, musical style is also an important characteristic of music. Recently, researchers have shown an increased interest in this area. Lu et al. [40] proposed an unsupervised music style transfer method without the need for parallel data. This method suits the wave and image data, but failed to handle sequence data, like MIDI files. To overcome the issue, Brunner et al. [9] devised a neural network model based on Variational Autoencoders to achieve style transfer between Classical and Jazz music. Though this model could handle sequence data, it needs massive parallel music data for training. Therefore, how to learn musical styles with unparalleled music data is a valuable problem.

To sum up, we compare our model with several related models and show the results in Table 1.

3 PRELIMINARIES AND PROBLEM DEFINITION

In this section, we first intuitively discuss the crucial influence of chord progression, rhythm pattern, instrument characteristic, and musical style in music generation, based on music knowledge

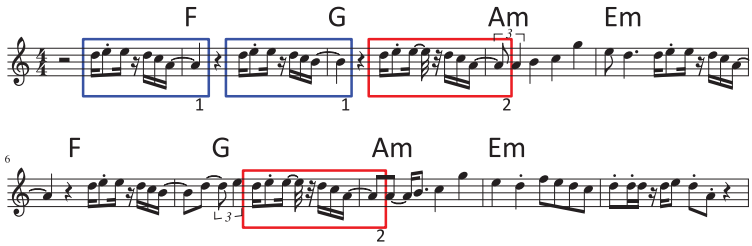
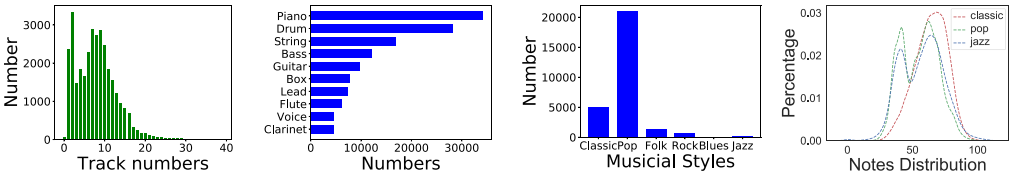


Fig. 2. Melody of the song "We Don't Talk Anymore" with chord progression labeled.



(a) Tracks distribution (b) Top 10 instruments (c) Musical styles distribution (d) Notes distribution

Fig. 3. Tracks, instruments and musical styles analysis of music.

with related statistical analysis to further support our motivation. Then we will present the music generation problem with a formulated problem definition, including melody, arrangement, and musical style.

3.1 Chord Progression

In music, chord is any harmonic set of pitches consisting of more notes that are heard as if sounding simultaneous. An ordered series of chords is called a chord progression. Chord progressions are frequently used in music and music often sounds harmonious and melodic if it follows certain chords patterns. As we can see from Figure 2, every period in melody has the corresponding chord, and "F-G-Am-Em" is the chord progression, which repeatedly appears in this music. In pop songs, the chord progression could influence the emotional tone and melody procession. For example, "C - G - Am - Em - F - C - F - G," one of the chord progressions in pop music, is applied in many songs, such as "Simple love," "Agreement," "Deep breath," and "Glory days."

3.2 Rhythm Pattern

Apart from the chords we mentioned above, rhythm pattern is another characteristic of pop music. Rhythm pattern could be defined as the duration of notes in a period. For example, the periods labeled by box in Figure 2, have the same rhythm pattern, which represents the duration of every note in a period. Different from the music generated note by note, pop music is a more structural task. However, previous works did not consider the structure of the music.

3.3 Instrument Characteristic

The another characteristic of the pop music is the arrangement, which means combing other instruments with the melody for making the whole music more contagious. In pop music, arrangement is a necessary section, and often includes drum, bass, string, and guitar to accompany the melody. We analyze the MIDI files, and the detailed statistics are shown in Figure 3(a), which indicates that the multi-track music widely exists in pop songs. Besides, as shown in Figure 3(b), piano

is usually used for representing melody and several other instruments, such as drum, bass, string, and guitar, are typically used for accompanying tracks.

3.4 Musical Style

Musical style is a recurring arrangement of features in musical events which is typical of an individual (composer and performer), a group of musicians, a genre, a place, and a period of time [20]. Music has own musical styles, such as pop, classic, jazz, and rock. It is necessary to learn the style of music, to enhance the diversity, specificity and facility of generated music. However, there are many challenges unsolved for music styles. For one thing, music data have been very unevenly distributed among different styles of music. As we can see from Figure 3(c), pop music is far more than any other styles of music. It is difficult to train the minor style music with the supervised model. For another, how to evaluate the musical style performance of generated music is still an open problem. But there are also valuable characteristics in musical style. For example, different styles of music has the similar distribution in notes distribution is shown in Figure 3(d). Therefore, we could learn the minor style of music with the help of pop music.

3.5 Problem Statement

Music generation means generating a series of notes, including pitch, duration, track, instrument, and so on. These could be defined as a form of sequence. Melody usually has a sequence and arrangement has multiple sequences. For different form of sequences, we divide our problem into the following two sections: melody generation and arrangement generation.

In melody generation, since each music usually has a specific chord progression, we consider the scenario of generating the music on the condition of given chord progression. Thus, we propose the *CRMCG* to generate melody and rhythm with the given chord progression C, showed in Section 4.1. The input of *CRMCG* is the given chord progression $C = \{c_1, c_2, \ldots, c_{l_c}\}$. Note that c_i is the one-hot representation of the chord and l_c is the length of the sequence. And outputs are suitable rhythm $R_i = \{r_{i1}, r_{i2}, \ldots, r_{il_r}\}$ and melody $M_i = \{m_{i1}, m_{i2}, \ldots, m_{il_m}\}$.

In arrangement generation, different from the single-track melody generation, we hope to generate multi-track music, which has more tracks with different instruments to accompany the melody, such as drum, bass, string, and guitar. We define this as a multi-sequence generation problem and devise a *MICA* for multi-track music in Section 4.2. The input of *MICA* is the melody M and rhythm R generated by *CRMCG* proposed above. We hope to get multi-track music output $A = \{(M_1^{'}, R_1^{'}), \ldots, (M_i^{'}, R_i^{'})\}$, where $M_i^{'}$, $R_i^{'}$ mean the i^{th} track melody and rhythm in generated arrangement. To deal with musical style, we further propose a *MSMICA* based on *MICA* to control musical style in Section 5.

In summary, we propose *CRMCG* for single track music, as well as *MICA* and *MSMICA* for multi-track music to tackle this issue. Figure 4 shows the flowchart overview of melody and arrangement generation model, which can be divided into the following four parts: (1) Data processing part; (2) *CRMCG* part for melody generation (single track); (3) *MICA* part for arrangement generation (multi-track); and (4) The display part. Figure 7 shows the multi-style arrangement model, *MSMICA*. We will introduce the structures and technical details of the proposed models in the following part, and the data processing part will be detailed in Section 6.

4 MELODY AND ARRANGEMENT GENERATION MODEL

In this section, we will introduce the structures and technical details of *CRMCG* for single track music, and *MICA* for multi-track music. For better illustration, Table 2 lists some mathematical notations used in this article.

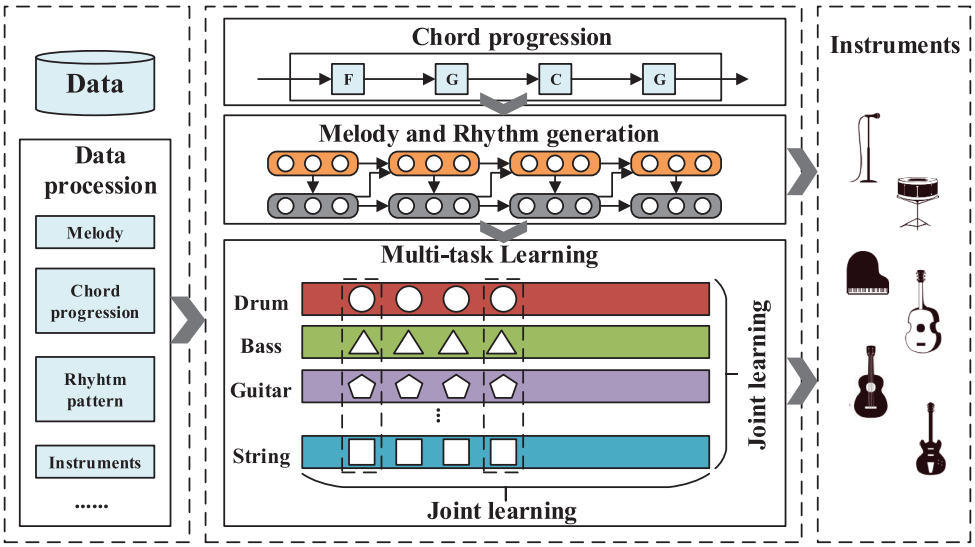


Fig. 4. The flowchart overview of melody and arrangement generation model.

Table 2. Notations Used in the Framework

Notations	Description
M	the melody sequence of pop music
R	the rhythm sequence of pop music
C	the chord progression of pop music
A	the arrangement sequence of pop music
p_i	the i-th period of pop music
m_{ij}	the j-th note in i-th period of pop music
r_{ij}	the j-th note duration in i-th period of pop music
c_i	the i-th chord of chord progression
l_m, l_r, l_c	the length of melody/rhythm/chord progression sequence respectively
$\bar{\boldsymbol{h}}^m_{i,j}, \bar{\boldsymbol{h}}^r_{i,j}, \bar{\boldsymbol{h}}^c_{i,j}$	the j-th hidden state in i-th period of melody/rhythm/chord progression sequence respectively
$\boldsymbol{h}^i_{t,k}$	the i-th task hidden state in period t at step k

4.1 Chord-based Rhythm and Melody Cross-Generation Model

As the fundamental part of music, melody is made up of a series of notes and the corresponding duration. However, it is still challenging to generate melody in harmony. Note-level generation methods [32, 48] have more randomness on the pause, which causes the music hard to sing. To address this issue, we propose *CRMCG* to generate a suitable rhythm for singing. Figure 5 shows the overall architecture of *CRMCG*.

Given a chord progression $C = \{c_1, c_2, \ldots, c_{l_c}\}$, we aim at generating the corresponding periods $\{p_1, p_2, \ldots, p_{l_p}\}$. The generated rhythm R_i and melody M_i in period p_i are closely related to the chord c_i. Since this is a sequence-to-sequence problem, we utilize encoder–decoder framework as our basic framework, which is flexible to adopt different networks to process sequence effectively.

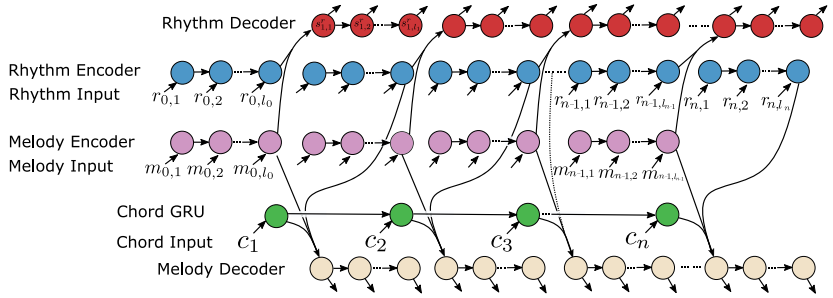


Fig. 5. CRMCG.

To better understand the chord progression and model the interaction and relation of these chords, we first utilize Gated Recurrent Units (GRU) [11] to process the low-dimension representation of chords. This process can be formulated as follows:

$$\begin{aligned} \bar{\boldsymbol{c}}_i &= \boldsymbol{E}_c \boldsymbol{c}_i, \quad \boldsymbol{E}_c \in \mathbb{R}^{V_c * d}, \\ \bar{\boldsymbol{h}}_i^c &= \mathrm{GRU}_c(\bar{\boldsymbol{c}}_i, \bar{\boldsymbol{h}}_{i-1}^c), \quad i = 1, 2, \ldots, l_c, \end{aligned} \tag{1}$$

where $\boldsymbol{E}_c$ is the embedding matrix for chord, $\bar{\boldsymbol{h}}_{i,0}^c$ is the i-step hidden state of the chord progression, and the chord embedding $\bar{\boldsymbol{c}}_i$ encodes each chord and sequence context around it. After contextual encoding, we intend to utilize these hidden states to help generate rhythm and melody. To be specific, our generation processing can be divided into two parts: rhythm generation and melody generation, which shown as follows:

Rhythm generation. It is critical that the generated rhythm should be in harmony with the existing part of music. Thus, in this part, we take the previous generation of music into consideration. To be specific, we first multiply previous rhythm R_{t-1} and melody M_{t-1} with embedding matrix $\boldsymbol{E}_r$ and $\boldsymbol{E}_m$. Then, we get the representations of $\bar{\boldsymbol{R}}_{t-1}$, $\bar{\boldsymbol{M}}_{t-1}$ as follows:

$$\begin{aligned} \bar{\boldsymbol{R}}_{t-1} &= \boldsymbol{E}_r R_{t-1}, \quad \boldsymbol{E}_r \in \mathbb{R}^{V_r * d}, \\ \bar{\boldsymbol{M}}_{t-1} &= \boldsymbol{E}_m M_{t-1}, \quad \boldsymbol{E}_m \in \mathbb{R}^{V_m * d}, \end{aligned} \tag{2}$$

where V_m and V_r are the vocabulary size of notes and beats. After getting the representations, we utilize two different GRUs to encode these inputs:

$$\begin{aligned} \bar{\boldsymbol{h}}_{t-1,i}^m &= \mathrm{GRU}_m(\bar{\boldsymbol{m}}_{t-1,i}, \bar{\boldsymbol{h}}_{t-1,i-1}^m), \quad i = 1, 2, \ldots, l_m, \\ \bar{\boldsymbol{h}}_{t-1,i}^r &= \mathrm{GRU}_r(\bar{\boldsymbol{r}}_{t-1,i}, \bar{\boldsymbol{h}}_{t-1,i-1}^r), \quad i = 1, 2, \ldots, l_r. \end{aligned} \tag{3}$$

Then, we separately concatenate the last hidden states of rhythm encoder $\bar{\boldsymbol{h}}_{t-1,l_r}^r$ and melody encoder $\bar{\boldsymbol{h}}_{t-1,l_m}^m$, and make a linear transformation. The result is treated as the initial state of rhythm decoder, which is made up by another GRU. The outputs of GRU are the probability of generated rhythm in current period, which can be formalized as follows:

$$\begin{aligned} \boldsymbol{s}_0^r &= ReLu(\boldsymbol{W}[\bar{\boldsymbol{h}}_{t-1,l_m}^m, \bar{\boldsymbol{h}}_{t-1,l_r}^r] + \boldsymbol{b}), \quad \boldsymbol{W} \in \mathbb{R}^{b*b}, \\ \boldsymbol{s}_i^r &= \mathrm{GRU}_r(\boldsymbol{y}_{i-1}^r, \boldsymbol{s}_{i-1}^r), \quad i > 0, \\ \boldsymbol{y}_i^r &= softmax(\boldsymbol{s}_i^r), \end{aligned} \tag{4}$$

where $\boldsymbol{s}_i^r$ is the hidden state of GRU for generating the i^{th} beat in the t^{th} period. With this operation, we are capable of generating the rhythm for the t^{th} period based on the previous generation.

ALGORITHM 1: CRMCG

Require: melody en-decoder GRU_m, rhythm en-decoder GRU_r, chord en-decoder GRU_c chord progression input $C = \{c_1, c_2, \ldots, c_l\}$

Output: generated melody $M = \{m_1, \ldots m_l\}$, generated rhythm $R = \{r_1, \ldots r_l\}$

```
Initialize GRU_m, GRU_r, GRU_c with random weights.
repeat
    for l-steps do
        Rhythm section training
            Generate rhythm sequence R^p_{1:T} = (r_1, ..., r_T) with previous melody M^{p-1}_{1:T} and rhythm R^{p-1}_{1:T}
            Train GRU_r via minimizing the cross-entropy loss L_r
        Melody section training
            Generate melody sequence M^p_{1:T} = (m_1, ..., m_T) with previous melody M^{p-1}_{1:T}, generated
        rhythm R^p_{1:T} and corresponding c_p
            Train GRU_m via minimizing the overall cross-entropy loss L_r + L_m
    end for
until CRMCG converges
```

Melody Generation. After generating the current rhythm, we can take advantage of this information to generate the corresponding melody. Similar to rhythm generation, we first concatenate previous melody M_{t-1}, currently generated rhythm R_t, and corresponding chords c_t. Then, we make a linear transformation in the concatenation. This process can be formulated as follows:

$$\boldsymbol{s}_0^m = ReLu(\boldsymbol{W}[\bar{\boldsymbol{h}}^m_{t-1,l_m}, \bar{\boldsymbol{h}}^r_{t,l_r}, \bar{\boldsymbol{h}}^c_l] + \boldsymbol{b}), \quad \boldsymbol{W} \in \mathbb{R}^{b*b}. \tag{5}$$

With the help of this operation, we get the initial hidden state of melody decoder $\boldsymbol{s}^m_{i-1}$. Then, we utilize GRU to process the result and generate the current melody for the whole generation:

$$\begin{aligned} \boldsymbol{s}_i^m &= \mathrm{GRU}_m(\boldsymbol{y}^m_{i-1}, \boldsymbol{s}^m_{i-1}), \quad i > 0, \\ \boldsymbol{y}_i^m &= softmax(\boldsymbol{s}^m_i). \end{aligned} \tag{6}$$

Loss Function. Since the generating process can be divided into two parts, we design two loss functions, respectively. Both of them are softmax *cross-entropy* functions. Based on the characteristic of the model, we can update the parameters alternately by parameter correlation. In rhythm section, we only update parameters related with rhythm loss L^r:

$$L^r = -\frac{1}{n}\sum_{i=0}^{n}(p \cdot log\,(p\,(y^r|x,\theta_r))) + (1-p)\,log\,(1-p(y^r|x,\theta_r)). \tag{7}$$

In melody section, considering rhythm is closely related to melody, every item in rhythm is corresponding duration of notes in melody. To better learn this relationship, we update all the parameters by melody loss L^m:

$$\begin{aligned} L^m = &-\frac{1}{n}\sum_{i=0}^{n}(p \cdot log\,(p\,(y^r|x,\theta_r))) + (1-p)\,log\,(1-p\,(y^r|x,\theta_r)) \\ &-\frac{1}{n}\sum_{i=0}^{n}(p \cdot log\,(p\,(y^m|x,\theta_m))) + (1-p)\,log\,(1-p(y^m|x,\theta_m)). \end{aligned} \tag{8}$$

4.2 Multi-task Arrangement Model

4.2.1 Multi-Instrument Co-Arrangement Model. In real-world applications, music contains more than one track, such as drum, bass, string, and guitar. However, most of existing methods

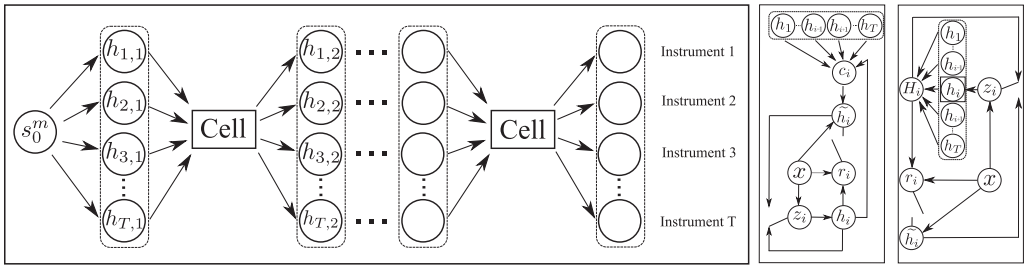


Fig. 6. (a): MICA; (b): Attention Cell; and (c): MLP Cell.

did not take this phenomenon into consideration [10, 56]. To this end, we formulate a *One-to-Many Sequences Generation (OMSG)* task. Different from conventional multiple sequences learning, the generated sequences in *OMSG* are closely related. When generating one of the sequences, we should take into account its harmony, rhythm matching, and instrument characteristic with other sequences. Previous works, such as hierarchical Recurrent Neural Network [13], did not consider the correlation between tracks. Therefore, they achieved good performance in single-track music generation, but failed in multi-track music generation. Encouraged by this evidence, we aim to model the information flow between different tracks during music generation and extend *CRMCG* to a novel *MICA*.

Given a melody, we focus on generating more tracks to accompany melody with different instruments. Different from usual encoder–decoder structure, the melody sequence is the input of the encoder, but the decoder outputs multiple sequences. As shown in Figure 6(a), the hidden $\boldsymbol{s}_0^m$ of decoder contains the melody information. MICA uses $\boldsymbol{s}_0^m$ as the initial state of decoder to generate more sequences of different instruments, where $\boldsymbol{h}_{T,i}$ represents the hidden state of GRU for task T in step i. However, how to better learn relationships and keep the harmony between different tracks is still a challenge. To this end, we designed two cooperate cells between the hidden layers of decoder to tackle this issue. These cooperate cells could improve generation performance of multiple sequences with other task information. The details of these two cells will be introduced in the following parts.

4.2.2 Attention Cell. Usually in single sequence generation model, such as GRU, the next hidden state is regard to the last hidden state and current input. As shown in Formula (9), $\boldsymbol{x}_{t,k}$ and $\boldsymbol{h}_{t,k}$ separately represents the input and hidden state at step k in the period t, where $r_{t,k}$, $z_{t,k}$ and $\widetilde{\boldsymbol{h}_{t,k}}$ are the reset gate, update gate, and state update in the GRU. However, in multiple sequence generation, the next hidden state need consider hidden states of other sequences to learn relationship of multiple sequences.

$$\begin{aligned} \boldsymbol{r}_{t,k} &= \sigma\left(\boldsymbol{W}_r \boldsymbol{x}_{t,k} + \boldsymbol{U}_r \boldsymbol{h}_{t,k-1} + \boldsymbol{b}_r\right), \\ \boldsymbol{z}_{t,k} &= \sigma\left(\boldsymbol{W}_z \boldsymbol{x}_{t,k} + \boldsymbol{U}_z \boldsymbol{h}_{t,k-1} + \boldsymbol{b}_z\right), \\ \widetilde{\boldsymbol{h}_{t,k}} &= \sigma\left(\boldsymbol{W} \boldsymbol{x}_{t,k} + \boldsymbol{U}\left[\boldsymbol{r}_{t,k} \cdot \boldsymbol{h}_{t,k-1}\right] + \boldsymbol{b}\right), \\ \boldsymbol{h}_{t,k} &= \left(1 - \boldsymbol{z}_{t,k}\right) \cdot \boldsymbol{h}_{t,k-1} + \boldsymbol{z}_{t,k} \cdot \widetilde{\boldsymbol{h}_{t,k}}. \end{aligned} \tag{9}$$

Motivated by attention mechanism [2], which can help the model focus on the most relevant parts of the input for the output, we design a creative attention cell to capture the relevant parts of other tasks for current task. Figure 6(b) shows the architecture of this attention cell. It can be formalized

as follows:

$$\begin{aligned} \boldsymbol{a}^i_{t,k} &= \sum_{j=1}^{T} \alpha_{t,ij} \boldsymbol{h}^j_{t,k-1}, \\ \boldsymbol{e}_{t,ij} &= \boldsymbol{v}^T tanh(\boldsymbol{W}\boldsymbol{h}^i_{t,k-1} + \boldsymbol{U}\boldsymbol{h}^j_{t,k-1}), \quad \boldsymbol{W}, \boldsymbol{U} \in \mathbb{R}^{b*b}, \\ \alpha_{t,ij} &= \frac{exp(\boldsymbol{e}_{t,ij})}{\sum_{s=1}^{T} exp(\boldsymbol{e}_{t,is})}, \end{aligned} \tag{10}$$

where $a^i_{t,k}$ represents the cooperate vector for task i at step k in the period t, $h^j_{t,k-1}$ represents the hidden state of the i^{th} task at step $k-1$ in the period t, and $\alpha_{t,ij}$ represents the weight of j^{th} task information captured by i^{th} task. The cooperate vector $a^i_{t,k}$ means i^{th} task capture the relevant parts of other tasks. Specially, we define the $e_{t,ij}$ the correlation value between i^{th} task and j^{th} task and use a multilayer perceptron to reduce computation rather than calculating $h^i_{t,k-1} \times h^j_{t,k-1}$. After getting the cooperation vector, we modify the cell of GRU to allow the current track generation to take full account of the impacts of the information from other tracks. The modifications are formulated as follows:

$$\begin{aligned} \boldsymbol{r}^i_{t,k} &= \sigma(\boldsymbol{W}^i_r \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i_r \boldsymbol{h}^i_{t,k-1} + \boldsymbol{A}^i_r \boldsymbol{a}^i_{t,k} + \boldsymbol{b}^i_r), \\ \boldsymbol{z}^i_{t,k} &= \sigma(\boldsymbol{W}^i_z \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i_z \boldsymbol{h}^i_{t,k-1} + \boldsymbol{A}^i_z \boldsymbol{a}^i_{t,k} + \boldsymbol{b}^i_z), \\ \widetilde{\boldsymbol{h}^i_{t,k}} &= \sigma(\boldsymbol{W}^i \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i \left[\boldsymbol{r}^i_{t,k} \cdot \boldsymbol{h}^i_{t,k-1}\right] + \boldsymbol{A}^i \boldsymbol{a}^i_{t,k} + \boldsymbol{b}^i), \\ \boldsymbol{h}^i_{t,k} &= (1 - \boldsymbol{z}^i_{t,k}) \cdot \boldsymbol{h}^i_{t,k-1} + \boldsymbol{z}^i_{t,k} \cdot \widetilde{\boldsymbol{h}^i_{t,k}}, \end{aligned} \tag{11}$$

where σ is the activate function and $\boldsymbol{W}^i_r$, $\boldsymbol{U}^i_r$, $\boldsymbol{A}^i_r$, $\boldsymbol{W}^i_z$, $\boldsymbol{U}^i_z$, $\boldsymbol{A}^i_z$, $\boldsymbol{W}^i$, $\boldsymbol{U}^i$, $\boldsymbol{A}^i$, $\boldsymbol{b}^i$ is corresponding weights of task i. To learn relationships between different tasks, we modify the structure of GRU, adding cooperation vector $a^i_{t,k}$ into state update $h^i_{t,k}$, update gate $z^i_{t,k}$, and reset gate $r^i_{t,k}$, to capture other tasks information in decoder procession. With the help of this operation, our model can generate every track for one instrument with the consideration of other instruments.

4.2.3 MLP Cell. Different from the above cell for sharing task information through input $\boldsymbol{x}^i_{t,k}$, we consider the individual hidden state of each instrument and integrate them by their importance for the whole music. Therefore, our model is capable of choosing the most relevant parts of each instrument to improve the overall performance. Figure 6(c) shows the structure of this cell, which can be formalized as follows:

$$\begin{aligned} \boldsymbol{r}^i_{t,k} &= \sigma(\boldsymbol{W}^i_r \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i_r \boldsymbol{H}^i_{t,k-1} + \boldsymbol{b}^i_r), \\ \boldsymbol{z}^i_{t,k} &= \sigma(\boldsymbol{W}^i_z \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i_z \boldsymbol{H}^i_{t,k-1} + \boldsymbol{b}^i_z), \\ \widetilde{\boldsymbol{h}^i_{t,k}} &= \sigma(\boldsymbol{W}^i_h \boldsymbol{x}^i_{t,k} + \boldsymbol{U}^i_h \left[\boldsymbol{r}^i_{t,k} \cdot \boldsymbol{H}^i_{t,k-1}\right]), \\ \boldsymbol{h}^i_{t,k} &= (1 - \boldsymbol{z}^i_{t,k}) \cdot \boldsymbol{H}^i_{t,k-1} + \boldsymbol{z}^i_{t,k} \cdot \widetilde{\boldsymbol{h}^i_{t,k}}, \\ \boldsymbol{H}^i_{t,k-1} &= \sigma(\boldsymbol{W}^i \left[\boldsymbol{h}^1_{t,k-1}, \ldots, \boldsymbol{h}^N_{t,k-1}\right] + \boldsymbol{b}^i), \end{aligned} \tag{12}$$

where $\boldsymbol{H}^i_{t,k-1}$ is the i^{th} task hidden state in period t at $k-1$ step which contains all tasks current information $\boldsymbol{h}^1_{t,k-1}, \ldots, \boldsymbol{h}^N_{t,k-1}$ by gate units. σ is the sigmoid activate function. $\boldsymbol{W}^i_r$, $\boldsymbol{U}^i_r$, $\boldsymbol{W}^i_z$, $\boldsymbol{U}^i_z$, $\boldsymbol{W}^i_h$, $\boldsymbol{U}^i_h$, $\boldsymbol{W}^i$, $\boldsymbol{b}^i$ is corresponding weights of task i. Since our model shares each track information at decoding step, it can obtain the overall music information and generate music in harmony.

4.2.4 Loss Function. Motivated by [17], we optimize the summation of several conditional probability terms conditioned on representation generated from the same encoder.

$$L(\theta) = \underset{\theta}{argmax}\left(\sum_{T_k}\left(\frac{1}{N_p}\sum_{i}^{N_p} logp\left(Y_i^{T_k}|X_i^{T_k};\theta\right)\right)\right),$$

where $\theta = \{\theta_{src}, \theta_{trg_{T_k}}|T_k = 1, 2, \ldots, T_m\}$, and m is the number of tasks. θ_{src} is collection of parameters for source encoder, and $\theta_{trg_{T_k}}$ is the parameter set of the ${T_k}^{th}$ target track. N_p is the size of parallel training corpus of the p^{th} sequence pair.

4.2.5 Generation. In generation part, we arrange for melody generated by *CRMCG*. With the help of *CRMCG*, we get a melody sequence $M = \{m_1, m_2, \ldots, m_{l_m}\}$, and the next step is to generate other instrument tracks to accompany it. Similarly, we utilize GRU to process the sequence and get the initial state $\boldsymbol{s}_{l_m}^m$ of multi-sequences decoder. They can be formulated as follows:

$$\begin{aligned} \bar{\boldsymbol{m}}_{\boldsymbol{i}} &= \boldsymbol{E}_m m_i, \quad \boldsymbol{E}_m \in \mathbb{R}^{V_m * d}, \\ \boldsymbol{s}_{l_m}^m &= \mathrm{GRU}_m\left(\bar{\boldsymbol{m}}_{l_m}, \boldsymbol{s}_{l_m-1}^m\right), \end{aligned} \tag{13}$$

the outputs of multi-sequences decoder correspond other instrument tracks, considering both melody and other accompanying tracks. They can be formalized as follows:

$$\begin{aligned} \boldsymbol{s}_t^i &= \mathrm{CellOperation}(\boldsymbol{y}_{t-1}^i, \boldsymbol{s}_{t-1}^i), \quad \mathrm{CellOperation} \in \{\mathrm{AttentionCell}, \mathrm{MLPCell}\}, t > 0, \\ \boldsymbol{y}_t^i &= softmax(\boldsymbol{s}_t^i), \end{aligned} \tag{14}$$

where $\boldsymbol{s}_t^i$ is the i^{th} task hidden state at step t. We utilize $\boldsymbol{s}_t^i$ as input to generate the i^{th} instrument sequence through $softmax$ layer. The proposed Attention Cell and MLP Cell are used to get a cooperation state, which contains self-instrument state as well as other instrument states, to keep all instruments in harmony.

5 MULTI-STYLE ARRANGEMENT MODEL

In the previous section, we utilize *CRMCG* to generate single track music, and *MICA* to generate multi-track music. However, the generated music belongs to the same musical style, which is insufficient for diversification requirements of music generation. How to control the style of generated music remains an open problem that needs to be solved urgently. Based on our *MICA* model proposed in Section 4.2, we propose a newly designed *MSMICA* to generate multi-track music with musical style. The input of *MSMICA* is rhythm R and melody M generated by *CRMCG*, and output is the arrangement sequence $A^s = \{(M_1^s, R_1^s), \ldots, (M_i^s, R_i^s)\}$, where M_i^s, R_i^s denotes the i^{th} track melody and rhythm in multi-track music with a style s. Figure 7 shows the flowchart overview of multi-style melody and arrangement generation model.

In the next part, we will introduce the structure and technical details of our newly designed *MSMICA* model.

5.1 Multi-Style Multi-Instrument Co-Arrangement Model

Though *MICA* could generate harmonious multi-track music, it fails to control the musical style. How to control the style of the generated music is still unsolved. As previously mentioned, there are the following two characteristics in music: (1) Music data has been very unevenly distributed among different styles of music. We could train a pop music generation model with massive pop music data, but could not get enough minor style music data, such as jazz, rock, and classic. (2) Different style of music has similar notes distributions, proving they have commonness in melody and

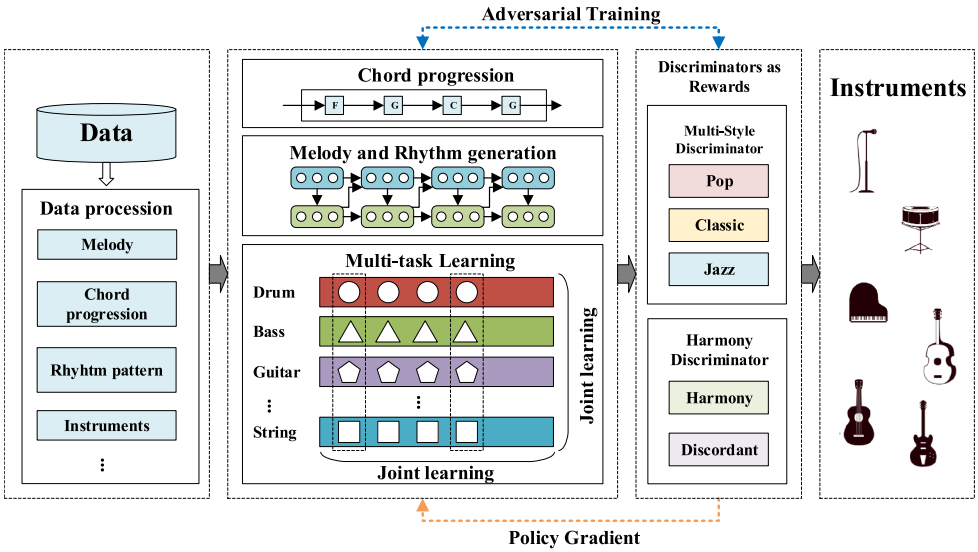


Fig. 7. The flowchart overview of multi-style melody and arrangement generation model.

arrangement. Therefore, we could learn the minor style of music from the major style of music. Motivated by [35, 50], we propose a new *MSMICA*, to control the style of the generated music. We first generate the harmonious multi-track music with *MICA*. Then, we devise *Multi-Style Discriminator* and *Harmony Discriminator* to control musical style. The *Multi-Style Discriminator* is designed to learn the style of the generated music, and *Harmony Discriminator* is used to keep the harmony of music. The technical details of the *MSMICA* will be introduced in the following parts.

5.2 Multi-SeqGAN

In order to learn the musical style with little unpaired data, we utilize an unsupervised (GAN [23] to deal with it. GAN usually utilizes a discriminative model to guide the training of generative model, which lets the generator constantly approach the real data. To be specific, we train a music generator G for musical style s by discriminative model D as follows:

$$\min_{G}\max_{D} V(D,G) = E_{s\sim p_{data}(s)}[logD(s)] + E_{x\sim p_x(x)}[log(1-D(G(x)))], \tag{15}$$

$p_x(x)$ denotes music x generated by G, we compare it with real music $p_{data}(s)$ based on the style s. We define it as a two-player minimax game with value function $V(D,G)$. However, the sequence is discrete, which is difficult for model to pass the gradient from the discriminative model to the generative model.

Inspired by SeqGAN [57], we utilize reinforcement learning [43] to train the model. Different from the original SeqGAN that usually handles single-sequence problems, multi-track music generation is a multiple sequences problem, which has more challenges. On the one hand, considering multi-track sequences are related to each other, we need to keep the harmony of music in the fine-tuning procession. On the other hand, how to design the reward for different sequences is a new problem. To solve above problems, we first develop a variant of SeqGAN (i.e., *Multi-SeqGAN*) to simultaneously deal with multiple sequences with a new sampling and reward method. Then, we devised multiple discriminators to control musical style as well as harmony. We will discuss these in the next sections.

5.3 Generator

Traditionally, sequence generation problems could be defined as follows: training a θ-parameterized generative model G_θ to produce a sequence $Y_{1:T} = (y_1, y_2, \ldots, y_T)$. Different from previous work [50], our arrangement generation model has multi-sequence output $A = \{(M_1^{'}, R_1^{'}), \ldots, (M_i^{'}, R_i^{'})\}$, where $M_i^{'}, R_i^{'}$ denote melody and rhythm of the i^{th} track. Thus, we optimize the generative model G_θ to output multi-sequence $\{Y_{1:T}{}^1, Y_{1:T}{}^2, \ldots, Y_{1:T}{}^I\}$, where $Y_{1:T}^i = ({y_1}^i, {y_2}^i, \ldots, {y_T}^i)$ and i denotes the i^{th} track. Similar to SeqGAN, we first use reward from discriminator to train the generator based on reinforcement learning. Then, we update parameters of generator and discriminator with adversarial training. We divide the model procession as three sections as follows:

5.3.1 Sampling. Considering discriminators only provide a reward after a finished sequence, we utilize Monte Carlo search with a roll-out policy G_β to sample the unknown last $T - t$ tokens:

$$\left\{Y_{1:T}^1, \ldots, Y_{1:T}^N\right\} = MC^{G_\beta}(Y_{1:t}; N). \tag{16}$$

Motivated by this, the sampling matrix of multiple sequences could be defined as:

$$Y^s = \begin{bmatrix} Y_{1:1}^1 & Y_{1:2}^1 \cdots & Y_{1:T}^1 \\ Y_{1:1}^2 & Y_{1:2}^2 \cdots & Y_{1:T}^2 \\ \vdots & \vdots & \vdots \\ Y_{1:1}^i & Y_{1:2}^i \cdots & Y_{1:T}^i \end{bmatrix}, \quad S_t = Y_{1:t}^1 \oplus Y_{1:t}^2 \oplus \cdots \oplus Y_{1:t}^i, \tag{17}$$

where $Y_{1:t}^i$ is sampling result of the i^{th} sequence at step t, which denotes sampling unknown $T - t$ tokens based on existing t-tokens sequence. Considering the relationships among tracks, we concatenate multiple sequences as S_t, where t represents the t^{th} sampling step.

5.3.2 Reward. Similar with SeqGAN, we use discriminator $D_\phi(S_t^i)$ as the reward of the generative model. Due to the reward of policy gradient can be computed as the sum over all sequences of valid actions and treated as the expected future reward, we have:

$$\begin{aligned} Q_{D_\phi}^{G_\theta}(s = S_{t-1}, a = a_t) &= \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi\left(S_T^n\right), S_T^n \in MC^{G_\beta}(S_t; N) & for \quad t < T \\ D_\phi(S_t) & for \quad t = T \end{cases}, \\ a_t &= y_t^1 \oplus y_t^2 \oplus \cdots \oplus y_t^i, \end{aligned} \tag{18}$$

where $Q_{D_\phi}^{G_\theta}$ is the reward from discriminator D_ϕ at the t^{th} step time. S_{t-1} includes existing multiple sequences at $t - 1$ step and a_t is the t^{th} step action, which has multiple actions of sequences. Considering the relationships among tracks, we define multiple unknown tokens as actions and share the reward between them at the same step. The reward matrix could be defined as follows:

$$G(Y^s) = \begin{bmatrix} G\left(Y_{1:1}^1\right) & G\left(Y_{1:2}^1\right) \cdots & G\left(Y_{1:T}^1\right) \\ G\left(Y_{1:1}^2\right) & G\left(Y_{1:2}^2\right) \cdots & G\left(Y_{1:T}^2\right) \\ \vdots & \vdots & \vdots \\ G\left(Y_{1:1}^i\right) & G\left(Y_{1:2}^i\right) \cdots & G\left(Y_{1:T}^i\right) \end{bmatrix} = \begin{bmatrix} Q_{D_\phi}^{G_\theta}(S_0, a_1) & Q_{D_\phi}^{G_\theta}(S_1, a_2) \cdots & Q_{D_\phi}^{G_\theta}(S_{T-1}, a_T) \\ Q_{D_\phi}^{G_\theta}(S_0, a_1) & Q_{D_\phi}^{G_\theta}(S_1, a_2) \cdots & Q_{D_\phi}^{G_\theta}(S_{T-1}, a_T) \\ \vdots & \vdots & \vdots \\ Q_{D_\phi}^{G_\theta}(S_0, a_1) & Q_{D_\phi}^{G_\theta}(S_1, a_2) \cdots & Q_{D_\phi}^{G_\theta}(S_{T-1}, a_T) \end{bmatrix} \tag{19}$$

5.3.3 Loss Function. We set the generator as the environment, expected token is action and reward is the result from discriminator. To improve all reward of the environment, we maximize all samplings reward as follows:

$$J(\theta) = \sum_{i \in I} \sum_{y^i_{1:T} \in Y} P_\theta \left(y^i_{1:T}\right) R\left(y^i_{1:T}\right) = \sum_{i \in I} \left(E_{y^i_{1:T} \sim p_\theta} \sum_{t=1}^{T} r\left(y^i_{1:t}\right) \right), \tag{20}$$

where i denotes the i^{th} sequence, $r(y^i_{1:t})$ represents the reward achieved at time t, and $R(y^i_{1:T})$ is the cumulative reward. Let $p_\theta(y^i_t | y^i_{1:(t-1)})$ be a parametric conditional probability of selecting y^i_t at time step t given all the previous tokens $y^i_{1:(t-1)}$. p_θ is defined as a parametric function of policy θ.

5.4 Discriminators

To generate multi-track music with a specific style, the following two important requirements should be satisfied: (1) the specific musical style should be controlled, such as pop, classic, and jazz; and (2) the harmony of the generated multi-style music should be kept. To meet these requirements, we propose two discriminators, i.e., *Multi-Style Discriminator* and *Harmony Discriminator*. The *Multi-Style Discriminator* is designed to learn the musical style and *Harmony Discriminator* is used to maintaining the harmony of different tracks. Two discriminators both utilize classifiers as base models with different training data. The details of these two discriminators will be introduced in the following parts.

5.4.1 Multi-Style Discriminator. This discriminator is designed to control the style of generated music. We utilize this module to recognize music with three different categories: pop, jazz, and classic. For example, we define jazz as positive examples, and pop and classic as negative examples when to generate jazz music. In order to achieve this goal, we first use GRU to get the hidden state c of the multi-track music sequence y. Then, we use multi-layer perception and softmax function to get the classification result. This process can be formulated as follows:

$$\begin{aligned} \boldsymbol{c} &= GRU(y), \\ \boldsymbol{f} &= tanh\left(\boldsymbol{W}_c \odot \boldsymbol{c} + \boldsymbol{b}_c\right), \\ C_m &= softmax(\boldsymbol{W}_m \cdot \boldsymbol{f} + \boldsymbol{b}_m), \end{aligned} \tag{21}$$

where $\boldsymbol{W}_c, \boldsymbol{b}_c, \boldsymbol{W}_m, \boldsymbol{b}_m$ are trainable parameters, $\odot$ is element-wise multiplication, and C_m denotes the probabilities over three classes of the multi-modal discriminator. We utilize GRU encoder as a discriminator to generate the probability $C_m(c|x, y)$ where $c \in \{pop, jazz, classic\}$.

5.4.2 Harmony Discriminator. With multi-style discriminator, we are capable of controlling the style of the generated music after fine-tuning with policy gradient. However, the harmony of generated music may not be kept. In order to achieve musical style controlling while giving consideration to harmony, we propose another discriminator to keep the harmony of generated music, called *Harmony Discriminator*. In concrete details, we prepare harmony and discordant music data. Harmony music is the real data, melody with corresponding arrangement. Additionally, we choose other music track to replace the original music to disorder the harmony of music for making discordant music data. Then, we propose a harmony discriminator D_h to guide generated music towards harmony music. Generated music will be classified into the following three classes: harmony, generated, and discordant. Additionally, harmony music is addressed as positive examples and other music are defined as negative examples. Similar as above discriminator, we also utilize GRU and fully connected layer to get the hidden state of music sequence y:

$$C_h = softmax\left(\boldsymbol{W}_h \odot GRU(\boldsymbol{y}) + \boldsymbol{b}_h\right), \tag{22}$$

ALGORITHM 2: MSMICA

Require: A generator G_θ, discriminator D_ϕ, policy G_β, rhythm input $R = \{r_1, r_2, \ldots, r_l\}$ and melody input $M = \{m_1, m_2, \ldots, m_l\}$, style s

Output: $A^s = \left\{\left(M_1^s, R_1^s\right), \ldots, \left(M_i^s, R_i^s\right)\right\}$, where M_i^s, R_i^s mean the i^{th} track melody and rhythm

```
Initialize G_θ, D_φ with random weights θ, φ.
Pre-train G_θ using MLE on R and M.
Generate negative samples using G_θ for training D_φ.
Pre-train D_φ via minimizing the cross entropy
repeat
    for g-steps do
        Generate multiple sequences Y^s ~ G_θ
        for t in 1:T do
            Compute Q(a = a_t; s = S_{t-1}) by Equation (18)
        end for
        Update generator parameters via policy gradient Equation (20)
    end for
    for d-steps do
        Use current G_θ to generate negative examples and combine with real data given as positive
        examples
        Train discriminator D_φ for k epochs
    end for
until MSMICA converges
```

where $\boldsymbol{W}_h$, $\boldsymbol{b}_h$ are parameters to be learned. The probability of classifying generated music y to a class c is formulated as $C_h(c|y)$ where $c \in \{harmony, generated, discordant\}$.

5.5 Model Learning

In this section, we will introduce the training details of our *MSMICA* model. As previously mentioned, we propose two discriminators to get rewards of style and harmony. Therefore, we utilize rewards as policy gradients to train our *MSMICA* model. Here, we define the reward function for policy gradient as a linear combination of probability of classifying generated music y weighted by parameter λ:

$$R(y|\cdot) = \lambda C_s(c = style|y) + (1 - \lambda) C_h(c = harmony|y). \tag{23}$$

Since our method also belongs to adversarial methods, we train our model as a minimax game between a generator G and a discriminator D with value function $V(G, D)$. Motivated by [35], we use multiple discriminators as D:

$$\min_G \max F(V(D_1, G), \ldots, V(D_n, G)), \tag{24}$$

where $n = 2$, and F denotes linear combination of discriminators as shown in Equation (23). Before we train our generation model with policy gradients from discriminators, we need pre-train our generator G based on maximum likelihood estimation (MLE). As mentioned in [50], we also pre-train our discriminators, which could help adjust the generator efficiently. After the pre-training, we train the generator and discriminators alternately. The generator aims to fool discriminators until they can't distinguish generated and real music. The overall flow is shown in Algorithm 2.

Table 3. Dataset Description

Statistics	Pop music	Classic music	Jazz music
# of music	28,772	4,593	1,730
# of all tracks	327,908	27,697	16,156
# of drum tracks	37,964	1,249	2,872
# of bass tracks	33,164	1,115	1,756
# of string tracks	46,602	3,848	896
# of guitar tracks	58,266	1,178	2,344
# of piano tracks	34,926	8,560	2,665
# of other instruments tracks	116,986	14,747	5,713
# Time of all tracks (hours)	21,282	3,186	1,552

Table 4. Musical Style Analysis

Music Style	Instrument	Track	Tone	Note Range
Pop	Piano, Guitar, String, Drum, Bass	9,8,7,10,1	C major, A minor, C minor, G major, F major	C6-B, C4-B, C5-B, C7-B, C3-B
Classic	Piano, String, Violin, Bass, Drum	4,3,5,9,8	C major, G major, D major, F major, D minor	C6-B, C5-B, C7-B, C4-B, C8-B
Jazz	Piano, Bass, Guitar, Trumpet, Sax	7,8,9,6,5	C major, A minor, C minor, G major, F major	C6-B, C5-B, C4-B, C3-B, C7-B

6 EXPERIMENT

In this section, we first introduce the dataset and training details. Then, we evaluate the effectiveness of *CRMCG* on the **Melody Generation** task, *MICA* on **Arrangement Generation** task, and *MSMICA* on **Musical Style Evaluation** task, respectively.

6.1 Data Description

In this subsection, we introduce a real-world dataset that we conducted our experiments on. This dataset consists of more than 94,770 MIDI (a digital score format) files downloaded from midi music website. MIDI file is a digital score format of music, which records the pitch, duration, tempo, style, instrument, and so on. To avoid biases, those incomplete MIDI files, e.g., music without vocal track was removed. Finally, 28,772 MIDI files were kept in our dataset. Specifically, each MIDI file contains various categories of audio tracks, such as melody, drum, bass, and string. Some basic statistics of the pruned dataset are summarized in Table 3. We exposed our dataset online.[4]

To explore the characteristic of musical style, we analyzed commonness and differentia among pop, classic, and jazz music from the following four aspects: Instrument, Track, Tone, and Note Range:

— **Instrument.** Different style of music usually uses different instruments to perform, but they have common instruments, such as Piano, String, Bass, and Guitar, which can be observed from Table 4. We use these common instruments to learn the change of musical style.

[4] https://data.bdaa.pro/datasets/KDD18-Zhu/.

—**Track.** Table 4 shows that different style of music usually has multiple tracks. In this way, we choose to learn the musical style in arrangement.
—**Tone.** Music has different tones and will cause chaos with different tones. Thus, we will convert all music to the same tone, and C major is the best choice as shown in Table 4.
—**Note Range.** From the note range of different style of music, we found different music has similar notes distribution of pop, jazz, and classic. Thus, we could learn minor music with massive major music.

Motivated by the data analysis, in our work, we choose several common instruments to do experiment, such as piano, drum, bass, string, and guitar. Besides, we also analyze track numbers, tone, note range and duration of different style music, which show similar distribution between them, proving that we could learn minor style music from major style music.

Additionally, we make massive pretreatment of MIDI files, including *Tone correction*, *BPM normalization*, *Track extraction*, and *Period division* as follows:

—**Tone correction.** Music has different tones. We converted all MiDI files to C major or A minor to keep all the music in the same tune.
—**BPM normalization.** We set BPM (Beats Per Minute) as 60 to control duration of one beat could be 1 second, ensuring all notes correspond to an integer beat.
—**Track extraction.** Multi-track music has different tracks in different instruments. We extract melody and other instrument tracks to make arrangement data.
—**Period division.** We merged every two bars into a period as sequence data to train our model.
—**Chord progression extraction.** We extract serial chords from period music by detecting chord for every bar and define this as the chord progression in the training part of *CRMCG*. In the generation part, we choose more than one hundred existing chord progression widely used in real music as the model input to improve generation performance.

6.2 Training Details

In the melody and arrangement generation section, we randomly select 17,263 instances from the dataset as the training data, another 5,754 for tuning the parameters, and the rest 5,755 as test data to validate the performance. In our model, the number of recurrent hidden units are set to 256 for each GRU layer in encoder and decoder. The dimensions of parameters to calculate the hidden vector in Attention Cell and MLP Cell are set as 256. The model is updated with the Stochastic Gradient Descent [6] algorithm where batch size set is 64, and the final model is selected according to the cross-entropy loss on the validation set. To better learn the relationship between rhythm and melody, *CRMCG* trains rhythm section and melody section crosswise. Specially, the model updates rhythm part only with rhythm parameters, while updating melody part with all parameters.

In musical style generation section, we firstly pre-train generator and discriminators, which can provide a better policy initialization. In detail, we set the *MICA* as base model of generator and GRU as discriminator. For discriminators, we utilize different style music as positive samples and generated music as negative samples for style discriminator, while we use real music as positive samples and disordered multi-track music as negative samples to train harmony discriminator. We set hidden state size of GRU as 512 and use Stochastic Gradient Descent to update discriminators. After pre-training procession, we train the generator and discriminators crosswise with times of 1 to 5 in adversarial training. Discriminators first use generated music as negative data to train model parameters. Then generator uses the reward from discriminators to update the generator model. We use Adam optimizer with learning rate 10^{-3} to fine-tuning generator as suggested by [57]. In order to balance the weight of reward from two discriminators, we do experiments to find the λ of

Table 5. Human Evaluation of Melody Generation

Methods	Rhythm	Melody	Integrity	Singability	Average
Magenta (RNN) [10]	3.1875	2.8125	2.8000	2.6000	2.8500
GANMidi (GAN) [23]	1.7125	1.7625	1.3500	1.4250	1.5625
CRMCG (full)	**3.7125**	**3.8125**	**3.7125**	**3.9000**	**3.7844**
CRMCG (w/o chord progression)	3.7000	3.5875	3.4375	3.8000	3.6312
CRMCG (w/o cross-training)	3.6375	3.5500	3.3500	3.6250	3.5406

linear combination parameter is set 0.2, which could improve performance of style and harmony better.

6.3 Melody Generation

In this subsection, we conduct Melody Generation Task to validate the performance of our *CRMCG* model. That is, we only use the melody track extracted from the original MIDI music to train the models and evaluate the aesthetic quality of the melody track generation results.

6.3.1 Baseline Methods. We select two state-of-the-art models for music generation as baselines:

- **Magenta (RNN).** A RNN-based model [10], which is designed to model polyphonic music with expressive timing and dynamics.
- **GANMidi (GAN).** A novel GAN-based model [56], which uses conditional mechanism to exploit versatile prior knowledge of music.

In addition to the proposed *CRMCG* model, we evaluate two variants of the model to validate the importance of chord progression and cross-training methods on melody generation:

- **CRMCG (full).** Proposed model, which generates melody and rhythm crosswise with chord progression information.
- **CRMCG (w/o chord progression).** Based on CRMCG (full), without chord information.
- **CRMCG (w/o cross-training).** Based on CRMCG (full), we train melody and rhythm patterns respectively based on L^m and L^r during the training processing.

6.3.2 Overall Performance. Considering the uniqueness of the music generation, there is not a suitable quantitative metric to evaluate the melody generation result. Thus, we validate the performance of models based on human study. Following some point concepts in [49], we use the metrics listed below:

- **Rhythm.** Does the music sounds fluent and pause suitably?
- **Melody.** Are the music notes relationships natural and harmonious?
- **Integrity.** Is the music structure complete and not interrupted suddenly?
- **Singability.** Is the music suitable for singing with lyrics?

We invited eight volunteers, who are experts in music appreciation, to evaluate the results of various methods. Volunteers rated every generated music with a score from 1 to 5 based on above evaluation metrics. The performance is shown in Table 5. According to the results, we realize that our *CRMCG* model outperforms all the baselines with a significant margin on all the metrics, demonstrating the effectiveness of our *CRMCG* model on Melody Generation. Especially, *CRMCG* (full) performs better than *CRMCG* (w/o chord), which verifies that the chord information can

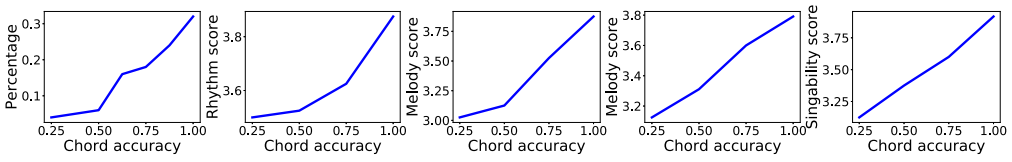


Fig. 8. Chord progression analysis compared with human study.

enhance the quality of melody. In addition, we also observe that cross-training can improve the quality of 6.9% on average, which proves effectiveness of our cross-training algorithm. At the same time, we find that the RNN-based baseline outperforms the GAN-based model which uses convolutional neural networks to generate melody. This phenomenon indicates that RNN-based model is more suitable for Melody Generation, which is the reason why we design *CRMCG* based on RNN.

6.3.3 Chord Progression Analysis. Here we further analyze the performance of chord progression in our *CRMCG* model. We define *Chord Accuracy* to evaluate whether chords of generated melodies match the input chord sequence:

$$Chord\ Accuracy = \sum_{i=1}^{P} e(y_i, \widetilde{y_i})/P,$$

$$e\left(y_i, \widetilde{y_i}\right) = \begin{cases} 1, & if\ y_i = \widetilde{y_i} \\ 0, & if\ y_i \neq \widetilde{y_i} \end{cases},$$

where P is the number of the periods, y_i is the i^{th} chord of generated melody detected through [28], and $\widetilde{y_i}$ is the i^{th} corresponding chord in given chord progression.

The performance is shown in Figure 8(a). Specially, the average Chord Accuracy of our generated melody is 82.25%. Moreover, we show the impact of Chord Accuracy of generated melody on different metrics in Figure 8(b)–(e). From the results, we realize that as the chord accuracy increases, the quality of melody generation improves significantly, which also confirms the importance of using the chord information on Melody Generation.

6.3.4 Rest Analysis. Rests are intervals of silence in pieces of music, and divide a melody sequence into music segments of different lengths. It is important to provide spaces to allow listeners to absorb each musical phrase before the next one starts. To create satisfying music, it is necessary to keep a good dynamic balance between musical activity and rest. Therefore, we evaluate the performance of rests in our generated music by contrasting the differences between distributions of the length of the music segments in generated music and original ones. Figure 9 shows the length distributions of the music segments, where minimum, average and maximum represent the different length distributions of real music and minimum_g, average_g and maximum_g represent the different length distributions of generated music. From the figure, our generated music has similar distributions on music segment lengths with original ones, which verifies that our CRMCG model can learn the appropriate rests in pieces of music and ensure the rhythm of generated music is suitable to listen.

6.4 Arrangement Generation

In this subsection, we conduct Multi-track Music Generation to validate the performance of our *MICA* model. We select five most important tasks in Multi-track Music Generation, i.e., melody, drum, bass, string, and guitar, shown in Figure 3(b).

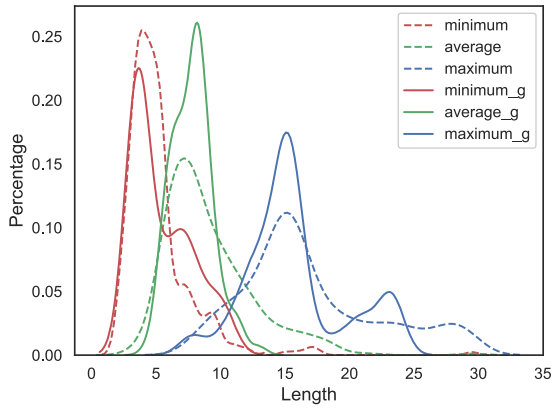


Fig. 9. Rhythm distribution.

Table 6. Human Evaluation of Arrangement Generation

Methods	Overall	Drum	Bass	String	Guitar
HRNN[13]	3.2500	2.9875	3.0875	2.8000	2.8625
MICA (w/ att)	3.6625	3.0750	2.8000	3.2125	3.0000
MICA (w/ mlp)	**3.8125**	**3.1000**	**3.4625**	**3.3125**	**3.3500**

6.4.1 Baseline Methods. To validate the performance of our two *MICA* models, a relevant model HRNN [13] is selected as baseline method. Specifically, we set the comparison methods as follows:

- **HRNN.** A hierarchical RNN-based model [13], which is designed to generate multi-track music. In particular, it uses a low-level structure to generate melody and higher-level structures to produce the tracks of different instruments.
- **MICA w/ Attention Cell.** The proposed model, which uses Attention Cell to share information between different tracks.
- **MICA w/ MLP Cell.** The proposed model, which uses MLP Cell to share information between different tracks.

6.4.2 Overall Performance. Different from the Melody Generation task, we ask volunteers to evaluate the quality of generated music in a holistic dimension. They score the performance of different instruments in music as well as the overall performance, which music is played with all instruments. The performance is shown in Table 6. According to the results, we observe that our *MICA* model performs better than current method HRNN both on single-track and multi-track, which means *MICA* has a significant improvement on Multi-track Music Generation task. Specially, we find that multi-track has higher score than single track score, which indicates that multi-track music sounds better than single-track music and confirms the importance of the arrangement. Meanwhile, we also observe that the drum track has the worst performance compared to other single-track, which is because the drum track only plays an accessorial role in a piece of multi-track music. Furthermore, our MLP Cell-based *MICA* model performs better than Attention Cell-based MICA model, and it seems that our MLP Cell mechanism can better utilize the information from other tracks.

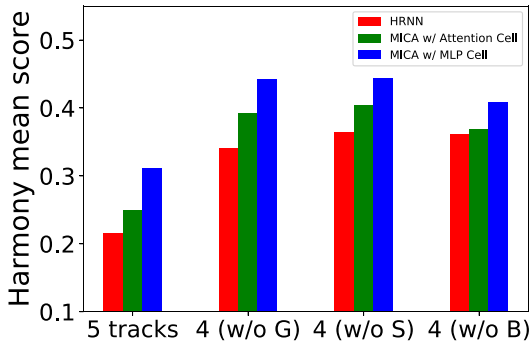


Fig. 10. The harmony analysis of arrangement (**G**: Guitar, **S**: String, and **B**: Bass).

6.4.3 Harmony Analysis. Besides human study on Multi-track Music Generation, we further evaluate whether melodies between different tracks are harmonious. Here we consider that two tracks are harmonious if they have similar chord progression [26]. Thus, we use chord similarity to represent harmony among multi-tracks. Formally, we define Harmony Score as:

$$Harmony\ Score = \sum_{p=1}^{P} \delta\left(\bigcap_{k=1}^{K} C_p^k\right),$$

$$\delta(a) = \begin{cases} 1, & if\ a \neq \varnothing \\ 0, & if\ a = \varnothing \end{cases},$$

where P and K denote the number of periods and tracks of generated music respectively, and C_p^k denotes the k^{th} track p^{th} corresponding chord.

As shown in Figure 10, we evaluate the harmony of multi-track music on different models. Every model generates several music given the same chord progression, and then we calculate the harmony score of these generated music. We observe that our MLP Cell-based *MICA* model achieves the best performance, with an improvement by up to 24.4% compared to HRNN. It indicates that *MICA* model improves the harmony of multi-track music through utilizing the useful information of other tasks. Besides, three models all achieve better score on four tracks task without guitar or string track, means guitar and string tracks are harder to learn. Specially, we find that harmony of music with fewer tracks is higher than music with more tracks. For this result, we specular that more tracks music have higher harmony requirements.

6.4.4 Arrangement Analysis. To observe our model performs at multi-track music arrangement, we generate each track while fixing melody track as source melody sequence. Here we validate the performance based on four metrics as follows:

—**Note accuracy.** Note accuracy is the fraction of matched generated notes and source notes over the total amount of source notes in a piece of music, that is:

$$Notes\ Accuracy = \sum_{i=1}^{N} e(y_i, \widetilde{y_i})/N,$$

where $y_i, \widetilde{y_i}$ denote the i^{th} source note and generated note, respectively. As sequence generation, the higher accuracy of note prediction means better generation performance. To this end, we utilize note accuracy to evaluate multi-track generation.

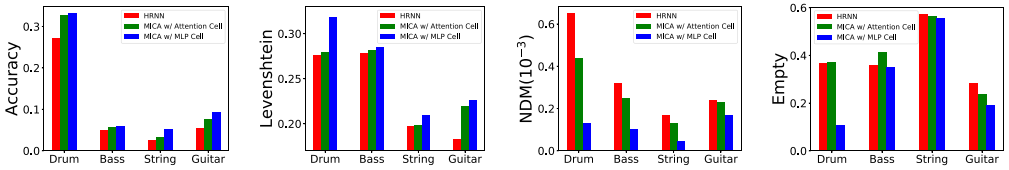


Fig. 11. The analysis of arrangement from four parts.

—**Levenshtein similarity.** We define the music generation as a sequence generation problem. It is necessary to evaluate the sequence similarity between real music and generated music. Compared with other sequence similar metrics, such as Mahalanobis distance [41] and Euclidean distance [1], Levenshtein distance [34] could handle unequal length sequences. Levenshtein distance is calculated by counting the minimum number of single-character edits (insertions, deletions or substitutions) required to change one sequence into the other. Therefore, we calculate the Levenshtein similarity by Levenshtein distance, and evaluate the similarity of generated musical notes sequences and original ones as follows:

$$Levenshtein\ similarity = 1 - \frac{Levenshtein\ distance}{N + \widetilde{N}},$$

where N, $\widetilde{N}$ denote the length of generated musical notes sequences and original musical notes sequences, respectively. The greater value of Levenshtein distance, the higher the coincidence of sequence.

—**Notes distribution MSE.** Notes distribution MSE is used to analyze the notes distribution between generated and original ones, which can be formulated by:

$$Notes\ distribution\ MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\frac{y_i}{N} - \frac{\widetilde{y_i}}{\widetilde{N}}\right)^2}{MN},$$

where M, N denote the number of pieces of music and note categories, respectively. Actually, every instrument has its own characteristic in terms of note range. For example, bass usually uses low notes and drum has fixed notes.

—**Empty.** It is bad for generation results to be empty while a real result has notes. We use it to evaluate generation results and a lower score indicates better performance.

The performance is shown in Figure 11. According to the results, our MLP Cell-based *MICA* model achieves best performance across all metrics. Specially, from Figure 11(a), it can be concluded that the drum task has the greatest note accuracy, which confirms that drum is easier to learn than other instruments. Moreover, as shown in Figure 11(b), our MLP Cell-based *MICA* model could improve the quality of 6.9% on average compared with HRNN. Meanwhile, from Figure 11(c), we observe that our MLP Cell-based *MICA* model has the most stable effect on Notes distribution MSE, which proves our model can learn instrument characteristics better. At last, Figure 11(d) illustrates the robustness of our MLP Cell-based *MICA* model, which can maintain a high level of generation result.

6.4.5 Multi-task Performance Analysis. We also found the multi-task learning could accelerate the training procession in multi-sequence generation. Here we compared base model HRNN and two multi-task learning models, including Attention Cell and MLP Cell-based MICA. Figure 12(a)–12(d) separately represent different arrangement models' training procession in drum, bass, string, and guitar, where MLP and Attention achieve convergence faster and better, compared with model

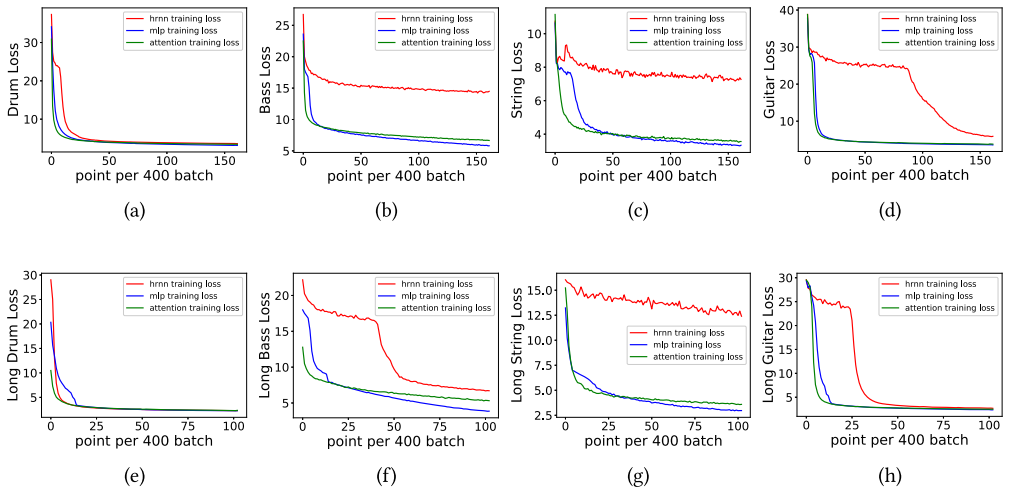


Fig. 12. Instrument loss analysis of arrangement.

without multi-task learning. Besides, Figure 12(b)–12(d) show that the model with multi-task learning could handle more complex task better. For example, in bass, string, and guitar generation tasks, which are more complexed than drum task, the multi-task learning model has a significant improvement compared with HRNN without multi-task learning. To further evaluate the robustness of our model, we experimented on longer music sequences. As showed in Figure 12(e)–12(h), generation models with multi-task learning still perform better than model without multi-task learning. Similarly, multi-task learning model has a higher improvement in complex tasks, such as bass, string, and guitar.

6.5 Musical Style Generation

In this subsection, we will analyze the musical style generation performance in our *MSMICA* model compared with other baselines. Meanwhile, we will discuss the fine-tuning method and how to control the style of generated multi-track music. Besides, to evaluate the influence of discriminators for controlling music generation, we analyze in detail the single and multiple discriminators in aspects of musical style and harmony.

6.5.1 Baseline Methods. To validate the performance of musical style generation model, we utilize *MSMICA* and several variations of model as baseline methods. Specifically, we set the comparison methods as follows:

—**MSMICA.** Proposed model, which controls musical style by both harmony and style discriminators with *MICA* as generator.

In addition to analyze the influence of different generators, we evaluate two variants of the model to validate the importance of style and harmony discriminators on music generation:

—**MSMICA w/o Style Discriminator.** Based on *MSMICA*, model controls musical style without style discriminator.
—**MSMICA w/o Harmony Discriminator.** Based on *MSMICA*, model controls musical style without harmony discriminator.

Table 7. Human Evaluation of Musical Style

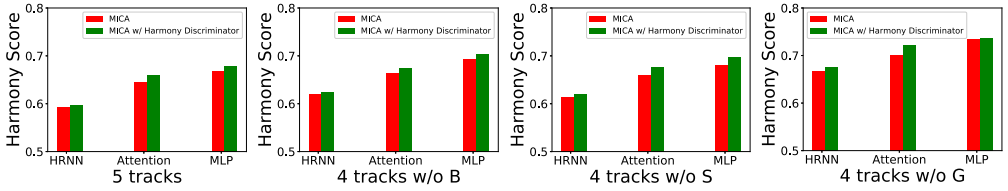
Methods	Overall	Rhythm	Integrity	Harmony	Style
HRNN[13]	3.2500	2.9875	3.0875	2.8000	2.0625
MICA[63]	3.4625	3.0750	2.8000	2.9125	2.2500
MSMICA	**3.7125**	3.0750	3.1250	**3.2625**	**3.3500**
MSMICA (w/o style)	3.5625	**3.1750**	**3.2000**	3.0125	2.8125
MSMICA (w/o harmony)	3.6125	3.1000	3.0625	2.9750	3.1500

6.5.2 Musical Style Analysis. We evaluate the performance by inviting human with music knowledge to validate the style of generated music. Similar in previous work [63], we utilize several metrics, including *Overall*, *Rhythm*, *Integrity*, and *Harmony*. Besides, we proposed a new *Style* to evaluate the performance of musical style:

—**Style.** We choose music data with a musical stye s as the training data for above models. After getting generated music, we let human with music knowledge to choose music best related to the style s with a rank score from one-to-five, which the score of five means corresponding music is highest related to style s.

Table 7 shows the results on above metrics with different models. HRNN and MICA are our baselines, while MSMICA and two variants are proposed model with fine-tuning method. From the table, we found the MSMICA performances better than HRNN and MICA in Overall, proving that the fine-tuning method could improve the generation performance. Besides, for the score of *style*, MSMICA has a best score and MSMICA(w/o style) has lowest score in variants of MSMICA. This evaluates that the style discriminator could guide the generator to output specific music with corresponding style. Specially, the model without harmony discriminator achieves the worst result on Integrity, Harmony and Style metrics. This supports our argument that when we try to fine-tuning the generator to control the musical style, fine-tuning will destroy the harmony among the multi-track music.

6.5.3 Harmony Analysis. As we proposed above, we specular the procession of the fine-tuning of generator will make generated music approaches constantly to the specific music, but also change the characteristics learned of music, such as harmony. In order to learn the style and harmony simultaneously, we utilize the harmony discriminator to ensure the harmony of generated music after fine-tuning with multi-style discriminator. To validate this idea, we compared the music harmony both before and after fine-tuning with harmony discriminator in multi-track music. MICA and MICA with harmony discriminator are used to experiment on different tracks music. Additionally, HRNN, Attention Cell, and MLP Cell are used to validate multi-task learning whether influence the fine-tuning performance. From Figure 13, we found the harmony score of generated music improved after fine-tuning, which means the discriminator could guide the generator achieve a more harmonious state. Besides, for different track music tasks, Attention Cell and MLP Cell performance are always better than HRNN, which shows the robustness of multi-task learning and significant importance for multiple sequence generation. We showed the Harmony Score in Table 8. From the experiment results, fine-tuning of discriminator improved the harmony score of generated music, proving the effectiveness of adversarial training. We also found the MLP cell has the best harmony score on every task, which also validates the effectiveness of this information exchange strategy compared with HRNN and Attention Cell. Specially, multi-task learning model has higher improvement, where Attention Cell achieves 2.90% score and MLP Cell achieves 2.45%.



(a) (b) (c) (d)

Fig. 13. Harmony analysis in fine tuning (**G**: Guitar, **S**: String, **B**: Bass).

Table 8. Harmony Score of Arrangement After Fine Tuning

Methods	MICA				MICA w/ Harmony Discriminator				Improvement			
	5	w/o B	w/o S	w/o G	5	w/o B	w/o S	w/o G	5	w/o B	w/o S	w/o G
HRNN	1.779	1.859	1.839	2.004	1.794	1.876	1.858	2.03	0.84%	0.91%	1.03%	1.30%
Attention	1.934	1.990	1.982	2.104	1.979	2.023	2.027	2.165	4.60%	1.66%	2.27%	2.90%
MLP	**2.003**	**2.081**	**2.042**	**2.201**	**2.036**	**2.109**	**2.092**	**2.211**	1.64%	1.35%	2.45%	0.45%

Table 9. Parameter Experiment of Multiple Discriminators

Methods	HRNN				MICA w/ Attention Cell				MICA w/ MLP Cell			
	5	w/o B	w/o S	w/o G	5	w/o B	w/o S	w/o G	5	w/o B	w/o S	w/o G
$\lambda = 1.0$	1.759	1.758	1.779	1.951	1.902	1.942	1.923	1.962	1.780	1.830	1.810	1.980
$\lambda = 0.8$	1.752	1.826	1.835	1.999	1.939	2.019	1.985	2.084	1.810	1.870	1.900	2.020
$\lambda = 0.5$	1.761	1.786	1.828	1.972	2.006	2.083	2.061	2.118	1.870	1.940	1.940	2.030
$\lambda = 0.2$	**1.982**	**2.068**	**2.045**	**2.097**	**2.032**	**2.103**	**2.089**	**2.147**	**1.984**	**2.093**	**2.094**	**2.107**
$\lambda = 0.0$	1.782	1.872	1.847	2.002	1.993	2.086	2.047	2.106	1.830	1.910	1.920	2.000

6.5.4 Multiple Discriminator Analysis. As we mentioned in Section 5.5, we define the reward function for policy gradient as a linear combination of probability of classifying generated music y weighted by parameter λ:

$$R(y|\cdot) = \lambda C_s(c = style|y) + (1 - \lambda) C_h(c = harmony|y). \tag{25}$$

However, how to define a suitable value of λ to balance performance of musical style and harmony is a problem. In this section, we conduct experiments on different parameters of λ. Besides, we also analyze influence of the policy reward on different generators.

We set five different parameters of λ, including 1.0, 0.8, 0.5, 0.2, 0.0. From Table 9, we compared harmony score of different generators, including HRNN, MICA w/ Attention Cell, and MICA w/ MLP Cell in different parameters of λ. When λ set as 1.0, all models only use style discriminator to guide music generation. We found all models achieve worst harmony score compared with other values of λ. This indicates harmony discriminator has an important influence on music harmony. Besides, the model only with harmony discriminator does not get the best performance, while the model with multiple discriminators gets higher harmony score. This might be due to the model could capture more training data with different musical styles, which have commonness on harmony. Specially, the experiment results show that the generator with multi-task learning method has better performance, proving that the multi-task learning method could help share reward among different tasks when computing policy rewards.

7 DISCUSSION

From the experimental results of the music generation, we can observe that CRMCG, MICA, and MSMICA outperform the baselines on multi-track music generation and musical style generation problems. Harmony and musical style analysis demonstrated that our model could generate harmonious multi-track music with different musical styles.

Nevertheless, there is still some room for improvement. First, our data currently suffers from the limit of musical style numbers, and we will try to collect more music data in the future. Second, We can design a model to choose the instruments automatically. Third, there may be some other problems in music generation, like longer music generation and emotions in music, that will be considered in future work.

8 CONCLUSIONS

In this article, we proposed an end-to-end melody and arrangement generation framework to generate a melody with several instruments accompanying simultaneously. To be specific, we first developed a novel *CRMCG*, which utilized chord progression to guide the melody and rhythm generation. Then, in order to enrich the music content and improve the quality of generation, we proposed *MICA* for multi-track music arrangement, which used other task states at each step in the decoder layer to improve the whole generation performance and ensure the harmony of multi-track music. One step further, we proposed to control the musical style and meet the diversified demand in the real world, we designed *MSMICA* to learn the musical style with adversarial training. Therefore, we can not only maintain the harmony of the generated music but also control the musical style for better utilization. By massive experiments provided, our system showed better performance compared with other models in human evaluation and we have completed the Turing test and achieved good results. Since there has been the relatively little work on pop music generation, we hope our work could inspire the relative research studies and lead many future works.

REFERENCES

[1] Howard Anton and Chris Rorres. 2013. *Elementary Linear Algebra, Binder Ready Version: Applications Version*. John Wiley & Sons.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR'15)*.

[3] Pierre Baldi, Yves Chauvin, Tim Hunkapiller, and Marcella A McClure. 1994. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences* 91, 3 (1994), 1059–1063.

[4] Judith O. Becker. 2004. *Deep Listeners: Music, Emotion, and Trancing*. Vol. 2. Indiana University Press.

[5] Geoffray Bonnin and Dietmar Jannach. 2015. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys* 47, 2 (2015), 26.

[6] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics*. Springer, 177–186.

[7] Mason Bretan, Gil Weinberg, and Larry Heck. 2016. A unit selection methodology for music generation using deep neural networks. *arXiv preprint arXiv:1612.03789* (2016).

[8] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. 2017. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620* (2017).

[9] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. 2018. MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. In *19th International Society for Music Information Retrieval Conference (ISMIR'18)*.

[10] Pietro Casella and Ana Paiva. 2001. Magenta: An architecture for real time automatic composition of background music. In *Proceedings of theInternational Workshop on Intelligent Virtual Agents*. Springer, 224–232.

[11] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–Decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.
[12] Parag Chordia, Avinash Sastry, and Sertan Şentürk. 2011. Predictive tabla modelling using variable-length Markov and hidden Markov models. *Journal of New Music Research* 40, 2 (2011), 105–118.
[13] Hang Chu, Raquel Urtasun, and Sanja Fidler. 2016. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477* (2016).
[14] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 160–167.
[15] Darrell Conklin. 2003. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*. 30–35.
[16] Shuqi Dai, Zheng Zhang, and Gus Xia. 2018. Music style transfer issues: A position paper. *arXiv preprint arXiv:1803.06841* (2018).
[17] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. 1723–1732.
[18] 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*.
[19] Sean R. Eddy. 1996. Hidden Markov models. *Current Opinion in Structural Biology* 6, 3 (1996), 361–365.
[20] Franco Fabbri. 2007. Browsing music spaces: Categories and the musical mind. In *Proceedings of the International Association for the Study of Popular Music*.
[21] Yanjie Fu, Hui Xiong, Yong Ge, Zijun Yao, Yu Zheng, and Zhi-Hua Zhou. 2014. Exploiting geographic dependencies for real estate appraisal: A mutual perspective of ranking and clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1047–1056.
[22] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.
[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems*. 2672–2680.
[24] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*. IEEE, 6645–6649.
[25] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. 2017. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. JMLR. org, 1362–1371.
[26] Christopher Harte, Mark Sandler, and Martin Gasser. 2006. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*. ACM, 21–26.
[27] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1923–1933.
[28] Nanzhu Jiang, Peter Grosche, Verena Konz, and Meinard Müller. 2011. Analyzing chroma feature types for automated chord recognition. In *Proceedings of the 42nd Audio Engineering Society Conference*. Audio Engineering Society.
[29] Daniel Johnson. 2015. Composing music with recurrent neural networks.
[30] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 655–665.
[31] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7482–7491.
[32] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[33] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* (2017).
[34] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Doklady Akademii Nauk SSSR* 163, 4 (1966), 707–710.
[35] Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. 2018. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *Proceedings of the 2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 783–791.
[36] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2873–2879.

[37] Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, Guoping Hu. 2019. EKT: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[38] Qi Liu, Guifeng Wang, Hongke Zhao, Chuanren Liu, Tong Xu, and Enhong Chen. 2017. Enhancing campaign design in crowdfunding: A product supply optimization perspective. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 695–702.

[39] Mingsheng Long and Jianmin Wang. 2015. Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117* (2015).

[40] Chien-Yu Lu, Min-Xin Xue, Chia-Che Chang, Che-Rung Lee, and Li Su. 2019. Play as you like: Timbre-enhanced multi-modal music style transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1061–1068.

[41] Prasanta Chandra Mahalanobis. 1936. On the generalized distance in statistics. In *Proceedings of the National Institute of Science of India*.

[42] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3994–4003.

[43] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[44] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

[45] François Pachet, Sony CSL Paris, Alexandre Papadopoulos, and Pierre Roy. 2017. Sampling variations of sequences for structured music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'17)*. 167–173.

[46] François Pachet and Pierre Roy. 2011. Markov constraints: Steerable generation of Markov sequences. *Constraints* 16, 2 (2011), 148–172.

[47] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142* (2017).

[48] Romain Sabathé, Eduardo Coutinho, and Björn Schuller. 2017. Deep recurrent music writer: Memory-enhanced variational autoencoder-based musical score composition and an objective measure. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'17)*. IEEE, 3467–3474.

[49] Paul Schmeling. 2011. *Berklee Music Theory*. Berklee Press.

[50] Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology and Electronic Engineering* 19, 1 (2018), 10–26.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 3104–3112.

[52] Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. 2000. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. IEEE, 1315–1318.

[53] Andries Van Der Merwe and Walter Schulze. 2011. Music generation with Markov models. *IEEE MultiMedia* 18, 3 (2011), 78–85.

[54] Dominique T. Vuvan and Bryn Hughes. 2019. Musical style affects the strength of harmonic expectancy. *Music & Science* 2 (2019), 2059204318816066.

[55] Yanan Wang, Qi Liu, Chuan Qin, Tong Xu, Yijun Wang, Enhong Chen, and Hui Xiong. 2018. Exploiting topic-based adversarial neural network for cross-domain keyphrase extraction. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM'18)*. IEEE, 597–606.

[56] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. 2017. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'17)*.

[57] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

[58] Kun Zhang, Guangyi Lv, Le Wu, Enhong Chen, Qi Liu, Han Wu, and Fangzhao Wu. 2018. Image-enhanced multi-level sentence representation net for natural language inference. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM'18)*. IEEE, 747–756.

[59] Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[60] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. 2016. Embedding label structures for fine-grained feature representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1114–1123.

[61] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* (2017).

[62] Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian. 2012. Mining personal context-aware preferences for mobile users. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*. IEEE, 1212–1217.

[63] Hongyuan Zhu, Qi Liu, Nicholas Jing Yuan, Chuan Qin, Jiawei Li, Kun Zhang, Guang Zhou, Furu Wei, Yuanchun Xu, and Enhong Chen. 2018. Xiaoice band: A melody and arrangement generation framework for pop music. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2837–2846.

Received April 2019; revised October 2019; accepted December 2019