XiaoIce Band: A Melody and Arrangement Generation Framework for Pop Music

Hongyuan Zhu[1,2*], Qi Liu[1†], Nicholas Jing Yuan[2†], Chuan Qin[1], Jiawei Li[2,3*], Kun Zhang[1], Guang Zhou[2], Furu Wei[2], Yuanchun Xu[2], Enhong Chen[1]
[1]University of Science and Technology of China , [2] AI and Research Microsoft
[3] Soochow University

ABSTRACT

With the development of knowledge of music composition and the recent increase in demand, an increasing number of companies and research institutes have begun to study the automatic generation of music. However, previous models have limitations when applying to song generation, which requires both the melody and arrangement. Besides, many critical factors related to the quality of a song such as chord progression and rhythm patterns are not well addressed. In particular, the problem of how to ensure the harmony of multi-track music is still underexplored. To this end, we present a focused study on pop music generation, in which we take both chord and rhythm influence of melody generation and the harmony of music arrangement into consideration. We propose an end-to-end melody and arrangement generation framework, called XiaoIce Band, which generates a melody track with several accompany tracks played by several types of instruments. Specifically, we devise a *Chord based Rhythm and Melody Cross-Generation Model (CRMCG)* to generate melody with chord progressions. Then, we propose a *Multi-Instrument Co-Arrangement Model (MICA)* using multi-task learning for multi-track music arrangement. Finally, we conduct extensive experiments on a real-world dataset, where the results demonstrate the effectiveness of XiaoIce Band.

KEYWORDS

Music generation, Melody and arrangement generation, Multi-task joint learning, Harmony evaluation

ACM Reference Format:

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 KDD'18: The 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, August 19–23, 2018, London, UK. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220105

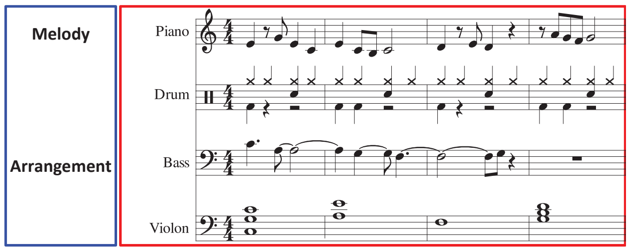


Figure 1: The example of our generated song.

1 INTRODUCTION

Music is one of the greatest invention in human history and has a vital influence on human 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. [8] 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 [5, 26] were applied in music generation. These methods could generate novel music, but require manual inspection of the features. Recently, Google Magenta[1] [3] created piano music with Deep Recurrent Neural Network [12] (DRNN) by learning MIDI (a digital score format) data. However, this method can only deal with single track music.

Indeed, generating a song for singing has more challenges, which are not well addressed in existing approaches. As shown in Figure 1, a typical song consists of melody, arrangement in addition to lyrics. Whether a song is pleasant to listen depends on several critical characteristics. Specifically,

- Chord progression generally exists in pop songs, which could guide melody procession. Thus, it is beneficial to capture chord progression as input for song generation. Besides, a

[†]Corresponding authors
[*]This work was accomplished when the first and fifth authors working as interns in Microsoft supervised by the third author
[1]https://magenta.tensorflow.org/

pop song has several fixed rhythm patterns, which make the song more structural and pause suitably. However, existing studies [17, 19] usually generate music note-by-note and without considering the rhythm pattern. On the other hand, though several works [13, 25] utilize chord for music generation, they only use single chord as a feature of input and without considering the progression of chords when generating melody.

- A complete song typically has multi-track arrangement[2] considering chord, beats and rhythm patterns, etc, with accompanying background music played with other instruments, such as drum, bass, string and guitar. Recent works [11, 25, 28] could generate melody of songs, however, they fail to take into account the multi-track arrangement.
- Different tracks and instruments have their own characteristics, while they should be in harmony with each other. A few existing works tackled the generation of multi-track music [6], but none of them considered the harmony between multiple tracks.

To this end, in this paper, we propose the XiaoIce Band [3], an end-to-end melody and arrangement generation framework for song generation. To be specific, we propose 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 introduce *Multi-Instrument Co-Arrangement Model (MICA)* for multi-track music. Here, two information-sharing strategies, Attention Cell and MLP Cell, are designed to capture other task's useful information. The former model utilizes chord progression to guide the note relationships between periods based on music knowledge. The latter shares the information among different tracks to ensure the harmony of arrangement and improve the quality of song generation. Extensive experiments on real-world dataset demonstrate our model's superiority over baselines on single-track and multi-track music generation. Specifically, our model [30] has created many pop songs and passed the Turing test in CCTV1[4]. The contributions of this paper are summarized as follows.

- We propose an end-to-end multi-track song generation system, including both the melody and arrangement.
- Based on the knowledge of music, we propose to 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.
- 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.
- By massive experiments provided, our system shows better performance compared with other models as well as human evaluations.

[2]http://mnsongwriters.org/accelsite/media/1051/Elements%20of%20a%20Song.pdf
[3]XiaoIce is a Microsoft AI product popular on various social platforms, focusing on emotional engagement and content creation [30]
[4]http://tv.cctv.com/2017/11/24/VIDEo7JWp0u0oWRmPbM4uCBt171124.shtml

Table 1: Comparing music generation models (G: Generation, Mt: Multi-track, M: Melody, Cp: Chord progression, Ar: Arrangement, Sa: Singability).

Methods	G	Mt	M	Cp	Ar	Sa
Markov music [31]	√		√			
Music unit selection [2]			√			
Magenta [3]	√		√			
Song from PI [6]	√	√	√			√
DeepBach [13]	√		√	√		
GANMidi [32]	√		√			
Sampling music sequences [25]			√	√		
XiaoIce Band (this paper)	√	√	√	√	√	√

2 RELATED WORK

The related work can be grouped into two categories, i.e., music generation and multi-task learning.

2.1 Music Generation

Music generation has been a challenging task over the last decades. A variety of approaches have been proposed. Typical data-driven statistical methods usually employed N-gram or Markov models [5, 26, 31]. Besides, a unit selection methodology for music generation was used in [2] which spliced music units with ranking methods. Moreover, a similar idea was also proposed by [25], 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. Among them, Johnson et al. [17] combined one recurrent neural network and one nonrecurrent neural network to represent the possibility of more than one note at the same time. An RNN-based Bach generation model was proposed in [13], which was capable producing four-part chorales by using a Gibbs-like sampling procedure. Contrary to models based on RNNs, Sabathe et al. [28] used VAEs [19] to learn the distribution of musical pieces. Besides, Yang and Mogren et al. [24, 32] adopted GANs [11] to generate music, which treated random noises as inputs to generate melodies from scratch. Different from single track music, Chu et al. [6] used hierarchical Recurrent Neural Network to generate both the melody as well as accompanying effects such as chords and drums.

Although extensive research has been carried out on music generation, no single study exists considering the specificity of music. 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.

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

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

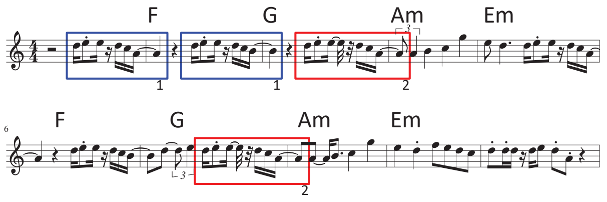


Figure 2: Melody of the song "We Don't Talk Anymore" with chord progression labeled.

others. In previous works, multi-task learning has been used successfully across all applications of machine learning, from natural language processing [7, 21] to computer vision [10, 33]. For example, Zhang et al. [34] proposed to improve generalization performance by leveraging the domain-specific information of the training data in related tasks. In the work [15], the authors pre-defined a hierarchical architecture consisting of several NLP tasks and designed a simple regularization term to allow for optimizing all model weights to improve one task's loss without exhibiting catastrophic interference in other tasks. Another work [18] in computer vision, adjusted each task's relative weight in the cost function by deriving a multi-task loss function based on maximizing the Gaussian likelihood with task-dependant uncertainty. More multi-task learning works applied in deep learning are proposed in [22, 23, 27].

3 PRELIMINARIES

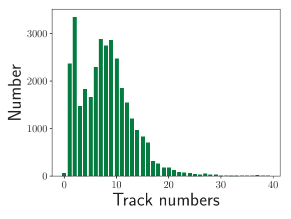
In this section, we will intuitively discuss the crucial influence of chord progression, rhythm pattern and instrument characteristic in pop song generation, based on music knowledge with related statistical analysis to further support our motivation.

3.1 Chord Progression

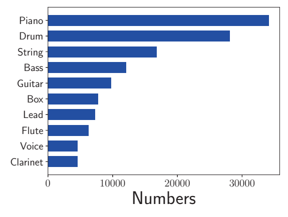
In music, chord is any harmonic set of pitches consisting of two or more notes that are heard as if sounding simultaneously. An ordered series of chords is called a chord progression. Chord progressions are frequently used in songs and a song often sounds harmonious and melodic if it follows certain 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 song. 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", "Glory days" and so on.

3.2 Rhythm Pattern

Apart from the chords we mentioned above, rhythm pattern is another characteristic of pop songs. Rhythm pattern could be defined as the notes duration 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 song is a more structural task. However, previous works didn't consider the structure of the song.



(a) Tracks distribution



(b) Top 10 instruments

Figure 3: Tracks and instruments analysis of pop song.

3.3 Instrument Characteristic

The last characteristic of the song 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, 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 show 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.

4 PROBLEM STATEMENT AND MODEL STRUCTURE

In this section, we will first present the music generation problem with a formulated problem definition and then introduce the structures and technical details of *Chord based Rhythm and Melody Cross-Generation Model (CRMCG)* for single track music, as well as *Multi-Instrument Co-Arrangement Model (MICA)* for multi-track music. For better illustration, Table 2 lists some mathematical notations used in this paper.

4.1 Problem Statement

Since each pop music has a specific chord progression, we consider the scenario of generating the pop music on the condition of given chord progression. Thus, the input of music generation task is the given chord progression $C = \{c_1, c_2, ..., c_{l_c}\}$. Note that c_i is the one-hot representation of the chord and l_c is the length of the sequence. We target at generating the suitable rhythm $R_i = \{r_{i1}, r_{i2}, ..., r_{il_r}\}$ and melody $M_i = \{m_{i1}, m_{i2}, ..., m_{il_m}\}$. To this end, we propose *CRMCG* for single track music, as well as *MICA* for multi-track music to tackle this issue.

Figure 4 shows the overall framework of XiaoIce Band, which can be divided into four parts: 1) Data processing part; 2) *CRMCG* part for melody generation (single track); 3) *MICA* part for arrangement generation (multi-track); 4) The display part. We will introduce the second and third part in detail. Data processing part will be detailed in experiment section.

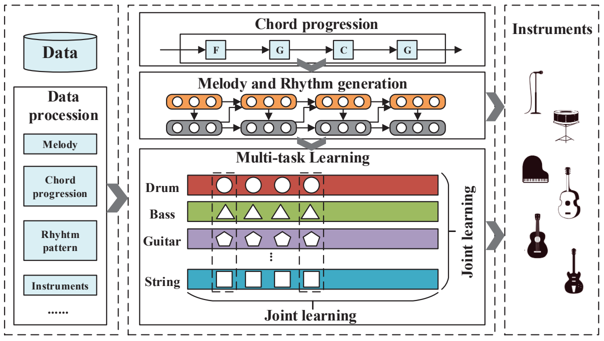


Figure 4: The flowchart overview of XiaoIce Band.

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
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
$\bar{h}^m_{i,j}, \bar{h}^r_{i,j}, \bar{h}^c_{i,j}$	the j-th hidden state in i-th period of melody/rhythm/chord progression sequence respectively
$h^i_{t,k}$	the i-th task hidden state in period t at step k

4.2 Chord based Rhythm and Melody Cross-Generation Model

Melody is made up of a series of notes and the corresponding duration. It's a fundamental part of pop music. However, it's still challenging to generate melody in harmony. Besides, note-level generation methods have more randomness on the pause, which causes the music hard to sing. Thus, we propose *CRMCG* to solve the problem and generate a suitable rhythm for singing. Figure 5 gives the architecture of *CRMCG*.

Given a chord progression $C = \{c_1, c_2, ..., c_N\}$, we aim at generating the corresponding periods $\{p_1, p_2, ..., p_N\}$. The generated rhythm R_i and melody M_i in period p_i are closely related to the chord c_i. We utilize encoder-decoder framework as our basic framework since it is flexible to use different neural networks, such as Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), to process sequence effectively.

In order to better understand the chord progression and model the interaction and relation of these chords, we utilize Gated Recurrent Units (GRU) [4] to process the low-dimension representation of chords. They can be formulated as follows:

$$\begin{aligned} \bar{C} &= E_c \boldsymbol{C}, \quad E_c \in \mathbb{R}^{V_c * d}, \\ \bar{h}^c_{i,0} &= \text{GRU}(\bar{c}_i), \quad i = 1, 2, ..., l_c, \end{aligned} \tag{1}$$

here, E_c is the embedding matrix for chord and hidden states $\bar{c}_i$ encode each chord and sequence context around it. Then we can use 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.

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

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

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

$$\begin{aligned} \bar{h}^m_{t-1,i} &= \text{GRU}(\{\bar{m}_{t-1,i}\}), \quad i = 1, 2, ...l_m, \\ \bar{h}^r_{t-1,i} &= \text{GRU}(\{\bar{r}_{t-1,i}\}), \quad i = 1, 2, ..., l_r. \end{aligned} \tag{3}$$

Then we separately concatenate the last hidden states of rhythm encoder and melody encoder, 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 of the current period. They can be formalized as follows:

$$\begin{aligned} s^r_0 &= g(W[\bar{h}^m_{t-1,l_m}, \bar{h}^r_{t-1,l_r}] + b), \quad W \in \mathbb{R}^{b*b}, \\ s^r_i &= \text{GRU}(y^r_{i-1}, s^r_{i-1}), \quad i > 0, \\ y^r_i &= softmax(s^r_i), \end{aligned} \tag{4}$$

here g is the Relu activation function and s^r_i is the hidden state of GRU for generating the i-th beat in t-th period. Thus we get the rhythm for the t-th period and turn to generate the melody.

Melody Generation. After generating the current rhythm, we can utilize this information to generate melody. Like rhythm generation, we first concat previous melody M_{t-1}, currently generated rhythm R_t and corresponding chords c_t. Second, we make a linear transformation in the concatenation, which can be formulated as follows:

$$s^m_0 = g(W[\bar{h}^m_{t-1,l_m}, \bar{h}^r_{t,l_r}, \bar{h}^c_t] + b), \quad W \in \mathbb{R}^{b*b}. \tag{5}$$

Then we get the initial hidden state of melody decoder. Finally, we utilize GRU to process the result and generate the current melody for the whole generation as follows:

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

Loss Function. Since the generating process can be divided into two parts, we design two loss functions for each part. The loss functions are both softmax cross-entropy functions. Based on the characteristic of the model, we can update the parameters alternately

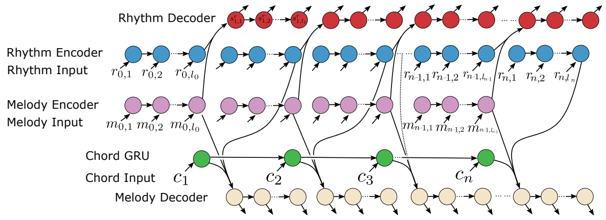


Figure 5: CRMCG.

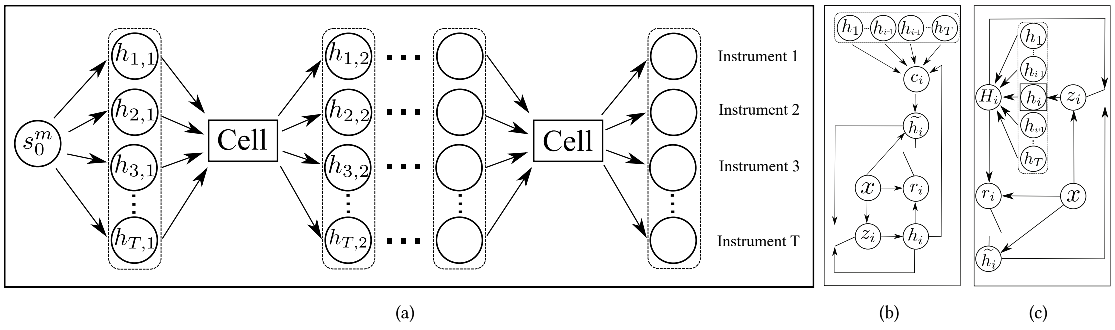


Figure 6: (a): MICA (b): Attention Cell (c): MLP Cell.

by parameter correlation. In rhythm section, we only update parameters related with rhythm loss L^r. Differently, we update all the parameters by melody loss L^m in melody section.

4.3 Multi-task arrangement Model

4.3.1 Multi-Instrument Co-Arrangement Model. In real-world applications, music contains more than one track, such as drum, bass, string and guitar. To this end, we formulate a *One-to-Many Sequences Generation (OMSG)* task. Different from conventional multiple sequences learning, the generated sequences in *OSMG* 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 proposed by [6], did not consider the correlation between tracks. Therefore, they could achieve good performance in single track generation, but failed in multi-track generation. Encouraged by this evidence, we aim to model the information flow between different tracks during music generation and propose the *Multi-Instrument Co-Arrangement Model (MICA)* based on *CRMCG*.

Given a melody, we focus on generating more tracks to accompany melody with different instruments. As shown in Figure 6(a), the hidden state of decoder contains sequence information. Hence, it naturally introduces the hidden state of other tracks when generating note for one of the tracks, but how to integrate them effectively is still a challenge. To this end, we designed two cooperate cells between the hidden layer of decoder to tackle this issue. The details of these two cells are in the following parts.

4.3.2 Attention Cell. Motivated by attention mechanism, which can help the model focus on the most relevant part of the input, we design a creative attention cell showed in Figure 6(b) to capture the relevant part of other tasks' states for current task. The attention mechanism can be formalized as follows:

$$\begin{aligned} a^i_{t,k} &= \sum_{j=1}^{T} \alpha_{t,ij} h^j_{t,k-1}, \\ e_{t,ij} &= v^T tanh(W h^i_{t,k-1} + U h^j_{t,k-1}), \quad W, U \in \mathbb{R}^{b*b}, \\ \alpha_{t,ij} &= \frac{exp(e_{t,ij})}{\sum_{s=1}^{T} exp(e_{t,is})}, \end{aligned} \tag{7}$$

note that, $a^i_{t,k}$ represents the cooperate vector for task i at step k in the period t, and $h^j_{t,k-1}$ represents the hidden state of j-th task at step $k-1$ in the period t. After getting the cooperation vector, we modify the cell of GRU to allow the current track generation take full account of the impacts of other tracks' information. The modifications are as follows:

$$\begin{aligned} r^i_{t,k} &= \sigma(W^i_r x^i_{t,k} + U^i_r h^i_{t,k-1} + A^i_r a^i_{t,k} + b^i_r), \\ z^i_{t,k} &= \sigma(W^i_z x^i_{t,k} + U^i_z h^i_{t,k-1} + A^i_z a^i_{t,k} + b^i_z), \\ \widetilde{h^i_{t,k}} &= \sigma(W^i x^i_{t,k} + U^i \left[r^i_{t,k} \cdot h^i_{t,k-1}\right] + A^i a^i_{t,k} + b^i), \\ h^i_{t,k} &= (1 - z^i_{t,k}) \cdot h^i_{t,k-1} + z^i_{t,k} \cdot \widetilde{h^i_{t,k}}, \end{aligned} \tag{8}$$

by combining attention mechanism and GRU cell, our model can generate every track for one instrument with the consideration of other instruments.

4.3.3 MLP Cell. Different from the above cell for sharing task information through input $x^i_{t,k}$, we consider the individual hidden state of each instrument and integrate them by their importance for the whole music, which is achieved by gate units. Therefore, our model can choose the most relevant parts of each instrument's information to improve the overall performance. Figure 6(c) shows the structure of this cell, which can be formalized as follows:

$$\begin{aligned} r^i_{t,k} &= \sigma(W^i_r x^i_{t,k} + U^i_r H^i_{t,k-1} + b^i_r), \\ z^i_{t,k} &= \sigma(W^i_z x^i_{t,k} + U^i_z H^i_{t,k-1} + b^i_z), \\ \widetilde{h^i_{t,k}} &= \sigma(W^i_h x^i_{t,k} + U^i_h \left[r^i_{t,k} \cdot H^i_{t,k-1}\right]), \\ h^i_{t,k} &= (1 - z^i_{t,k}) \cdot H^i_{t,k-1} + z^i_{t,k} \cdot \widetilde{h^i_{t,k}}, \\ H^i_{t,k-1} &= \sigma(W^i \left[h^1_{t,k-1}, ..., h^N_{t,k-1}\right] + b^i), \end{aligned} \tag{9}$$

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

4.3.4 Loss Function. Motivated by [9], we optimize the summation of several conditional probability terms conditioned on representation generated from the same encoder.

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

where $\theta = \left\{\theta_{src}, \theta_{trg_{T_k}}, T_k = 1, 2, ..., T_m\right\}$, 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 p-th sequence pair.

4.3.5 Generation. In generation part, we arrange for melody generated by *CRMCG*. We will discuss this part in details. With the help of *CRMCG*, we get a melody sequence $M_i = \{m_{i1}, m_{i2},...,m_{il_m}\}$, 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 s_0^m of multi-sequences decoder. They can be formulated as follows:

$$\begin{aligned} \bar{M} &= E_m M, \quad E_m \in \mathbb{R}^{V_m * d}, \\ s_0^m &= \text{GRU}(\bar{m}_{i,l_m}), \end{aligned} \tag{10}$$

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

$$\begin{aligned} s_t^i &= \text{AttentionCell}(y_{t-1}^i, s_{t-1}^i), \quad t > 0, \quad or \\ s_t^i &= \text{MLPCell}(y_{t-1}^i, s_{t-1}^i), \quad t > 0, \\ y_t^i &= softmax(s_t^i), \end{aligned} \tag{11}$$

where, s_t^i is the i-th task hidden state at step t. We utilize s_t^i to get i-th instrument sequences through $softmax$ layer. The Attention Cell and MLP Cell, we proposed above, 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 EXPERIMENTS

To investigate the effectiveness of the CRMCG and MICA, we conducted experiments with the collected dataset on two tasks: **Melody Generation** and **Arrangement Generation**.

5.1 Data Description

In this paper, we conducted our experiments on a real-world dataset, which consists of more than fifty thousand MIDI (a digital score format) files, and to avoid biases, those incomplete MIDI files, e.g., music without vocal track were removed. Finally, 14,077 MIDI files were kept in our dataset. Specifically, each MIDI file contains various categories of audio tracks, such as melody, drum, bass and string.

To guarantee the reliability of the experimental results, we made some preprocessing on the dataset as follows. Firstly, we converted all MIDI files to C major or A minor to keep all the music in the same tune. Then we set the BPM (Beats Per Minute) to 60 for all the music, which ensures that all notes correspond to an integer beat. Finally, we merged every 2 bars into a period. Some basic statistics of the pruned dataset are summarized in Table 3.

Table 3: Data Set Description.

Statistics	Values
# of popular songs	14,077
# of all tracks	164,234
# of drum tracks	18,466
# of bass tracks	16,316
# of string tracks	23,906
# of guitar tracks	28,200
# of piano tracks	18,172
# of other instruments tracks	59,174
Time of all tracks (hours)	10,128

5.2 Training Details

We randomly select 9,855 instances from the dataset as the training data, another 2,815 for tuning the parameters, and the last 1,407 as test data to validate the performance as well as more generated music. 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 [1] algorithm where batch size set is 64, and the final model is selected according to the cross entropy loss on the validation set.

5.3 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 result.

5.3.1 Baseline Methods. As the music generation task could be generally regarded as a sequence generation problem, we select two state-of-the-art models for sequence generation as baselines:

- **Magenta (RNN).** A RNN based model [3], which is designed to model polyphonic music with expressive timing and dynamics.
- **GANMidi (GAN).** A novel generative adversarial network (GAN) based model [32], 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 chords information.
- **CRMCG (w/o chord progression).** Based on CRMCG (full), the chords information is removed.
- **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.

5.3.2 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 [29], we use the metrics listed blow:

Table 4: Human evaluation of melody generation.

Methods	Rhythm	Melody	Integrity	Singability	Average
Magenta (RNN) [3]	3.1875	2.8125	2.8000	2.6000	2.8500
GANMidi (GAN) [11]	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

- **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 4. According to the results, we realize that our CRMCG model outperforms all the baselines with a significant margin on all the metrics, which demonstrate 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 enhance the quality of melody. In addition, we also find that cross-training can improve the quality of 6.9% on average, which proves effectiveness of our cross-training algorithm on melody generation.

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.

5.3.3 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 [16], and $\widetilde{y_i}$ is the i-th corresponding chord in given chord progression.

The performance is shown in Figure 7(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 7(b), 7(c), 7(d) and 7(e). From the result, 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.

5.3.4 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 8 shows the distributions of the minimum, maximum and average length of the music segments of the generated music and original ones. We realize our generated music have similar distributions on music segments lengths with original ones, which verifies that our CRMCG model can generate the appropriate rests in pieces of music.

Table 5: Human evaluation of arrangement generation.

Methods	Overall	Drum	Bass	String	Guitar
HRNN[6]	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**

5.4 Arrangement Generation

In this subsection, we conduct Multi-track Music Generation to validate the performance of our MICA model. Here we select five most important tasks in Multi-track Music Generation, i.e., melody, drum, bass, string and guitar.

5.4.1 Baseline Methods. To validate the performance of our two MICA models, a relevant model **HRNN** [6] is selected as baseline method. Specifically, we set the comparison methods as follows:

- **HRNN.** A hierarchical RNN based model [6], 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.

5.4.2 Overall Performance. Different from Melody Generation task, we ask volunteers to evaluate the quality of generated music in a holistic dimension. The performance is shown in Table 5. According to the results, we realize that our MICA model performs better than current method HRNN both on single-track and multi-track, which means MICA has 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 observe that the drum tracks 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 among the multiple tracks.

5.4.3 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 [14]. Thus,

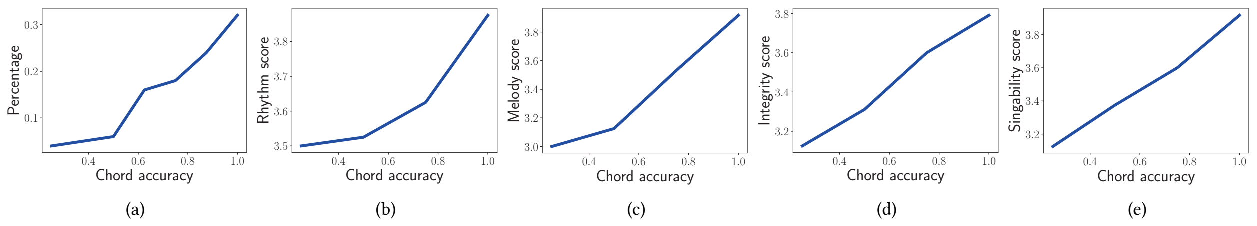


Figure 7: Chord progression analysis compared with human study.

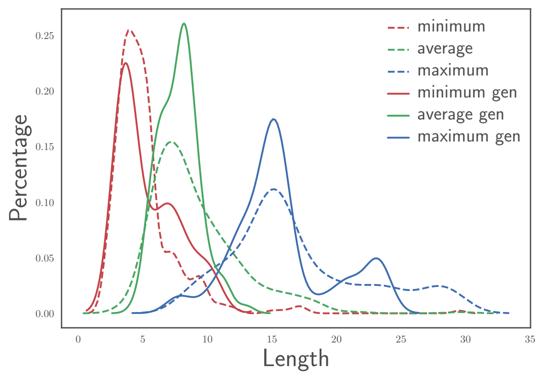


Figure 8: Rhythm distribution.

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 realize that our MLP Cell based MICA model achieves the best performance, with an improvement by up to 24.4% compared to HRNN. It indicates our MICA model improves the harmony of multi-track music through utilizing the useful information of other tasks. Specially, we find that less tracks music harmony is higher than more tracks music. For this result, we think more tracks music have higher harmony requirements.

5.4.4 Arrangement Analysis. To observer how 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.

- **Levenshtein similarity.** 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. And it is usually used to measure the difference between two sequences [20]. Here we calculate the Levenshtein similarity by Levenshtein distance, and it can evaluate the similarity of generated musical notes sequences and original ones. That is

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

where N, $\widetilde{N}$ denote the length of generated musical notes sequences and original musical notes sequences respectively.

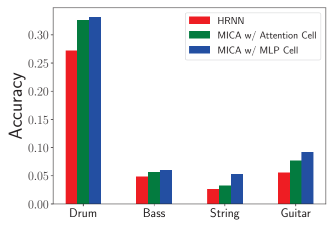
- **Notes distribution MSE.** Notes distribution MSE is used to analyze the notes distribution between generated and original ones, which can be formulated by:

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

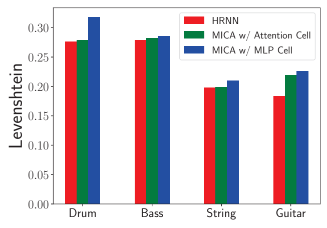
where M,N denote the number of pieces of music and note categories respectively. Actually, every instrument has its own characteristic in terms of note range. For example, bass usually uses low notes and drum has fixed notes.

- **Empty.** It's 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.

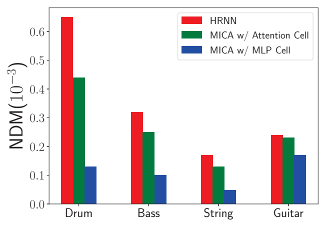
The performance is shown in Figure 9. According to the results, generally, our MLP Cell based MICA model achieves best performance across all metrics. Specially, from Figure 9(a), it can be concluded that the drum task has the greatest note accuracy, which confirms that drum is easier to learn than other instruments. And, as shown in Figure 9(b), our MLP Cell based MICA model could improve the quality of 6.9% on average compared with HRNN. Meanwhile, from Figure 9(c), we observe that our MLP Cell based MICA model has the most stable effect on Notes distribution MSE, which proves our model can do a better job in learning instrument characteristics. At last, the Figure 9(d) illustrates the robustness of our MLP Cell based MICA model, which can maintain a high level of generation result.



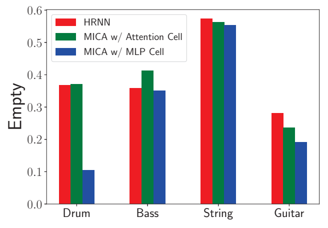
(a)



(b)



(c)



(d)

Figure 9: The analysis of arrangement from four parts.

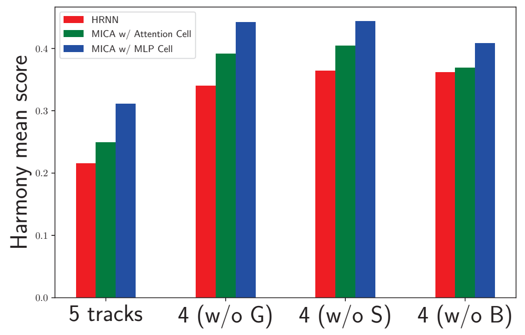


Figure 10: The harmony analysis of arrangement (G: Guitar, S: String, B: Bass).

6 CONCLUSIONS

In this paper, we proposed a melody and arrangement generation framework based on music knowledge, called XiaoIce Band, which generated a melody with several instruments accompanying simultaneously. For melody generation, we devised a *Chord based Rhythm and Melody Cross-Generation Model (CRMCG)*, which utilizes chord progression to guide the melody procession, and rhythm pattern to learn the structure of song crosswise. For arrangement generation, motivated by multi-task learning, we proposed a *Multi-Instrument Co-Arrangement Model (MICA)* for multi-track music arrangement, which used other task states at every step in the decoder layer to improve the whole generation performance and ensure the harmony of multi-track music. 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. Moreover, we generated pop music examples on the Internet, showing the application value of our model.

7 ACKNOWLEDGMENTS

This research was partially supported by grants from the National Natural Science Foundation of China (No.s 61672483 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.

REFERENCES

[1] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.
[2] 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).
[3] Pietro Casella and Ana Paiva. 2001. Magenta: An architecture for real time automatic composition of background music. In *International Workshop on Intelligent Virtual Agents*. Springer, 224–232.
[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[5] 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.
[6] 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).
[7] 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.
[8] 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*. Citeseer, 30–35.
[9] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task Learning for Multiple Language Translation.. In *ACL (1)*. 1723–1732.
[10] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.
[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.
[13] Gaëtan Hadjeres and François Pachet. 2016. DeepBach: a Steerable Model for Bach chorales generation. *arXiv preprint arXiv:1612.01010* (2016).
[14] 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.
[15] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. *arXiv preprint arXiv:1611.01587* (2016).
[16] Nanzhu Jiang, Peter Grosche, Verena Konz, and Meinard Müller. 2011. Analyzing chroma feature types for automated chord recognition. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*. Audio Engineering Society.
[17] Daniel Johnson. 2015. Composing music with recurrent neural networks. (2015).
[18] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *arXiv preprint arXiv:1705.07115* (2017).
[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[20] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.
[21] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* (2016).
[22] Mingsheng Long and Jianmin Wang. 2015. Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117* (2015).
[23] 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.
[24] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

[25] 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'2017), Suzhou, China*. 167–173.

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

[27] 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).

[28] 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 *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 3467–3474.

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

[30] Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots. *arXiv preprint arXiv:1801.01957* (2018).

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

[32] 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'2017), Suzhou, China*.

[33] 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.

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