Contour-Preserving Melody Conversion

Satoru Fukayama and Masataka Goto
National Institute of Advanced Industrial Science and Technology (AIST)
{s.fukayama, m.goto}@aist.go.jp

ABSTRACT

This paper presents contour-preserving melody conversion which enables a melody to be brought from one accompaniment to another while preserving the melodic contour. The conversion automatically adjusts pitches that are inconsistent with the accompaniment and enables convenient interactive composition. The conversion can be achieved by an optimization using two different probabilistic models: a contour model to preserve the melodic contour, and a consistency model to maintain consistency between the melody and the accompaniment. The contour model consists of time-varying transition probabilities, and the consistency model is governed by bi-directional recurrent neural networks. For evaluation, we have calculated two measures: the root mean squared difference of intervals to check how the conversion preserves the original melodic contour, and the negative log-probability on test data to see the consistency between the melody and the accompaniment. The evaluation results indicate that a composer can control how strictly to preserve the melodic contour by balancing between the contour model and the consistency model.

1. INTRODUCTION

The recent development of fully automatic music generation enables an interactive computer-aided composition where a composer and an automatic composition collaborate in composing music. Automatic music generation outputs various musical pieces and inspires the composer through the iterative uses of automatic music generation. It can also generate music from specific fragments of music such as melodies or chords that are sketched manually by a composer, and completes or adjusts the remaining parts of the music. Our proposed method contour-preserving melody conversion is a novel way to compose music interactively with a computer.

Contour-preserving melody conversion regards a melody by a composer as a source melody and adjusts the pitches to fit different accompaniments. It enables a composer to freely explore melodies aside from constraints of the accompaniment. The overview of our method is shown in Figure 1. To prevent diminishing the character of the source melody through the conversion, it preserves the melodic contour and the rhythm of the source melody. A melodic contour is a sequence of single pitches that resembles the original intervallic patterns of a melody, and the contour can appear with changes in intervallic detail [1]. The experiments show that listeners usually find it easy to respond positively to all comparison melodies that share a melodic contour and respond negatively to melodies with different melodic contours [2, 3].

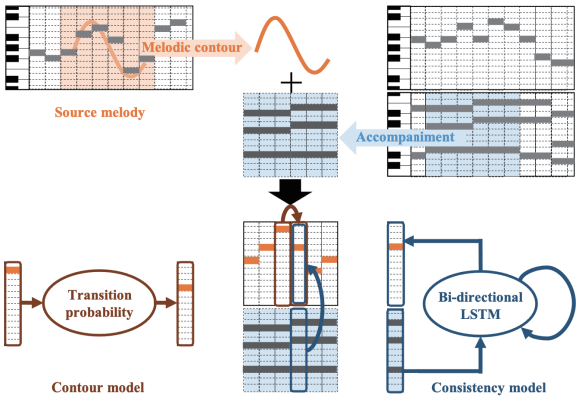

Figure 1. Overview of contour-preserving melody conversion to generate a novel melody by adjusting pitches in the source melody to fit a different accompaniment.

A technical issue for the contour-preserving melody conversion is to achieve consistency between a melody and the accompaniment while preserving the original melodic contour. The accompaniment contains chords, modes, and tonality, and it governs the available pitches in the melody. The conversion needs to generate a melody which satisfies both the constraints given by a melodic contour and the consistency between the melody and the accompaniment.

The contribution of this paper is to show that contour-preserving melody conversion can be achieved by solving an optimization problem. The objective function to maximize consists of two probabilistic models: a contour model and a consistency model. The contour model comprises a sequence of transition probabilities that represents the constraints of intervals within a melody. The consistency model is a recurrent neural network (RNN) with long short-term memory (LSTM) recurrent units [4] that can infer conditional probabilities of pitches when an accompaniment is given. The adjusted pitches for a new melody are generated by searching for the optimal pitches with dynamic programming.

2. RELATED WORK

2.1 Generating melody from melodic contour

A system that generates a solo melody from a hand-drawn curve while satisfying constraints given by a chord sequence in blues style has been proposed [5]. Compared to this work that allows users to input an abstracted form of melodic contour to control the results, our work provides a method to extract it from the existing example, which enables us to inherit an attractive melodic contour in an existing melody.

2.2 Generating melodic variations

Our work was inspired by the previous work on generating variations of a melody [6]. A variation of a melody is probabilistically sampled to have a differentiated rhythm and melody notes but an impression similar to that of the original melody. Our work shares the philosophy of making use of melodic structure in existing melodies rather than generating it from scratch. Using chord sequences to constraint generation of melodic variations has also been proposed [7]. Our work copes with both generating variations and handling constraints by the accompaniment in a unified framework.

2.3 Generating sequences with constraints

Our research was also inspired by attempts to create a long-term structure by a Markov chain. A method based on exact sampling with belief propagation to handle dependencies between remote words or melody notes in a sequence has been proposed [8]. Imposing constraints when sampling from a Markov chain to create long-term repetitive structures and phrases has also been proposed [9]. Our method imposes constraints on a probabilistic sequence model to generate a melodic contour. However, rather than trying to generate a structure by learning from data, our approach relies on an existing melody and reuses its melodic contour to generate a new melody.

2.4 Modeling melody-accompaniment consistency

Our method models the consistency between a melody and an accompaniment. Melody harmonization is the major research theme from this viewpoint. Expert systems [10], neural networks [11, 12], Markov chains [13], hidden Markov models [14], and log-linear interpolation of probabilistic models [15] have been used. Our method generates a melody rather than chords, and it could be regarded as a different application of modeling the consistency.

2.5 Generating sequence with neural networks

Our work could also be regarded as a melody generation using neural networks. In recent years, neural networks, especially Transformers and RNNs have been used for generating polyphonic music [16, 17, 18, 19], chord sequences [12], guitar tabs [20], and chorales [21, 19, 22]. DeepBach generates voices of chorales in the style of Johan Sebastian Bach with bi-directional LSTM and Gibbs sampling [23]. SequenceTutor is a system to adapt RNNs to generate more preferable melody sequences by applying reinforcement learning [24]. MidiNet and MuseGan train convolutional neural networks (CNNs) as generative adversarial networks (GANs) [25, 26]. Counterpoint by convolution achieves counterpoint with CNNs [27]. Our method generates a melody by using not only the probability given by the neural networks but also using probabilities from the melodic contour, which provides us with essential constraints to generate melodic variations.

3. CONTOUR-PRESERVING MELODY CONVERSION

3.1 Composition with contour-preserving melody conversion

Let us assume that a composer wants to use a melody (the source melody) with a different accompaniment. The melody initially with an original accompaniment is now combined with a different accompaniment, and there could be pitches in a melody that are inconsistent with the newly attached accompaniment. The preferable method for computer-aided composition is to automatically adjust pitches in a melody to be consistent with the new accompaniment. However, the pitches could be changed drastically, and this could change the characteristics of the source melody.

Our method keeps the original characteristics of the melody by preserving the melodic contour while adjusting pitches to be consistent with the accompaniment. The method firstly creates a contour model from intervals of the source model. The model comprises a set of transition probabilities to achieve a pitch transition which resembles the intervallic patterns of the source melody. Then the method constructs a consistency model by training a bi-directional LSTM from paired melody and accompaniment data. Finally, the method executes melody conversion to adjust pitches to meet constrains given by the contour model and the consistency model.

3.2 Contour model

A sequence of intervals can be represented as transition probabilities between consecutive melody notes. Let us represent the pitches in a melody as $x_1, x_2, \ldots, x_N$, where x_n is the MIDI note number of the n^{th} note and N is the total number of notes. A transition probability $q(x_n|x_{n-1})$ that achieves interval k in semitones between x_{n-1} and $x_n = x_{n-1} + k$ is

$$q(x_n|x_{n-1}) = \begin{cases} 1 & (x_n = x_{n-1} + k) \\ 0 & (\text{otherwise}). \end{cases} \quad (1)$$

Since a melodic contour can appear with changes in intervallic detail, the transition probabilities need to allow the case when the intervals are slightly different from the original. To achieve this, we design the transition probability for interval k as:

$$q(x_n|x_{n-1}) = \frac{1}{W} \exp\left(-\frac{1}{2}(x_n - (x_{n-1} + k))^2\right), \quad (2)$$

$$W = \varepsilon + \sum_{x_n} \exp\left(-\frac{1}{2}(x_n - (x_{n-1} + k))^2\right), \quad (3)$$

where ε is a very small constant for numerical stability.

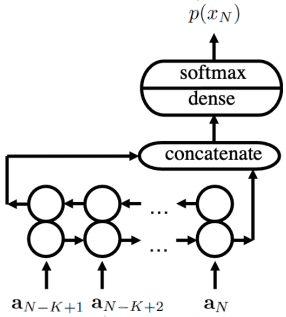

Figure 2. Network architecture of consistency model which outputs probability for $P(x_N)$ from accompaniment vector inputs $\mathbf{a}_{N-K+1}, \mathbf{a}_{N-K+2}, \ldots, \mathbf{a}_N$ (step size K).

3.3 Consistency model

We create a consistency model so that we can measure to what extent the pitches are consistent with the accompaniment. We take a machine learning approach for creating this model. We chose (1-layer) bi-directional RNNs with LSTM recurrent units [4] for the neural network structure. RNNs are used because they can capture the dependencies between the consecutive melody notes.

An accompaniment behind the melody contains notes coming from chords and other voices. We denote the probability of having the particular pitch behind the n^{th} melody note as accompaniment vector $\mathbf{a}_n$. The dimension of this vector is the same as the number of available pitches in MIDI, which is 128. The i^{th} element is defined to be the probability of having a pitch of MIDI note number i. Examples of the accompaniment vectors are shown on the right in Figure 3. Each value in an accompaniment vector is the ratio of the duration of an accompaniment note to the duration of a melody note. Suppose we observe a melody note whose onset time is 0 and the offset time is 100 in MIDI ticks. If we observe an accompaniment consists of single note in pitch 48 where the onset time is 50 and offset time is 200, an accompaniment vector will contain zeros except for the 48-th element, which will be 0.5. Since the accompaniment note overlaps the melody note at onset time 50 to offset time 100, the duration of overlapping is 50 whereas the duration of the melody is 100 and the value for the 48-th element is $0.5\ (= (100-50)/100)$.

When an accompaniment has multiple parts or tracks, an accompaniment vector is first created for each accompaniment part. Then the vectors for all parts are combined to create a single vector for every melody note by taking the maximum value among the multiple parts.

The bi-directional RNNs take K accompaniment vectors $\mathbf{a}_{N-K+1}, \mathbf{a}_{N-K+2}, \ldots, \mathbf{a}_N$ as inputs, and they output probability $p(x_N)$ as a 128 dimensional vector. To achieve this non-linear transformation from the input to the output, we set the dimension of the hidden units to 32, and set the activation function for input and for the recurrent input to the rectified linear unit and sigmoid, respectively. To ensure the output is a probability distribution, we set a softmax at the output layer. The network structure of the consistency model is illustrated in Fig. 2.

We trained the RNNs by minimizing the categorical cross-entropy loss between the output and the 128-dimensional one-hot vectors created from the pitches of the melody notes. Examples of the one-hot vectors are shown on the left in Figure 3. The probability of dropout was set to 0.3 to avoid overfitting, and the step size of the input was set to $K = 10$. We used the Adam optimizer [29] to minimize the loss while training with 300 epochs (learning rate $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^8$).

3.4 Melody conversion

Pitch-contour-preserving melody conversion generates a novel pitch sequence by searching for the optimal pitches $x_1, \ldots, x_N$ that maximize the following objective function J, which combines the contour model and the consistency model:

$$J = \ln p(x_1) + \sum_{n=2}^{N} \ln\left((1-\lambda)p(x_n) + \lambda q(x_n|x_{n-1})\right), \tag{4}$$

where $\lambda (0 \leq \lambda \leq 1)$ is a parameter to balance between the contour model $q(x_n|x_{n-1})$ and the consistency model $p(x_n)$.

The optimal pitches can be obtained by an efficient ($\mathcal{O}(N)$) algorithm based on dynamic programming. We show that the maximum value of the objective function J can be written as a recursive formula and therefore we can use dynamic programming. Let us denote the maximum value of J when $x_n = j$ by $\delta_n(j)$. We can derive

$$\begin{aligned} \delta_n(j) &= \max J(x_1, \ldots, x_{n-1}, x_n = j) && (5)\\ &= \max_i \left[\max J(x_1, \ldots, x_{n-1} = i) + r_n^{i,j}\right] && (6)\\ &= \max_i \left[\delta_{n-1}(i) + r_n^{i,j}\right], && (7) \end{aligned}$$

where

$$r_n^{i,j} = \ln\left((1-\lambda)p(x_n = j) + \lambda q(x_n = j|x_{n-1} = i)\right) \tag{8}$$

and $\delta_n(j)$ is recursively described by using $\delta_{n-1}(i)$. The resulting algorithm is shown in Alg. 1.

Algorithm 1 Melody conversion

Input: log probabilities $\ln p(x_n)$ and $r_n^{i,j}$

Initialization :

for $j = 1$ to 128 **do**
  $\delta_1(j) = \ln p(x_1 = j)$
  $\psi_1(j) = 0$
end for

Forward calculation :

for $n = 2$ to N **do**
  for $j = 1$ to 128 **do**
    $\delta_n(j) = \max_i[\delta_{n-1}(i) + r_n^{i,j}]$
    $\psi_n(j) = \text{argmax}_i[\delta_{n-1}(i) + r_n^{i,j}]$
  end for
end for

Backward tracking :

$x_N = \text{argmax}_j\, \delta_N(j)$
for $n = N-1$ to 1 **do**
  $x_n = \psi_{n+1}(x_{n+1})$
end for

Output: converted pitches $x_1, \ldots, x_N$

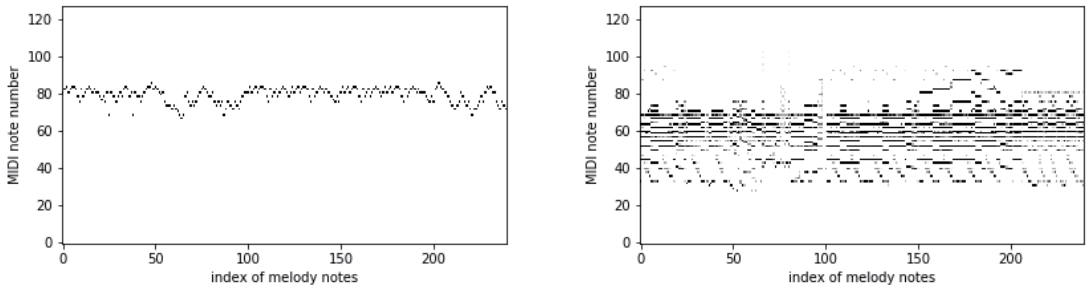

Figure 3. Examples of one-hot vectors of pitches in a melody (on the left) and the corresponding accompaniment vectors (on the right) in an excerpt of RWC-MDB-P-2001 No.97 from the RWC music database [28].

4. MELODY CONVERSION EXPERIMENT

4.1 Balancing between contour and consistency

In order to show contour-preserving melody conversion is an useful and a feasible method, we conducted an experiment to find whether our method can generate melody pitches that preserve the melodic contour and also maintain the consistency between the melody and the accompaniment. The possible failures are the two extreme cases. One is the case that generated pitches preserves the melodic contour but does not fit the accompaniment. The other extreme case is when the melodic contour is drastically changed by maintaining the consistency given by the consistency model. We investigated the results generated with different choices of parameter λ, which balances between the contour model and the consistency model in the objective function (eq. (4)).

4.2 Metrics to evaluate contour and consistency

Two quantitative measures were used for evaluation. To see how a generated melody follows the melodic contour of the source melody, we calculated the root mean squared difference (RMS) between the intervals of the generated pitches and the pitches of the source melody. The smaller the RMS is, the better the generated pitches preserve the melodic contour.

The other evaluation measure was to see how well a generated pitch fits the accompaniment. We used negative log-probability (NLP) calculated by using only the consistency model (last term in eq.(4)) divided by the length of the generated pitches. NLP measures how likely it is to observe a generated pitch given the trained model with the training data. NLP could be increased by forcing the pitches to preserve the melodic contour. The smaller the NLP is, the more likely the generated pitches maintain consistency with the accompaniment as it is in the training data.

4.3 Melodies used in experiment

We used MIDI files of popular music (RWC-MDB-P-2001 No. 1-30) from the RWC Music Database [28]. Vocal tracks containing melodies were manually detected. The remaining tracks except for drums were collected to prepare accompaniment vectors.

We trained parameters of RNNs by using all melody notes in RWC-MDB-P-2001 No. 1-20 (training data). To test the contour-preserved melody conversion, we used 8 bars from the beginning of verse A of RWC-MDB-P-2001 No. 21-30 (test data). For all combinations of 10 songs ($90 = 10 \times (10-1)$ possibilities excluding the conversion using the original accompaniment), one song was used for the melodic contour, and the other was used for the accompaniment.

The evaluation metrics RMS and NLP were calculated for every combination of songs, and statistics of the metrics (mean and standard deviation) were calculated. We tried 6 different settings for $\lambda = 0.0, 0.2, 0.4, 0.6, 0.8, \text{and } 1.0$.

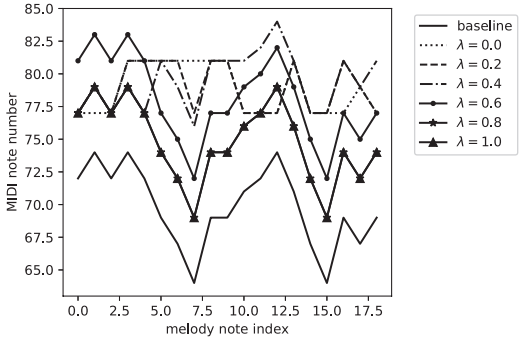

Figure 4. Pitches generating when converting pitches of RWC-MDB-P-2001 No. 27 to fit the accompaniment of RWC-MDB-P-2001 No. 26 with six different λ. The melodic contour follows that of the source melody (baseline) when λ is close to 1.0.

4.4 Melody conversion results

The results obtained when converting pitches of RWC-MDB-P-2001 No. 27 to fit the accompaniment of RWC-MDB-P-2001 No. 26 are shown in Figure 4. We observed that the melodic contour gradually followed the original melodic contour of the source melody more and more closely as λ was gradually increased from 0.0 to 1.0.

Mean and standard deviation of evaluation metrics (RMS and NLP) calculated when varying λ are shown in Figure 5. By setting λ around 0.4, the method generated pitches preserving the original contour within 3 semitones of difference (RMS: 2.46 ± 1.18, NLP: 2.00 ± 0.733). The extreme case of ignoring the melodic contour appeared when we set λ to 0.0 resulting in around 5 semitones dif-

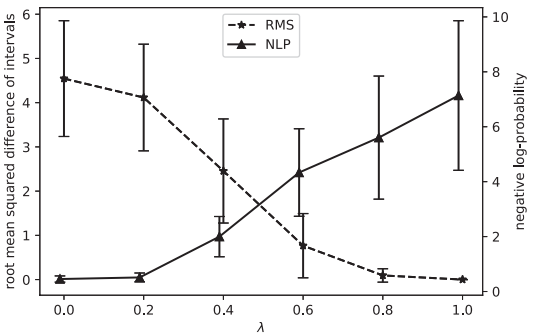

Figure 5. Mean and standard deviation of evaluation metrics (RMS and NLP) by varying λ to balance between preserving the melodic contour and maintaining the consistency between the melody and the accompaniment.

	NLP
proposed (λ=0.0)	0.454 ± 0.113
proposed (λ=0.4)	2.00 ± 0.733
proposed (λ=1.0)	7.13 ± 2.74
source melody	9.36 ± 3.58
human	7.01 ± 2.01

Table 1. Comparison of negative log-probability among proposed method (proposed), using the pitches of the source melody (*source melody*), and the human-composed melodies with the accompaniment (*human*).

ference (RMS: 4.54 ± 1.31). The other extreme case of ignoring the accompaniment was obtained by setting λ to 1.0, for which the NLP was the highest among all settings of λ (NLP: 7.13 ± 2.72). This highest NLP value was lower than the baseline result (NLP: 9.36 ± 3.58), which reuses the original pitches in the source melody.

5. DISCUSSION

We have found that contour-preserving melody conversion generates novel pitches based on an existing melodic contour. The experimental results indicate that a composer can balance between strictly preserving the melodic contour and maintaining consistency of melody and the accompaniment by setting λ.

There are issues we would like to deal with in future work. The first issue is the phrase boundaries within a melody. We discarded the length of melody notes and equally handled all intervals between consecutive notes. It seems more valuable not to preserve the interval between phrases that are separated far apart. The constraints of melodic contour between phrases could be relaxed by locally increasing λ for creating transition probabilities depending on the onset-to-onset duration of notes.

Rhythm is another important melodic structure we would like to consider. In order not to copy the rhythm but to inherit the rhythm structure, we need an abstracted form of rhythm. This could be done by estimating a tree structure behind the observed rhythm and using the tree to adapt to an arbitrary number of melody notes but still preserve the character of the rhythm.

The third issue is the listening test. While the NLP indicates how likely the generated pitches maintain consistency with the accompaniment, it depends on how well the consistency model is trained. The comparison of the negative log-probability of the generated pitches is shown in Table 1. The value for the proposed method at $\lambda = 0.4$ is lower than that for the human-composed melodies ($2.00 < 7.01$). Since the NLP values for the training and validation data were significantly lower (0.877 ± 0.398), the consistency model seemed to constrain too much on the pitches. This should be confirmed through listening test to obtain subjective evaluation.

Although using a stronger and better-trained consistency model is our future work, our contour-preserving melody conversion is useful for interactive computer-aided composition applications. The parameter λ provides a flexible control between the fully automatic composition and the manual composition. A composer can explore various possibilities of melodies, from the most recommended ones to the one strictly preserving the melodic contour of composer's manual composition.

6. CONCLUSION

We discussed contour-preserving melody conversion for the interactive computer-aided composition of melodies. We formalized the conversion as an optimization problem to maximize the log-probability of pitches with an algorithm based on dynamic programming. The experimental results indicated that composers could obtain various melodies by balancing between strictly preserving the melodic contour and maintaining the consistency between the melody and the accompaniment. In the future, we would like to generalize this approach to handle chords, rhythms, and musical structures.

Acknowledgments
This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan. We thank Yi-Hsuan Yang (Academia Sinica) for his valuable comments.

7. REFERENCES

[1] W. J. Dowling, "Melodic Contour in Hearing and Remembering Melodies," in *Rita Aiello (ed.) Musical Perception*. Oxford University Press, 1993, pp. 173–190.

[2] W. J. Dowling and D. S. Fujitani, "Contour, interval, and pitch recognition in memory for melodies," *Journal of the Acoustical Society of America*, vol. 49, no. 2, pp. 524–531.

[3] W. J. Dowling, "Scale and contour: Two components of a theory of memory for melodies," *Psychological Review*, vol. 85, no. 4, pp. 341–354.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] T. Kitahara, S. Giraldo, and R. Ramírez, "JamSketch: Improvisation Support System with GA-based Melody Creation from User' s Drawing," in *Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research*, 2017, pp. 352–363.

[6] F. Pachet, A. Papadopoulos, and P. Roy, "Sampling Variations of Sequences for Structured Music Generation," in *Proceedings of the International Society of Music Information Retrieval Conference*, 2017, pp. 167–173.

[7] J. Feulner and D. Hönel, "MELONET: Neural networks that learn Harmony-based melodic Variations," in *Proceedings of the International Computer Music Conference*, 1994, pp. 121–124.

[8] A. Papadopoulos, F. Pachet, P. Roy, and J. Sakellariou, "Exact sampling for regular and Markov constraints with belief propagation," in *G. Pesant (ed.) Constraint Programming 2015*, vol. 9255. Springer, Heidelberg, 2015, pp. 341–350.

[9] T. Collins and R. Laney, "Computer-Generated Stylistic Compositions with Long-Term Repetitive and Phrasal Structure," *Journal of Creative Music Systems*, vol. 1, no. 2, 2017.

[10] K. Ebcioğlu, "An expert system for harmonizing four-part chorales," in *Proceedings of the International Computer Music Conference*, 1986, pp. 447–449.

[11] H. Hild, J. Feulner, and W. Menzel, "A neural net for harmonizing chorales in style of J. S. Bach," in *Proceedings of Annual Conference on Neural Information Processing Systems*, 1991, pp. 267–274.

[12] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using BLSTM networks," in *Proceedings of the International Society of Music Information Retrieval Conference*, 2017, pp. 621–627.

[13] F. Pachet and P. Roy, "Musical Harmonization with constraints: A survey," *Constraints*, vol. 6, no. 1, pp. 7–19, 2001.

[14] M. Allan and C. K. I. Williams, "Harmonizing Chorales by Probabilistic Inference," *Advances in Neural Information Processing Systems*, vol. 17, pp. 25–32, 205.

[15] S. A. Raczyński, S. Fukayama, and E. Vincent, "Melody Harmonization with Interpolated Probabilistic Models," *Journal of New Music Research*, vol. 42, no. 3, pp. 223–235, 2013.

[16] K. Goel, R. Vohra, and J. Sahoo, "Polyphonic Music Generation by Modeling Temporal Dependencies Using a RNN-DBN," in *Proceedings of Artificial Neural Networks and Machine Learning*, 2014, pp. 217–224.

[17] Q. Lyu, Z. Wu, J. Zhu, and H. Meng, "Modeling high-dimensional sequences with LSTM-RTRBM: application to polyphonic music generation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 4138–4139.

[18] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[19] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating Music with Long-term Structure," in *Proceedings of the 7th International Conference on Learning Representations*, 2019, pp. 1–14.

[20] Y.-H. Chen, Y.-S. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by Transformers and groove modeling," in *Proceedings of the 21st International Society of Music Information Retrieval Conference*, 2020, pp. 756–763.

[21] F. Liang, M. Gotham, M. Johnson, and J. Shotton, "Automatic stylistic composition of Bach chorales with deep LSTM," in *Proceedings of the International Society of Music Information Retrieval Conference*, 2017, pp. 449–456.

[22] A. Liu, A. Fang, G. Hadjeres, P. Seetharaman, and B. Pardo, "Incorporating Music Knowledge in Continual Dataset Augmentation for Music Generation," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 1–3.

[23] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a Steerable Model for Bach Chorales Generation," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[24] N. Jaques, S. Gu, D. Bahdanau, J. Hernández-Lobato, R. E. Turner, and D. Eck, "SequenceTutor: Conservative Fine-Tuning of Sequence Generation Models with KL-Control," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[25] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proceedings of the 34th International Society of Music Information Retrieval Conference*, 2017, pp. 324–331.

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

[27] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings of the International Society of Music Information Retrieval Conference*, 2017, pp. 211–218.

[28] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: popular, classical, and jazz music databases," in *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002, pp. 287–288.

[29] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015, pp. 1–13.