DEEP COMPOSER: DEEP NEURAL HASHING AND RETRIEVAL APPROACH TO AUTOMATIC MUSIC GENERATION

Brandon Royal, Kien Hua, Brenton Zhang

University of Central Florida
brandonroyal@knights.ucf.edu, kienhua@cs.ucf.edu, brenton.a.zhang@gmail.com

ABSTRACT

With recent advances in artificial intelligence, recurrent neural networks have successfully generated pleasing melodies; however, they have struggled to create a full song that has structure, theme, and uniqueness. To overcome this limitation, we introduce Deep Composer, a music generation system based on music retrieval using deep neural hashing to encode the music building blocks. Music generation is performed by using the current music segment to retrieve the next segment from the database until the whole composition is complete. We present performance comparisons with multiple recent state-of-the-art music generation methods to show that the songs generated by Deep Composer are unique, musically pleasing and contain more structure and theme features like that of a professionally composed song.

Index Terms— Music Generation, Music Retrieval, Hashing, Deep Learning

1. INTRODUCTION

Music has been called the "universal language of mankind" because it is a diverse art form that connects people all over the world by allowing them to share their common appreciation for pleasing harmonies and styles. In recent years, music has become a driving force in fields ranging from entertainment and economics to music therapy for both physical and mental health. However, music's benefits are limited because songs previously have only successfully been created by experts.

To eliminate the "experts-only" barrier to music composition, researchers have explored various methods for computer-generated music. Rule-based methods [1, 2, 3] have attempted to generate new songs based on defined constraints that focus on following music theory principles. These methods often compose a song with structure, but the results lack creativity because of the user defined constraints. These methods also only work with specific styles due to the complexity of music. Concatenation-based methods [4, 5, 6, 7, 8] compose a song by piecing together tiny segments from many existing songs to create a new unique song. Given the enormous amount of music in the public domain, an endless number of new original songs could be generated with the concatenation-based approach. These methods also allow for more creativity and are not limited to a specific style of music. The challenge is how to create a song that is complete with a theme and structure. Even deep learning based approaches [9, 10, 11, 12, 13], which use recurrent neural networks to generate the next timestep of a song, have failed to compose a complete, unique song that has both a clear theme and structure.

To address the above limitations of current music generation research, we introduce Deep Composer, a concatenation-based music generation system capable of creating a complete song with theme, structure, and uniqueness. Deep Composer is composed of multiple long short-term memory networks (LSTMs) [14], a type of Recurrent Neural Network, each of which is focused on creating a different structural component of a song. We also introduce a new two-phase music segmentation approach that breaks a song into tiny segments to ensure uniqueness and groups these segments into one of the following structural groups: beginning, middle, end, and final. Each LSTM in Deep Composer learns a hash encoding for the musical segments in the forwards and backwards direction. During music generation, Deep Composer concatenates musical segments by retrieving pairs with the minimal Hamming distance between their encodings. The result is a complete, unique song with a distinctive theme and structure.

2. HASH BASED MUSIC GENERATION

2.1. Two-Phase Music Segmentation

A song has many different features that can be considered during segmentation. For Deep Composer, we want the segmentation process to consider the structure of the song while minimizing the segment length so that we can generate unique songs that do not resemble any of the original songs. Some segmentation methods, such as the ones used in [4] and [15], divide a song into one or two measure segments. These methods extract segments small enough to prevent significant overlap with the original songs, but the segmentation is based on timesteps and does not consider the structure of each song. Other methods perform segmentation based on the

978-1-7281-1331-9/20/$31.00

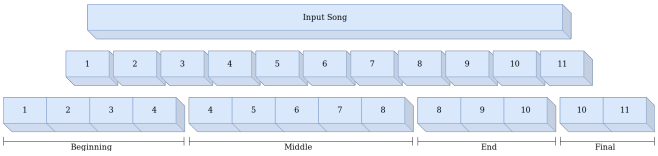

Fig. 1. Example segmentation of a song using our Two-Phase segmentation approach.

structure of the song [16, 17, 18]. These structure-focused approaches to segmentation result in just a few long segments for each song. Using long segments for music generation would greatly limit not only the uniqueness of each individual song but also the total number of new composable songs. To solve this problem, we propose a two-phase music segmentation process that considers the structure of each song while minimizing the segment length.

2.1.1. Phase One: Time-Based Music Segmentation.

In the first phase of segmentation, we follow the results of [4], which suggest that music generated from segments of one or two measure durations was more likable and natural to the listener. We choose to use segment lengths of one measure to maximize the uniqueness of the generated songs.

The measure length of a song is determined by its time signature. For example, the time signature $\frac{4}{4}$ indicates that a measure will contain 4 beats (top number) and that each beat has a quarter note value (bottom number). Using this information, we can define the total number of measures, and therefore the total number of time-based segments, in a song as $\mathbf{s} = \sum \frac{\mathbf{n}_t}{\mathbf{b}_t}$ where $\mathbf{n}_t$ is the total beats in the current time signature group and $\mathbf{b}_t$ is the number of beats per bar for the current time signature group. Since the time signature may change throughout a song, we sum over each time signature group. The result of this segmentation process is a segment vector of length $\mathbf{s}$.

2.1.2. Phase Two: Structure-Focused Music Segmentation.

In the second phase, we group the time-based segments of each song by their structural relative location. To ensure that we can generalize our approach for any set of songs, we simply define a song as having four structural components: a beginning, middle, end, and final section. A distinction is made between the end of a song and the final section because the last few measures of a song typically have a distinct theme that concludes the song and creates a pleasing ending.

We define four variables: $\mathbf{l}_b$, $\mathbf{l}_m$, $\mathbf{l}_e$, and $\mathbf{l}_f$, which represent the number of time-based segments in each structural group. Since the length of a song can vary greatly, we make $\mathbf{l}_b$, $\mathbf{l}_m$, and $\mathbf{l}_e$ proportional to the current song. The final section of a song is much smaller so we make $\mathbf{l}_f$ constant across all songs in a given datastet. Given a time-based segment vector $\mathbf{v}$, having a length of $\mathbf{s}$, the final section of a song is defined as the last $\mathbf{l}_f$ segments in the vector. The remaining segments are divided among the the beginning, middle, and end structure groups by using three multipliers, $\mathbf{m}_b$, $\mathbf{m}_m$, and $\mathbf{m}_e$ where $\mathbf{m}_b + \mathbf{m}_m + \mathbf{m}_e = 1$. The length of each remaining structure group is found using the following group of equations: $\mathbf{l}_b = (\mathbf{s} - \mathbf{l}_f) \times \mathbf{m}_b$, $\mathbf{l}_m = (\mathbf{s} - \mathbf{l}_f) \times \mathbf{m}_m$, and $\mathbf{l}_e = (\mathbf{s} - \mathbf{l}_f) \times \mathbf{m}_e$.

The first $\mathbf{l}_b$ segments are placed into the beginning group, the next $\mathbf{l}_m$ segments are placed into the middle group, and the remaining $\mathbf{l}_e$ segments are placed into the end group. For the segmentation phase, the number of final segments and $\mathbf{m}_b$, $\mathbf{m}_m$, and $\mathbf{m}_e$ are all user defined, tunable parameters.

Even with technical analysis, the exact location of the structural boundaries within a song is subjective by nature [17]. To account for this and to help create transitions between structures, we introduce an additional overlap variable, $\mathbf{o}_s$, into the structure-segmentation phase. After the structure groups are created, the last $\mathbf{o}_s$ time-based segments in each structure group are treated as members of the following structure group. For example, if $\mathbf{o}_s$=2, the last two time-based segments in the beginning group are also included in the middle group. Fig. 1 shows a visualization of the entire segmentation process.

2.2. Segment Hash Encoding

Both [7] and [15] show how hashing can be used to quickly find good candidate segments during music generation, however these works, along with other concatenation based generation methods, fail to address one of the most important aspects of song composition: structure. We propose a new hash-based encoding method which assigns each segment a set of structure-focused hash codes, in the forwards direction, denoted $\mathbf{H}_f$, and backwards direction, denoted $\mathbf{H}_b$.

In each hash code set, a hash code is defined for each structural group (beginning, middle, end, final). Therefore, the forwards and backwards sets are defined as $\mathbf{H}_f = \{\mathbf{h}_{f_b}, \mathbf{h}_{f_m}, \mathbf{h}_{f_e}, \mathbf{h}_{f_f}\}$ and $\mathbf{H}_b = \{\mathbf{h}_{b_b}, \mathbf{h}_{b_m}, \mathbf{h}_{b_e}, \mathbf{h}_{b_f}\}$:

We define the composability of a pair of segments (s_i, s_j), to focus on not only whether s_j is a suitable candidate to follow s_i based on musical features but also whether s_j is a suitable candidate to follow s_i based on its structural features. Considering structural features has many advantages; for example, two segments may sound pleasing together and be theoretically fitting from a music theory analysis, but if the second segment is from the final part of a song, it will make the overall song sound abrupt or choppy. The same issue arises if a segment which starts building a musical idea is placed last in the generated song. Therefore, based on this new definition of the composability of a pair of segments, the hash codes are assigned in a way such that the Hamming distance between the forward hash code of a given structure group of s_i and the backwards hash code of the same structure group of s_j is

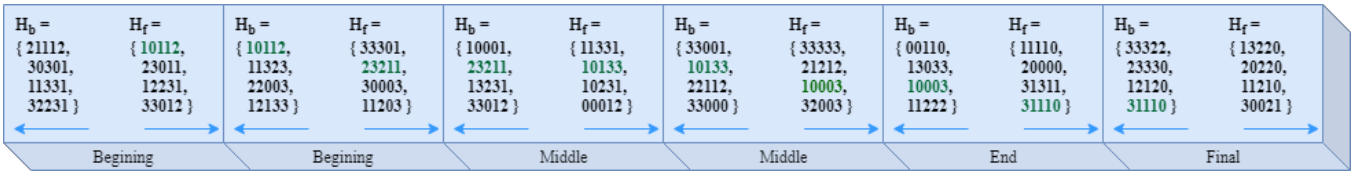

Fig. 2. Segment concatenation using structure-focused hash codes.

minimized if the pair musically and structurally fit. The Hamming distance is maximized if the pair does not fit musically or structurally. During music generation, the hash code used for determining candidacy is selected based on which structure (beginning, middle, end, or final) is currently being built. Fig. 2 shows a visualization of the concatenation of segments based on the hash codes.

2.3. Multi-LSTM System: Deep Composer

To learn our structured focus hash encoding for each music segment, we designed a LSTM based network which contains forwards and backwards LSTMs working side-by-side. The forward LSTM takes a pair of segments in the forwards directions and the backward LSTM takes the same segments but in backwards direction. We calculate the inner product loss using both the result of the forward and backward LSTMs and update our network's weights. Fig. 3 shows a visual representation of our network.

The input to our network is a pair of musical segments with either a composable or non-composable label. As discussed in the previous section, we want to learn which pairs of segments have a high composability. Two segments are defined as having a high composability if they are found together in the training data or if, based on their extracted features, they could possibly be composable. We determine if pairs are possibly composable by analyzing the known composable pairs and finding if another given pair of segments has similar features within a threshold. All other pairs, which are not found to be composable, are treated as negative training examples.

We train our network in two phases. First, we train our network for time step prediction. In this phase, we train a single network to predict the features of the next segment of a song in the forwards and backwards direction. Here we do not consider the structure, but rather only train using the measure long segments extracted during the time based segmentation. A list of the features extracted for each segment and their description can be found in Table 1.

We then use these pre-trained weights to train multiple networks, one for each structural group (beginning, middle, end, final). In this phase, we add the distinction that, for a given structural group network, if the pair of segments are not both from the same structural group, they are treated as a negative pair. We use a similar hashing approach to [19], which

Table 1. Musical Segment Feature Representation

Feature	Description
Chord Class	one-hot chord representation
Melody Articulation	one if the melody is articulated
Melody Octave	one-hot octave representation
Melody Pitch	one-hot pitch representation
Whole Note Count	total number of whole notes
Half Note Count	total number of half notes
Quarter Note Count	total number of quarter notes
Eighth Note Count	total number of eighth notes
Sixteenth Note Count	total number of sixteenth notes

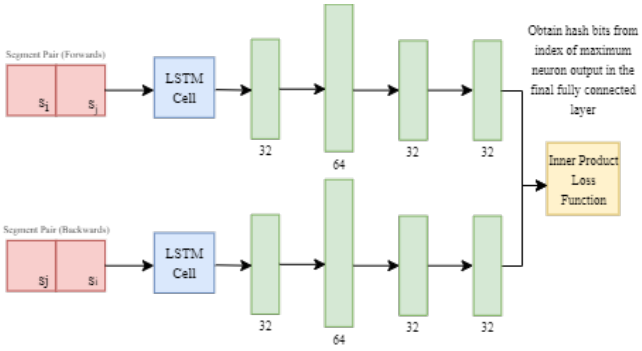

Fig. 3. Network Architecture

learns multiple independent multi-dimensional embeddings of the input using the last hidden layer of the network. In this phase, the network learns to minimize the Hamming distance between composable pairs, and maximize the Hamming distance between non-composable pairs. The result is four distinct networks, which make up our system, Deep Composer.

Deep Composer is generalizable and not limited by the training data. After training, Deep Composer generates the hash encodings for additional music segments from the database. The system generates music by randomly selecting a starting segment and then finding a next segment that has a high composability with the next. Deep Composer's flexibility allows it to generate music of all different lengths.

3. EXPERIMENTS AND RESULTS

3.1. Dataset

To evaluate Deep Composer, we use a publicly available, cleaned version of the Nottingham dataset [1]. This dataset consists of 1034 American and British folk tunes represented in the MIDI file format. For our experiments, we select 496 songs from the dataset by only considering songs that are in a major key and in either $\frac{2}{4}$ time or $\frac{4}{4}$ time. The selected song are transposed into C Major. We chose the Nottingham dataset because of its frequent use in recent music generation research [9, 15, 20, 10, 21]. Our approach is generic so that it can be applied to different music datasets and generate all styles of music.

We do time-based segmentation using one measure for each segment. Similar to [15], we use a two beat overlap between segments. In total, we extract 11,886 distinct time-based segments. Structure-focused segmentation is performed with the following values for each parameter: $\mathbf{l}_f$=1, $\mathbf{m}_b$=0.4, $\mathbf{m}_m$=0.4, $\mathbf{m}_e$=0.2, and $\mathbf{o}_s$=1. After structural-segmentation, the beginning, middle, end, and final structure groups consist of 6,755 distinct segments, 8,620 distinct segments, 5475 distinct segments, and 322 distinct segments respectively.

3.2. Survey Overview

To properly evaluate Deep Composer, we created two surveys in which we compare our system to multiple recent state-of-the-art approaches. In the first survey, we compare our method to another music segment concatenation method DSHL, proposed by Joslyn *et al.* [15]. In the second survey we compare our method to multiple non-concatenation based methods [9, 10, 12, 22]. We chose to do two separate surveys, because in the first, we compare our method to another concatenation based method so we can analyze the differences when using the same segment database. In the second survey, the songs are various lengths and the methods are largely different, so we only focus on comparing musical quality. For example, in [9], the first few measures of the melody are copied from the original song and the whole bass is from the original song. We perform the second survey to ensure that our method is superior in relation to not only concatenation based music generation but also all other types of music generation. In both surveys, all methods use the Nottingham dataset.

The first survey is composed of both individuals with musical experience and individuals without musical experience. We defined someone as having musical experience if: 1) the individual has received a formal musical education and 2) the individual has been playing a musical instrument for longer than five years. In total, we surveyed n = 20 people (10 with musical experience, 10 without musical experience). The first survey consisted of 9 sample songs generated by Deep Composer and 6 sample songs generated by DSHL. All songs consisted of either 32 or 33 musical segments extracted from the Nottingham dataset and each segment consisted of 4 beats. To prevent bias, the order of the songs was randomized and a blind study approach was taken.

[1] https://github.com/jukedeck/nottingham-dataset

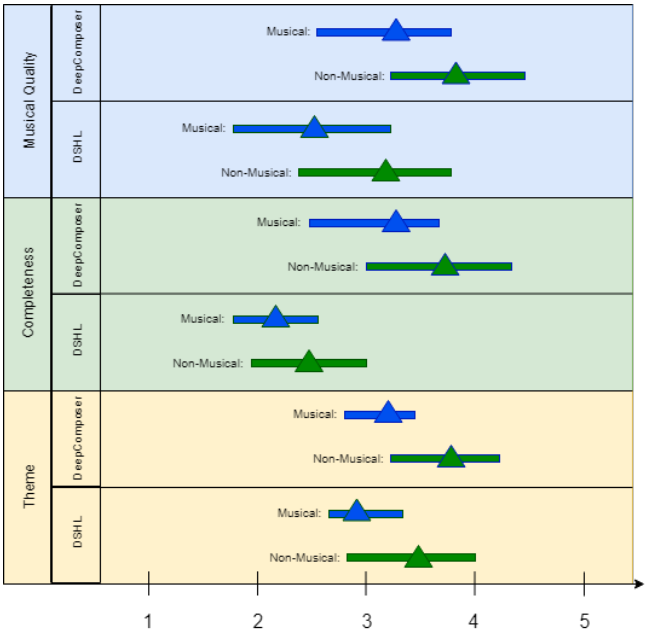

Fig. 4. Average Score for Each Method Given by Individuals.

We defined three metrics for participants to rate the songs: Theme, Completeness, and Musical Quality. The metrics were defined the same for all participants by supplying a "guidelines" sheet before and during the survey. For each song, We told the participants to give each metric a score using a scale from 1 (low) to 5 (high).

In the second survey, we surveyed an additional n = 14 people (7 with musical experience, 7 without musical experience). In this survey we randomly selected 2 songs from our method and 2 from each of the other methods. In this survey the songs ranged from 20 seconds long to over a minute. Some of the methods only generated a melody without a bass. Because of this, we asked participants to only rate each song based on the musical quality using a scale from 1 (low) to 5 (high). Comparing theme and completeness would not be fair in this survey, as some of the pieces were very short.

3.3. First Survey Results

The first survey's results for individuals with musical experience and individuals without musical experience are showed in Fig. 4. This figure shows the range of each metric along with the average across the songs for each method. The blue

Table 2. Average Musical Quality Score for Each Song in the Second Survey

Method	Average Musical Quality
Deep Composer (song 1)	4.14
Deep Composer (song 2)	**4.57**
Medeot *et al.* (song 1)	2.00
Medeot *et al.* (song 2)	2.86
Johnson *et al.* (song 1)	2.86
Johnson *et al.* (song 2)	3.07
Chen *et al.* (song 1)	3.21
Chen *et al.* (song 2)	3.93
Boulanger *et al.* (song 1)	2.71
Boulanger *et al.* (song 2)	3.07

line corresponds to the results from musical participants and the green line corresponds to the results from non-musical participants. Deep Composer not only had a higher average across all categories, but also had the highest rating for each.

The first survey's results for both individuals with musical experience and individuals without musical experience follow a similar trend. Songs generated by Deep Composer had higher average ratings across all metrics, as compared to songs generated by the baseline method. An increase of over 50% was observed for Deep Composer when comparing the average Completion scores across all songs. This significant increase was reinforced by participants frequently reporting that songs generated by our method had a clear structure and pleasing ending. We observed an increase of over 24% for Deep Composer when comparing the average Musical Quality scores across all songs. This increase is significant for arguably the most important category since the average music listener focuses more on if the music is pleasing to listen to rather than the technical aspects of the song. One interesting observation is found in the Theme category, in which Deep Composer only had an increase over the baseline of 7.72% (musical individuals' scores) and 11.40% (non-musical individuals' scores). This increase is arguably still significant. Based on participant verbal feedback, we hypothesize that, although Deep Composer consistently generates songs with a more clear theme, the theme is subject to frequent variation as song structure transitions. As a future solution, we could consider the previous n segments while generating a song to capture theme between each song structure.

3.4. Second Survey Results

In the second survey we focused only on musical quality. Musical quality is an important aspect since music is listened to for enjoyment the majority of the time. The second survey results also favored our method over all other methods. Both of the songs generated with our method had a higher average musical quality score than any of the other songs. Table 2 shows the results of the second survey.

3.5. Diversity

In addition to our subjective, human evaluation of the generated songs, we performed an objective analysis of the songs by considering how unique each song is compared to the original songs in the dataset. To evaluate the uniqueness of the songs generated by Deep Composer, we define a new metric: diversity. Diversity is calculated by the following equation where $\mathbf{n}_d$ is the number of distinct songs that the segments used for generation came from and $\mathbf{n}_s$ is the total number of segments in the generated song: $diversity = \frac{\mathbf{n}_d}{\mathbf{n}_s}$

To test the diversity of our method, we generate 100 songs consisting of 32 segments, and calculate the diversity of each. Fig. 5 shows the diversity for each song. Although the diversity score shows that multiple segments come from the same original song, further analysis shows that this is often because of the same segment appearing multiple times in a generated song. Repetition is a common technique used by composers to give a song structure and theme, so the multiple appearance of certain segments may be a direct result of a stronger structure and theme in the songs generated by Deep Composer.

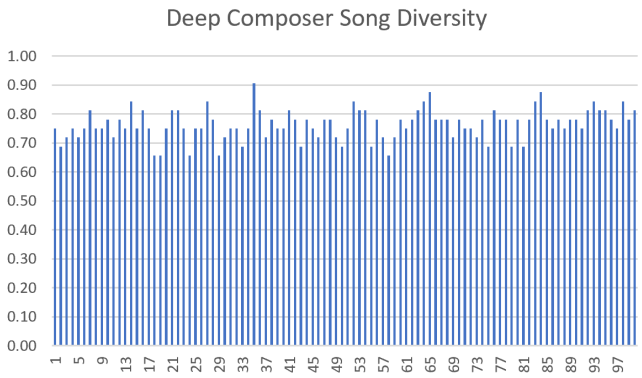

Fig. 5. Diversity of songs generated by Deep Composer.

3.6. Example Generation

Because of space limitations, full musical scores, along with mp3 files, of songs generated by Deep Composer are provided on DropBox [2].

4. CONCLUSION AND FUTURE WORK

In this paper, we introduce a novel neural system that learns hash encodings for musical segments. Music composition is performed by using the current segment's hash code as a query to retrieve the next composable segment from the database. Our experiments compare Deep Composer to multiple state-of-the-art methods, and show that songs generated

[2] https://www.dropbox.com/sh/3h0z1doolo07rqr/AAAvxj9cQRCvrPb0PMNFHWdja?dl=0

by Deep Composer are unique and perceived by participants to have important qualities, such as a clear theme, clear structure, pleasing transitions between structures, and a pleasing ending. In the future, we plan to investigate mapping the semantic relationship between music and the other modalities using a hash-based approach.

5. REFERENCES

[1] David Cope and Melanie J Mayer, *Experiments in musical intelligence*, vol. 12, AR editions Madison, 1996.

[2] Raymond P Whorley and Darrell Conklin, "Music generation from statistical models of harmony," *Journal of New Music Research*, vol. 45, no. 2, pp. 160–183, 2016.

[3] Derek Wells and Hala ElAarag, "A novel approach for automated music composition using memetic algorithms," in *Proceedings of the 49th Annual Southeast Regional Conference*. ACM, 2011, pp. 155–159.

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

[5] Jérôme Nika, Marc Chemillier, and Gérard Assayag, "Improtek: introducing scenarios into human-computer music improvisation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 4, 2016.

[6] Francois Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[7] Aleksey Charapko and Ching-Hua Chuan, "Indexing and retrieving continuations in musical time series data using relational databases," in *2014 IEEE International Symposium on Multimedia*. IEEE, 2014, pp. 341–346.

[8] Heng-Yi Lin, Yin-Tzu Lin, Ming-Chun Tien, and Ja-Ling Wu, "Music paste: Concatenating music clips based on chroma and rhythm features.," in *ISMIR*. Citeseer, 2009, pp. 213–218.

[9] Ke Chen, Weilin Zhang, Shlomo Dubnov, Gus Xia, and Wei 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.

[10] Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, S Abdalla, M Selvi, E Rex, and K Webster, "Structure net: Inducing structure in generated melodies," in *The 19th International Society for Music Information Retrieval Conference*, 2018.

[11] Hang Chu, Raquel Urtasun, and Sanja Fidler, "Song from pi: A musically plausible network for pop music generation," *arXiv preprint arXiv:1611.03477*, 2016.

[12] Daniel D Johnson, "Generating polyphonic music using tied parallel networks," in *International conference on evolutionary and biologically inspired music and art*. Springer, 2017, pp. 128–143.

[13] Michael Chan, John Potter, and Emery Schubert, "Improving algorithmic music composition with machine learning," in *Proceedings of the 9th International Conference on Music Perception and Cognition, ICMPC*, 2006.

[14] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Kevin Joslyn, Naifan Zhuang, and Kien A Hua, "Deep segment hash learning for music generation," *arXiv preprint arXiv:1805.12176*, 2018.

[16] Rongshu Sun, Jingiing Zhang, Wei Jiang, and Yuexin Hu, "Segmentation of pop music based on histogram clustering," in *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2018, pp. 1–5.

[17] Cheng-i Wang, Gautham J Mysore, and Shlomo Dubnov, "Re-visiting the music segmentation problem with crowdsourcing.," in *ISMIR*, 2017, pp. 738–744.

[18] Gabriel Sargent, Frédéric Bimbot, and Emmanuel Vincent, "Estimating the structural segmentation of popular music pieces under regularity constraints," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 344–358, 2017.

[19] Kevin Joslyn, Kai Li, and Kien A Hua, "Cross-modal retrieval using deep de-correlated subspace ranking hashing," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM, 2018, pp. 55–63.

[20] Sarthak Agarwal, Vaibhav Saxena, Vaibhav Singal, and Swati Aggarwal, "Lstm based music generation with dataset preprocessing and reconstruction techniques," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 455–462.

[21] Y Cem Subakan and Paris Smaragdis, "Diagonal rnns in symbolic music modeling," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 354–358.

[22] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.