electronics

MDPI

Article

Melody Extraction and Encoding Method for Generating Healthcare Music Automatically

Shuyu Li, Sejun Jang and Yunsick Sung *

Department of Multimedia Engineering, Dongguk University-Seoul, Seoul 04620, Korea; lishuyu@dongguk.edu (S.L.); sejun@dongguk.edu (S.J.)

* Correspondence: sung@dongguk.edu; Tel.: +82-2-2260-3338

Received: 28 August 2019; Accepted: 29 October 2019; Published: 31 October 2019

Abstract: The strong relationship between music and health has helped prove that soft and peaceful classical music can significantly reduce people's stress; however, it is difficult to identify and collect examples of such music to build a library. Therefore, a system is required that can automatically generate similar classical music selections from a small amount of input music. Melody is the main element that reflects the rhythms and emotions of musical works; therefore, most automatic music generation research is based on melody. Given that melody varies frequently within musical bars, the latter are used as the basic units of composition. As such, there is a requirement for melody extraction techniques and bar-based encoding methods for automatic generation of bar-based music using melodies. This paper proposes a method that handles melody track extraction and bar encoding. First, the melody track is extracted using a pitch-based term frequency–inverse document frequency (TFIDF) algorithm and a feature-based filter. Subsequently, four specific features of the notes within a bar are encoded into a fixed-size matrix during bar encoding. We conduct experiments to determine the accuracy of track extraction based on verification data obtained with the TFIDF algorithm and the filter; an accuracy of 94.7% was calculated based on whether the extracted track was a melody track. The estimated value demonstrates that the proposed method can accurately extract melody tracks. This paper discusses methods for automatically extracting melody tracks from MIDI files and encoding based on bars. The possibility of generating music through deep learning neural networks is facilitated by the methods we examine within this work. To help the neural networks generate higher quality music, which is good for human health, the data preprocessing methods contained herein should be improved in future works.

Keywords: deep learning; encoding; feature engineering; melody; music generation; healthcare; term frequency–inverse document frequency

1. Introduction

Many studies that have analyzed the relationship between music and health found that music positively affects human health [1–5]. Music has been shown to reduce human stress levels and has a considerable positive effect on psychological state [6–9]. Although serene classical music is known to be beneficial during patient treatment, it is time-consuming to collect specific classical music pieces that have been shown to positive affect health; therefore, presenting a need for a system is to automatically generate beneficial musical pieces. Recently, many researchers have begun to explore the possibilities of deep learning techniques in the field of artistic creation. Deep learning has had great achievement in search engine, data mining, machine translation, natural language processing, computer vision, and other related fields. The deep learning algorithm has given great support on interpretation of data such as text, images, and sounds by learning the rules in the sample data. Its goal is to enable machines to have the ability to analyze text [10], images [11], and sounds [12] like humans. The achievements

of this complex machine learning algorithm in sound and image processing far exceed the previous related technologies.

Currently, most deep learning-based music generation used real music as sample data. Melody is an important compositional element that reflects the rhythm and emotion of musical works. Although the non-melody compositional aspects also play important roles in the automatic generation of music [13], these non-melody elements correlate strongly with the melodic aspects. Therefore, the generation of melody compositional element is necessary and informs the generation of other compositional elements. To generate the melody, a training library of isolated melody tracks are required. Musical instrument digital interface (MIDI) files, often used in deep learning-based automatic music generation studies, contain multiple tracks. Beneficially, there is usually one track in a MIDI file that stores melody information. To generate music using these melody tracks, methodologies must be developed to accurately extract melody track information from among multiple present tracks [14–17]. Melody changes occur frequently within bars, where bars are used as basic units of composition; accordingly, composers typically produce music around this bar structure. However, to generate music based on bars, a matrix-based bar encoding method is first required.

In traditional music composition, the composer has developed specialized semantic knowledge of music and combines both emotion and creative experience into generating music. As computer technology has evolved, various music-related technologies have also been developed [18–22]. Music can be divided into several parts that are recombined based on semantic knowledge of musical composition to compose new music. The ILLIAC I computer has generated a novel musical piece known as the Illiac Suite based on a Markov chain algorithm [23]. The experiments in musical intelligence (EMI) used a recombinant algorithm to analyze music, then generated compositions by imitating the style of famous composers [24]. A considerable amount of time is required to create new music. Thus, recurrent neural network-based music generation systems are being developed to help composers to more rapidly produce new musical pieces. C-RNN-GAN [25] is an RNN-based GAN model for music generation. GAN models consist of two components: The generator and the discriminator. In general, GANs show high performance for image generation; similarly, the C-RNN-GAN shows great promise for the generation of music. However, there exist two problems requiring improvement in musical generation systems: (1) C-RNN-GAN utilizes a few melodic tracks or analyzed nonmelodic tracks to generate music. In most MIDI files, the melody tracks are not labeled, and using only manually labeled melody tracks under time constraints can result in insufficient learning data. However, without distinction, all the tracks are used as training data; hence, latent space features containing nonmelodic information will be generated. (2) Music is typically composed in units of bars, being organized sections of tonal notes. As composers compose music by the bar, this method for generating music, being close to the methods used by composers, should be studied.

This paper proposes the following methods to resolve issues with traditional music generation methodologies, thereby improving performance. The melody track must be extracted from the MIDI files to generate music based on melody tracks. The term frequency–inverse document frequency (TFIDF) algorithm [26] and feature-based filtering are used to extract the melody tracks. The tracks are extracted from the one MIDI file and the TFIDF value is calculated between the one of the extracted tracks and real melody track set. The track with minimum TFIDF value is selected to the melody track. In addition, a bar encoding method is studied to generate new music based on bars. All the note information contained in the bar is encoded into the bar matrix through the bar encoding method. The bar matrix can accurately represent the start time, duration, pitch, and intensity of each note.

The GAN model can train to generate music using the results of the preprocessing as follows. First, the generator output the music using recurrent neural networks based on bars. Second, the music generated from the generator and real music is input to the discriminator. Third, the discriminator outputs the compare result by compared two music. Fourth, the compared result is inputted to the generator and the generator is trained. The trained generator can generate music.

Additionally, the generator can be used to modulate the instruments and style of music. Instruments whose sounds are beneficial to health can be soloed or featured during playback of the generated music. In order to evaluate the generated music, audience-based evaluation can be leveraged to assess the quality of generated music [27]. However, in order to evaluate automatically, we utilized the pitch structure to determine if the generated music and the music within the training dataset are sufficiently similar.

Our contributions are as follows:

- Automatically melody extraction: In the automatic music generation research based on MIDI files, a melody track is required as the learning data when generating the melody. This research utilizes the TFIDF algorithm and feature-based filters to automatically extract melody tracks from MIDI files.
- Bar encoding method for more features: Bar encoding encodes the start time, duration, pitch, and intensity of the notes contained in the bar into the bar matrix. The bar matrix can accurately represent the detailed features of each note.
- Improved preprocessing for music generation based on deep learning: Automatic melody extraction and bar encoding can enable the deep learning neural network to train the composition method better.

The remainder of this paper is organized as follows. Section 2 introduces studies on deep learning-based music generation. Section 3 describes melody track extraction and the bar encoding methods. Section 4 describes the content and results of the experiments. Section 5 contains our concluding remarks.

2. Related Literature

Typical data representations used during the design and training of music automatic generation algorithms are signal, transformed signal (e.g., spectrum, via Fourier transformation), piano roll, MIDI, text, etc. The most recent research on deep learning methods for music generation use only the melody track contained within MIDI files as training data [25,28]. However, there are other, non-melody tracks within MIDI files. Traditional melody extraction research includes Skyline [29], Best Channel [30], Top Channel [31], Entropy Channel [32], Entropy Part [33], and Revised Skyline [34]. The most common method of extracting melody tracks of extracting melody tracks from MIDI files is the Skyline algorithm [26], which combines all notes into a single track and selects the highest pitch sounds (typically a representative feature of the melody portions of a musical piece). However, there are three disadvantages with the Skyline algorithm. First, the hold time of the note itself can change if a new note with a higher pitch appears; meaning that the playing of the first note ceases, and the new note is played instead. Consequently, the extracted melody track may be different from the true melody of the musical composition. Second, notes of low intensity in a melody track may be erased. Skyline automatically finds and fills all time periods with the highest previously played pitch if no sounds are played for a certain period; thereby, further altering the melody. Third, there are many compositions in which the melody contains pitches lower than those of other tracks. Therefore, the Skyline algorithm cannot derive melody tracks accurately.

Madsen et al. [35] proposed a method for measuring musical complexity and applied it to a melody extraction model. The music complexity model did not utilize musical expertise. TANG et al. [36] assumed that there is only one melody track in music, and [37] proved that this assumption is universal. When extracting the melody track, five methods were proposed. The first three methods, AvgVel, PMRatio, and SilenceRatio, sort the probabilities of the tracks considered to be melody by analyzing the track features. The other two methods, TrackName and Range, can directly divide the tracks into melody tracks or non-melody tracks. In the 16,012 collected MIDI files, 247 melody tracks were extracted by TrackName. TrackName is the most efficient way to extract melody tracks in five ways, but only a small number of MIDI files can be sorted by TrackName

Rizo et al. [38] proposed a feature-based melody filtering method for determining a melody track by analyzing features important to the melody track, differentiating it from other tracks. Their method uses track information, pitch, intensity, lasting time, and total number of notes as track features [39]. However, this method cannot be applied to large volumes of data as the labeling of features is performed manually. Herein, we apply the TFIDF algorithm to the traditional feature-based filtering method. By first using the pitch frequency-based TFIDF algorithm, we can remove most of the non-melody tracks both quickly and automatically.

Most methods of melody extraction only focus on the temporal continuity of the frequency, and seldom consider the temporal correlations between the frequency and amplitude, resulting in the poor extractions of the melody by many algorithms. Consequently, we propose a new melody extraction method [40] based on the frequency–amplitude correlations.

Within the automatic music generation literature using recurrent neural networks, there are different data representation encoding methods. Table 1 describes the difference between various data representation and encoding methods used by previous studies and proposed methods.

Table 1. Differences between traditional music generation systems and the proposed method.

	C-RNN-GAN [25]	**MidiNet [28]**	**MuseGAN [41]**	**Proposed Method**
Data representation	MIDI	MIDI	Piano-roll	MIDI
Encoding	Based notes	Based bars	Based notes	Based bars

Mogren [25] utilizes MIDI files as training data. The pitch, start time, hold time, and intensity of the note are all accurately represented. As it contains a note end event, it addresses this oversight of the Piano-roll representation. However, notes-based music generation method is unsuitable to generate the music [42]. There are limitations in performance compared to the bar encoding. In MidNet [28], MIDI files and bar-based encoding method are utilized; it may not be trivial to easily distinguish between a long note and two short repeating notes (i.e., consecutive notes with the same pitch). MuseGAN [41] utilizes the data represented by the Piano-roll [43] format for its training data. Piano-roll is one of the most commonly used data representations and can represent both the pitch and hold time of a sound; however, it has several limitations. First, note termination information does not exist. Therefore, it is impossible to distinguish between a long note and two short and equal notes. Second, Piano-roll cannot represent the intensity of a note.

In order to solve the above problem, this paper proposes two methods. First, the melody tracks extracted by TrackName are used as the comparison dataset. The difference between the non-melody track and the comparison dataset is calculated by the TFIDF algorithm. Then, the track with the minimum TFIDF is extracted as the melody track. Second, in order to represent more features of note, an encoding method based on bar is proposed to represent the start time, lasting time, pitch, and intensity of the note.

3. Proposed Method

3.1. Music Generation Process of the Proposed Method

Our proposed music generation process method is shown in Figure 1. First, during the melody track extraction phase, the TFIDF algorithm and feature-based filtering are used to extract multiple melody tracks from the collected MIDI files. Second, during the bar encoding phase, the extracted melody tracks are encoded based on bars. As the proposed method generates MIDI files based on bars, melody tracks are segmented into multiple bars. Each divided bar is encoded based on the pitch, start time, length, and intensity of the notes within the bar. Next, data are integrated into a single melody matrix. Third, during the MIDI file generation phase, the melody matrix sets are used to train a deep learning-based music generation model. Accordingly, new MIDI files are generated.

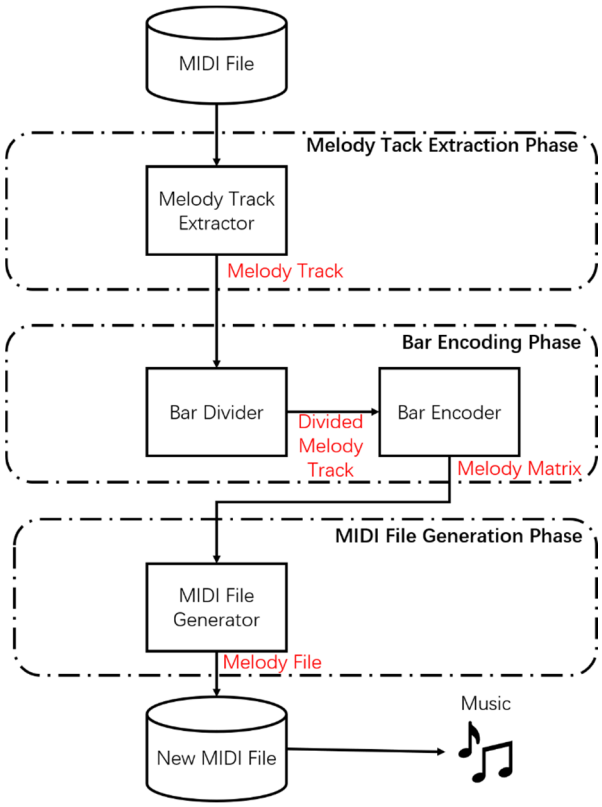

Figure 1. Proposed music generation system process.

3.2. TFIDF-Based Melody Track Extraction

The melody, chord, and drum accompaniment are stored in different tracks within the MIDI file. This section describes the methods used to extract the melody track from the MIDI file.

Typically, the *TrackNameEvent* attribute track metatags to the melody track. If the value is "Melody," the track is a melody track. If there is no *TrackNameEvent* attribute in the track, or if its value is not "Melody," it is difficult to determine whether the track is a melody track or not. Herein, we assume that there is at most one melody track in a MIDI file. The proposed method uses the TFIDF algorithm and feature-based filtering to extract the melody tracks from the MIDI files, as shown in Figure 2.

For example, $T = \{60, 70, 65, 79, 61, 62, 61, 64\}$, T is a set of pitches, which indicates a pitch included in one of the tracks in the MIDI. When using the TFIDF algorithm, the TF (term frequency) value of each note in each track must be calculated. The TF value represents the number of times a note appears in the current track. For example, if a 60 Hz pitch appears once in the first track, then the TF value corresponding to this pitch is 1. The IDF (inverse document frequency) value of each note in each track then needs to be calculated. The IDF value represents the reciprocal of the number of tracks containing this note within the collected melody tracks. If the number of tracks containing this note is 1000, then the IDF value of the note is 0.001. The TFIDF value of each note in the track is determined by multiplying the TF and IDF values. Based on the example note discussed, the TFIDF value of this note is 0.001. The average of the TFIDF values of each note within the track is the TFIDF value of the track. If the current track and melody track set are similar, the TFIDF average value will be low. Therefore, the track with the lowest average value of TFIDF will be extracted as the melody track.

First, the collected MIDI file set, M, is classified. The total MIDI file set contains files that are both labeled and unlabeled as melodies via the *TrackNameEvent* attribute; the unclassified files are represented as M^U and the classified MIDIs are referred to as M^M. The i^{th} MIDI files, m_i, m_i^U, and m_i^M obey the relationships $m_i \in M$, $m_i^U \in M^U$, and $m_i^M \in M^M$, respectively. If "Melody" appears in the *TrackNameEvent* of the tracks included in the i^{th} MIDI file m_i, it is added to the MIDI file set M^M.

If *TrackNameEvent* does not exist or if "Melody" does not appear, it is added to the unconfirmed MIDI file set, M^U.

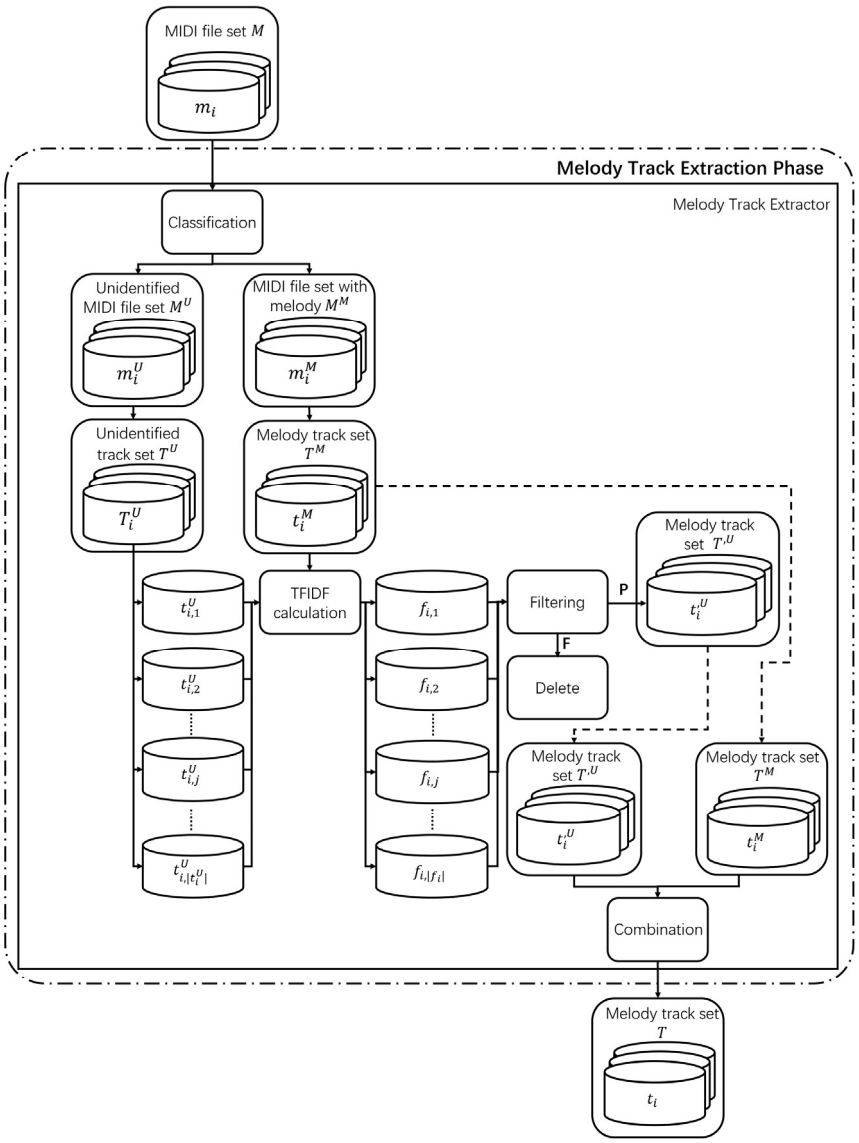

Figure 2. Melody track extraction process.

Second, the melody tracks of the MIDI files are separated. All tracks included in each unconfirmed MIDI file m_i^U in the unconfirmed MIDI file set M^U are defined as the unconfirmed track set T_i^U. Accordingly, $t_{i,j}^U$, which is the jth track in the unconfirmed MIDI file m_i^U, is added to the unconfirmed track set T_i^U. The jth track $t_{i,j}^U$ is $t_{i,j}^U \in T_i^U$, and $T_i^U \in T^U$. In the MIDI file set M^M that includes melody, the melody track set T^M is defined as the only melody track set. The melody tracks are extracted from the MIDI file set M^M that includes melodies, and they are added to the melody track set T^M. Correspondingly, $t_i^M \in T^M$. Third, the TFIDF value is calculated. The TFIDF algorithm is a method of calculating the difference between an element and a set [26]. It is used to (a) calculate the difference between track $t_{i,j}^U$ and the tracks in the melody track set T^M and (b) to use the calculated TFIDF value to represent the track. Pitch is the degree of highness or lowness of a note. Additionally, θ_{MIN} is the minimum value in the allowable pitch range and θ_{MAX} is the maximum value in the allowable pitch range. Pitch p is a pitch in the allowable pitch range and it is an integer between θ_{MIN} and θ_{MAX}. Equivalently, $n_p^{T^M}$ is the number of tracks in melody track set T^M in which pitch p appears at least

once. In turn, $s^U_{i,j}$ is the total number of notes in the individual track $t^U_{i,j}$. Additionally, $n^U_{i,j,p}$ is the frequency of pitch p in the separated track $t^U_{i,j}$. The number of tracks in the melody track set T^M is defined as the track quantity α, while $f^U_{i,j,p}$ is the TFIDF value of pitch p in the separated track $t^U_{i,j}$; $f^U_{i,j}$ is the average TFIDF value of pitch p corresponding to the TFIDF value of the separated track $t^U_{i,j}$. In the proposed method, the TFIDF value of track $t^U_{i,j}$ is calculated based on the pitch frequency, as shown in Algorithm 1.

Algorithm 1. Term frequency–inverse document frequency (TFIDF) value calculation algorithm.

FUNCTION Calculate TFIDFvalue($t^U_{i,j}$, α)
OUTPUT
    $f_{i,j}$//TFIDF value of $t^U_{i,j}$
BEGIN
    FOR p ← θ_{MIN} to SIZE(θ_{MAX})
        $n^{T^M}_p$ ← Calculate frequency of track containing p in T^M
    END FOR
    $s^U_{i,j}$ ← Size of $t^U_{i,j}$
    FOR p ← θ_{MIN} to SIZE(θ_{MAX})
        $n^U_{i,j,p}$ ← Calculate frequency of p in $t^U_{i,j}$
        $f^U_{i,j,p} \leftarrow \frac{n^U_{i,j,p}}{S^U_{i,j}} \times \log\left(\frac{\alpha}{n^{T^M}_p}\right)$
    END FOR
    $f^U_{i,j}$ ← Calculate average of $f^U_{i,j,p}$
END

Fourth, shallow structure description [38] of each unconfirmed MIDI file melody track is used to perform filtering. The shallow structure description includes the track's maximum pitch, minimum pitch, average pitch, maximum lasting time, minimum lasting time, average lasting time, maximum intensity, minimum intensity, and average intensity. The proposed method filters the tracks using the average pitch, average lasting time, and average intensity that were derived from the dataset analysis. For the kth note in the melody track t^M_i, the pitch is defined as $p^M_{i,k}$, the lasting time as $h^M_{i,k}$, and the intensity as $e^M_{i,k}$. The threshold values for each stage are defined as the average pitch $p^M_{i,A}$, average note lasting time $h^M_{i,A}$, and the average note intensity $e^M_{i,A}$ of the melody track t^M_i included in the melody track set T^M. Additionally, $p^M_{i,A}$, $h^M_{i,A}$, and $e^M_{i,A}$, are used as the filtering values for each feature. To find $p^M_{i,A}$, $h^M_{i,A}$, and $e^M_{i,A}$, the values $p^M_{i,k}$, $h^M_{i,k}$, and $e^M_{i,k}$, are used to find the averages for each feature, respectively, as indicated in Equation (1).

$$\begin{aligned} p^M_{i,A} &= \frac{p^M_{i,1}+p^M_{i,2}+\cdots\cdots+p^M_{i,k}}{k} \\ h^M_{i,A} &= \frac{h^M_{i,1}+h^M_{i,2}+\cdots\cdots+h^M_{i,k}}{k} \\ e^M_{i,A} &= \frac{e^M_{i,1}+e^M_{i,2}+\cdots\cdots+e^M_{i,k}}{k} \end{aligned} \quad (1)$$

The average pitch, lasting time, and intensity of each melody track feature are then used to determine the filtering values for each feature. In this respect, p_{MIN} is defined as the pitch filtering value setting the minimum value of the average pitch calculated in the melody tracks; h_{MIN} and h_{MAX} are defined as the lasting time filtering values, where h_{MIN} sets the minimum value of the average lasting time calculated in the melody tracks and h_{MAX}. sets the maximum value of the average lasting time calculated in the melody tracks. The e_{MIN} value is defined as the filtering value for the intensity of the notes and is set as the minimum value of the average intensity calculated in the melody tracks. Track $t^U_{i,j}$, which is found based on the lowest TFIDF value $f^U_{i,j}$, undergoes a three-stage filtering process. The average pitch of $t^U_{i,j}$ is checked to ascertain if it is less than p_{MIN}, and the average note lasting time

of $t_{i,j}^{U}$ is checked to assess whether it is less than h_{MIN} or greater than h_{MAX}. The average note intensity of track $t_{i,j}^{U}$ is checked to assess if it is less than e_{MIN}. If the average pitch, average note lasting time, and average note intensity of $t_{i,j}^{U}$ do not satisfy the set conditions, the track is deleted and the track with the next lowest TFIDF value is selected to perform the filtering process. Fifth, the melody tracks to be used in the unconfirmed MIDI file set are determined. As the TFIDF values of the tracks included in the unconfirmed MIDI files become smaller, the track set becomes more similar with the melody track set T^{M}. The unconfirmed MIDI files are used to define the preprocessed unconfirmed track set $T^{\prime U}$. It is defined as the preprocessed unconfirmed tracks $t_i^{\prime U}$ in the unconfirmed MIDI track files. Accordingly, $t_i^{\prime U} \in T^{\prime U}$. For each unconfirmed MIDI file, the track with the lowest TFIDF value is set as an unconfirmed track $t_i^{\prime U}$. The preprocessed unconfirmed track $t_i^{\prime U}$ that is selected in the unconfirmed MIDI file is set as the preprocessed unconfirmed track set $T^{\prime U}$. Sixth, the melody tracks to be used during music generation are created. The melody track set T is defined as the melody tracks used in music generation. The melody track set T is created by combining the preprocessed unconfirmed track set $T^{\prime U}$ and the melody track set T^{M}. The melody-containing track set T is ultimately used to generate music.

3.3. Bar Encoding

Encoding is used to show each bar contained in a melody track as a matrix. Traditional music generation studies have operated on insufficient musically relevant features as they perform note-based encoding. To resolve this, we encode tracks within out process based on bars, which are groups of notes. The proposed bar encoding method is shown in Figure 3.

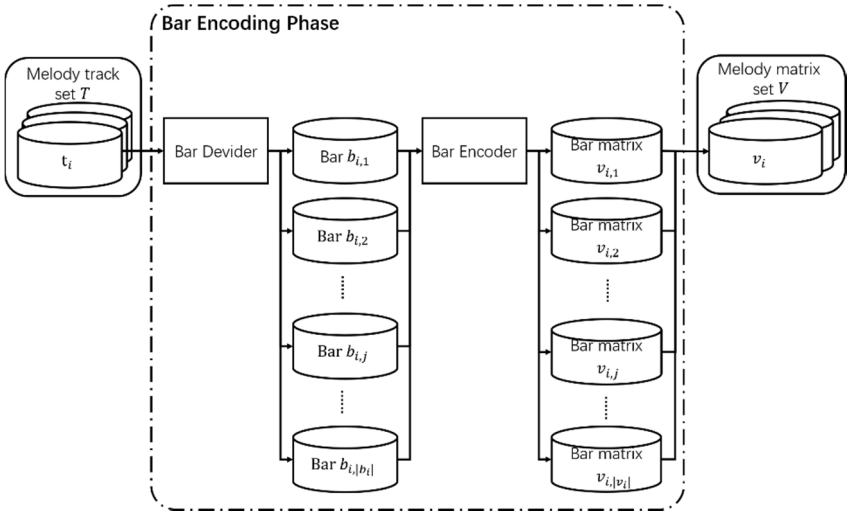

Figure 3. Bar encoding method.

First, the melody track is used to separate the bars. Additionally, the lasting time information of each track in the MIDI file is available. The melody track lasting time is l_i. The l_i value is the lasting time of the ith melody track, t_i, in the melody track set T during the bar division process. The lasting time of each bar in the same melody track is the same, and it is defined as $l_{i,B}$, which is calculated by beat. Bars are separated into $[b_{i,1}, b_{i,2}, \ldots, b_{i,j}, \ldots, b_{i,|b_r|}]$ according to the bar lasting time $l_{i,B}$, as shown in Figure 4. The melody is divided into bars and strictly divided according to the beat, which can be obtained from MIDI file. If bars are re-segmented, the structure of the melody will remain unchanged. The notes included in the jth bar of the melody track are defined as $b_{i,j}$. Additionally, $v_{i,j}$ is a two-dimensional (2D) matrix that shows the pitch, start time, length, and intensity of the notes in each bar, as shown in Figure 5.

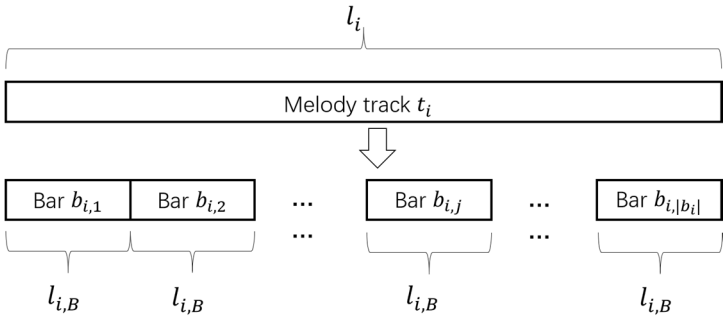

Figure 4. Bar segmentation.

Second, the divided bars are used to perform the process of encoding the track segments within a bar matrix. When encoding is performed, a 2D matrix is used to represent the pitch of each note within a bar, as well as the start time, length, and intensity, as shown in Figure 5.

The size of the x-axis of the 2D matrix is the bar's lasting time, $l_{i,B}$, which is the unit of lasting time in the MIDI file, referred to as a Tick. The time by counting the number of Ticks can be quantified. The variable y-dimension is representative of pitch range. The size of the y-axis determines the maximum value, θ_{MAX}, and the minimum value, θ_{MIN}, of the allowable pitch range in the MIDI files. The jth bar of the melody track to be encoded is $b_{i,j}$. The melody matrix, v_i, is the matrix storing the results of the encoding of all the bars that are included in the melody track t_i. The melody matrix, v_i, consists of $[v_{i,1}, v_{i,2}, \ldots, v_{i,j}, \ldots, v_{i,|v_r|}]$. The bar matrix, $v_{i,j}$, defines the results that are encoded based on each bar $b_{i,j}$ of the melody track t_i. For $v_{i,j}$, the notes included in each bar $b_{i,j}$ are converted to a bar matrix, as shown in Figure 5. In turn, $b_{i,j,k,F}$ is defined as the pitch of the kth note of $b_{i,j}$. The value of $b_{i,j,k,F}$ is set by the pitch of the kth note of $b_{i,j}$, and it is a value between θ_{MIN} and θ_{MAX} according to the allowed pitch range of the MIDI file. The $b_{i,j,k,F}$ value is used to determine the position of the current note corresponding to the y-axis. The onset of the kth note of $b_{i,j}$ is defined as $b_{i,j,k,S}$, while $b_{i,j,k,S}$ is set as the start time of the kth note in the $b_{i,j}$. The $b_{i,j,k,S}$ value is used to determine where the current note begins on the x-axis. Additionally, $b_{i,j,k,L}$, which is the length of the kth note of $b_{i,j}$, is defined as the note's duration time, and $b_{i,j,k,L}$ is set as the time during which the kth note of $b_{i,j}$ is sustained. $b_{i,j,k,L}$ is then used to determine the length of the current note on the x-axis. The intensity $b_{i,j,k,V}$ of the kth note of $b_{i,j}$ is defined as the volume of the sound. Furthermore, the variable $b_{i,j,k,V}$ sets the matrix as the intensity of the kth note of $b_{i,j}$. After the position of the current note in the measure matrix is determined, fill with $b_{i,j,k,V}$. Each bar is encoded as a matrix, as shown in Figure 5, and the empty spaces are filled with zeros.

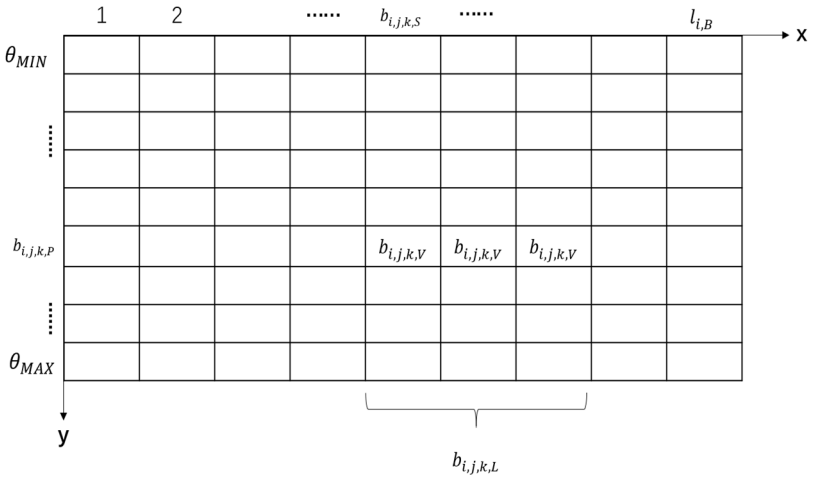

Figure 5. Bar encoding.

The encoded bar matrix, $v_{i,j}$, is integrated with the melody matrix v_i. The set of the melody matrices is defined as the melody matrix set V. The encoded melody matrix, v_i, is added to the melody matrix set V.

In the MIDI file, the start time and end time are clearly specified for each note. When using these data to encode the bar matrix, the start time and end time of the note will not be changed.

4. Experiments and Results

4.1. Experimental Objectives

We conducted several experiments including melody track extraction accuracy verification and bar-based encoding performance verification experiments. The melody track extraction accuracy verification experiments determine whether melody tracks are accurately extracted from MIDI files. Verified MIDI melody files marked with "Melody" in the *TrackNameEvent* attribute are used for melody track extracted verification, and these MIDI files are part of the dataset [44]. In the melody track extraction accuracy verification experiments, the level of accuracy was extracted as shown below to verify the performance of the proposed TFIDF-based melody track extraction method. First, the melody tracks were extracted from the verification MIDI file set. Second, the text parameter of the *TrackNameEvent* attribute, which is included in the extracted melody metatags, was checked to assess whether its value was "Melody," and the level of accuracy was determined. The bar encoding performance verification experiments assessed whether the bar information could be accurately represented by a matrix.

4.2. Experimental Data

The verification MIDI file set was used to support experiments intended to verify the melody track extraction accuracy. In the MIDI file quality verification experiment, the melody track set extracted by the melody track extraction method was encoded based on bars and used as the input data.

For the MIDI files, the data needed in the experiments were extracted by the Python MIDI library, as shown in Figure 6. The MIDI files consist of multiple tracks, and there is usually one melody track among the multiple tracks. A track has *TrackNameEvent*, *NoteOnEvent*, and *EndOfEvent* attributes. In Track #1, the *NoteOnEvent* and *NoteOffEvent* attributes show the start and end instants of a note. *NoteOnEvent* includes three types of information: Start time, pitch, and intensity. Conversely, *NoteOffEvent* includes duration time, pitch, and intensity, with the intensity is expressed as zero. *EndOfTrackEvent* indicates the end of the track. In experiments, it is assumed that the collected MIDI files are good for health and each MIDI file usually has one melody track.

During melody track extraction accuracy verification experiments, 150 of the 763 MIDI files, which the *TrackNameEvent* parameter value was "Melody" in all 2000 MIDI files, were used as the verification MIDI file set. One hundred and fifty melody tracks would be extracted from the verification MIDI file set. The melody tracks were extracted from the remaining 613 MIDI files and used as a comparison melody track set.

In the bar encoding performance verification experiments, the melody track set extracted by the TFIDF algorithm-based melody track extraction method was used as input. The melody track set consisted of 2000 melody tracks. As shown in Figure 6, the start time information included in *NoteOnEvent* is the time relative to the onset of the previous note. Because a single note is expressed as two events, *NoteOnEvent* and *NoteOffEvent*, a preprocessing stage is required before utilizing this information as the input in the quality verification experiments. Figure 7 shows the preprocessing stage. In the melody track, the start time, which includes the event logs of the melody track, is converted from relative time (red) to absolute time (black), and the start time (red), length (green), pitch (blue), and intensity (purple) of each note are contained in a single vector.

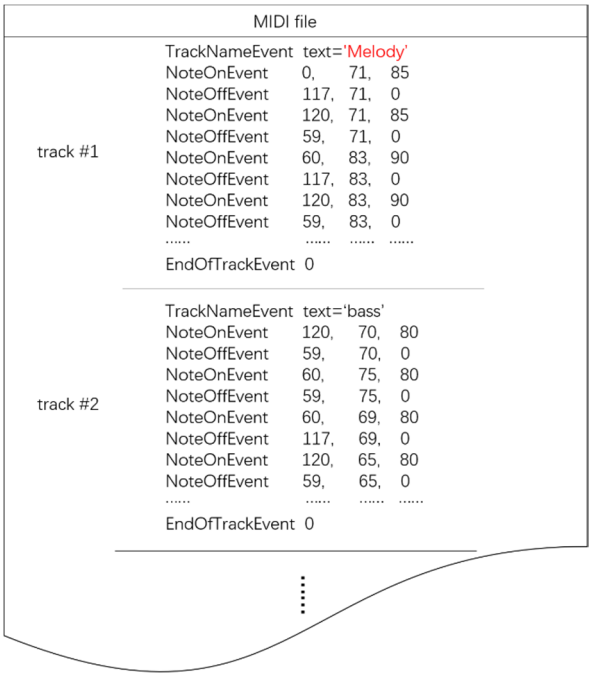

Figure 6. Information extracted from a musical instrument digital interface (MIDI) file.

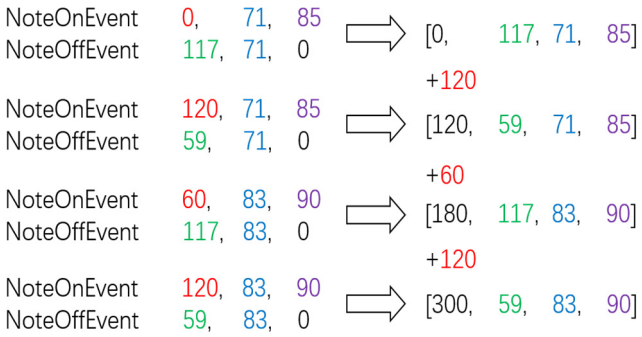

Figure 7. Preprocessed bar of melody track.

4.3. Experimental Results

This subsection describes the results of the extraction of the melody tracks from MIDI files using the TFIDF algorithm, as well as the performance of the proposed bar encoding method.

Table 2 shows the results derived from the TFIDF algorithm and the filtering stage. Three features were used to filter the tracks that were separated from the MIDI file, starting with the track that had the lowest TFIDF value based on the comparison with the melody tracks. Threshold values were found for the three features, including the average pitch, average note lasting time, and average note intensity. The purpose of providing an average value is to determine the range of the average pitch and average intensity for each melody track. During the related task of extracting melody tracks from MIDI files, the average value is often used as an important feature for analysis. It should be noted that since the pitch and intensity of the notes in the MIDI file range is from 0 to 127, the average pitch and average intensity must be between 0 and 127. The average pitch, average note lasting time, and average note intensity were extracted from the comparison melody track set to determine the threshold values for each of the features that the melody tracks had. When features such as the maximum pitch, minimum pitch, maximum lasting time, minimum lasting time, maximum intensity, and minimum intensity are used for filtering, the melody track cannot be distinguished from the other tracks. Therefore, these features have low information density in relation to the present tasks. Given that the minimum

value of the extracted average pitches was 66, the pitch filtering value was 66. Additionally, given that the average lasting time was between 0.3 and 1.3, for easy comparison, here the tick was converted to seconds and the filtering value range was between 0.3 s and 1.3 s. Furthermore, given that the average strengths were above 60, the filtering value was set to 60. If a track passed through the filtering stage without engaging the filters, it was considered a melody track and was extracted. If it did not pass the filtering stage, the track was skipped, and filtering was performed on the track with the next lowest TFIDF value. If no tracks passed through the filtering stage, the track with the lowest TFIDF value was considered to be the melody track. The tracks marked with Y are the accurately extracted melody tracks, and the tracks marked with N are the inaccurately extracted melody tracks.

Table 2. Results of extracted melody tracks.

Index	TFIDF (Minimum)	Average Pitch	Average Lasting Time	Average Intensity	Y/N
1	0.045	71.285	0.810	83.027	Y
2	0.041	71.285	0.350	103.163	Y
3	0.025	68.694	0.501	79.956	Y
4	0.054	73.078	0.520	106.039	Y
5	0.037	71.651	0.681	100	Y
6	0.044	72.291	0.835	90	Y
7	0.045	67.778	0.679	100	N
8	0.079	73.914	0.530	100	Y
...	...	...	...	...	...
150	0.049	71.953	0.696	90	Y

Table 3 shows the accuracy of the proposed TFIDF-based melody track extraction method. MIDI files were used, and 150 tracks were extracted with the proposed method. Of these melody tracks, 142 tracks, or 94.7%, were extracted accurately.

Table 3. Accuracy of extracted melody track.

MIDI File Number	True	False	Accuracy
150	142	8	94.7%

The bar encoding performance verification experiments confirmed that the bar encoding method can accurately represent the four features of each notes: Pitch, start time, length, and intensity. Figure 8 shows the results of the encoding of the output of Figure 7. The encoded results are formulated within a fixed-size matrix. This matrix represents a bar consisted of four notes. The orange- and yellow-colored boxes show the results associated with playing notes at different intensities for a certain amount of time. In the note-based encoding method, the numbers of notes in the bars are different. Therefore, the encoding matrix sizes are different. Encoding matrices with different sizes cannot be used as input for the neural networks with fixed input feature dimension sizes. In the MIDI file, the start and end times are clearly specified for each note. When using these data to encode the bar matrix, the start time and end time of the note should remain unchanged.

A comparative experiment was performed between music generated based on notes and music generated based on bars. Music produced with units of bars was perceived to be of higher quality than music produced by a note-by-note method. However, this paper focuses on the preparation process, which is the preprocessing method suitable for most automatic music generation research based on the recurrent neural network.

	0	1	···	···	117	118	119	120	121	···	179	180	181	···	297	298	299	300	301	···	359	···	479
···	0	0	···	···	0	0	0	0	0	···	0	0	0	···	0	0	0	0	0	···	0	···	0
71	85	85	···	···	85	0	0	85	85	···	85	0	0	···	0	0	0	0	0	···	0	···	0
···	0	0	···	···	0	0	0	0	0	···	0	0	0	···	0	0	0	0	0	···	0	···	0
···	0	0	···	···	0	0	0	0	0	···	0	0	0	···	0	0	0	0	0	···	0	···	0
83	0	0	···	···	0	0	0	0	0	···	0	90	90	···	90	0	0	85	85	···	85	···	0
···	0	0	···	···	0	0	0	0	0	···	0	0	0	···	0	0	0	0	0	···	0	···	0

Figure 8. Encoded bar matrix.

5. Conclusions

Comfortable music can help people relieve stress, but collecting music requires professional knowledge of music and needs a lot of time. Deep learning has had great achievements in various fields, and it has also been active in the field of automatic music generation. To help the deep automatic generation system based on deep learning generate music, which is beneficial to human health, this paper proposed new preprocessing methods that perform melody track extraction and bar encoding on MIDI file encoded music. Herein, we introduce a TFIDF-based melody track extraction method. To extract the melody track among various other tracks, the TFIDF algorithm and three types of feature-based filtering were used to accurately extract the melody track based on the pitch frequency. Furthermore, we explore a new bar-based encoding method. To express all the note information in a bar in the form of a matrix, four note features (i.e., pitch, start time, length, and sound intensity) were used to encode bars as the matrices with the same size. The 142 melody tracks were accurately extracted from 150 MIDI files by the proposed melody track extraction technique with an accuracy rate of 94.7%. The bar matrices that were encoded by the bar-based encoding method had the same size, unlike the matrices encoded by the note-based encoding methods. Correspondingly, the cost and standardization of the bar-based encoding method were verified.

In this paper, the TFIDF algorithm-based melody track extraction method and the bar encoding method were proposed to support the research of automatic music generation based on deep learning. In future research, the comparison datasets will be classified by genre or composer. By this way, features for each genre or composer will be obtained and utilizing these features can extract music more accurately; and for bar encoding, in addition to the information of the notes, it is necessary to improve the existing methods to express the rhythm and speed of the music to express more musical features.

Author Contributions: Conceptualization, S.L., S.J., and Y.S.; methodology, S.L., J.S., and Y.S.; software, S.L., S.J., and Y.S.; validation, S.L., S.J., and Y.S.

Funding: This work was supported by the Dongguk University Research Fund of 2017.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Westen, D. Clinical assessment of object relations using the TAT. *J. Personal. Assess.* **1991**, *56*, 56–74. [CrossRef] [PubMed]
2. Arnon, S.; Shapsa, A.; Forman, L.; Regev, R.; Litmanovitz, I.; Dolfin, T. Live music is beneficial to preterm infants in the neonatal intensive care unit environment. *Birth* **2006**, *33*, 131–136. [CrossRef] [PubMed]
3. Jancke, L. Music, memory and emotion. *J. Biol.* **2008**, *7*, 82–86. [CrossRef] [PubMed]
4. Cepeda, M.S.; Carr, D.B.; Lau, J.; Alvarez, H. Music for pain relief. *Cochrane Database Syst. Rev.* **2006**, *2*. [CrossRef]
5. Trappe, H.J. The effects of music on the cardiovascular system and cardiovascular health. *Heart* **2010**, *96*, 1868–1871. [CrossRef] [PubMed]

6. Leard, S.; Pietroletti, R.; Angeloni, G.; Necozione, S.; Ranalletta, G.; Gusto, B.D. Randomized clinical trial examining the effect of music therapy in stress response to day surgery. *Br. J. Surg.* **2007**, *94*, 943–947. [CrossRef] [PubMed]
7. Spintge, R. *Clinical Use of Music in Operating Theatres*; Oxford University Press: Oxford, UK, 2012; pp. 277–286.
8. Miranda, D.; Gaudreau, D.; Debrosse, D.; Morizot, D.; Kirmayer, L.J. Music listening and mental health: Variations on internalizing psychopathology. *Music. Health Wellbeing* **2012**, 513–529. [CrossRef]
9. Wiggins, G.A. Searching for computational creativity. *New Gener. Comput.* **2006**, *24*, 209–222. [CrossRef]
10. Noaman, H.M.; Sarhan, S.S.; Rashwan, M.A.A. Enhancing recurrent neural network-based language models by word tokenization. *Hum. Cent. Comput. Inf. Sci.* **2018**, *8*, 12–24. [CrossRef]
11. Ghrabat, M.J.J.; Ma, G.; Maolood, I.Y.; Alresheedi, S.S.; Abduljabbar, Z.A. An effective image retrieval based on optimized genetic algorithm utilized a novel SVM-based convolutional neural network classifier. *Hum. Cent. Comput. Inf. Sci.* **2019**, *9*, 31–59. [CrossRef]
12. You, S.D.; Liu, C.H.; Chen, W.K. Comparative study of singing voice detection based on deep neural networks and ensemble learning. *Hum. Cent. Comput. Inf. Sci.* **2018**, *8*, 34–51. [CrossRef]
13. De Prisco, R.; Malandrino, D.; Zaccagnino, G.; Zaccagnino, R. *An Evolutionary Composer for Real-Time Background Music*; Springer: Cham, Switzerland, 2016; pp. 135–151.
14. Chou, H.; Chen, M.T.; Chi, T.S. A hybrid neural network based on the duplex model of pitch perception for singing melody extraction. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018.
15. Danayi, A.; Seyedin, S. A novel algorithm based on time-frequency analysis for extracting melody from human whistling. In Proceedings of the 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), Tehran, Iran, 25–27 December 2018.
16. Chen, L.; Ma, Y.J.; Zhang, J.; Wan, G.C.; Tong, M.S. A Novel Extraction Method for Melodic Features from MIDI Files Based on Probabilistic Graphical Models. In Proceedings of the 2018 Progress in Electromagnetics Research Symposium (PIERS-Toyama), Toyama, Japan, 1–4 August 2018.
17. Lu, W.T.; Su, L. Deep Learning Models for Melody Perception: An Investigation on Symbolic Music Data. In Proceedings of the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Honolulu, HI, USA, 12–15 November 2018.
18. Chenchen, K.; Yu, Y. Main Melody Extraction Using the Auditory Scene Analysis for the Humming Music Retrieval. In Proceedings of the 2018 14th IEEE International Conference on Signal Processing (ICSP), Beijing, China, 12–16 August 2018.
19. Young, H. A categorial grammar for music and its use in automatic melody generation. In Proceedings of the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design, Oxford, UK, 9 September 2017.
20. Prudente, L.; Coronel, A. Towards Automated Counter-Melody Generation for Monophonic Melodies. In Proceedings of the 2017 International Conference on Machine Learning and Soft Computing, Ho Chi Minh City, Vietnam, 13–16 January 2017.
21. Zhu, H.; Liu, Q.; Yuan, N.J.; Qin, C.; Li, J.; Zhang, K.; Zhou, G.; Wei, F.; Xu, Y.; Chen, E. 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, London, UK, 19–23 August 2018.
22. Chen, K.; Zhang, W.; Dubnov, S.; Xia, G.; Li, W. The effect of explicit structure encoding of deep neural networks for symbolic music generation. In Proceedings of the 2019 International Workshop on Multilayer Music Representation and Processing (MMRP), Milano, Italy, 23–24 January 2019.
23. Sandred, O.; Laurson, M.; Kuuskankare, M. Revisiting the Illiac Suite—A rule-based approach to stochastic processes. *Sonic Ideas Ideas Sonicas* **2009**, *2*, 42–46.
24. Cope, D. Experiments in musical intelligence (EMI): Non-linear linguistic-based composition. *J. New Music. Res.* **1989**, *18*, 117–139. [CrossRef]
25. Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In Proceedings of the Constructive Machine Learning Workshop (CML) at NIPS 2016, Barcelona, Spain, 10 December 2016. stage of accepted.
26. Kim, S.W.; Gi, M.J. Research paper classification systems based on TF-IDF and LDA schemes. *Hum. Cent. Comput. Inf. Sci.* **2019**, *9*, 30–50. [CrossRef]

27. Pearce, M.T.; Wiggins, G.A. Towards a framework for the evaluation of machine compositions. In Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences, York, UK, 21–24 March 2001.
28. Yang, L.C.; Chou, S.Y.; Yang, Y.H. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In Proceedings of the ISMIR Conference 2017, Suzhou, China, 23–27 October 2017.
29. Uitdenbogerd, A.; Zobel, J. Melodic matching techniques for large music databases. In Proceedings of the Seventh ACM International Conference on Multimedia (Part 1), Orlando, FL, USA, 30 October–5 November 1999; pp. 57–66.
30. Isikhan, C.; Ozcan, G. A survey of melody extraction techniques for music information retrieval. In Proceedings of the 4th Conference on Interdisciplinary Musicology (SIM'08), Thessaloniki, Greece, 3–6 July 2008.
31. Ozcan, G.; Isikhan, C.; Alpkocak, A. Melody extraction on MIDI music files. In Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05), Irvine, CA, USA, 14 December 2005.
32. Uitdenbogerd, A.L.; Zobel, J. Music ranking techniques evaluated. *Aust. Comput. Sci. Commun.* **2002**, *24*, 275–283.
33. Shan, M.K.; Kuo, F.F. Music style mining and classification by melody. *IEICE Trans. Inf. Syst.* **2003**, *86*, 655–659.
34. Velusamy, S.; Thoshkahna, B.; Ramakrishnan, K.A. A novel melody line identification algorithm for polyphonic MIDI music. In *International Conference on Multimedia Modeling*; Springer: Berlin, Germany, 2007.
35. Madsen, S.T.; Widmer, G. A complexity-based approach to melody track identification in midi files. In Proceedings of the International Workshop on Artificial Intelligence and Music 2007, Valencia, Spain, 11–13 April 2007.
36. Michael, T.; Lap, Y.C.; Kao, B. Selection of melody lines for music databases. In Proceedings of the 24th Annual International Computer Software and Applications Conference, COMPSAC2000, Taipei, Taiwan, 25–27 October 2000.
37. Li, J.; Yang, X.; Chen, Q. MIDI melody extraction based on improved neural network. In Proceedings of the 2009 International Conference on Machine Learning and Cybernetics, Baoding, China, 12–15 July 2009; pp. 1133–1138.
38. Rizo, D.; De León, P.J.P.; Pérez-Sancho, C.; Pertusa, A.; Quereda, J.M.I. A pattern recognition approach for melody track selection in MIDI files. In Proceedings of the ISMIR, Victoria, BC, Canada, 8–12 October 2006; pp. 61–66.
39. De León PJ, P.; Pérez-Sancho, C.; Inesta, J.M. A shallow description framework for musical style recognition. In Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Lisbon, Portugal, 18–20 August 2004; pp. 876–884.
40. Liang, Y.; Li, C.; Tian, L. Melody extraction from polyphonic music based on the amplitude relation. In Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing, Guangzhou, China, 10–12 May 2019; pp. 84–88.
41. Dong, H.W.; Hsiao, W.I.; Yang, L.C.; Yang, Y.H. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 34–41.
42. Chu, H.; Urtasun, R.; Fidler, S. Song from PI: A musically plausible network for pop music generation. In Proceedings of the International Conference on Learning Representations (ICLR), Scottsdale, AZ, USA, 24–26 April 2017.
43. Bryan, N.J.; Mysore, G.J.; Wang, G. Source Separation of Polyphonic Music with Interactive User-Feedback on a Piano Roll Display. In Proceedings of the International Society for Music Information Retrieval (ISMIR), Curitiba, Brazil, 4–8 November 2013; pp. 119–124.
44. Available online: http://www.classicalmidi.co.uk/ (accessed on 31 October 2019).