IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol. 003, LIIR081
DOI: https://doi.org/10.52731/liir.v003.081

Generating Melodies from Melodic Outlines Towards an Improvisation Support System for Non-musicians

Tetsuro Kitahara [*]

Abstract

One promising approach for supporting non-musicians' music improvisation is to enable them to input the coarse shape of melodies they want to play. Therefore, drawing a *melodic outline* is adopted as an input method of existing improvisation support systems. However, generating melodies from melodic outlines with deep neural networks has not been fully exploited. In this paper, we report an attempt to generate melodies from melodic outlines using a *convolutional neural network*. Given a melodic outline, it is reduced with two convolution layers and then coverted to a sequence of notes with two deconvolution layers. Objective and subjective evaluations imply that our model with sufficient filter width and channels could generate melodies of moderate, close quality to human melodies.

Keywords: convolutional neural network, improvisation, melodic outline, melody generation

1 Introduction

Improvisation is an exciting but difficult form of musical performance because players must create melodies simultaneously while playing an instrument. Therefore, some researchers have been developing systems that enable non-musician users to enjoy improvisation with the computer's support. For example, while users play improvisation with *ism* [1], it auto-matically corrects their musically unnatural notes in real time based on anxs N-gram melody model. *Theremoscore* [2], a music keyboard in which a Peltier device is built on each key, presents the musical availability of each key by controlling its temperature. *coJIVE* [3], a collaborative improvisation system, controls the width of each virtual key (a target to hit with the baton) according to the musical availability. *JamSketch* [4], as the user draws a *melodic outline* on the screen, creates melodies along the outline drawn in real time. Such systems allow users to improvise without expert knowledge.

The main task in JamSketch is to create melodies from given melodic outlines. The original JamSketch creates melodies using a genetic algorithm (GA). The fitness function for the GA is designed as a linear combination of several factors, including the similarity

[*] Nihon University, Tokyo, Japan

to the given melodic outline and the N-gram-based melodic likelihood. However, the GA-based optimization takes a long time; therefore, to create melodies immediately, the system must stop the optimization before obtaining sufficiently optimized melodies.

There have been attempts to create improvisations using deep neural networks. JazzGAN [5] used a generative adversarial network (GAN), which generates content based on a random seed vector. Other models, such as MINGUS [6], BebopNet [7], and Jazz Transformer [8], are next-event-prediction models based on a long short-term memory (LSTM) and/or Transofmer-XL.

This paper reports an attempt to create melodies from melodic outlines with a *convolutional neural network* (CNN). CNNs are a promising approach for creating melodies. In fact, many researchers have used CNNs to create melodies [9, 10, 11]. A CNN can hierarchically reduce the temporal information in a melody by stacking convolution layers. Our CNN model reduces a melodic outline and a chord progression with two convolution layers and generates a melody (a sequence of notes) with two transposed convolution layers.

2 Method

2.1 Dataset

We used 96 pieces with Tonality Type of BLUES taken from the Weimer Jazz Database[1]. Each piece was divided into every 24 measures, and each division was treated as a separate piece. For data augmentation, we prepared editions where we moved up and down the melody's pitch by one octave for each piece. Each piece was transposed into the C-major or C-minor key in advance. In total, we prepared 539 pieces. We assigned half for training and the other half for testing.

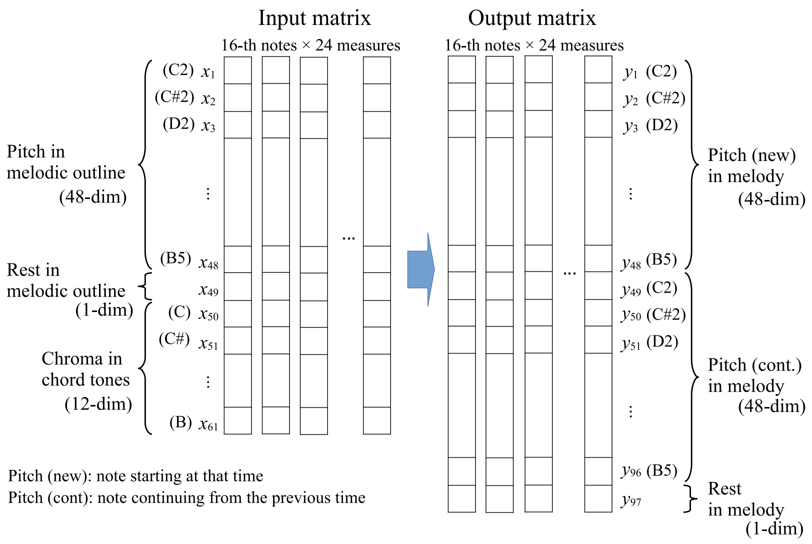
2.2 Data Representation

Since our task is to create a melody from a given melodic outline, the input is a melodic outline (and a chord progression), and the output is a melody. Both the input and output are temporal sequences, the time resolution of which is a 16th-note long. The details are illustrated in Figure 1. An input vector, $x[n]$, for time n is a combination of a one-hot vector representing the pitch of a melodic outline and a many-hot vector representing chord tones. An output vector, $y[n]$, for time n is a one-hot vector representing the pitch of a melody. In this vector, a note starting at the time n and a note continued from the previous time are represented by different elements to distinguish a long note (e.g., C—–) and a repetition of a short note (e.g., CCCC). An example is shown in Figure 2.

2.3 Preprocessing

Each MIDI sequence taken from the database (transposed into the C key in advance) is converted into sequences of $x[n]$ and $y[n]$. Because the time resolution is a 16th-note long, some notes may be reduced if the melody includes notes shorter than a 16th-note long. The pitch trajectory taken from the melody is smoothed via a moving average to obtain the melodic outline. The smoothing width is nine time steps (one measure has 16 time steps). After that, each data point is rounded to an integer because one-hot encoding is adopted for representing a melodic outline.

[1]https://jazzomat.hfm-weimar.de/dbformat/dboverview.html



Input matrix
16-th notes × 24 measures
Output matrix
16-th notes × 24 measures
Pitch in melodic outline (48-dim)
Rest in melodic outline (1-dim)
Chroma in chord tones (12-dim)
Pitch (new) in melody (48-dim)
Pitch (cont.) in melody (48-dim)
Rest in melody (1-dim)
Pitch (new): note starting at that time
Pitch (cont): note continuing from the previous time

Figure 1: Representation of input and output data

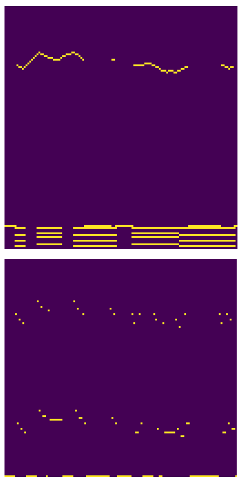


Figure 2: Example of inputs (upper) and outputs (lower)

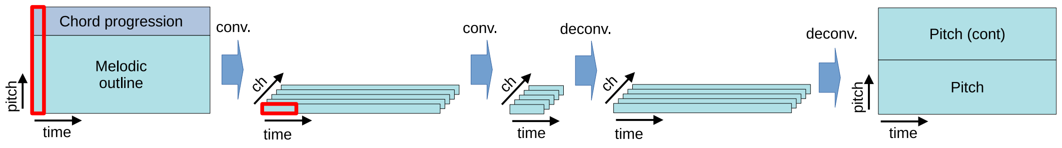


Figure 3: CNN architecture (Red rectangles represent filter sizes)

2.4 Learning a CNN

The architecture of our CNN is shown in Figure 3. Let's denote the size of an input matrix by M dimensions $\times$ N time steps (in the current implementation, $M = 48+1+12, N = 16 \times 24$). Because the first convolution layer aims at convolving pitch information for each time step, we set the filter size to $M \times 1$. The size of the output of this layer is $1 \times N \times C$, where C is the number of filter channels. The second convolution layer aims at reducing the temporal information, in which for every w beats ($w = 1,2,4$), the sequence of vectors is reduced to one vector. The filter size and stride are therefore $1 \times 4w$. The third and fourth layers are transposed convolution layers, corresponding to the second and first layers, respectively. The size of the output matrix is $M' \times N$ where $M' = 48+48+1$ in the current implementation. We introduced drop out (rate: 0.2) and batch normalization for each layer. We used the ReLU function for activation, except the sigmoid function is used at the last layer. This model was constructed and trained with TensorFlow (batch size: 16, epochs: 1000, optimization method: Adam).

2.5 Generating MIDI Data

From each melody (called a *given melody*) taken from the test dataset, a melodic outline is extracted via smoothing. Then, a new sequence of notes (called a *generated melody*) is generated with the trained CNN. From this, a MIDI sequence is generated. When the *Pitch (cont.)* element for a certain pitch at time n is hot but the *Pitch (new)* element for the same pitch at time $n-1$ is not hot in the output, we consider that the *Pitch (new)* element for the

Table 1: Participants' musical experience

	Playing	Improvisation	Composition
0 year	1	12	19
1 to 5 years	7	8	8
6 to 10 years	5	1	1
11 years or longer	17	9	2
Total	30	30	30

same pitch is hot instead at time n.

2.6 Evaluating Generated Melodies

2.6.1 Objective evaluation

The objective evaluation was performed by comparing generated and given melodies. First, we calculated the note-level concordance rate between them (i.e., how many notes in the generated melodies are the same as in the given melodies). Note that the concordance rate in general tends to be low because there may be more than one musically available melody for each melodic outline. Second, we calculated the dissimilarity of the pitch class histograms (called *pitch class dissimilarity*, PCD) between the generated and given melodies. We also calculated the dissimilarity of the pitch motion interval histograms (called *pitch motion dissimilarity*, PMD). The PCD and PMD were calculated as KL-divergences.

2.6.2 Subjective evaluation

We conducted a subjective evaluation using a Web-based crowdsourcing service. In advance, we selected 11 pieces with typical Blues chord progressions and collected the generated and given melodies for each piece (in total, we collected 22 melodies). Note that the given melodies are not purely human performances because they were quantized at the 16th-note level. The filter width w was 4, and the number of filter channels C was 1024. Each participant accessed our website on which eight melodies were selected randomly and listed in random order. Each participant listened to each melody and evaluated it via the following six criteria on a seven-grade scale (1–7):

- Overall quality
- Fewness of dissonant harmonies
- Variation between 1st and 2nd loops
- Originality
- Blues-likeness
- Human-likeness

The participants were 30 persons. The details of their musical experience are listed in Table 1.

3 Results

3.1 Objective Evaluation

The objective evaluation results are listed in Table 2.

The note-level concordance rate was about 20–25%. Considering that there is more than one musically appropriate melody for the same melodic outline, this low concordance

Table 2: Results of objective evaluation

w	1			2			4		
C	64	256	1024	64	256	1024	64	256	1024
Conc.	0.2477	0.2149	0.1927	0.2507	0.2047	0.1931	0.2320	0.1956	0.1979
PCD	0.3414	0.0878	0.0314	0.5012	0.1016	0.0294	2.2294	0.1863	0.0701
PMD	18.6550	0.0618	0.0332	14.3351	0.1022	0.0478	18.4874	0.1463	0.0710

Conc.: note-level concordance rate, PCD: pitch class dissimilarity, PMD: pitch motion dissimilarity

Table 3: Results of subjective evaluation (all participants)

		Overall	Dissonant	Variation	Originality	Blues	Human
Given	Mean	3.88	3.82	4.09	4.82	4.02	3.97
	SD	0.82	0.70	0.66	0.51	0.60	0.82
	Lowest	2.67	2.89	3.00	3.92	3.33	2.78
	Highest	5.15	5.15	5.18	5.64	5.18	5.36
	5 or higher	2	1	1	4	1	2
Generated	Mean	3.95	3.97	4.17	4.52	3.69	3.97
	SD	0.95	0.74	0.77	0.59	0.78	1.01
	Lowest	2.67	3.07	3.11	3.44	2.60	2.89
	Highest	5.86	5.29	5.86	5.57	5.29	6.14
	5 or higher	2	2	1	2	1	2

rate can be considered unavoidable. It was also seen that the concordance rate tends to be higher as the number of filter channels is lowered.

When the number of filter channels was 64, the pitch class dissimilarity and pitch motion dissimilarity were high. In fact, 60–70% of the generated notes were C or G; in addition, M2, m2, M7, and m7 account for 90% of the pitch motion intervals. These results show that the generated melodies from the model with the 64 filter channels are too monotonous.

When the number of filter channels was 1024, the pitch class dissimilarity and pitch motion dissimilarity were sufficiently low. In particular, the frequencies of D♯ (E♭) and F♯ (G♭) tend to increase as the number of filter channels increases. These notes are commonly used in Blues melodies, so we can consider that the models adequately learned characteristics in Blues melodies.

3.2 Subjective Evaluation

Some statistics of the subjective evaluation are listed in Table 3. Because the 30 participants rated eight melodies selected randomly, each melody had about 11 ratings on average (max: 19, min: 6). The ratings for each melody were averaged. The statistics in Table 3 were calculated melody-wise (for example, “5 or higher” represents how many melodies were rated 5 or higher on average). Moreover, the same statistics were calculated after removing the ratings by the participants with no (0-year) experience of improvisation (Table 4), in which about 6.5 ratings on average (max: 12, min: 3) were given for each melody.

The rating for the overall quality of the generated melodies is approximately 4 on average and is not sufficiently different from the given melodies. This means that generated melodies have moderate, close quality to human melodies.

The rating for the fewness of dissonant harmonies of the given melodies was lower than that for the generated melodies. This could be because the given melodies were playback of quantized MIDI transcriptions with the piano sound. This could cause a dissonant im-

Table 4: Results of subjective evaluation (non-experienced participants removed)

		Overall	Dissonant	Variation	Originality	Blues	Human
Given	Mean	4.02	3.68	4.19	4.78	4.07	4.10
	SD	0.92	0.76	0.60	0.65	0.56	0.91
	Lowest	2.67	2.40	3.14	3.83	3.29	2.67
	Highest	5.50	4.88	5.13	5.57	5.00	5.43
	5 or higher	2	0	1	5	1	2
Generated	Mean	4.05	3.94	4.33	4.59	3.77	4.17
	SD	1.20	0.90	0.89	0.67	0.83	1.02
	Lowest	2.57	2.63	3.00	3.29	2.33	2.86
	Highest	6.00	5.25	5.60	5.40	5.20	6.00
	5 or higher	3	3	3	4	1	3

pression because almost all improvisations in this dataset were originally performed on a wind instrument (e.g., trumpet, saxophone).

The Blues-likeness of the generated melodies was lower than that of the given melodies. Although the generated melodies contain Blue notes, we need to analyze more profound reasons why those are less Blues-like.

3.3 Examples

We show an example of the generated melodies (Figure 4), which implies the following:

- The melody generated with w of 1 contains many rests and short phrases (mostly about a quarter-note long). This is because the melody generation process is executed independently for every quarter note in this model.

- The melody generated with w of 4 contains fewer rests and longer phrases. Also, the melody has a musically natural rhythm because it contains various note durations such as half and quarter notes as well as 16th notes. In the fifth to sixth measures, it has a long rest. This is because there is no melodic outline in this interval since the given melody also has no notes.

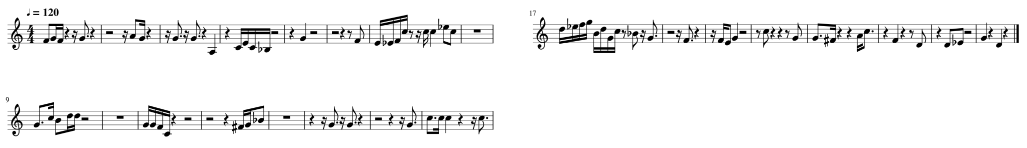
- Comparing Figure 4 (b) and (c), we can see that the overall pitch movements of these melodies are close. This means that the generated melody reflects the given melodic outline because the melodic outline was generated by smoothing Figure 4 (c)'s melody.

- The melody shown in Figure 4 (b) uses the E♭ and B♭ notes as well as the E and B notes. This means that melodies were generated based on the Blues note scale (which has the III♭, V♭, and VII♭ notes and the notes in the major scale).

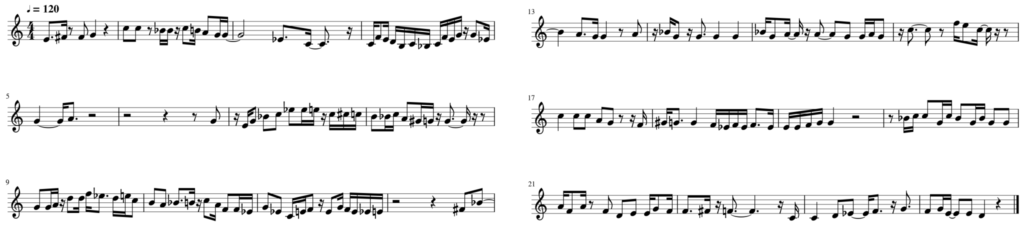
Other examples are available on our web page: `https://bit.ly/3tmDt5a`[2]

In addition, the demonstration web page, in which users can try to draw a melodic outline and obtain a melody, is available at: `https://bit.ly/3zB2Tja`

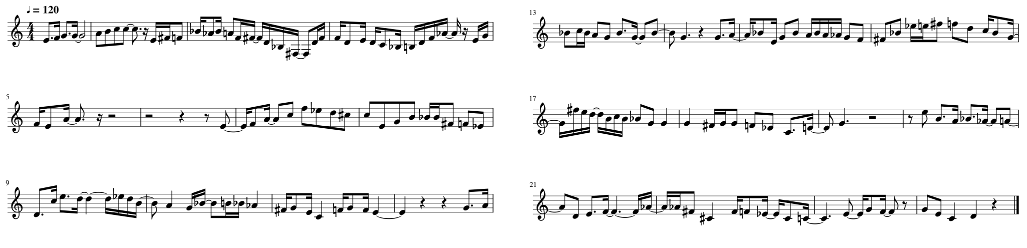
[2]If you cannot access this page, please try it again after logging out from all Google accounts.



(a) Generated melody ($w = 1, C = 1024$)



(b) Generated melody ($w = 4, C = 1024$)



(c) Given melody

Figure 4: An example of genereated and given melodies

4 Conclusion

This paper reported an attempt to generate melodies from melodic outlines with a CNN. Given a melodic outline, it is reduced with two convolution layers and then a melody (a sequence of notes) is generated with two deconvolution layers. Experimental results with different filter widths and different numbers of channels showed that too small filter widths and/or too small number of channels generate musically inappropriate melodies. Subjective evaluation showed no clear differences between the quality of generated melodies and human melodies, although the human melodies were quantized in advance.

However, the melodic outlines used in the experiment are not actually drawn by non-musicians; they were obtained by smoothing the pitch trajectories of melodies included in the Weimer Jazz Database. We will plan to conduct experiments with melodic outlines drawn by non-musicians as well as futher improvment of our model.

Acknowledgments

This work was supported by JSPS Kakenhi 22H03711 and 21H03572.

References

[1] Katsuhisa Ishida, Tetsuro Kitahara, and Masayuki Takeda. ism: Improvisation supporting system based on melody correction. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 177–180, June 2004.

[2] Homei Miyashita and Kazushi Nishimoto. Theremoscore: A new-type musical score with temperature sensation. In *Int'l Conf. New Interface for Musical Expression*, pages 104–107, 2004.

[3] Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers. coJIVE: a system to support collaborative jazz improvisation. Technical Report AIB-2007-04, Aachener Informatik-Berichte RWTH Aachen, Department of Computer Science, http://www.informatik.rwth-aachen.de/go/id/lolj/lidx/1/file/47944, 2007.

[4] Tetsuro Kitahara, Sergio Giraldo, and Rafael Ramírez. JamSketch: Improvisation support system with GA-based melody creation from user's drawing. In *Proceedings of the 2017 International Symposium on Computer Music Multidisciplinary Research*, pages 352–363, Matoshinhos, Portugal, 2017.

[5] Nicholas Trieu and Robert M. Keller. JazzGAN: Improvising with generative adversarial networks. In *Proceedings of the 2018 Workshop on Musical Metacreation (MUME 2018)*, 2018.

[6] Vincenzo Madaghiele, Pasquale Lisena, and Raphael Troncy. MINGUS: Melodic improvisation neural generator using seq2seq. In *Proceedings of the 22nd International Society for Music Information Retrieval (ISMIR 2021)*, pages 412–419, 2021.

[7] Shunit Haviv Hakimi, Nadav Bhonker, and Ran El-Yaniv. BebopNet: Deep neural models for personalized jazz improvisations. In *Proceedings of the 21st International Society for Music Information Retrieval (ISMIR 2020)*, pages 828–836, 2020.

[8] Shih-Lun Wu and Yi-Hsuan Yang. The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through guantitavie measures. In *Proceedings of the 21st Information Society for Music Information Retrieval (ISMIR 2020)*, 2020.

[9] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of Int'l Soc. for Music Information Retrieval Conf.*, pages 324–331, 2017.

[10] Kosuke Nakamura, Takashi Nose, Yuya Chiba, and Akinori Ito. A symbolic-level melody completion based on a convolutional neural network with generative adversarial learning. *Journal of Information Processing*, 28:248–257, 2020.

[11] Yongjie Huang, Xiaofeng Huang, and Qiakai Cai. Music generation based on convolution-LSTM. *Computer and Information Science*, 11(3):50–56, 2018.