FLAGNET : FEATURE LABEL BASED AUTOMATIC GENERATION NETWORK FOR SYMBOLIC MUSIC

Anonymous authors
Paper under double-blind review

ABSTRACT

The technology for automatic music generation has been very actively studied in recent years. However, almost in these studies, handling domain knowledge of music was omitted or considered a difficult task. In particular, research that analyzes and utilizes the characteristics of each bar of music is rare, even though it is essential in the human composition. We propose a model that generates music with musical characteristics of bars by the conditional generative adversarial network and analyzes the good combination of the sequence of which characterized bars for symbolic-domain music generation by Recurrent Neural Network with Long short term memory layer. Also, by analyzing symbolic music data as image-like based on a relational pitch approach, it increases the utilization of the data set with arbitrary chord scales and enables the use of generational results extensively. The resulting model FLAGNet generates music with the understanding of musical domain knowledge while handling inputs like a minimum unit of note, length of music, chart scales, and chord condition.

1 INTRODUCTION

The importance of computing technology related to automatic music generation has been studied for quite a long time and has been emphasized more recently. With increased access to a wide range of music data sets, various generation models based on WAV-based technologies, Symbolic presentation-based technologies, etc. have been proposed, and they have sought to find models that produce ”successful” music with various deep learning technologies such as Recurrent Neural Network, Convolution Neural Network, Reinforcement Learning, etc. However, the generation proposed by these technologies is somewhat different from producing ”good” music. In fact, Automatic composition often constructs a model by identifying only the suitability of notes and the degree of similarity with the dataset, whereas human composition constructs a bar by using each note’s skillful application and uses them to create phrases of music. In other words, the existing generation models have difficulty making proper use of domain knowledge of music.

There is a recent study (Akama, 2019) that aims to apply the musical domain knowledge to the generation model. In this case, high-level domain knowledge, such as rhythm, contour, and fragment & consolidation, is applied as a relatively simple algorithm with MIDI. It is utilized in music generation to achieve very successful results. This study of applying domain knowledge of music at the human level not only increases the scalability of the generation Model but also helps to generate more music that is good to listen to.

Nowadays, RNN and its applications are the technologies that are being used most often in the Neural Network Model for Music Generation. (Wu et al., 2019) The biggest reason for this is that the creation of notes over time is important because music is a time sequence. However, studies like MIDINet (Yang et al., 2017) and the MuseGAN (Dong et al., 2017) model are using CNN, especially GAN. They have proven to use CNN has quite valid and has better performance for Learning time.

So, the idea we came up with to apply the Musical Domain Knowledge to the Generation Model is to label the musical skill in the bar-level Notes and to build an environment where MIDI can be Generation by using cGAN (Mirza & Osindero, 2014). But there is a limit to creating ’music’ simply by organizing a bar-level MIDI with cGAN because music is time sequence, so difficult to utilize the relationship between bars. Thus, based on a given music dataset, we combine how the sequence of musical skills can be attractive and how the bars created by using the simple RNN model, i.e.

handling the flow of music. Also, to make image size can be properly reduced, and the Musical Skill can be maintained while processing MIDI Bar with the image we utilize image processing based on Relational Pitch Change. This approach allows the use of the music in the train without relying on the chord scale of given music in the dataset and provides a wider range of possibilities for the music produced by matching the music to the 12 basic chart scale (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Also, to distinguish major scale and minor scale, we can generate music with chord conditions with a major scale. The resulting model, FLAGNet can use the musical skill contained in the music bar to understand the musical domain knowledge, analyze the sequence of the musical skills to control the overall flow of music, and process the generated images to make symbolic music, as well as to utilize all 12 basic chart scales and handling chord condition to distinguish major/minor scale.

2 METHODOLOGY

In this paragraph, we explain the theoretical background of FLAGNet.

2.1 MUSICAL NOTATIONS

Before explaining the model, we define some music-related words and explain the musical theories used in this study.

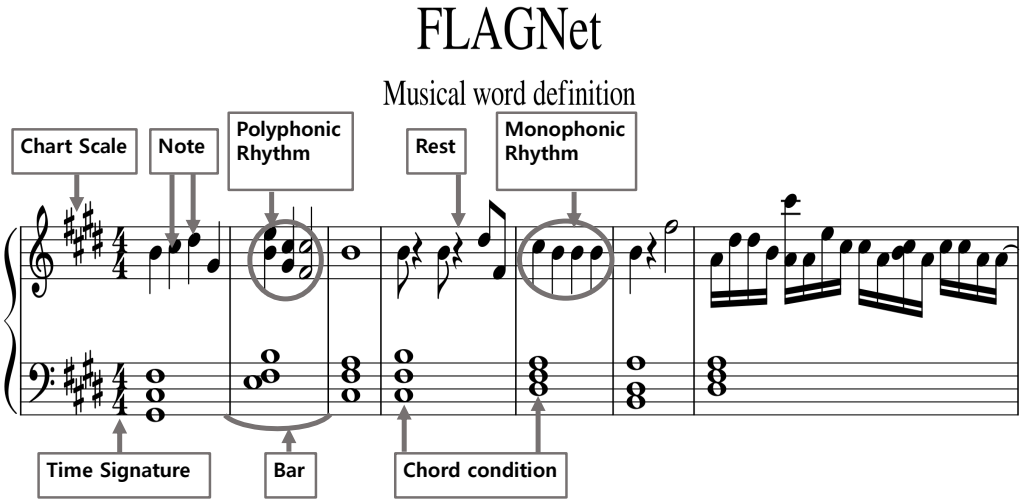

Figure 1: Simple music sheets and words which are used in this paper. Note that these sheets have no artistic meaning. It is to explain the theoretical background.

Most words can be explained using Figure 1.n_{th} note is related to time signature and bar. Time signature $3/4$ means that 3 numbers of 4_{th} note construct 1 bar. So, if one bar at 4/4 beats is divided into 16 number of parts, one minimum unit is divided into 16th notes and the triplet note of 16th note is the smallest unit in the case of 24 number of parts. our study uses these 2 minimum units.

There are some words which are not in Figure 1. Note's duration means 'How long do notes play', Note's timing means 'When do notes play', Note's pitch means 'How high or low note is', and Note's velocity nearly means 'How strong do notes play'. These words are the musical words frequently used in this paper.

2.2 ARCHITECTURE OF FLAGNET

The basic structure of FLAGNet is shown in Figure 2. We first process a given MIDI Dataset to configure Bar Matrices and attach the corresponding Skill Labels and Direction Labels depend on MIDI information. The skill labels contain the corresponding musical skill information for a given bar matrix. These labels are defined by heuristic musical algorithm, detailed label defining method is explained in appendix A. Direction Labels will specify the direction of pitch change between right next bar. This Direction label will simply use in the last of MIDI Processing. We can find skill label sequences for each music easily because music contains a sequence of the bar, and each bar has skill labels. These label sequences are used for the learning of RNN. Since music is a sequence of bars, so finding relations between bars is important, and the RNN model is good for this. In the process of learning cGAN, we did blur processing with binary matrices to the 'blurred' matrix by using a Gaussian filter, to increase the learning performance of cGAN. Overall, RNN provides the skill sequence prediction, the cGAN will then generate the matrices. Based on the musical information inputs such as chord scale, music length, and prediction of pitch change. We finally do tuning and processing with matrix sets with controlling some musical elements to create a complete MIDI.

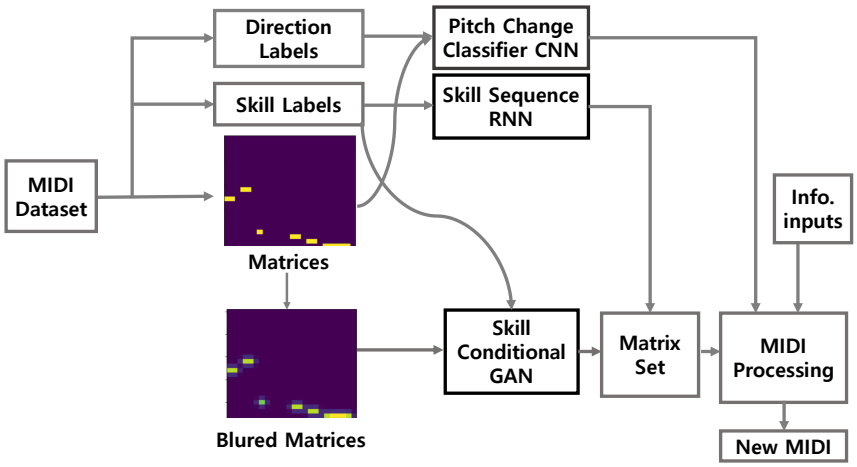

Figure 2: Overview of the FLAGNet model. It starts with the MIDI dataset, and finally generates new MIDI. Bolded box means models that are actually learning.

2.3 MATRIX REPRESENTATION OF MIDI

The encoding method which separates MIDI by bars and uses as a matrix has been used quite extensively in music generation research. (Li et al., 2019) Also in this study, a matrix based on the 'piano-roll' form is used, but the data form is optimized by processing the pitch information. We just remain the pitch information only relational difference. With this method, the musical flow is maintained so we just need to determine which pitch to use during the decoding process. Pitch defining method is explained later. MIDI bars are expressed in binary matrix $\boldsymbol{M} \in \{0,1\}^{n \times m}$. Matrix $\boldsymbol{M} \in$'s column change means a change in time, and row change means a change in Pitch. By matching the lowest pitch note to the lowest row at each matrix, we can use a smaller number of pitches to optimize the matrix size small while not losing the bar's skill properties and pitch change information of the bars. We create MIDI by matching matrices to a given chart scale during the MIDI processing. With this 'piano-roll' like method, we can use MIDI as an image, and can intuitively grasp the music flow just by watching images.

2.4 CONDITIONAL GAN

First of all, there is a explain of general Generative Adversarial Nets (Goodfellow et al., 2014). This contains two adversarial models, Generator Model G and Discriminator Model D. Generative model G learns the distribution of a given data, and based on that, it learns to generate data in a structure similar to a given data, from low dimensional noise $z \in \mathbb{R}$. Discriminator model D proceeds with learning in the form of distinguishing whether a given data is generated from a Generative model G or real data. Based on the given data X, G and D will proceed with the learning based on the minimax algorithm, and the detailed equation is as follows.

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim P_X}[\log(D(x))] + \mathbb{E}_{z\sim P_Z}[\log(1-D(G(z)))], \tag{1}$$

Where $x \sim P_X$ denotes the distribution of real data, and $z \sim P_Z$ denotes the distribution of random noise.

In conditional generative adversarial nets, Condition $y \in \{f_1, f_2, \ldots, f_n\}$ is added to the GAN model. G and D will be trained about a given Label y. Detailed equation may be represented by some adding of condition about y at the GAN's equation form as shown above equation (1),

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim P_X}[\log(D(x|y))] + \mathbb{E}_{z\sim P_Z}[\log(1-D(G(z|y)))], \tag{2}$$

In conclusion, this allows Generator G to generates the MIDI image corresponding to a given label f_i.

2.5 IMAGE DECODING AND MIDI PROCESSING

First, normalizing the generated matrix image to have the minimum elements equal to 0, and maximum elements equal to 1. We leave only the elements that have a larger value than some fixed value and change the rest to 0. After that, we use the local maximum filter to detect the peak matrix $\boldsymbol{M}_{(P)}$. Let the 3×3 matrix $[i-1, i+1] \times [j-1, j+1]$ as $\boldsymbol{C}_{ij}$,(if $i-1, i+1$ or $j-1, j+1$ are out of matrix size, then just consider 2 rows or columns) the peak matrix that filtered by local maximum filter is following the below form:

$$M_{(P)i,j} = \begin{cases} 1, & \text{if } \max(\boldsymbol{C}_{ij}) = M_{i,j} \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

Use each peak at the beginning of the note and for each peak, determine the duration of the note by checking elements at the right to the peak note, whether the size of the right elements of the peak is larger than some fixed value which we consider in implementation. And determine the velocity of the value the same as $M_{j,l}$ where (k, l) is the position of the peak. In this way, the matrix image generated by generator G can be expressed again in the form of 'piano-roll' like binary matrix $\boldsymbol{M} \in \{0,1\}^{m\times n}$.

Then, to identify the starting pitch for a given relative pitch matrix, define the available set $\mathbb{S}_n$ of each bar matrix $\boldsymbol{B}_n$'s starting pitch as follows:

$$\mathbb{S}_n = \begin{cases} \{x | SP_n \le x \le SP_n + 11\}, & \text{if pitch change classifier's prediction of} \\ & \text{previous bar is 'up' or } \boldsymbol{B}_n \text{is starting bar} \\ \{x | SP_n - 11 \le x \le SP_n & \text{otherwise} \end{cases}, \tag{4}$$

where Standard Pitch, SP_n is the last note pitch of the previous bar, or 48(C4) if $\boldsymbol{B}_n$ is the starting bar. Note that we use 12 elements in set $\mathbb{S}_n$ because 1 Octave in music has 12 notes, with 7 white notes and 5 black notes.

If the first note is determined, we can easily decide the pitch of the remaining notes in the bar. So, available note pitches set for each bar can be easily obtained with each first note in $\mathbb{S}_n$. For example, if relative pitch change of bar is form of $[0, +3, -2]$, and $\mathbb{S}_n$ is $[48, 49, 50, \ldots, 59]$, then available note pitches set contains $[48, 51, 49], [49, 52, 50], \ldots, [59, 62, 60]$. Based on a given chart scale input, we define the scale score SC as the Number of notes that matches with the given scale. We use the note pitch set that has the highest SC value. If some available note pitch sets have the same scale score, select the set closer to the Standard Pitch set.

3 IMPLEMENTATION

Based on the above methodology, this paragraph describes how the actual implementation was carried out. You can see implementation in `https://github.com/slslslrhfem/FLAGNet`

3.1 DATASET

We use The Patterns for Prediction Development Dataset(PPDD) in MIREX 2019 (Downie, 2006), which is a subset of the Lakh dataset (Raffel, 2016). Lakh dataset is widely used in symbolic music-related studies because it contains a lot of symbolic music with relatively simple and easy-to-handle structures. The largest version of the PPDD dataset contains 10000 monophonic-only MIDI, 10000 homophonic MIDI, etc. We only use monophonic-only MIDI in this study. Since we can use these data without relying on the chart scale by our encoding method, use a monophonic dataset of all chart scales. However, we only use MIDI only with $4/4$ time signature. The reason is that we judging that handling all kinds of time signature can reduce the model's performance a lot.

3.2 FEATURE LABEL DEFINING

It is necessary to designate the skill labels separately for each bar because there are no labels in MIDI data that can explain the bar's musical skill directly It is general to choose clustering in defining the label of the data, but we thought that this could be against the objective of applying the musical domain knowledge. Therefore, we labeled each bar by utilizing the statistical information obtained using MIDI data in a Naïve algorithm. We first collected pitch change information, notes duration information, notes timing information, and rest information for each bar based on MIDI data. Labels were then attached to each bar based on the Naïve algorithm using this information. We defined label such as 'up_stepping', 'repeating', 'one_rhythm', 'leapping_twisting', and so on. You can see the detailed algorithm in appendix A. . Since it is difficult to handle some noisy cases by directly using labels by Naïve algorithm, we used multilabel classifier (Tsoumakas & Katakis, 2007) based on given matrices and labels. Label smoothing technology was applied to handling the noisy case, and weight balancing technology was applied to carry out learning because the number of matrices corresponding to labels could be different. Then again, matrices were re-categorized by this classifier to organize the dataset so that each matrix could have one of the most suitable skill labels. Then, we use this data in FLAGNet.

3.3 FLAGNET

Depending on the shape of the bar given, it is important to determine whether the pitch will go up or down on the next bar. We use a simple CNN-based model for the pitch change classifier. There are four-pitch change labels, which are Up, Down, Meaning_less, and Last_bar. The Meaning Less label is used if the Pitch change does not occur in the next bar, or if the next bar is completely Resting Bar, and the Last_bar label is used for the last bar in each song. Although this classifier model trained with 4 labels, we use only two labels of 'Up' and 'Down' in the prediction of this model.

In learning cGAN, there are several ways to improve the performance of this model. It is known that GAN performance can be improved by controlling the learning balance between the generator and the discriminator, and by applying label smoothing technology. (Salimans et al., 2016) To maintain the balance between the generator and the discriminator, we set the degree of learning of the discriminator to be only 0.8 times that of the generator, and if the loss of the generator rises above a certain level, we made the discriminator learn 0.4 times of the generator to adjust the balance between the relatively slow generator and the fast discriminator. In the case of our study, a simple form of label smoothing can also improve performance, but since a given data body is Binary Data consisting of 1 or 0, it is more flexible to learn image data through Gaussian Smoothing, rather than simply using label smoothing.

The Skill Sequence RNN uses the LSTM (Hochreiter & Schmidhuber, 1997) layer to further improve performance. After receiving the skill sequence input of length 5, the following skill sequence is pre-dictated, and the case that length less than 5 is handled by 0-padding. When defining the next skill in the actual generation process, the prediction of the RNN model is 1-normalized and we use this as probability distribution to determine the next skill to be generated. Also, to prevent the

excessive repetition of the same kill, reduce the prediction value by 1/3 times when the same skill is repeated.

In the MIDI processing process after image generation and decoding, we suggest several tricks for the musical detail in addition to the methods described in 2.4. If the generated note has a too-short length to make the rest of the music, the duration of the note has been increased to fill the rest. This prevents the song from being over-empty. We can also control notes that deviate from the chart scale. We can ignore or shift them. And if the generated notes create homophonic notes, we select only the note which has the largest velocity value. And make it a monophonic note. This is used because the dataset contains monophonic MIDI only, so homophonic rhythm may not be stable. All these tricks are flexible to use, so If we want to remain homophonic notes, note deviating from the scale, or short-duration notes, then we may not use these tricks.

We chose two forms of generation. The first one is just using implementations and the tricks described above to create a monophonic rhythm. Now we call this the FLAGNet-1 model. However, there is a limit to the clear distinction between the major scale and minor scale. In other words, it was difficult to distinguish between major and minor chart scale using the same pitch set, such as C major and A minor.(Both are using [C,D,E,F,G,A,B].) To handle this and also increase musical completeness, we decided to add a chord condition with a given chart scale. Seven diatonic scales based on Roman numeral analysis are I, ii, iii, IV, V, vi, vii^o. (Mehegan, 1984) We use these chords, excluding vii^o. These are randomly selected for each bar and used as base chords. Also, in the process of obtaining the Scale Score, it is not matched to the whole chart scale but matched only to the six chords above. (Each chord has 3 elements) Through this, we can generate more complete music with 12 basic major scales. Minor scale can be implemented in a similar way, but in this study, we remain this task as future work and generate music only within the range of major scale. Now, we call this the FLAGNet-2 model.

4 RESULT

In this paragraph, we describe the results of the FLAGNet's creation and explain the method for the survey by real people and its result.

With shifting notes deviating from scale and other tricks applied, one of the good sample music sheet produced by the FLAGNet-2 model is in Figure 3.

As shown in Figure 3, music generation is carried out well and the creation of music itself is quite creative, but it also appears that it seems somewhere less stable. And also, musical features like repeating, or melody leaping are evident in some bars, but there are bars that are difficult to define clearly.

To check the performance of the music generator, 15 people were invited to participate in the survey. The actual survey is conducted as follows: 2 monophonic symbolic music made of the FLAGNet-1 model was created with a minimum unit of 16th note and 24th note, respectively. And there is one MIDI sample that the MidiNet model composed and one song composed by a real person, were presented in the monophonic symbolic music. All the songs were extracted into wav using the same virtual piano environment. Participants have no idea which model each song was created from or who composed it, but just know what music is with only monophonic rhythm or with chord condition. They evaluate this as a maximum of five points for four categories: Creativity, Comfortability, Reality, and Musicality. We explained each category as:

◯ **Creativity** : If the possibility of using a given rhythm is enormous, or if you think it has not existed before, it has high creativity points.

◯ **Comfortability** : For any reason, if there are many inconvenient elements to listen to, the song may be less comfortability points.

◯ **Reality** : If a given song seems to be composed by a person, it has high reality points.

◯ **Musicality** : Including the above elements, you can judge the level of music by considering all the factors you can think of. 5 points for best song.

Table 1 shows the average score of the survey. Bolded value means the highest values for each metric in monophonic music, and music with chord condition.

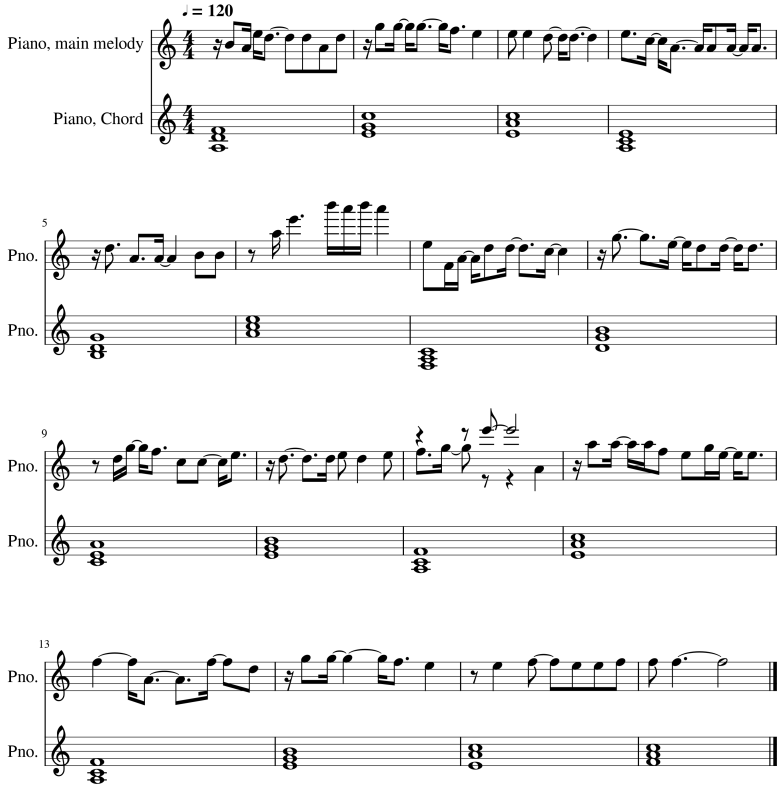

Figure 3: Good sample music generated by FLAGNet-2 model

Table 1: Human survey comparison of model performance against true data: 15 participants were surveyed and asked to score examples on a scale of 1-5 for the qualities listed. The rows listed as human music.

Songs	Creativity	Comfortability	Reality	Musicality
monophonic human music	2.4	2.8	2.467	2.533
monophonic midinet music	2.933	**2.867**	**3.2**	**3.2**
monophonic FLAGNet music(16th)	2.933	2.8	2.933	2.733
monophonic FLAGNet music(24th)	**3.267**	1.733	2.067	2.267
with chord human music	2.733	**4.4**	**4.133**	**4**
with chord midinet music	2.667	3.4	3.333	3.067
with chord FLAGNet music(16th)	**3.733**	3.133	3.333	3.4
with chord FLAGNet music(24th)	3.4	3.467	3.467	3.4

Judging from the results, FLAGNet's generation for only monophonic rhythm was hardly better than MIDINet's music and human music. We think this is because comfortability and musicality can be relatively low because the problem is caused by the unclear handle of the major/minor scale and the method of structure that cannot directly follow the priming melody, so the structure of music is not stable. However, in the generation with the given chord, the scale can be clearly distinguished, and the problem related to the existence of the priming melody also some resolve because a combination of the chord is reasonable in domain music knowledge. So it is performing well on musicality indicators, especially in the area of creativity. It was also found that the use of the minimum unit of 24th note in monophonic rhythm composition was very unstable compared to the use of 16th note or others, but these phenomenons were found to be resolved in situations supported by the chord. We think this is because too many offbeats are directly heard in the ear when using the 24th note alone by Main Melody.

Overall, FLAGNet can produce creative music extensively, and their musicality is well-received, but somewhat lacking in comfortability. However, this can be solved to some extent in addition to the chord, and we think it will be a very meaningful model musically if research on scale and priming melody is conducted through future work.

5 CONCLUSION

We have proposed FLAGNet, a generator model that processes symbolic music by the bar and applies the cGAN, and RNN given to each bar to capture the musical domain knowledge, which has succeeded in creating symbolic music, which is more creative and musical than the existing generation model like MIDINet. By proposing this model, we have shown that the completed music can be automatically generated through the application of skills that a small unit of music has.

In the above study, the definition of skill was carried out based on a Naive algorithm for each bar, but it is expected that performance can be raised or utilized using labeling with a more 'precise' method, or using a more practical label such as 'buildup-bar' or 'verse-bar' etc. Also, there was difficulty in major/minor scale controlling in the FLAGNet-1 model, which is considered complementary through musical studies. In addition, if learning is applied not only to monophonic rhythm and solo-track music, it is possible to create multi-track music with various instruments as well as a generation of music with homophonic rhythms.

In addition to the areas related to Music Generation, it is thought that the approach in this study could be useful in the areas dealing with large units of data. It is expected that further studies on how to divide the data into smaller pieces, analyze it based on smaller units, and identify their relationship.

REFERENCES

Taketo Akama. Controlling symbolic music generation based on concept learning from domain knowledge. In *ISMIR*, pp. 816–823, 2019.

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. *arXiv preprint arXiv:1709.06298*, 2017.

J Stephen Downie. The music information retrieval evaluation exchange (mirex). *D-Lib Magazine*, 12(12), 2006.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Shuyu Li, Sejun Jang, and Yunsick Sung. Melody extraction and encoding method for generating healthcare music automatically. *Electronics*, 8(11):1250, 2019.

John Mehegan. *Tonal and Rhythmic Principales: Jazz Improvisation I*. Watson-Guptill Publications, 1984.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. A hierarchical recurrent neural network for symbolic melody generation. *IEEE Transactions on Cybernetics*, 50(6):2749–2757, 2019.

Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

A APPENDIX

First, get information about the bar: Change of Note pitch, Duration of notes and rests, Timing of notes, and Real Playtime of Notes, 'Real Playtime' means the length between one notes timing and right next note's timing. If there is a rest between two notes, it means it has longer 'real playtime' than duration.

The following contents are labels and explanations. n is a hyper-parameter that we can control to make labels balanced. In implementation, we used 66% for rare case like 'up_leaping', 'down_leaping'. and others use 75%. Some case like 'twisting' has some different condition, but same as others, n is hyper_parameter.

'repeating': defined as repeating when all but $n\%$ or more of the total notes are the same.

'up_stepping': defined as up_stepping if more than $n\%$ of the total notes, or excluding n, are stepping up or the same note, i.e. rising below 3Note on a semi-negative basis.

'down_stepping': defined as down_stepping if more than $n\%$ of the total notes, or excluding n, are stepping down or the same sound, i.e. falling below 3Note on a semi-negative basis.

'up_leaping': defined as up_leaving if more than $n\%$ of the total notes, or excluding n, are leaping up, i.e. if they rise above 3Note on a semi-negative basis.

'down_leaping': defined as down_leaving if more than $n\%$ of the total notes, or excluding n, are leaping down, i.e. lower than 3Note on a semi-negative basis.

'stepping_twisting': defined as stepping_twisting if there are more than four notes, excluding n, in the form of repeated rises and falls of less than 2Note.

'leaping_twisting': defined as leaping_twisting if there are more than four notes, excluding n, in the form of repeating the rise and fall of more than 3Note.

'fast_rhythm': defined as fast_rhythm if there are more than 9 notes within 1 bar.

'One_rhythm': The real-time of all notes, that is, if the time until the next note is the same, define it as One_rhythm.

'triplet': defined as a triplet if a triplet exists based on real-time. For a generation with a minimum unit as 16th notes, this skill is ignored.

'Staccato': Based on Duration Time, if the negative Duration of n% or more is less than 0.16667 (minimum unit for this study.), defined as Staccato.

'continuning_rhythm': If the ratio of 'Rest' in the pitch change list is less than 25%, it is defined as continuing_rhythm.

'no_skills': if the bar has no property among these skills, then we label this bar as 'no_skills' and do not use for generation.

While constructing multi-label sets, we clear the impossible cases like case which have both 'up_stepping' and 'down_stepping', to improve the performance of the classifier. Also, this labeling method occurs noise for some cases, so we considered this classifier as a noisy label classifier. and try to denoise.

B APPENDIX

You can download 8 samples for the human survey. https://drive.google.com/file/d/1tn5vORxt0zs_q5Uma93G89pAje-PkDj9/view?usp=sharing

We decided The Composer of Human music is familiar with a team member to prevent license issues. Monophonic music is the first version of esSeL-Animate, and homophonic music is esSeL-Slow Snow. Both musics are available at https://grafolio.naver.com/searchList.grfl?query=esSeL#middleTab