mathematics

MDPI

Article

INCO-GAN: Variable-Length Music Generation Method Based on Inception Model-Based Conditional GAN

Shuyu Li and Yunsick Sung *

Department of Multimedia Engineering, Dongguk University-Seoul, Seoul 04620, Korea; lishuyu@dongguk.edu
* Correspondence: sung@dongguk.edu

Abstract: Deep learning has made significant progress in the field of automatic music generation. At present, the research on music generation via deep learning can be divided into two categories: predictive models and generative models. However, both categories have the same problems that need to be resolved. First, the length of the music must be determined artificially prior to generation. Second, although the convolutional neural network (CNN) is unexpectedly superior to the recurrent neural network (RNN), CNN still has several disadvantages. This paper proposes a conditional generative adversarial network approach using an inception model (INCO-GAN), which enables the generation of complete variable-length music automatically. By adding a time distribution layer that considers sequential data, CNN considers the time relationship in a manner similar to RNN. In addition, the inception model obtains richer features, which improves the quality of the generated music. In experiments conducted, the music generated by the proposed method and that by human composers were compared. High cosine similarity of up to 0.987 was achieved between the frequency vectors, indicating that the music generated by the proposed method is very similar to that created by a human composer.

Keywords: convolutional neural network; deep learning; conditional generative adversarial network; music composition; inception model

check for updates

Citation: Li, S.; Sung, Y. INCO-GAN: Variable-Length Music Generation Method Based on Inception Model-Based Conditional GAN. *Mathematics* **2021**, *9*, 387. https://doi.org/10.3390/math9040387

Academic Editor: Bo-Hao Chen
Received: 29 December 2020
Accepted: 10 February 2021
Published: 15 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Music composition is a creative task for humans that requires some familiarity with music theory. To enable machines to compose music like human composers, many studies utilize deep learning techniques. However, there are two major problems that need to be overcome for effective automatic music generation via machine learning [1]. First, the temporal relationship of notes or bars in music must be considered. In music, a single note or bar has no meaning, as in the case of a sentence containing only a single word. As in the case of a sentence, wherein nouns, verbs, adjectives, and other elements are arranged in grammatical order to constitute a meaningful sentence [2], in music, some distinct or identical notes are arranged in a certain order to constitute a beautiful melody. The task of automatic music generation is to learn how to arrange the selected notes. Second, one must consider the connection between multiple tracks in music. When music is being played by different players or instruments, it is divided into different tracks. The relationship between tracks is sometimes difficult to grasp; either they can be completely independent, or they can complement each other [3]. Hence, when multiple tracks are played together, the interrelationship between their respective notes becomes very complicated.

At present, the research on automatic music generation via deep learning is divided into two main types. The first type involves predictive models. This type of model mainly utilizes recurrent neural networks (RNN) to predict the next note based on the previous notes. In CONCERT [4], the pitch, note duration, and harmonic chord are encoded with musical rules and then fed into the RNN model. The RNN model then predicts the next note by analyzing all the information that has been extracted. Subsequently, the

predicted note will be fed back into the RNN as input again to generate a sequence. In the RL-Tuner melody generation system [5], both RNN and deep Q-learning networks are combined to generate music. In this model, reinforcement learning is utilized to add and optimize specific rewards for improving sequence prediction. MiniBach [6] is also one of the automatic music generation systems, utilizing one-hot encoding and feedforward networks. However, because the model's architecture is too simple, the performance is not ideal. DeepBach builds upon the design philosophy of MiniBach. DeepBach added long short-term memory (LSTM) to the original feedforward network. Compared with the basic RNN, LSTM performs better at processing sequence data. Song from Pi [7] draws inspiration from the music generated based on Pi and achieves the goal of generating multitrack music by utilizing a multilayer LSTM. In their final test, more than 80% of the 27 listeners perceived that the effect was better than that of Magenta [8].

The second common type is based upon a generative model, most often constructed from generative adversarial networks (GAN). A GAN is composed of a generator and a discriminator. The music produced by the generator is passed to the discriminator, and the discriminator updates itself and the generator by comparing the generated music with a sample of real target music. The generator generates higher quality music by competing with the discriminator. C-RNN-GAN [9] was the first research to utilize GAN to generate music; both the generator and discriminator utilized LSTM networks. The difference is that, for better comparison from two directions, the discriminator utilizes the Bidirectional-LSTM (Bi-LSTM) network. Compared with the traditional RNN-based note prediction model, C-RNN-GAN utilizes the LSTM-based generative adversarial model to generate music with high accuracy and diversity. The generator of the generative adversarial model does not get feedback directly from the training data but continuously learns composition through the feedback of the discriminator, which is the inspiration obtained from the application of GAN in image generation. The goal of the GAN model is never to generate samples the same as the training data but to generate new objects similar to the training data. SeqGAN [10] utilizes the Seq2Seq model [11] as a generator, which consists of two parts: the encoder and decoder based on RNN. The discriminator utilizes a convolutional neural network (CNN) because the discriminator is intended to classify both real and generated samples. In the classification task, the performance of CNN may often be better than that of RNN. MidiNet's [12] generator and discriminator each utilize CNN, and the results show that CNN can be a powerful alternative to RNN. In addition, the structure of conditional GAN [13] supports human input of the conditional vectors to influence music generation instead of utilizing only randomly generated noise vectors. Additionally, MuseGAN [14] only utilizes CNN, but a notable difference from other research is that music is generated in complete bars rather than note-by-note.

Presently, regardless of the predictive model or generative model being utilized, there are some common problems that need to be resolved. First, when generating music, the model cannot judge when to end the generation by itself. In other words, the model only can generate fixed-length music [15,16]. This means that the model cannot determine whether the currently generated segment belongs to the beginning, middle or end of the music. Next, CNN has shown high potential in the music generation given that RNN is normally suitable for sequence data processing. However, it still suffers from some disadvantages [17]. For example, there is no apparent time relationship between the notes or bars generated by a CNN because, unlike RNN, CNN cannot store information in a memory cell.

In this paper, an inception model-based conditional GAN is proposed to generate variable-length music automatically. The proposed network is composed of four stages: First, a conditional vector generator (CVG) is proposed to deduct conditional vectors. The conditional vectors include two parts, one is utilized to control the structure of the music generation, and the other one is utilized to determine whether the music generation has ended. Next, an inception model-based conditional GAN (INCO-GAN) is introduced to process the conditional vectors for generating variable-length music to enable the INCO-

GAN to decide the portion of the whole composition being generated. Next, a time distribution layer that considers sequential data is added to the CNN to consider time relationships in like manner to RNN. Finally, an inception model [18] is utilized to construct a GAN discriminator to obtain richer features. Our contributions are as follows:

- Using conditional vectors to control the structure of music generation. The structure comprises intro, verse, chorus, and outro;
- Inception model-based conditional GAN controls the structure of music generation to generate variable-length music;
- A time distributed layer is added to CNN to share the weight of each timestamp so that it considers the context relationship in like manner to RNN;

The remainder of this paper is organized as follows: Section 2 overviews studies on deep learning-based music generation. Section 3 describes the system proposed for variable-length music generation. Section 4 presents the experiment results. Section 5 analyzes and discusses the results. Section 6 presents concluding remarks.

2. Related Work

In predictive model-based music generation, CONCERT [4] is an exemplar of early music generation systems. Before the advent of deep learning, the representation was designed with rich handcrafted features. One advantage of utilizing a deep learning architecture is that such rich and profound features can be automatically extracted and managed by the architecture. Although CONCERT is now obsolete, it was a pioneering work at the time it was proposed. The reinforcement strategy was first proposed by the RL-Tuner melody generation system [5], and the goal was to control the generation of melody with user constraints. This reinforcement strategy allowed the combination of user control along with the RNN. Notably, in the general reinforcement learning model, rewards are not predefined. For RL-Tuner, there are two rewards that are predefined: handcrafted rules based on music theory and those learned from the dataset regarding the musical style. However, designing rewards is often harsh and inaccurate. MiniBach [6] is a melody-based accompaniment generation system that consists only of an input layer, an output layer, and only one hidden layer. The hidden layer utilizes ReLu as the activation function, the output layer utilizes a softmax activation function, and the data representation is PianoRoll [19] with one-hot encoding. Furthermore, the length of all notes is standardized to one-16th note. Even if the simplest network structure and strategy is utilized, the result is still acceptable. Some limitations of MiniBach are its determinism and the fixed duration of the generation. MiniBach is an extreme simplification of DeepBach [6], only relying on feedforward. DeepBach has a more complex structure, and it combines two LSTM layers and two feedforward networks. Unlike standard LSTM, which only considers a single time direction, DeepBach considers both forward and backward time directions. Therefore, two LSTM layers are utilized, one to summarize information from the past and the other to summarize information from the future. The outputs of the two LSTM layers and the feedforward network are combined and then passed into another feedforward network to predict notes relying on pseudo-Gibbs incremental sampling of variables. Song from Pi [7] proposed a hierarchical generation method for popular songs based upon musical theory. Surmounting past research, this method can generate multitrack music. However, like most existing methods, this system is still learning only to generate music at the note level. This can be unsuitable for music, as music is flexible and intentionally made to be unpredictable when it is composed.

Generative models have made greater progress in automatic music generation than predictive models. However, limitations still exist when the goal is to generate discrete sequences. The main reason is that the output of the generative model is discrete, and it is difficult to transfer the gradient update of the discriminator to the generator. In addition, the discriminator can only evaluate a complete sequence, but for a partially generated sequence, after the entire sequence is generated, it is important to balance its current score and future score. SeqGAN [10] is a proposed sequence generation framework to solve

these problems. The generator of C-RNN-GAN [9] also utilizes RNN to generate discrete sequences. The representation chosen by C-RNN-GAN is inspired by MIDI and models each musical note via four attributes: duration, pitch, intensity, and time elapsed relative to the previous event. This allows the representation of simultaneous notes. C-RNN-GAN utilizes feature matching when training the model. MidiNet [12] is both an adversarial and a convolutional architecture to generate pop music melody. The structure also utilizes chords as an additional input to provide conditions for music generation. Like MidiNet, MuseGAN's [14] generator and discriminator are also composed of CNN. Furthermore, a bar generator is included in the generator to generate bars, and then the generator combines the generated bars. Chords, style, melody, and groove, are utilized together as input to the generator. Although it may still be below the level of human musicians in terms of musical esthetics, it has sparked much inspiration for follow-up research.

A comparison of all related research and proposed methods is presented in Table 1. Ten music generation research works are compared by considering the representation of data, type of model, architecture, etc.

Table 1. Differences between related research work and the proposed method.

Research Work	Representation of Data	Type of Model	Architecture	Neural Network	Self-Control Structure
CONCERT	One-Hot	Predictive	Single	RNN	X
RL-Tuner	Midi	Predictive	Compound	RNN, Deep Q Network	X
MiniBach	One-Hot	Predictive	Single	Feedforward Network	X
DeepBach	Note Real Names	Predictive	Compound	Feedforward Network, LSTM	X
Song from Pi	Midi	Predictive	Compound	Multi-LSTM Layer	X
SeqGAN	Midi	Generative	GAN	LSTM, CNN	X
C-RNN-GAN	Midi	Generative	GAN	LSTM, Bi-LSTM	X
MidiNet	Midi	Generative	Conditional GAN	CNN	X
MuseGAN	Midi	Generative	GAN	CNN	X
Proposed Method	Midi	Generative	Conditional GAN	CNN	√

3. Music Generation System Based on INCO-GAN

This paper proposes an inception model-based conditional GAN to generate variable-length music automatically. Automatic music generation is divided into two phases: training and generation.

3.1. Overview

As shown in Figure 1, the training phase consists of three training steps: Preprocessing, CVG training, and conditional GAN training. The preprocessing step receives and parses MIDI files and extracts the four elements, t, p_t, c_t, e_t, required during training. t is a constant ranging from one to n, which indicates the time step index in one MIDI file, which is converted to vector by one-hot encoding. p_t represents the musical content of one phrase at time t and is a combination of several tracks where each track is a combination of bars, given that every bar will be encoded into a matrix with normalization. c_t represents the position of p_t in the MIDI file. t can be regarded as a constant representing the absolute position, but c_t is a percentage that represents a relative position, which will be encoded into a vector by one-hot encoding. e_t is a binary value encoded by one-hot encoding. It is used to indicate whether the p_t is the last phrase in the MIDI file.

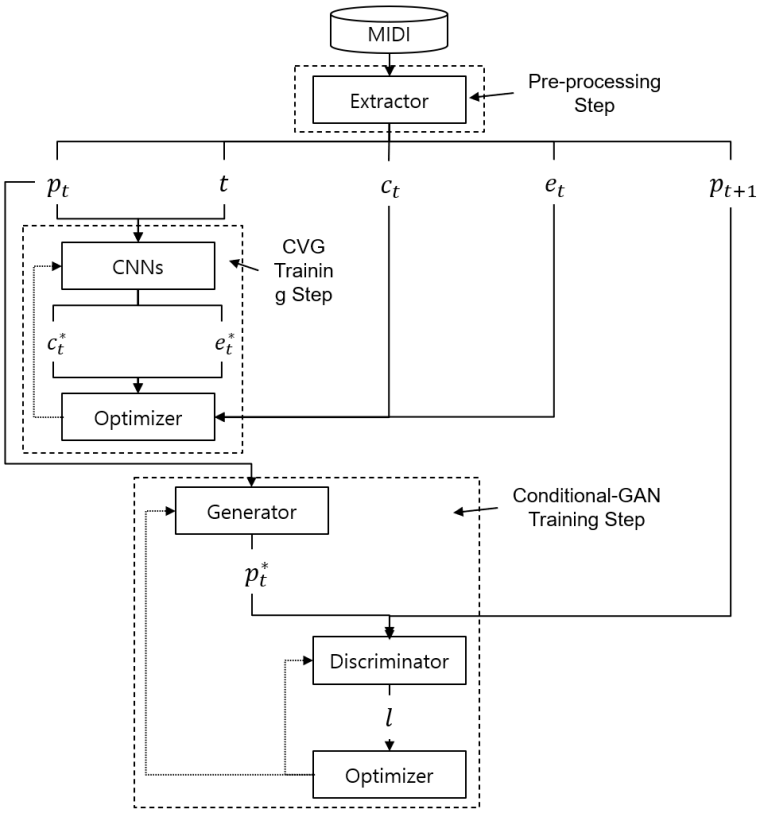

Figure 1. Training phase of melody composition system.

Second, p_t and t will be input to the conditional vector generator (CVG). CVG consists of two parts: one part is utilized to generate the relative position vector to represent the generation process, and the other part can predict whether the generation is to end. CVG generates conditional vectors c_t^* and e_t^* based on CNN, the conditional vectors are the simulation conditional vector different from c_t and e_t extracted in training data. The optimizer continuously updates the CVG by comparing the generated result with the actual label extracted from the MIDI file. Although c_t^* and e_t^* are generated, they do not participate in the music generation step during the training phase. However, they are necessary for the generation phase, so CVG should be trained in advance.

Third, p_t and c_t are input into the generator. Because the proposed method utilizes the conditional GAN model, the generator can accept c_t as an additional relative position, vector to be input. The generator generates p_{t+1}^* at time $t+1$ based on the p_t under the influence of c_t. Finally, the discriminator compares the p_{t+1}^* generated by the generator with the p_{t+1}. The loss l of the comparison will be converted into the gradient by the optimizer to update the discriminator as well as the generator. When the training phase concludes, the trained CVG, generator, and discriminator are obtained. In the training phase, the CVG training and conditional GAN training are independent of each other. Although the two modules are trained separately, they have a strong correlation because of the same input data.

As shown in Figure 2, the generation phase comprises three steps: CVG executing, phrase generation, and postprocessing. First, p_t^* and t are input to the CVG to generate c_t^* and e_t^*. Because in the generation phase, there is no way to obtain the c_t and e_t directly like that in the training phase, the CVG must predict and generate c_t^* and e_t^*. The generated c_t^* and e_t^* are input into the checker together. The checker judges whether to end music generation based on the e_t^*. In addition, the c_t^* will also be utilized as input to the checker to assist in judgment. This c_t^* can indicate the current music generation point—i.e., whether intro, verse, chorus, or outro. Through this relative position vector, the checker will try to avoid ending the music generation if the system is not at the outro. Second, the p_{t+1}^*

generated by the generator is utilized as an input to loop through this process again, and t is also continuously incremented through the Counter. Because the shape and value representation of the phrase as input and output differ, reshape and value transform is required when $Phrase^*_{t+1}$ is utilized as an input again. Third, when the generation process is completed, all generated phrases are integrated into a new musical composition through postprocessing.

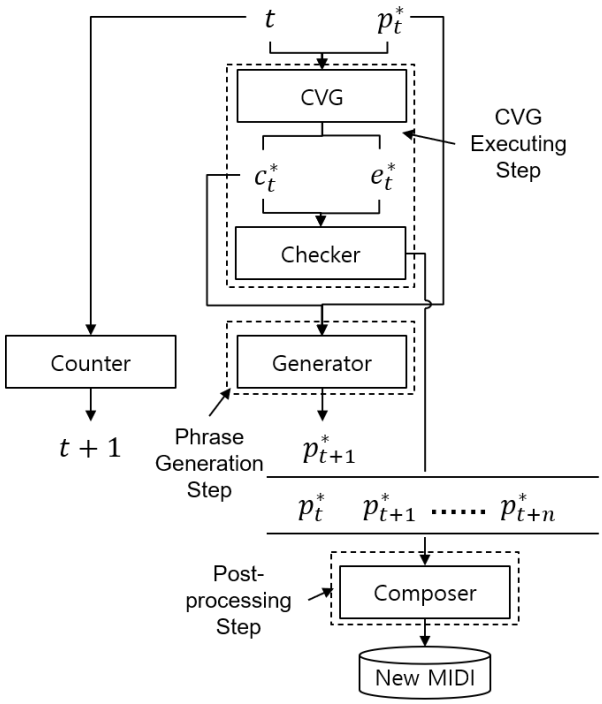

Figure 2. Generation phase of melody composition system.

3.2. Extractor

During preprocessing, there are four tasks to be completed. First, the tracks are divided into several phrases in order. Second, the phrases at time t and $t + 1$ are selected to be encoded into a matrix. Third, during encoding, each phrase is converted into a matrix in units of the bar, as shown in Figure 3. Each bar can be treated as a sub-matrix of the phrase matrix. Because the MIDI file may contain multiple tracks, multiple bars are arranged independently in the track-dimension. The vertical axis of the bar matrix is utilized to represent the pitch of the note, the horizontal axis is the time axis of the bar, and the unit is tick. Finally, the bars are connected to compose the encoded p_t and p_{t+1}.

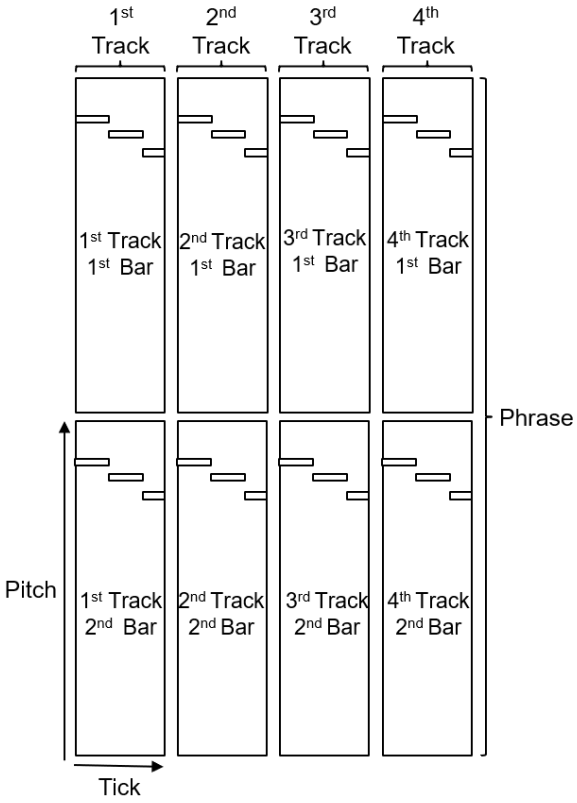

Figure 3. Result of encoding.

3.3. Structure of the Conditional Vector Generator

CVG provides the conditional vector required for music generation for the generator. CVG can control music generation, including structure and end. Hence, CVG is also composed of two parts, as shown in Figure 4. One is utilized to control the music generation structure c_t^*, and the other one to control the generation's ending e_t^*. CVG generates conditional vectors according to the following algorithm.

Algorithm 1 Conditional Vector Generator Training

Input: p_t, t, c_t, e_t
Output: c_t^*, e_t^*
For $t \leftarrow 1$ to Input $(p_{1,2,\ldots,t})$
    $h_1 \leftarrow Covolution(p_t)$
    $h_2 \leftarrow Concatenation(h_1 \oplus t)$
    $c_t^* \leftarrow Softmax(h_2)$
    $loss_c \leftarrow Mean\ Square\ Error(c_t^*, c_t)$
    $e_t^* \leftarrow Sigmoid(h_2)$
    $loss_e \leftarrow Mean\ Square\ Error(e_t^*, e_t)$
$Optimizer(loss_c, loss_e)$
Output c_t^*, e_t^*

There are obvious characteristics in each structure of music, especially the phrase at the end part of music is easy to be identified. CVG predicts the music structure of the current phrase by analyzing the phrase p_t and time step index t; at the same time, by changing the activation function of the output layer to sigmoid, it outputs the probability of predicting the end of the music.

CVG utilizes the two simple CNN to achieve this goal. P_t and t are input into CNN, and then the vector of the control structure and the judgment result of the end generation output, respectively.

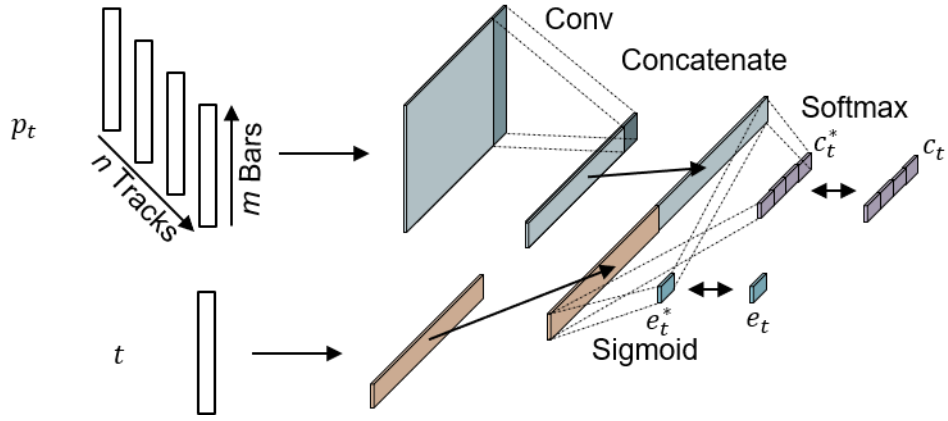

Figure 4. Structure of the conditional vector generator.

3.4. Structure of the Generator of INCO-GAN

Part of the conditional GAN model is the generator. In this section, the structure of the generator is described in detail. As shown in Figure 5, p_t and c_t are inputted into the Generator. After entering the generator, the p_t will be broken down into several tracks, with each track containing several bars. At the same time, three noise vectors will be generated to match each bar. The *Chords* vector could control anything about the music that changes per bar, such as general rhythmic style, without being specific to any track. The *Style* vector's job is to control the general dynamic nature of the music over time. The *Groove* vectors are not passed through the temporal network but are instead fed straight through to the bar generator unchanged. However, unlike in the *style* vector, there is a distinct *Groove* input for every track, meaning that the generator can utilize these vectors to adjust the overall output for each track independently [9].

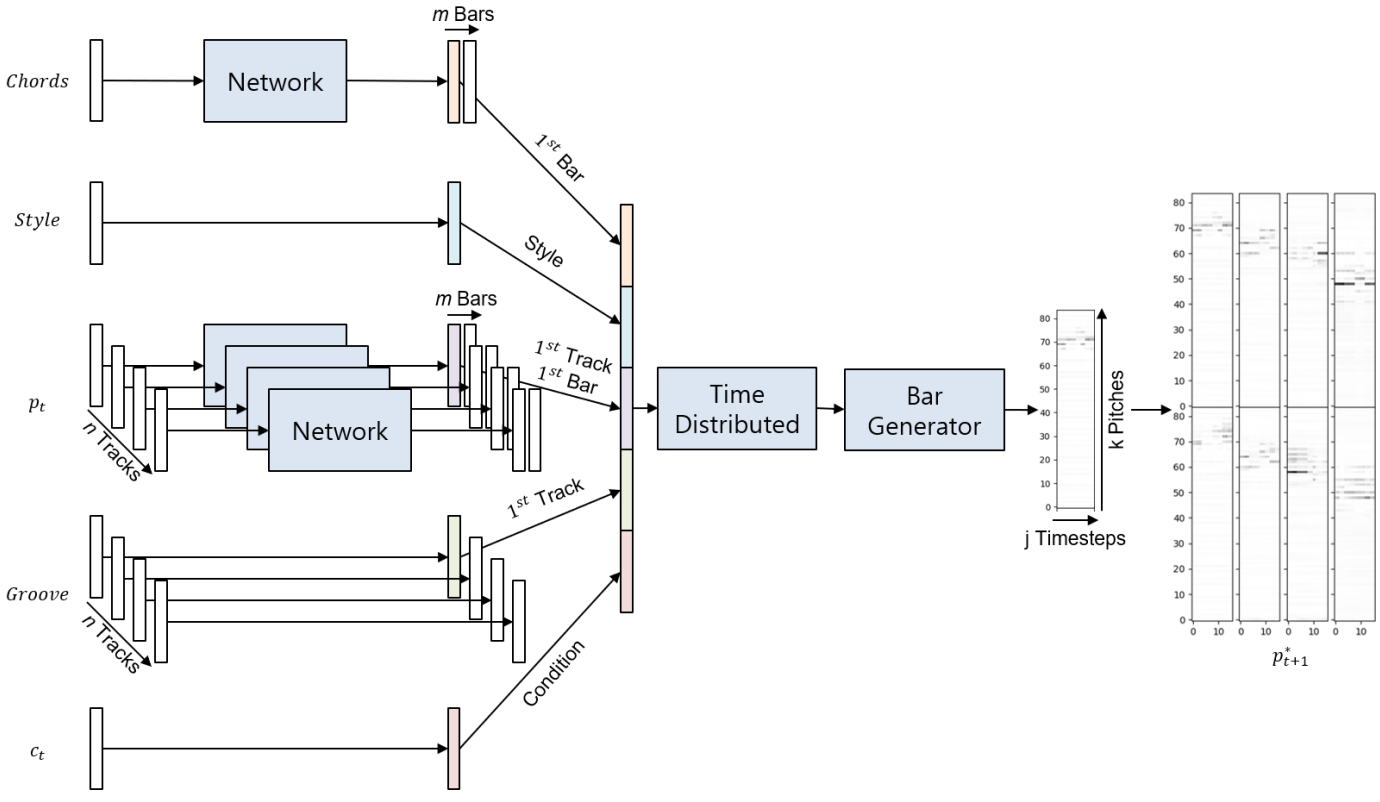

Figure 5. Structure of the generator.

In the process of generating p_{t+1}, MuseGAN's Bar Generator [9] was borrowed. Therefore, $Phrase_{t+1}$ is not directly generated but is a combination of multiple bars that are given time relationships generated by Bar Generator. Before that, the time-distributed dense layer adds time relationships to the input at each time by sharing weights.

3.5. Structure of the Discriminator of INCO-GAN

As shown in Figure 6, the input of the discriminator can be p_t or p_t^*, and output is the judgment result. The dimension of p_t or p_t^* is (n, m, j, k), where n is the number of tracks, m is the number of bars in one track, j is the number of timesteps in a bar, and k is the range of pitch. The *phrase* needs to be reshaped into (m, j, k, n) before being input to the first convolutional layer.

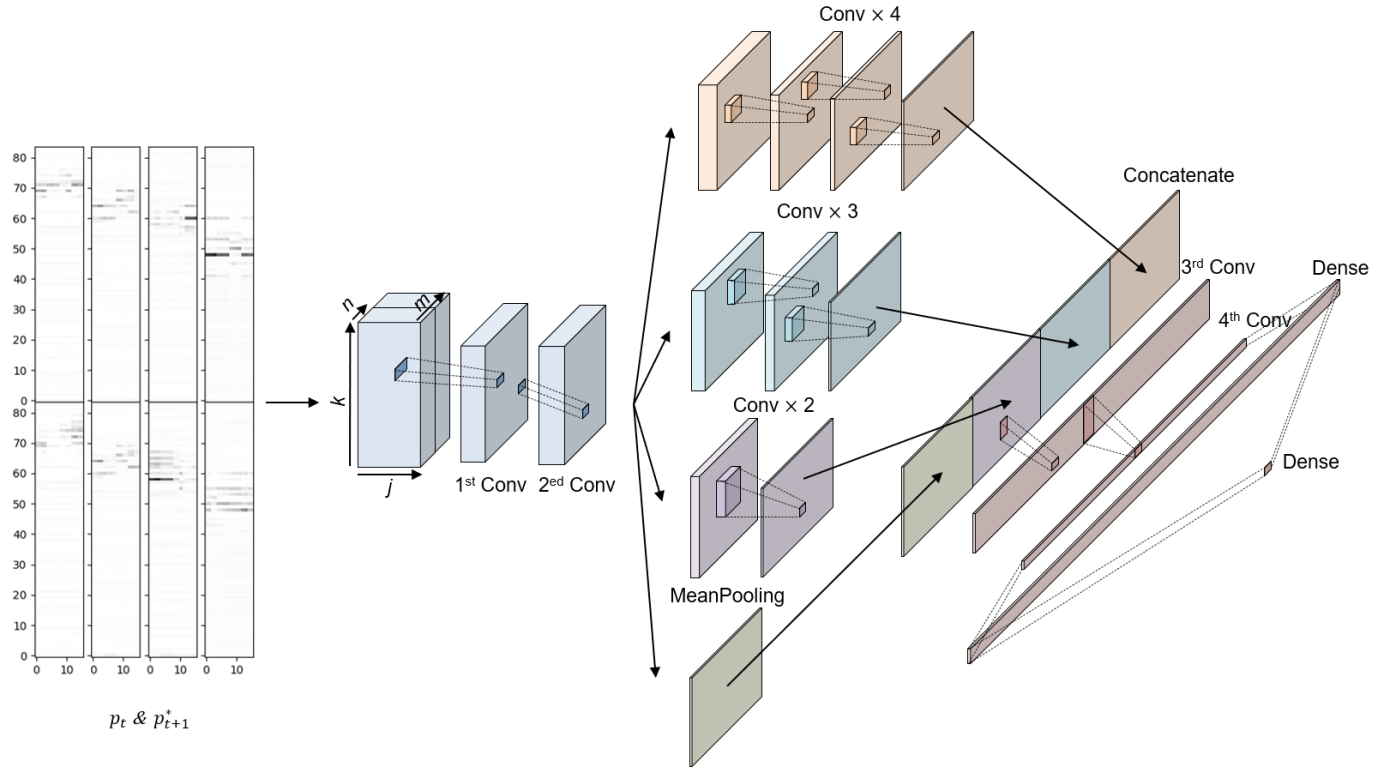

Figure 6. Structure of the discriminator.

The p_t or p_t^* is folded into $(1, j, k, n \times f)$ in m-dimension through the first convolutional layer, where f is the number of filters in the convolutional layer. The second convolutional layer does not change the dimension of the input, but it can add more nonlinear relationships to it, which is helpful for training.

The inception model subsequently folds the input on the j-dimension because the relationships among notes are complicated on the timesteps. Multiple filters of different sizes are utilized to convolve the input, in addition to adding a mean pooling layer to consider larger, more global features. These features are then combined through the Concatenate layer. The third convolutional layer folds the input according to the scale on the k-dimension. The fourth convolutional layer continues to fold on the k-dimension based on the output of the third layer. Finally, the judgment result is obtained through two fully connected layers based on the flattened output of the convolutional layer.

4. Experiment and Results

In the experiments conducted, the process and result of the conditional vectors generation method based on CVG and the music generation method based on INCO-GAN

were extracted to verify the proposed method. The results of music generation were then obtained.

4.1. Experimental Environment

The model of the proposed method can be divided into the CVG and conditional GAN. In the training phase, because the CVG and conditional GAN were trained, respectively, they need to be verified separately. The accuracy and loss value obtained during the training process were utilized to determine whether the CVG and INCO-GAN were trained well. Table 2 shows the parameters of the CVG and conditional GAN during training.

Table 2. The parameters of the conditional generative adversarial network approach using an inception model (INCO-GAN) and conditional vector generator (CVG) during training.

INCO-GAN		**CVG**	
Input dim	(2, 16, 84, 4)	Learning rate	0.01
Critic learning rate	0.001	Optimizer	SGD
Generator learning rate	0.001	Batch size	128
Optimizer	Adam		
Grad weight	10		
z dim	32		
Batch size	64		
n tracks	4		
n bars	2		
n steps per bar	16		
n pitches	84		

When the training was completed, test data were utilized to evaluate the trained model. In CVG evaluation, p_t and t of music by human composers was input into the trained CVG. The CVG predicted $Condition_t^*$ and End_t^* based on the input. Subsequently, the accuracy was obtained by comparing it with the real $Condition_t$ and End_t^*. It appeared that the testing process is like the training process, but by utilizing real data that are not used in the training phase, it can be verified that the trained CVG has good generalization.

Objectively testing the conditional GAN is a challenge, owing to both the diversity of music, and the purpose of this paper is to generate music that sounds like that from a human composer without producing the same music. In this paper, an evaluation method based on frequency and time was utilized, which is a more appropriate method to compare the music from human composers and the generated music.

The experimental environment was composed of Windows 10, i5-10400, NVIDIA GeForce GTX 1650 4 GB, DDR4 32 GB. The proposed system was developed in Python, and the conditional GAN model was implemented using the deep Keras backend on TensorFlow. MIDI files were processed by the music21 library.

4.2. Experimental Data

Experiments were conducted on the Lakh MIDI dataset [20]. The Lakh MIDI dataset is a collection of 176,581 unique MIDI files, 45,129 of which have 13 genre labels since they were matched and aligned to entries in the Million Song dataset, such as pop/rock, electronic, country R&B, jazz, Latin, and international.

4.3. Experimental Results

Figure 7 shows the accuracy and loss of the CVG. Because the output of CVG consists of two parts. They were conditional vectors that control the music generation process and the judgment result of whether the music generation is over.

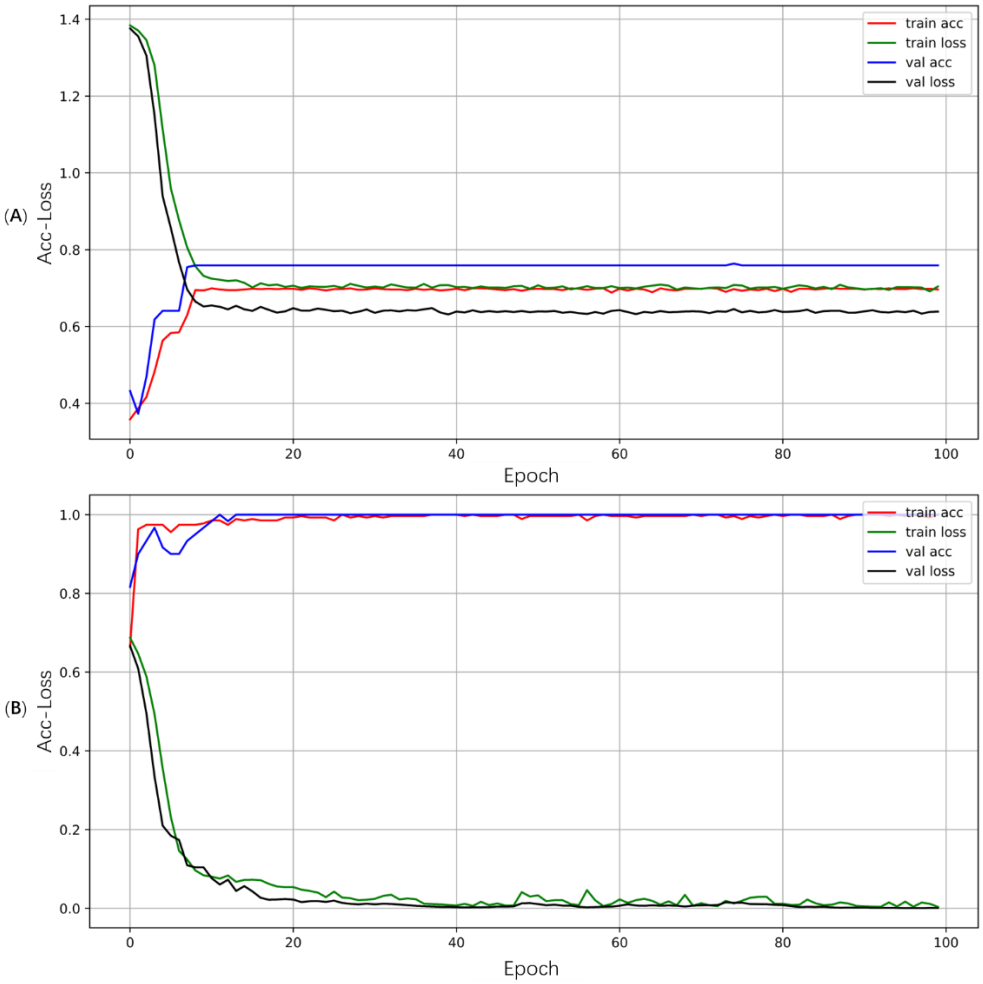

Figure 7. Accuracy and loss of the structure of generation and judgment of end. (**A**) Structure of generation; (**B**) Judgment of end.

In Figure 7A, the accuracy of the training set was finally floating around 70%, while the accuracy of the validation set was around 78%. Because the complexity of music and the same combination of notes may appear at any portion of the music, therefore, this result can be accepted, and it is helpful for generating diverse music.

In Figure 7B, the accuracy of the training data and validation data remained above 99% at the end. This result shows that the music has the same features at the end portion, and the feature is easy to be recognized.

Figure 8 shows the loss of the conditional GAN. The discriminator calculated loss from three aspects are for music by human composers, music generated by INCO-GAN and construct weighted average between human-composed music and generated music. The generator had only one loss.

The black line (valid) represented the loss of the discriminator's judgment of music by human composers. The black line quickly converged near 0, which indicated that the discriminator had a good discrimination ability for music by human composers. The green line (fake) gradually moved towards 0 in the first 500 epochs, which indicated that the generator was learning how to generate music during this period. After 500 epochs, the green line fitted around 0, which indicated that it was difficult for the discriminator to distinguish the music generated by the generator. However, even so, there is still a gap between the green line and the black line. The red (validity interpolated) line represented the result of combining the black line and the green line. The loss of the

generator represented by the yellow (generator) line can be regarded as the gap with the music by human composers.

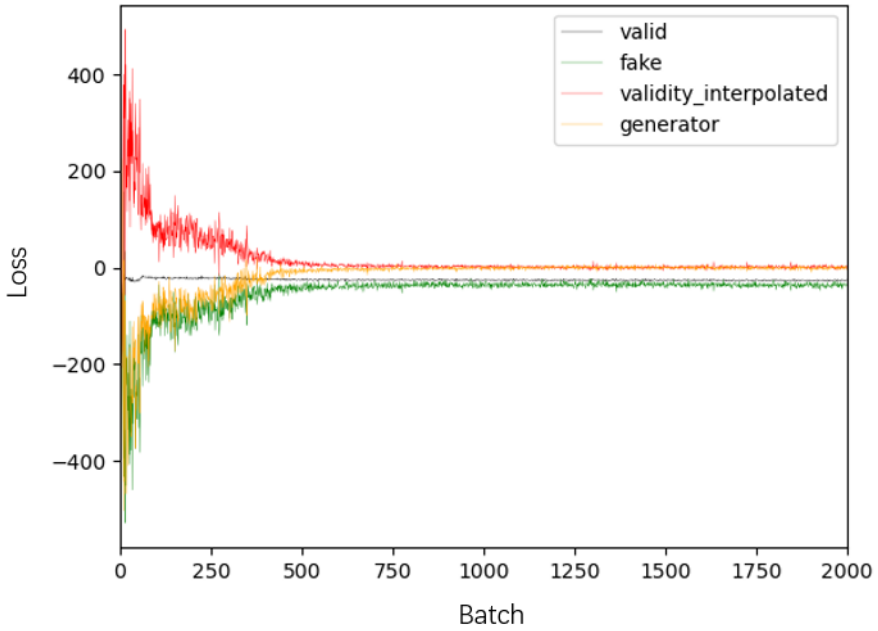

Figure 8. Loss of the conditional generative adversarial networks (GAN).

Table 3 shows the test results of the CVG by utilizing real music as test data. The test results show that the accuracy of music structure and end prediction has achieved 77.2% and 99.9%, respectively. These results proved that the trained model was generalized and can provide correct conditional vectors for INCO-GAN to generate music.

Table 3. The musical characteristics and their musical meanings.

Model	Test (%)
Structure	77.2
End	99.9

To evaluate the generated music, the pitch frequency of the music generated by the proposed method was compared with human composers' music. Figure 9 shows the comparison result of music by human composers and the music generated by the proposed method and MuseGAN [14]. MuseGAN is the main reference research in this paper, so it was utilized as a comparison. The frequency corresponding to each pitch of the music generated by INCO-GAN and MuseGAN was converted into the vector, and the cosine similarity between the music by human composers was calculated to be 0.987 and 0.978, respectively. These values show that the generated music by INCO-GAN was very similar to the music created by human composers in characteristic. The similarity of INCO-GAN was 0.09 higher than that of MuseGAN, which proved that the discriminator based on the inception model could better grasp the music features and give feedback to the generator.

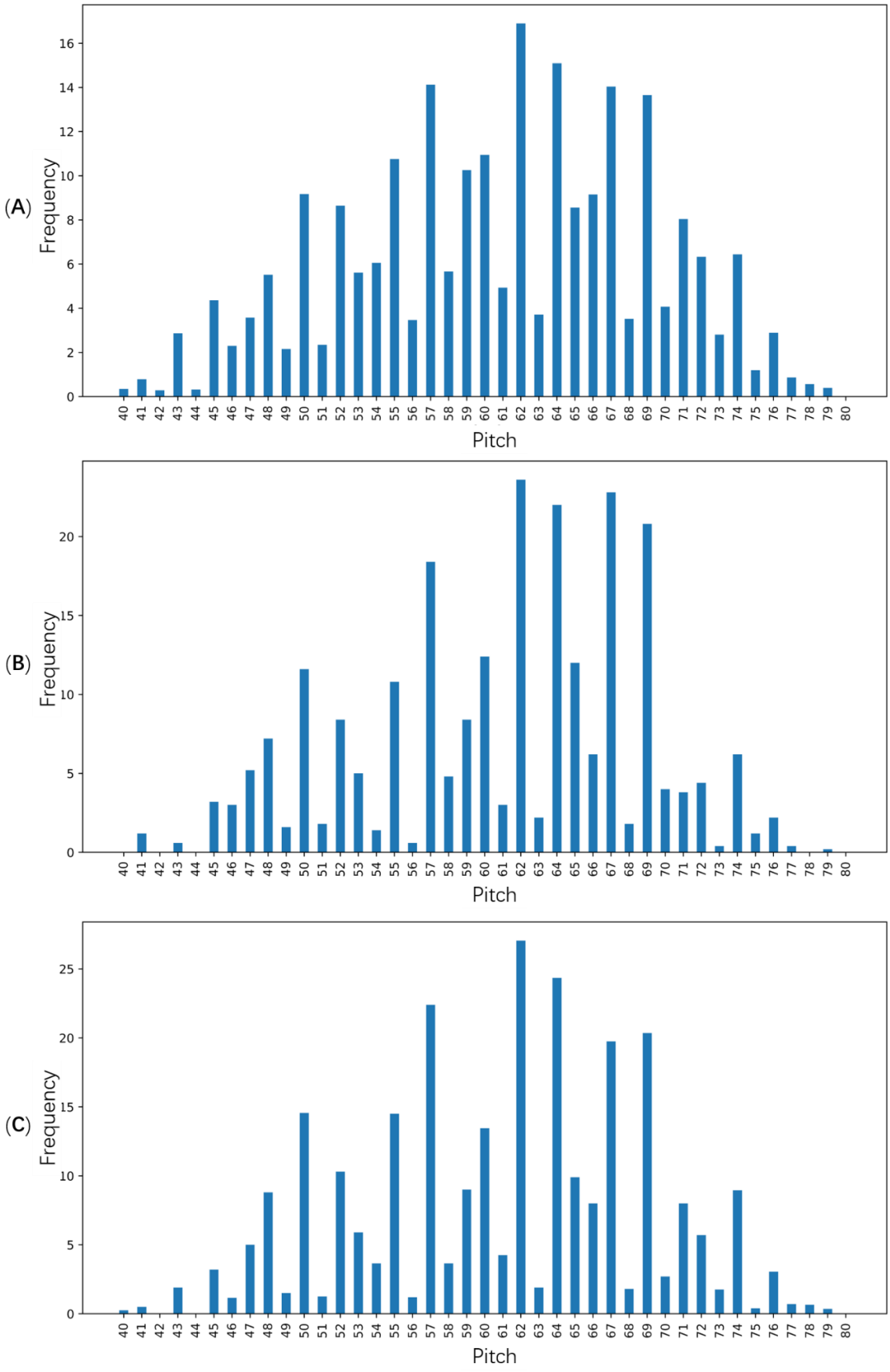

Figure 9. Comparison based on pitch frequency. (**A**) The pitch frequency of music by human composers; (**B**) The pitch frequency of generated music by MuseGAN; (**C**) The pitch frequency of music by INCO-GAN.

In addition, human composers' music and generated music can be more easily compared through visualization as shown in Figure 10a was converted from 4 music produced by human composers, Figure 10b is converted from 4 generated music.

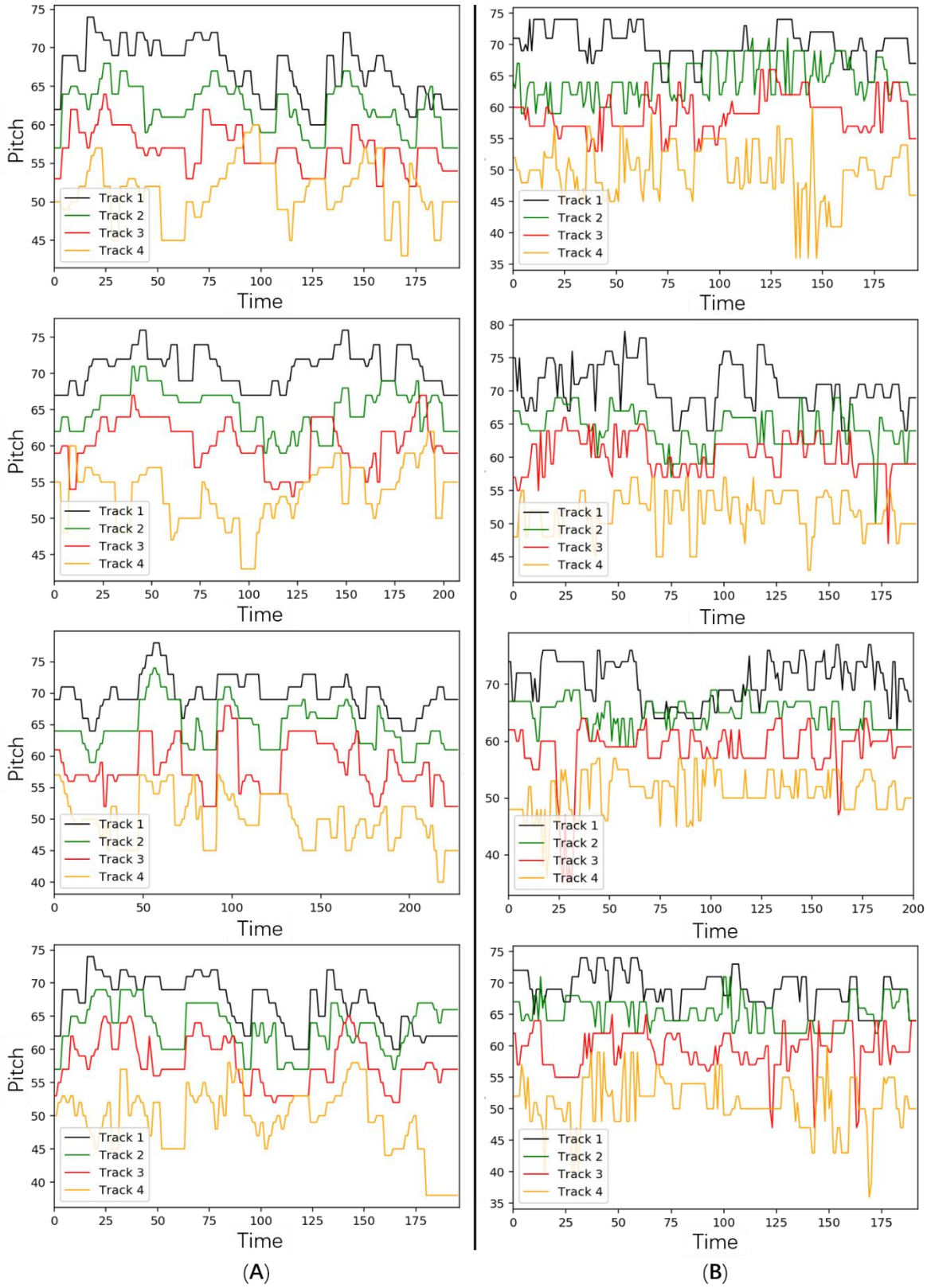

Figure 10. Comparison based on visualization. (**A**) Visualization of midi by human composers; (**B**) Visualization of generated midi by INCO-GAN.

5. Discussion

Through the comparisons, the length of human composers' music and that of generated music are usually different, but both distributions of length are similar, which means that CVG controlled the length of the generated music by the conditional vector by analyzing the generated music by human composers. In music by human composers, the tracks were clearly layered, and there was rarely any crossover. The same characteristics were shown in generated music.

However, in music by human composers, the four tracks show the same trend at the same time point. This feature was not obvious in the generated music. Moreover, the pitch of the generated music occasionally changes frequently, which was not common in music by human composers. This shows that in the generated music, the relationship between the tracks and the duration of the notes are still some gaps with music by human composers.

6. Conclusions

The paper proposed an automatic music generation method based on the conditional GAN (INCO-GAN) model with an inception model. In the proposed method, INCO-GAN is completely autonomous and capable of generating whole musical compositions with variable-length by controlling the input conditions. Moreover, the series of operations such as adding a time distribution layer that considers the time relationship of a data sequence and utilizing the inception model improved the quality of the generated music. To verify the proposed method, the generated MIDI files were evaluated. In the experiment, an evaluation method based on frequency and time was utilized, which was a more appropriate way to compare the musical characteristics in the extracted music by human composers and the generated music. In particular, it could be seen from the cosine similarity of up to 0.987 between the frequency vectors that the music produced by this method is very similar to that made by human composers.

Author Contributions: Conceptualization, S.L., and Y.S.; methodology, S.L., and Y.S.; software, S.L., and Y.S.; validation, S.L., and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Science, ICT, Korea, under the High-Potential Individuals Global Training Program (MSIT) (2019-0-01585, 2020-0-01576) supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from Colin Raffel and are available https://colinraffel.com/projects/lmd/ accessed on 1 December 2020 with the permission of Colin Raffel.

Acknowledgments: This research was supported by the Ministry of Science, ICT, Korea, under the High-Potential Individuals Global Training Program (MSIT) (2019-0-01585, 2020-0-01576) supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP).

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

References

1. Li, S.; Jang, S.; Sung, Y. Automatic melody composition using enhanced GAN. *Mathematics* **2019**, *7*, 883. [CrossRef]
2. Kim, E.; Jang, S.; Li, S.; Sung, Y. Newspaper article-based agent control in smart city simulations. *Hum. Cent. Comput. Inf. Sci.* **2020**, *10*, 44. [CrossRef]
3. Li, S.; Jang, S.; Sung, Y. Melody extraction and encoding method for generating healthcare music automatically. *Electronics* **2019**, *8*, 1250. [CrossRef]
4. Mozer, M.C. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connect. Sci* **1994**, *6*, 247–280. [CrossRef]
5. Jaques, N.; Gu, S.; Turner, R.E.; Eck, D. Generating music by fine-tuning recurrent neural networks with reinforcement learning. In Proceedings of the 3rd Deep Reinforcement Learning Workshop, Barcelona, Spain, 9 December 2016.

6. Hadjeres, G.; Pachet, F.; Nielsen, F. Deepbach: A steerable model for bach chorales generation. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1362–1371.
7. Chu, H.; Urtasun, R.; Fidler, S. Song from PI: A musically plausible network for pop music generation. *arXiv* **2016**, arXiv:1611.03477.
8. Magenta. Available online: https://magenta.tensorflow.org (accessed on 16 November 2020).
9. Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv* **2016**, arXiv:1611.09904.
10. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
11. Amiriparian, S.; Freitag, M.; Cummins, N.; Schuller, B. Sequence to sequence autoencoders for unsupervised representation learning from audio. In Proceedings of the DCASE 2017 Workshop, Munich, Germany, 16–17 November 2017.
12. Yang, L.C.; Chou, S.Y.; Yang, Y.H. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In Proceedings of the 2017 International Society of Music Information Retrieval Conference (ISMIR), Suzhou, China, 24–27 October 2017.
13. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
14. Dong, H.W.; Hsiao, W.Y.; 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.
15. Billard, M.; Bishop, R.; Elsisy, M.; Graves, L.; Kolokolova, A.; Nagisetty, V.; Northcott, Z.; Patey, H. Non-sequential melody generation. In Proceedings of the ICLR 2020 Conference, Addis Ababa, Ethiopia, 20–30 April 2020.
16. Huang, Y.; Huang, X.; Cai, Q. Music generation based on convolution-LSTM. *Comput. Inf. Sci.* **2018**, *11*, 50–56. [CrossRef]
17. Han, Z.; Lu, H.; Liu, Z.; Vong, C.M.; Liu, Y.S.; Zwicker, M.; Chen, C.P. 3d2seqviews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation. *IEEE Trans. Image Process.* **2019**, *28*, 3986–3999. [CrossRef] [PubMed]
18. Szegedy, C.; Loffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
19. Lakh MIDI Dataset. Available online: https://salu133445.github.io/lakh-pianoroll-dataset/dataset.html (accessed on 16 November 2020).
20. Liu, H.M.; Yang, Y.H. Lead sheet generation and arrangement by conditional generative adversarial network. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018.