Artificial Intelligence 286 (2020) 103303

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

PopMNet: Generating structured pop music melodies using neural networks

Jian Wu[a], Xiaoguang Liu[b], Xiaolin Hu[a,*], Jun Zhu[a]

[a] *Institute for Artificial Intelligence, Beijing National Research Center for Information Science and Technology (BNRist), the State Key Laboratory of Intelligent Technology and Systems, and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[b] *LingDongYin Technoloy Co., Ltd., Beijing, 100084, China*

A R T I C L E   I N F O

Article history:
Received 19 August 2019
Received in revised form 12 May 2020
Accepted 15 May 2020
Available online 26 May 2020

Keywords:
Melody generation
Melody structure
Artificial neural network
Generative adversarial network
LSTM

A B S T R A C T

Recently, many deep learning models have been proposed to generate symbolic melodies. However, generating pop music melodies with well organized structures remains to be challenging. In this paper, we present a melody structure-based model called PopMNet to generate structured pop music melodies. The melody structure is defined by pairwise relations, specifically, *repetition* and *sequence*, between all bars in a melody. PopMNet consists of a Convolutional Neural Network (CNN)-based Structure Generation Net (SGN) and a Recurrent Neural Network (RNN)-based Melody Generation Net (MGN). The former generates melody structures and the latter generates melodies conditioned on the structures and chord progressions. The proposed model is compared with four existing models AttentionRNN, LookbackRNN, MidiNet and Music Transformer. The results indicate that the melodies generated by our model contain much clearer structures compared to those generated by other models, as confirmed by human behavior experiments.

1. Introduction

Automatic generation of music is a long-standing theme in artificial intelligence. Early researches can date back to half a century ago [1,2]. However, the progress was slow till recent advances in deep learning. For melody generation, one of the most important parts of music generation, many deep learning models have been proposed. Some models are based on recurrent neural networks (RNN) [3–5] while some other models are based on convolutional neural networks (CNN) [6,7]. However, most models generate melodies with local temporal structures whose dependencies span only several bars. Integrating long temporal structures into automatically generated melodies remains to be challenging.

Essentially, a melody is a linear succession of musical notes and these notes exhibit complex dependencies in different time-scales. Fig. 1 shows the melody of a pop song "Simple Love". It is seen that a beat consists of several notes and a bar consists of several beats. These musical units determine the rhythm of the melody. Different parts of a melody often have certain relations, which determine the structure of the melody. For example, the 25th - 32nd bars are repetitions of the 17th - 24th bars in Fig. 1. Existing melody generation methods are insufficient for modeling such long temporal structures.

We propose to represent the melody structure with pairwise relations between all bars in a melody and construct a model for these relations. Specifically, we consider two important relations — *repetition* and *sequence*, which play critical roles in the formation of pop music melody structures [8,9]. A repetition indicates repetition of a segment (e.g., the 3rd

[*] Corresponding author.
E-mail address: xlhu@tsinghua.edu.cn (X. Hu).

https://doi.org/10.1016/j.artint.2020.103303
0004-3702/© 2020 Elsevier B.V. All rights reserved.

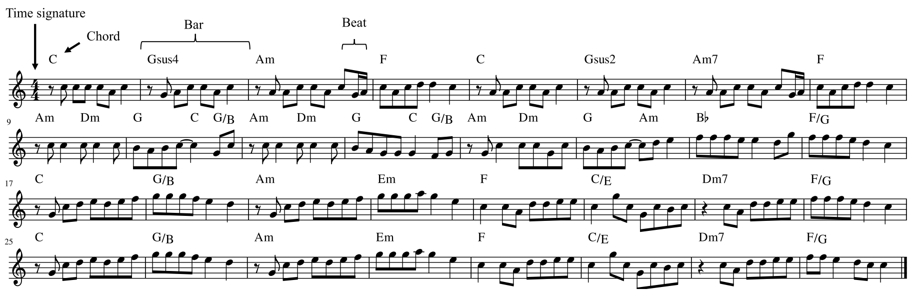

Time signature
Chord
Bar
Beat
C
Gsus4
Am
F
C
Gsus2
Am7
F
Am Dm G C G/B Am Dm G C G/B Am Dm G Am B♭ F/G
C G/B Am Em F C/E Dm7 F/G
C G/B Am Em F C/E Dm7 F/G

Fig. 1. A piece of melody of "Simple Love", which is a pop song by Chinese singer Jay Chou, released on 14 September 2001.

bar and the 6th bar in Fig. 1) while a sequence indicates the restatement of a segment at a different pitch (e.g., the 1st bar and the 2nd bar in Fig. 1). Two of the most typical sequences are rhythmic sequence and tonal sequence. A rhythmic sequence is the repetition of a rhythm with free use of pitches. A tonal sequence requires the subsequent segment to be diatonic transpositions of the first segment (e.g., the 16th bar and the 18th bar). We use a directed acyclic graph to represent relations between bars in which nodes denote bars and edges between nodes denote relations (repetition and sequence). Unfortunately, an analysis on our dataset shows that only 3.35% of pairs have the tonal sequence relation — too few to learn an accurate model. As a result, we decided to treat tonal sequence pairs as rhythmic sequence pairs in this work (note that a tonal sequence is also a rhythmic sequence).

We propose a melody structure-based melody generation model, called *PopMNet*, where the structure is characterized by repetition and sequence between pairs of bars. The model consists of two modules and the melodies are generated in a two-stage manner. First, structures are generated by the Structure Generation Net (SGN), which is a convolutional generative adversarial network (GAN) (see Section 3 for details). Second, melodies are generated conditioned on the structure by the Melody Generation Net (MGN), which consists of three RNNs (see Section 4 for details). Experiments show that melodies generated by our PopMNet have better structures compared to many existing models.

2. Related works

2.1. Melody generation

Melody generation can be considered as a stochastic process from a mathematical perspective. Over half a century ago, a Markov chain of order m was employed to sample the next event given previous $m-1$ events in a melody [1].

Melody can be also regarded as a language. From this perspective, many grammar-based approaches and rule-based approaches are proposed. Based on Chomsky grammars of type 0 to 3, the Generative Grammar achieved automatic generation of music [10,11]. A rule-based system used a series of rules summarized from the music of Alice Tegner (1864-1943) to generate melodies [2].

Evolutionary algorithms have also been used in music generation. GenJam is a genetic algorithm-based model that produces jazz solos over a given chord progression [12]. Its fitness function requires interaction with human, greatly limiting its efficiency in evaluating. Another kind of fitness function adopts explicit rules to evaluate generated compositions. For instance, some music theory-based evaluation rules were used to generate traditional musical harmony using genetic algorithms [13].

With the development of deep learning, many RNN-based methods are proposed for melody generation. Long Short Term Memory (LSTM) is a widely used model in this field. Over 10 years ago, it was used to learn a form of blues music [14]. Two more advanced versions were proposed recently [4] — the Lookback RNN, where a lookback feature is employed to model the repetition in melodies, and the Attention RNN, where an attention mechanism is added to LSTM. Deep Artificial Composer (DAC) divides pitches and durations of notes into two sequences and employs two RNN to learn them jointly [15]. Song From Pi employs a hierarchical RNN that uses multiple recurrent layers to generate melodies, drums and chords [16]. A framework called XiaoIce Band is proposed to generate rhythm and melody separately [5]. These RNN-based methods are mainly based on modeling the conditional distribution of notes given previous notes. In these methods note is the smallest unit in melody. Distributions of larger melody segments such as bars can also be modeled explicitly. A typical model is the hierarchical variational auto-encoder named MusicVAE [3]. RNN-based and CNN-based GANs are also proposed to learn the distribution of melody clips [17,6,7]. Music Transformer uses a self-attention mechanism to generate music with long-term structure [18]. To generate melodies with well-organized structures, StructureNet imposed structural restrictions on generated melodies [19]. However, such post-processing restrictions usually conflict with the generating procedure and require tedious parameter tuning to make the algorithm converge [20]. Some of the deep learning models are either open-sourced or have generated samples released [4,6,3,18], which facilitate the development of new models because the performances of the models can be compared easily.

Melody structure is the foundation of a melody. Many rule-based methods use structure-related constraints to generate melodies. A composition system represents melody structures with grammar trees and generates melodies with several structure templates [2]. Many types of music such as pop music have high-level units such as phrases and periods — a phrase consists of several bars and a period consists of several phrases. An emotional harmony generation system generates chord progressions that have predefined four-period structures, i.e., ABAB, ABAA, and ABCA where A, B, and C denote different periods [21]. A music composition system represents melody structures in a hierarchical way, which segments melodies into periods and segments periods into phrases [22]. MorpheuS represents melody structures with recurring patterns such as themes and motives from a template piece and fixes these structural elements in a new composition [23]. In these approaches, the structures are predefined. A generative adversarial network focusing on self-repetition is proposed to generate compositions with long term repetition structures [24]. It represents self-repetition with a self-similarity matrix and employs a discriminator to help the generator generate structured melodies by distinguishing self repetitions of generated melodies and real melodies. In this work, we use a generative model to generate structures directly to guide the generation of structured melodies.

2.2. Sequence learning

With the development of deep learning, many neural network-based models are used in the sequence learning domain. An RNN-based language model is proposed for speech recognition [25]. An encoder-decoder framework is proposed to map an input sequence to a target sequence for machine translation [26]. A transfer learning model DATNet is proposed to address representation difference and resource data imbalance in sequence labeling [27]. A task-aware neural language model termed LM-LSTM-CRF is proposed to extract character-level embedding under a multi-task framework [28].

2.3. Graph generation

Graph generation has applications in many domains, such as discovering new chemical structures [29] and modeling social interactions [30]. Traditional generative models for graph include random graphs [31,32], stochastic block models [33], etc. Recently, deep learning has been used to generate graphs. GraphVAE employs CNNs to encode, reconstruct and generate graphs in an end-to-end way [34]. Since it represents graphs as adjacency matrices, the number of nodes in generated graphs is limited by the matrix size. GraphRNN models graphs as sequences via breadth-first search and generates the sequence of node and edge with a deep auto-regressive model [35]. Similar to GraphVAE, we represent melody structures as adjacency matrices and utilize GAN to generate adjacency matrices.

2.4. Hierarchical generation

The idea of hierarchical or two-stage generation has been explored in generating other forms of data. Li et al. propose an LSTM model that hierarchically builds embedding of a paragraph from the embedding of sentences and words to reconstruct the original paragraph [36]. Jacob et al. model the video forecasting problem in two-stages: first generates the future human poses and then uses future poses as conditional information to predict the future frames [37]. To generate images with high diversity, FaceFeat-GAN generates diverse features first and then renders features to images [38]. PopMNet shares a similar idea with the above models: generates high-level features (melody structure) first and then uses high-level features as conditional information to generate melodies. Due to the different properties of data, the concrete methods in these works are quite different.

3. Melody structure generation

In this section, we introduce an algorithm to extract structures from melodies and the SGN.

3.1. Pairwise relation-based melody structure

As stated in Section 1, the repetition and rhythmic sequence between bars are employed to represent the structure of a melody. If we treat bars as nodes and relations between bars as edges, we can use a directed acyclic graph to represent melody structures. Since a bar can have the same relation with many other bars, to avoid redundancies in representation, each bar is matched with its previous nearest bar that is in the relation of repetition or rhythmic sequence with it. Algorithm 1 shows the algorithm to extract the melody structure given a series of melody bars. As an example, the structure graph of the melody "Simple Love" is shown in Fig. 2.

Each bar is matched with its previous bars from near to far to find out whether they are in the relation of repetition or rhythmic sequence.

3.2. Structure generation net

The structure graphs can be represented by adjacency matrices. Since the edges in graphs have an attribute *type* (repetition or rhythmic sequence), we use a three dimensional adjacency matrix x to represent a melody structure graph (V, A) where

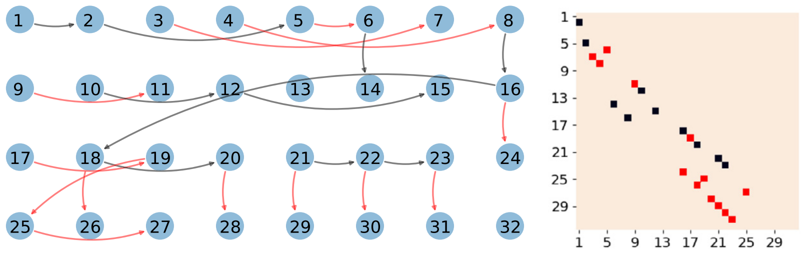


Fig. 2. The melody structure of "Simple Love". Melodies were segmented by bars. **Left:** The graph representation of melody structure (red line: repetition; black line: rhythmic sequence). **Right:** The adjacency matrix of the melody structure graph. The red dots represent repetitions and the black dots represent rhythmic sequences. The element in the i-th row and j-th column denotes the i-th bar is a repetition or rhythmic sequence of the j-th bar, where $i > j$. The same convention for plotting adjacency matrices is used throughout the paper. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Algorithm 1: Extracting structure from a melody.

```
   Input  : sequential bars $B = \{b_1, ..., b_n\}$
   Output: melody structure graph $(V, A)$
1  for $i \leftarrow 1$ to $n$ do
2      insert node $i$ to $A$
3      for $j \leftarrow i-1$ to $0$ do
4          if $b_i$ is a repetition of $b_j$ then
5              edge $e \leftarrow (j, i)$
6              set $e.type = 0$
7              insert $e$ to $V$
8              Break
9      end
10     if $b_i$ is not a repetition of previous bars then
11         for $j \leftarrow i-1$ to $0$ do
12             if $b_i$ is a rhythmic sequence of $b_j$ then
13                 edge $e \leftarrow (j, i)$
14                 set $e.type = 1$
15                 insert $e$ to $V$
16                 Break
17         end
18 end
```

$$x[i, j, k] = \begin{cases} 1, e = (j, k) \text{ in } (V, A) \text{ and } e.type = i \\ 0, \text{otherwise}. \end{cases} \tag{1}$$

If we restrict the maximum number of melody bars to be N, the size of the tensor x is $(2, N, N)$. For a melody having Q bars with $Q < N$, its adjacency matrix can be padded to have size $(2, N, N)$ with zeros (in this paper $N = 32$). Fig. 5a shows the adjacency matrices of some melody structures in our dataset. CNN-based models have been proved to be powerful for generating images that are always represented as 3D matrices. Since adjacency matrices are also 3-dimensional, we employed CNN to model them.

As illustrated in Fig. 3, a convolutional GAN is designed to generate the adjacency matrix of the melody structure graph. It consists of a critic D and a generator G. The critic D aims to distinguish between real melody structure graphs and generated melody structure graphs while the generator G aims to generate melody structure graphs that can fool D.

The input of G is a random vector $z \in \mathbb{R}^l$ sampled from an isotropic Gaussian distribution $N(0, I)$. The generator G consists of a fully-connected layer, three deconvolution layers, and a sigmoid layer. The output of the generator G is a matrix $\widetilde{x}$ and its size is $(2, n, n)$. The value $\widetilde{x}[i, j, k]$ indicates the probability of existence of edge $e = (j, k)$ with type i. The critic D is a CNN with three convolutional layers and a fully-connected layer.

The generator G and critic D are trained jointly under the Wasserstein GAN with gradient penalty framework [39]. The objective function for the critic to minimize is:

$$\underset{\widetilde{x}\sim\mathbb{P}_g}{\mathbb{E}}[D(\widetilde{x})] - \underset{x\sim\mathbb{P}_r}{\mathbb{E}}[D(x)] + \lambda \underset{\hat{x}\sim\mathbb{P}_{\hat{x}}}{\mathbb{E}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{2}$$

where $\mathbb{P}_g$ is the distribution of generated data and $\mathbb{P}_r$ is the distribution of real data. λ is the gradient penalty coefficient. The distribution $\mathbb{P}_{\hat{x}}$ is defined by taking straight lines between points in the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$: $\hat{x} = \epsilon x + (1-\epsilon)\widetilde{x}, \epsilon \sim U[0, 1], x \sim \mathbb{P}_r, \widetilde{x} \sim \mathbb{P}_g$.

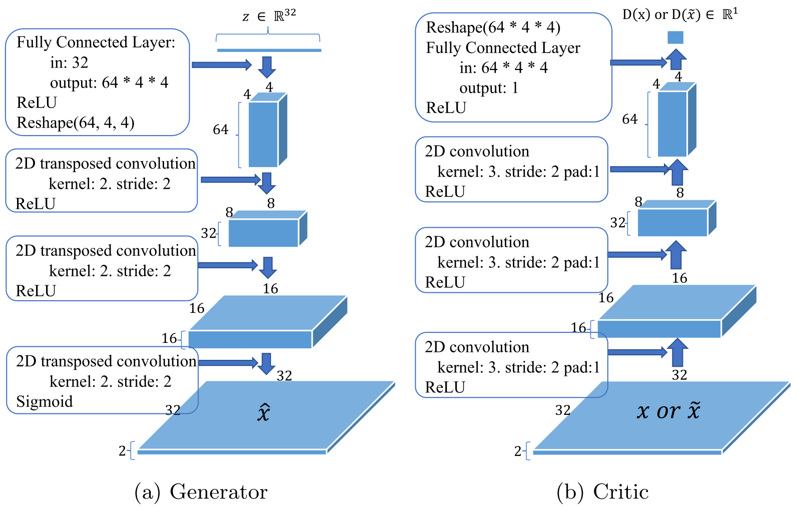

Fully Connected Layer: in: 32 output: 64 * 4 * 4 ReLU Reshape(64, 4, 4)
2D transposed convolution kernel: 2. stride: 2 ReLU
2D transposed convolution kernel: 2. stride: 2 ReLU
2D transposed convolution kernel: 2. stride: 2 Sigmoid
z ∈ ℝ^32
x̂
(a) Generator
Reshape(64 * 4 * 4) Fully Connected Layer in: 64 * 4 * 4 output: 1 ReLU
2D convolution kernel: 3. stride: 2 pad:1 ReLU
D(x) or D(x̃) ∈ ℝ^1
x or x̃
(b) Critic

Fig. 3. Architecture of SGN. Here $z, x, \widetilde{x}, D$ denote a random vector, real data, generated data and the critic respectively.

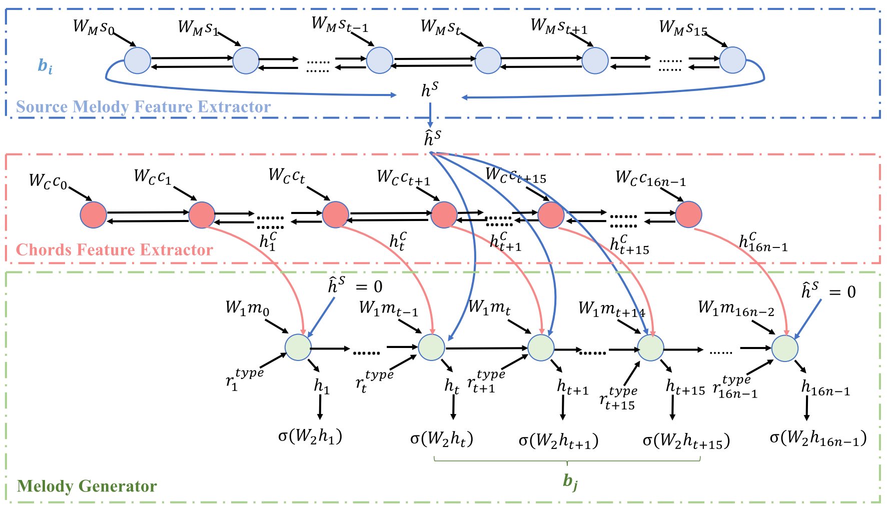


Fig. 4. Architecture of MGN. Here we assume that the bar b_j ranges from t to $t+15$ and it has a relation with another bar $b_i = \{s_0, ..., s_{15}\}$. So in the generation at every time step between t and $t+16$, we have $\hat{h}^S = h^S$. We also assume that the last melody bar does not have any relation with other melody bars, so $\hat{h}^S$ at the last time step is a zero vector.

The objective function for the generator G is

$$-\mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_g}[D(\widetilde{x})]. \tag{3}$$

In a generated adjacency matrix $\tilde{x}$, any $\widetilde{x}[i, j, k] \geq 0.5$ indicates the existence of a relation of type i between bars b_j and b_k. In this way, we can extract relations from the adjacency matrix $\widetilde{x}$ to form the melody structure.

The choice of GAN for generating the adjacency matrix is somehow arbitrary. The reason is that we empirically verified that it generated good enough results (see Section 5). Other methods may perform as well as GAN, or even better, but that is not the focus of this paper.

4. Melody generation conditioned on melody structure and chord progression

In this section, we will first describe the methods to represent melody and chord progression. Then we will introduce the MGN.

4.1. Melody representation

The representation of the melody is based on the General MIDI standard [40]. In a monophonic melody, all notes and rests are represented by 128 "note-on" tokens and one "note-off" token. Statistics on our dataset shows that about 99.83% of pitches are between C2 to C5. So all melodies are octave-shifted into this range so that only 36 "note-on" tokens in this range are used. As for timing, all notes and rests are quantized to 16th note intervals so that every bar is divided into 16 time steps and represented by 16 tokens. A special token "no-event" is used to represent that there is neither "note-on" nor "note-off" happened in a time step. Therefore a token can be denoted by a 38-dimensional one-hot vector (one element equal to one and other elements equal to zero). In this way, a melody M with the length of n bars results in a token sequence with the length of $16n$ ($M = \{m_0, m_1, ..., m_{16n-1}\}$, m_i is the one-hot representation of a token).

4.2. Condition representation

4.2.1. Melody structure condition

The melody structure provides relations between different bars. If there is a relation r between bars (b_i, b_j), the relation type $r.type$ and the source bar b_i are used in the generation of b_j. The $r.type$ is represented by a one-hot vector with a length of three (repetition, rhythmic sequence, and no-relation).

4.2.2. Chord progression condition

The chord progression is also quantized to 16th note intervals. For a chord progression with length of n bars, the quantization results in a chord sequence $C = \{c_0, c_1, ..., c_{16n}\}$ where each c_i is a 36-dimensional vector concatenated from a 12-dimensional one-hot encoding of root pitch, a 12 dimensional one-hot encoding of bass pitch and a 12-dimensional vector that indicates pitch class presented in the chord.

4.3. Melody generation net

Assume we aim to generate a melody with the length of n bars. The conditions provided to the model are a melody structure represented by a collection of relations $R = \{r_0, r_1, ..., r_k\}, k \leq n$ and a chord progression $C = \{c_1, c_2, ..., c_{16n}\}$. If $r = (b_i, b_j)$ is a repetition, the MGN learns to copy the source b_i to generate b_j. If $r = (b_i, b_j)$ is a rhythmic sequence, the MGN learns to generate b_j with the same rhythm to b_i.

The architecture of MGN is illustrated in Fig. 4. It consists of three submodules: a chord progression feature extractor, a source melody feature extractor, and a melody generator.

4.3.1. Chord progression feature extractor

A bidirectional Gated Recurrent Units (GRU) [41] is used to extract feature from chord progression C,

$$h^C = \mathrm{GRU}(W_C C), \tag{4}$$

where W_C is the embedding matrix for chords, and h^C is the hidden states at every time step which is used as feature for melody generation.

4.3.2. Source melody feature extractor

Given a relation $r = (b_i, b_j)$, a bidirectional GRU is used to summarize the source melody b_i:

$$h^S = \mathrm{GRU}(W_M b_i), \tag{5}$$

where W_S is the embedding matrix for source melody, and h^S is the final state of the GRU.

4.3.3. Melody generator

Another GRU is used as the melody generator. It takes features h^C, h^S and previous tokens as input and generates the token at the current time step.

At time step t, the previous token vector m_{t-1} is first embedded by the embedding matrix W_1:

$$\widetilde{m}_{t-1} = W_1 m_{t-1}. \tag{6}$$

If t is in bar b_j, the distribution y_t over all possible tokens is:

$$\begin{aligned} h_t &= \mathrm{GRU}(\widetilde{m}_{t-1}, h_{t-1}, h_t^C, \hat{h}^S, r.type) \\ y_t &= \sigma(W_2 h_t), \end{aligned} \tag{7}$$

where σ is a softmax layer and W_2 is a fully connected layer. Also, $\hat{h}^S$ equals to h^S if b_j is in a relation with b_i. Otherwise $\hat{h}^S$ equals to a zero vector with the same length as h^S.

4.4. Objective function

This MGN is trained by solving the following optimization problem:

$$\max_{\theta} \sum_{M \in \mathcal{M}} \sum_{t=1}^{\text{len}(M)} \log y_t(m_t; \theta), \tag{8}$$

where θ is the parameters of the MGN and $\mathcal{M}$ is the set of all melodies.

4.5. Generation

Given chord progression, melody structure and a primer sequence (the initial part of the sequence to be generated), the MGN generates the distribution y_t over all candidate tokens. The token m_t was chosen by sampling over y_t. Other tokens are generated in a successive way.

5. Experiments

5.1. Dataset

MIDI files typically represent chords with notes and it is difficult to distinguish between chord progressions and melodies. To utilize chord progression, we have collected 3,859 lead sheets in the format of MusicXML from http://www.wikifonia.org, which saves melody and chord progression separately. The time signatures of these lead sheets are 4/4. All melodies and chord progressions in these files were transposed to key C. The melody structures extracted from these melodies were used in the training of the SGN. Besides, in the training of the MGN, 90% of the lead sheets were used as the training set and the other 10% were used as the validation set. The dataset is public available.[1]

5.2. Implementation details

The hyper-parameters of SGN and MGN are detailed below. We did not exhaustively search for the optimal hyper-parameters, since it is difficult to quantitatively evaluate a music generation model.

5.2.1. Structure generation net

The generator G and critic D were trained with the Adam optimizer [42]. The initial learning rate was 0.0001. And β_1, β_2 and ϵ were $0.5, 0.9$ and 10^{-8}. The batch size was 50. The gradient penalty coefficient λ was 10. In each iteration, the critic D was updated five times while the generator G was updated once. Training stopped after 20,000 iterations.

5.2.2. Melody generation net

The embedding size and the hidden size of all RNNs were set to 256. Each RNN consists of two recurrent layers.

The MGN was trained with the Adam optimizer. The initial learning rate was 0.001. The β_1, β_2 and ϵ were $0.9, 0.999$ and 10^{-8}. The batch size was 64 and the weight decay was 10^{-5}. Dropout with the ratio of 0.5 was used for the outputs of the first recurrent layer of all RNNs. The validation-based early stopping was adopted to prevent over-fitting.

5.3. Structure generation results

5.3.1. Graph visualization

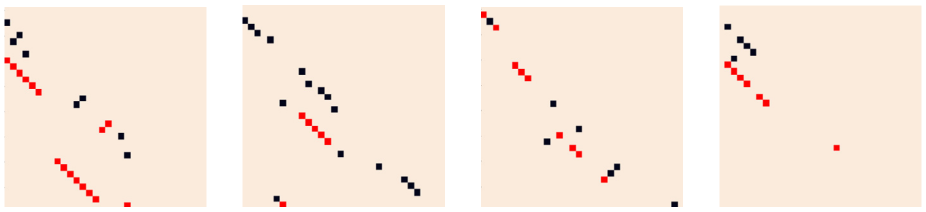
Fig. 5 visualizes the structures of several melodies in the dataset and the structures generated by SGN. It is seen that SGN can capture the characteristics of melody structures. For instance, the learned relations often exhibited group pattern: a set of consecutive bars were repetitions of another set of consecutive bars.

5.3.2. Statistics

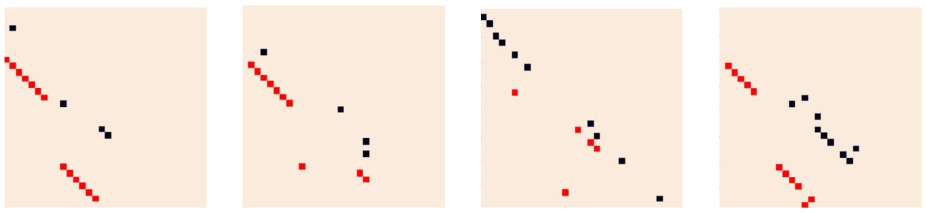
We wanted to know if the generated structures shared similar properties to real structures. We extracted the structure of the first 32 bars of each lead sheet in the dataset using Algorithm 1, which resulted in 3,859 structures in total. The distribution of distances (measured in the number of bars) between bars that were in a relation is defined as relation distance distribution. Fig. 6 illustrated different melodies' relation distance distributions. There were two observations:

- The majority of repetition distances were 1, 2, 4, 8, and 16 and the majority of rhythmic sequence distances were 1, 2, 4, and 8. Only a small part of relations had odd distances other than 1.

[1] https://www.dropbox.com/s/x5yc5cwjcx2zuvf/Wikifonia.zip?dl=0.



(a) The melody structures of real melodies.



(b) The melody structures generated by the SGN.

Fig. 5. The real melody structures and generated melody structures represented in the format of adjacency matrices. Best viewed in color.

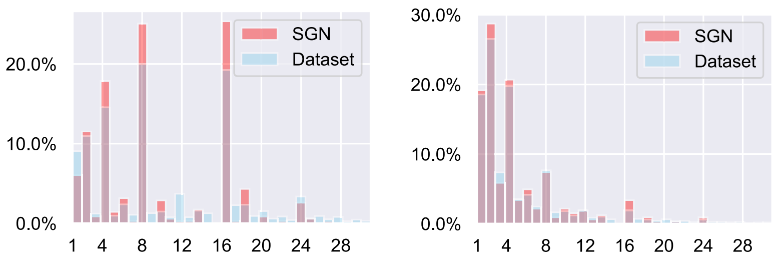


Fig. 6. The relation distance distributions of real structures and generated structures. **Left**: repetition. **Right**: rhythmic sequence. Best viewed in color.

- Most relations in large time scale melody structure (distance 8 and 16) were repetition, which indicates that repetition is more important for the formation of a large time scale melody structure.

We then generated 1000 structures with a length of $N = 32$ using SGN. The relation distance distribution of these generated structures had similar characteristics with that of real melody structures (Fig. 6).

5.4. *Melody generation results*

5.4.1. *Our models*

The PopMNet and its two variants were used in the experiments:

- **PopMNet:** the model proposed in this paper. The melody structures used in the melody generation are generated by the SGN.
- **PopMNet-Real:** same as PopMNet but the melody structures used in the melody generation are extracted from the dataset.
- **PopMNet-NC:** the PopMNet without chord progressions as a condition.

5.4.2. *Existing models for comparison*

Four state-of-the-art models were compared with our model, whose source codes are publicly available.

- **AttentionRNN:** an RNN-based model with attention mechanism [4]. Chord progressions are used as a condition.
- **LookbackRNN:** an RNN-based model with Lookback feature [4]. Chord progressions are used as a condition.
- **MidiNet:** a CNN-based GAN [6].
- **Music Transformer:** an attention-based neural network [18].

The AttentionRNN, LookbackRNN, and MidiNet were trained on our dataset with the same setting in their original papers. The Music Transformer was also trained on our dataset but the setting of the model is a little bit different from the original work. The original Music Transformer was used to generate piano performances. To generate melodies for comparison, we made a simple extension by letting its encoder encode chord progressions and its decoder decode melodies.

MusicVAE [3] is also available open source, but not included in comparison here. All the models we compared are variable-length autoregressive models, while MusicVAE is a fixed-length latent variable model. Despite efforts in tuning its hyperparameters, we failed to obtain satisfactory results with MusicVAE after training it on our dataset. To obtain good results, MusicVAE may require a large dataset for training. In the original work [3], the training dataset consisted of 1.5 million MIDI files.

5.4.3. Input of models

MGN, LookbackRNN, AttentionRNN, MidiNet, and Music Transformer needed chord progressions and primer melodies as input to generate melodies. The chord progressions and primer melodies were randomly sampled from lead sheets in the dataset. The length of the used primer melody was 1 bar.

5.4.4. Melody structure analysis

Each model generated many melodies. Then Algorithm 1 was used to extract their structures. The melodies generated by AttentionRNN, LookbackRNN, and MidiNet hardly had any clear structures. There were almost no repetitions in melodies of AttentionRNN while melodies generated by PopMNet and Music Transformer contained clear structures. One observation is that the melodies generated by PopMNet and Music Transformer often contain long-term relations, which is indicated by red or black dots towards the lower left of the adjacency matrices (e.g., the first and fourth column in Figs. 7d, 7e). However, such dots are scarce in the adjacency matrices of the other three models.

For quantitative comparison, each model generated 200 melodies. The percentages of relations in the melodies are shown in Table 1. The percents of PopMNet and Music Transformer are close to the real data. The percents of repetition and rhythmic sequence of MidiNet and LookbackRNN are comparable with those of PopMNet. However, most repetitions of MidiNet and LookbackRNN had distances within 4 bars (Fig. 8). The distributions of PopMNet and Music Transformer were similar to the distributions of real melody structures (Fig. 8).

We employed wasserstein distance to quantitatively measure the distance between two relation distance distributions u and v:

$$l(u, v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| \mathrm{d}\pi(x, y) \tag{9}$$

where $\Gamma(u, v)$ is the set of probability distributions whose marginals are u and v on the first and second factors respectively. Let u denote the distribution of melodies generated by a model and v denote the distribution of the dataset. Then $l(u, v)$ denotes the structure metric of the model on the dataset. If u denotes the distribution of a single melody, then $l(u, v)$ denotes the structure metric of this melody. The wasserstein distance between u and v was calculated with a python package SciPy [43]. Table 2 shows different models' structure metrics. Among the four existing models, Music Transformer performed the best in capturing repetitions in the dataset. However, it was inferior to LookbackRNN and MidiNet in capturing rhythmic sequences. Our models outperformed all existing models, which indicates that melodies generated by our models have better structures compared to melodies generated by other models. Structure metrics of PopMNet and PopMNet-NC were similar, suggesting that the structure of the generated melodies was mainly brought by the guidance of the generated melody structure instead of the chord progression.

5.4.5. Human behavior experiment

Evaluating the generation quality of generative models is a challenging task. So far no convincing metric can measure the quality of melodies. Some works [44–46] employed the likelihood to evaluate generative models for sequence generation. We did not use it as a metric to evaluate models because we found that a higher likelihood on the training set or the evaluation set does not imply better performance by a preliminary behavior experiment by the first author. Similar to many melody generation works [6,5,3], we conducted two human behavior experiments to evaluate the melodies generated by different models. In experiments, subjects were asked to rate melodies on a scale from 0 to 5 on the following four metrics:

- Pleasure: do you like the melody?
- Reality: is the melody composed by a human?
- Smoothness: is the transition between notes smooth?
- Integrity: does the melody have a clear structure?

These metrics are similar to the metrics used in previous studies [5,6]. Each model generated ten melodies for rating. The lengths of melodies were 32 bars. All melodies were played at the speed of 120 beats per minute, and they were presented to the subjects randomly. All midi files and audio files can be found in **Supplementary Materials**.

Experiment 1. In this experiment, we evaluated five models, PopMNet, PopMNet-real, AttentionRNN, LookbackRNN, and MidiNet. Music Transformer was evaluated in a separate experiment (see Experiment 2) because it was added to the paper

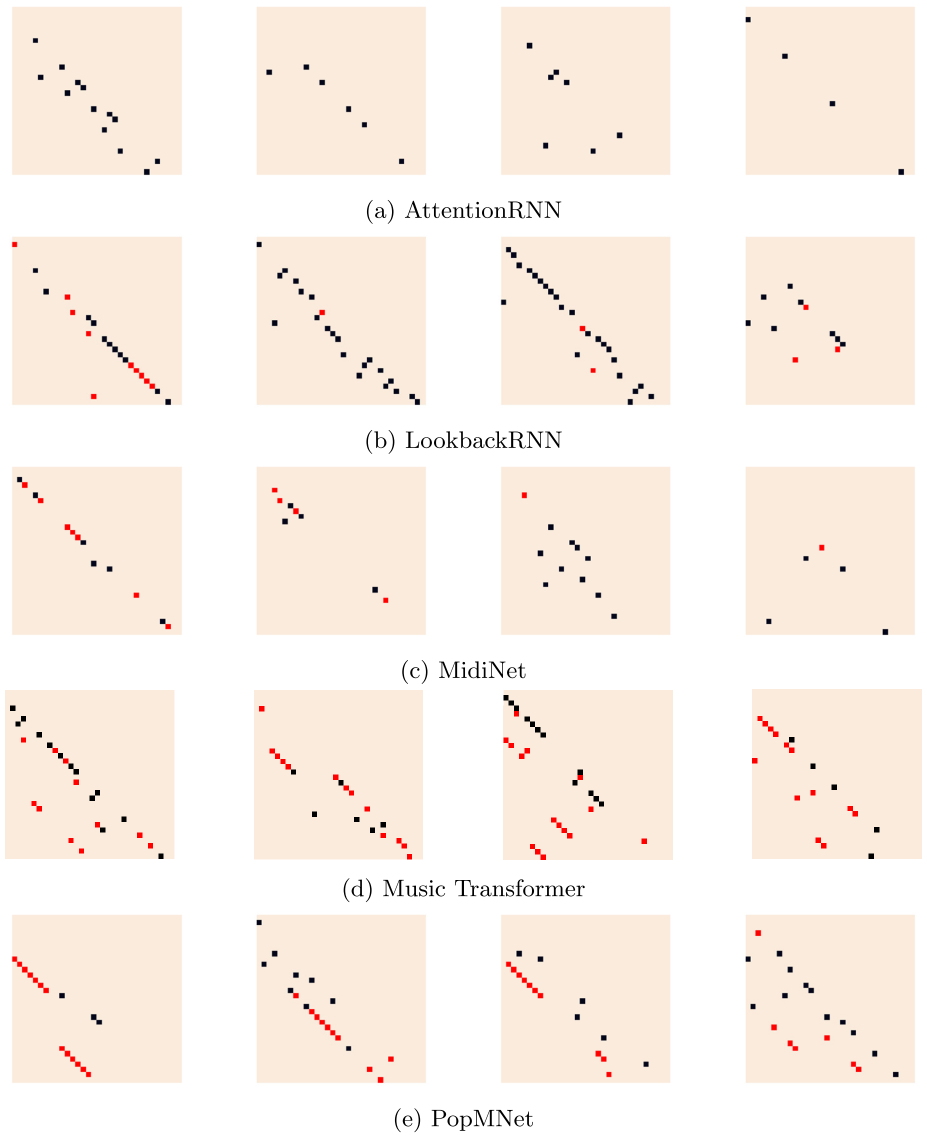
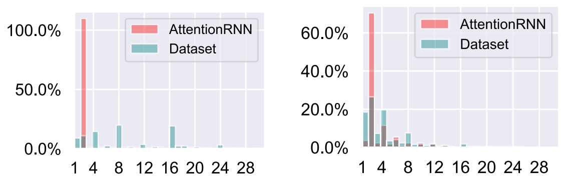


Fig. 7. The structures of example melodies generated by different models. The melodies of structures shown in the first two columns are presented in Supplementary Figs. 1–5, respectively. Best viewed in color.

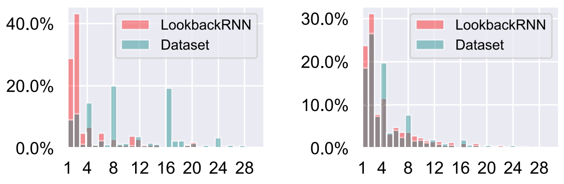
Table 1
The percentages of relations in the melodies in the dataset and the melodies generated by different models.

	Repetition	Rhythmic sequence	No relation
Real data	29.06%	32.64%	38.30%
AttentionRNN	0.42%	24.73%	74.85%
LookbackRNN	9.13 %	37.78%	53.09%
MidiNet	18.75%	18.752%	62.29%
Music Transformer	35.64%	25.15%	39.21%
PopMNet	27.05%	28.91%	44.04%
PopMNet-Real	31.45%	31.33%	37.22%
PopMNet-NC	26.28%	29.45%	44.27%

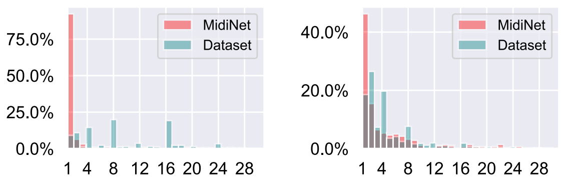
in the revision phase and it would be costly to redo this experiment by incorporating a new model. 17 subjects (6 female and 11 male, ages between 19 and 42) were paid to participate in this experiment. 8 subjects among them were professional music practitioners. The whole experiment for each subject took about 55 minutes.



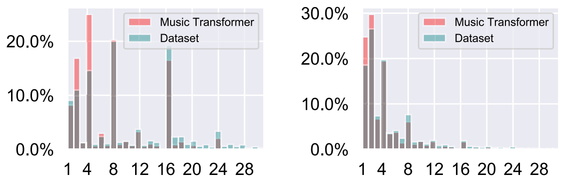
(a) AttentionRNN



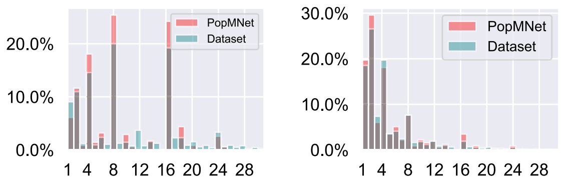
(b) LookbackRNN



(c) MidiNet



(d) Music Transformer



(e) PopMNet

Fig. 8. The relation distance distributions of the melodies generated by different models. **Left:** Repetition. **Right:** Rhythmic sequence. Best viewed in color.

Table 2
Structure metrics of melodies generated by different models.

	Repetition	Rhythmic sequence
AttentionRNN	8.272	1.973
LookbackRNN	6.300	0.508
MidiNet	8.731	0.727
Music Transformer	2.597	0.848
PopMNet	1.743	**0.484**
PopMNet-Real	**1.565**	0.665
PopMNet-NC	1.738	0.562

Table 3 shows the human evaluation scores given by all subjects. PopMNet outperformed the four existing models on all metrics by a large margin (Pleasure: $p \leq 0.0043$; Reality: $p \leq 0.00054$; Smooth: $p \leq 0.0011$; Integrity, $p \leq 9.00 \times 10^{-5}$; one-tailed t-test, $N = 170$). This demonstrates the effectiveness of the melody structure for melody generation. The average

Table 3
Human evaluation scores of melodies in Experiment 1 (mean±std, all 17 subjects).

	Pleasure	Reality	Smooth	Integrity
AttentionRNN	2.57 ± 1.43	2.39 ± 1.43	2.53 ± 1.39	2.60 ± 1.41
LookbackRNN	3.18 ± 1.17	2.99 ± 1.17	3.08 ± 1.13	3.18 ± 1.16
MidiNet	2.70 ± 1.36	2.57 ± 1.39	2.81 ± 1.32	2.93 ± 1.34
PopMNet	3.50 ± 1.00	3.40 ± 1.02	3.46 ± 1.07	3.64 ± 1.03
PopMNet-Real	3.55 ± 1.12	3.38 ± 1.12	3.50 ± 1.04	3.61 ± 1.09

Table 4
Human evaluation scores of melodies in Experiment 1 (mean±std, 8 music practitioners).

	Pleasure	Reality	Smooth	Integrity
AttentionRNN	1.59 ± 0.93	1.32 ± 0.81	1.62 ± 0.92	1.75 ± 1.16
LookbackRNN	2.68 ± 1.15	2.37 ± 1.00	2.64 ± 1.04	2.75 ± 1.18
MidiNet	2.02 ± 1.10	1.70 ± 0.97	2.19 ± 1.16	2.21 ± 1.20
PopMNet	3.39 ± 0.83	3.10 ± 1.02	3.26 ± 1.05	3.33 ± 1.03
PopMNet-Real	3.31 ± 1.08	3.03 ± 1.08	3.25 ± 1.05	3.38 ± 1.18

Table 5
Human evaluation scores of melodies in Experiment 2 (mean±std, all 18 subjects).

	Pleasure	Reality	Smooth	Integrity
Music Transformer	2.98 ± 1.08	3.01 ± 1.16	3.10 ± 1.97	3.02 ± 1.23
PopMNet	3.41 ± 0.83	3.29 ± 0.82	3.39 ± 0.94	3.31 ± 0.94
Human	3.92 ± 0.92	3.89 ± 0.94	3.85 ± 0.91	3.98 ± 0.89

scores of the PopMNet and PopMNet-Real were not significantly different ($p \geq 0.34$ on all metrics; two-tailed t-test, $N = 170$). This verified that the SGN could generate realistic melody structures.

Similar conclusions can be obtained from the scores given by 8 music practitioners (Table 4). PopMNet outperformed the four existing models on all metrics by a large margin (Pleasure: $p \leq 3.73 \times 10^{-5}$; Reality: $p \leq 9.86 \times 10^{-6}$; Smooth: $p \leq 1.19 \times 10^{-5}$; Integrity, $p \leq 7.21 \times 10^{-4}$; one-tailed t-test, $N = 80$). The average scores of the PopMNet and PopMNet-Real were not significantly different ($p \geq 0.738$ on all metrics; two-tailed t-test, $N = 80$). But scores given by 8 music practitioners were lower than scores given by all subjects significantly. This indicated that music practitioners have higher standards compared to ordinary subjects.

Experiment 2. In this experiment, we evaluated PopMNet and Music Transformer. 18 subjects (11 female and 7 male, ages between 19 and 34, four among which were professional music practitioners) were paid to participate in this experiment. Besides melodies generated by PopMNet and Music Transformer, 10 melodies sampled from the dataset were also evaluated. These 10 melodies were labeled as “Human”. The whole experiment for each subject took about 35 minutes.

PopMNet performed significantly better than Music Transformer (Pleasure: $p = 2.25 \times 10^{-5}$; Reality: $p = 0.0059$; Smooth: $p = 0.0038$; Integrity, $p = 0.0070$; one-tailed t-test, $N = 180$), but performed worse than human (Pleasure: $p \leq 2.66 \times 10^{-8}$; Reality: $p \leq 1.87 \times 10^{-10}$; Smooth: $p \leq 1.34 \times 10^{-6}$; Integrity, $p \leq 4.91 \times 10^{-12}$; one-tailed t-test, $N = 180$) (Table 5).

5.4.6. *Correlation between score and structure*

We have shown that, statistically, our proposed model generated melodies with better structures and higher human evaluation scores than existing models. However, it is unclear if a higher score is actually due to a better structure, because it is possible that some melodies with bad structures had high evaluation scores and some melodies with good structures had low evaluation scores. Fig. 9 shows the structures of some of the melodies with the highest and lowest scores in the first human behavior experiment. Most melodies that got high scores have rich and long-distance relations between bars. On the other hand, the melodies that got low scores did not have clear structures.

We calculated the correlation between structure metrics and human evaluation scores of all 50 melodies evaluated in Section 5.4.5 (see Fig. 10). A strong correlation (r=-0.757) was found between the repetition metric and the evaluation score. However, only a weak correlation (r=-0.128) was found between the rhythmic sequence metric and the evaluation score. One reason is that almost all structures of generated melodies contained many rhythmic sequences (Figs. 7, 9) and their distributions had small wasserstein distances to the distribution of the rhythmic sequence of the dataset (Table 2).

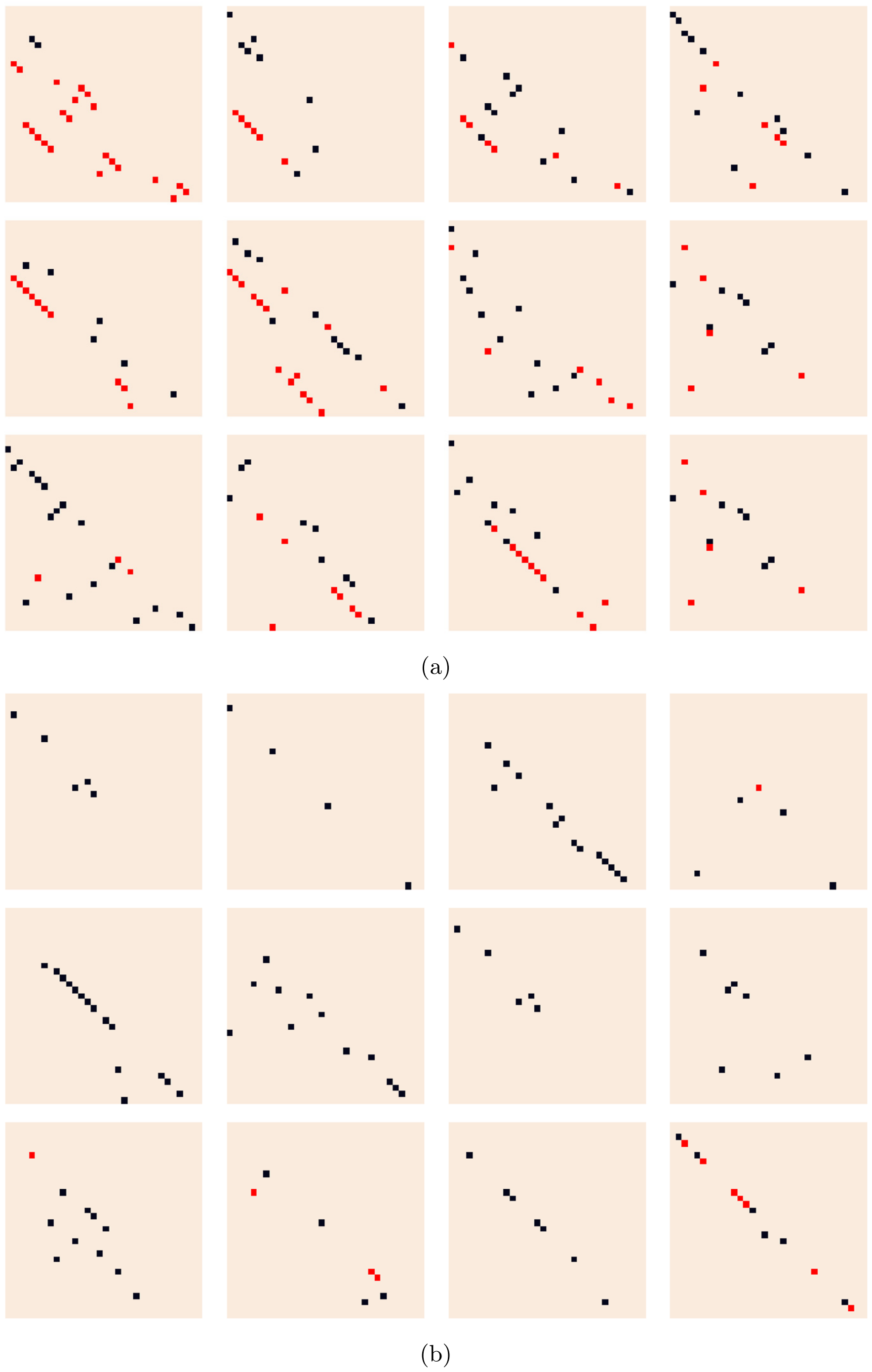


Fig. 9. The structures of 24 melodies with highest (a) and lowest (b) average scores, averaged across subjects and the four described metrics. Best viewed in color.

6. Discussion

We present the PopMNet to generate structured pop music melodies, which integrates melody structure into the generation process. This model consists of a Structure Generation Net and a Melody Generation Net. The former generates melody structures and the latter generates melodies conditioned on the structures and chord progressions. In experiments, we compared PopMNet with four existing models AttentionRNN, LookbackRNN, MidiNet, and Music Transformer. The results indicate that PopMNet can generate higher quality melodies with clear structures than these existing models.

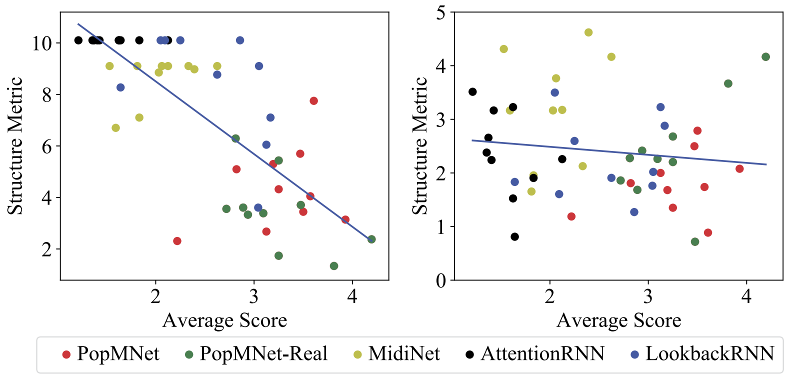


Fig. 10. Correlation between structure metrics and average scores of melodies. **Left:** Repeat; **Right:** Rhythmic sequence. Each point denotes a generated melody and blue lines represent the best fit to all points. Best viewed in color.

This is only the first step towards generating rich and compelling pop music melodies. First, only repetition and rhythmic sequence between bars were considered as structures, while real, human-composed pop music melodies contain much more complex relations between melody segments, which will certainly be studied. Second, in this paper our focus is the idea of two-stage generation: from structure to melody, not specific models we have used. It is possible that more advanced models could be devised to further improve the generation of musical structure and melody, respectively. These two aspects are worth more exploration

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by National Key Research and Development Program of China under Grant 2017YFA0700904 and National Natural Science Foundation of China under Grant U19B2034, Grant 61836014, and Grant 61620106010. The authors thank Runxuan Yang for useful suggestions.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2020.103303.

References

[1] F.P. Brooks, A.L. Hopkins, P.G. Neumann, W.V. Wright, An experiment in musical composition, IEEE Trans. Comput. EC-6 (1957) 175–182.
[2] B. Lindblom, J. Sundberg, Towards a generative theory of melody, STL-QPSR 10 (1969) 053–086.
[3] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, D. Eck, A hierarchical latent vector model for learning long-term structure in music, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 2018, pp. 4364–4373.
[4] E. Waite, D. Eck, A. Roberts, D. Abolafia, Project magenta: generating long-term structure in songs and storie, 2016.
[5] H. Zhu, Q. Liu, N.J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, E. Chen, 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, New York, NY, USA, 2018, pp. 2837–2846.
[6] L.-C. Yang, S.-Y. Chou, Y.-H. Yang, Midinet: a convolutional generative adversarial network for symbolic-domain music generation, in: Proceedings of the 18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017.
[7] H. Dong, W. Hsiao, L. Yang, Y. Yang Musegan, Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2018, pp. 34–41.
[8] R. Middleton, 'Play it again sam': some notes on the productivity of repetition in popular music, Pop. Music 3 (1983) 235–270.
[9] C. Drake, C. Palmer, Accent structures in music performance, Music Percept. 10 (1993) 343–378.
[10] S.R. Holtzman, A generative grammar definition language for music, J. New Music Res. 9 (1980) 1–48.
[11] S. Holtzman, Using generative grammars for music composition, Comput. Music J. 5 (1981) 51–64.
[12] J.A. Biles, et al., GenJam: a genetic algorithm for generating jazz solos, in: International Computer Music Conference, vol. 94, 1994, pp. 131–137.
[13] S. Phon-Amnuaisuk, A. Tuson, G. Wiggins, Evolving musical harmonisation, in: Artificial Neural Nets and Genetic Algorithms, Springer, 1999, pp. 229–234.
[14] D. Eck, J. Schmidhuber, A first look at music composition using LSTM recurrent neural networks, in: Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, vol. 103, 2002.
[15] F. Colombo, A. Seeholzer, W. Gerstner, Deep artificial composer: a creative neural network model for automated melody generation, in: Computational Intelligence in Music, Sound, Art and Design, Springer International Publishing, Cham, 2017, pp. 81–96.
[16] H. Chu, R. Urtasun, S. Fidler, Song from pi: a musically plausible network for pop music generation, in: International Conference on Learning Representations (ICLR), 2017, workshop track.

[17] O. Mogren, C-rnn-gan: a continuous recurrent neural network with adversarial training, in: International Conference on Neural Information Processing Systems, Constructive Machine Learning Workshop, 2016.
[18] C.-Z.A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A.M. Dai, M.D. Hoffman, M. Dinculescu, D. Eck, Music transformer, in: International Conference on Learning Representations, 2019.
[19] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, K. Webster, Structurenet: inducing structure in generated melodies, in: The 19th International Society for Music Information Retrieval Conference, 2018, pp. 725–731.
[20] K. Chen, W. Zhang, S. Dubnov, G. Xia, W. Li, The effect of explicit structure encoding of deep neural networks for symbolic music generation, in: 2019 International Workshop on Multilayer Music Representation and Processing (MMRP), IEEE, 2019, pp. 77–84.
[21] B. Xu, S. Wang, X. Li, An emotional harmony generation system, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–7.
[22] S. Chiu, M. Shan, Computer music composition based on discovered music patterns, in: Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics, IEEE, Taipei, 2006, pp. 4401–4406.
[23] D. Herremans, E. Chew, Morpheus: generating structured music with constrained patterns and tension, IEEE Trans. Affect. Comput. (2017).
[24] H. Jhamtani, T. Berg-Kirkpatrick, Modeling self-repetition in music generation using generative adversarial networks, in: Machine Learning for Music Discovery Workshop, ICML, 2019.
[25] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur, Recurrent neural network based language model, in: T. Kobayashi, K. Hirose, S. Nakamura (Eds.), INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010, ISCA, 2010, pp. 1045–1048.
[26] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014, pp. 3104–3112.
[27] J.T. Zhou, H. Zhang, D. Jin, X. Peng, Dual adversarial transfer for sequence labeling, IEEE Trans. Pattern Anal. Mach. Intell. (2019) 1.
[28] L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, J. Han, Empower sequence labeling with task-aware neural language model, in: AAAI, 2018.
[29] N. De Cao, T. Kipf, MolGAN: an implicit generative model for small molecular graphs, in: International Conference on Machine Learning, Workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.
[30] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, B.Y. Zhao, Measurement-calibrated graph models for social network experiments, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 861–870.
[31] P. Erdos, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci 5 (1960) 17–60.
[32] G. Robins, P. Pattison, Y. Kalish, D. Lusher, An introduction to exponential random graph (p^*) models for social networks, Soc. Netw. 29 (2007) 173–191.
[33] T.A. Snijders, K. Nowicki, Estimation and prediction for stochastic blockmodels for graphs with latent block structure, J. Classif. 14 (1997) 75–100.
[34] M. Simonovsky, N. Komodakis, Graphvae: towards generation of small graphs using variational autoencoders, arXiv preprint, arXiv:1802.03480, 2018.
[35] J. You, R. Ying, X. Ren, W. Hamilton, J. Leskovec, GraphRNN: generating realistic graphs with deep auto-regressive models, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 2018, pp. 5708–5717.
[36] J. Li, M.-T. Luong, D. Jurafsky, A hierarchical neural autoencoder for paragraphs and documents, arXiv preprint, arXiv:1506.01057, 2015.
[37] J. Walker, K. Marino, A. Gupta, M. Hebert, The pose knows: video forecasting by generating pose futures, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3332–3341.
[38] Y. Shen, B. Zhou, P. Luo, X. Tang, Facefeat-gan: a two-stage approach for identity-preserving face synthesis, arXiv preprint, arXiv:1812.01288, 2018.
[39] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of wasserstein gans, in: Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5767–5777.
[40] T. International MIDI Association, General MIDI level 1 specification, 1991.
[41] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 2014, pp. 1724–1734.
[42] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations, San Diego, 2015.
[43] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E.W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, S... Contributors, SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python, arXiv: e-prints arXiv:1907.10121, 2019.
[44] A.M. Lamb, A.G.A.P. Goyal, Y. Zhang, S. Zhang, A.C. Courville, Y. Bengio, Professor forcing: a new algorithm for training recurrent networks, in: Advances in Neural Information Processing Systems, 2016, pp. 4601–4609.
[45] D. Serdyuk, N.R. Ke, A. Sordoni, A. Trischler, C. Pal, Y. Bengio, Twin networks: matching the future for sequence generation, in: International Conference on Learning Representations, 2018.
[46] Z. Dai, Z. Yang, Y. Yang, W.W. Cohen, J. Carbonell, Q.V. Le, R. Salakhutdinov, Transformer-xl: attentive language models beyond a fixed-length context, arXiv preprint, arXiv:1901.02860, 2019.