A study on conditional music generation with GAN

Seong-bin Yoon, *Hyun-Chul Choi
ICVSLab, Department of Electronic Engineering

Yeungnam University

Gyeongsan, Republic Of Korea

ysb120883@yu.ac.kr, pogary@ynu.ac.kr

Abstract—**This paper is about a study using musegan [1] that succeeded in generating music by generating multi-track music according to the framework of GAN. Musegan has a limitation in that it cannot control the output through conditions. Therefore, we offer models that can generate music through genre conditions. We show that the suggested models can generate music with conditional genres through qualitative and quantitative examination of experimental results.**

Keywords; Music Generation; Generative Adversarial Network; MuseGAN

I. INTRODUCTION

There are few variances because most contemporary music uses Western 12 tone scales, and money chords are somewhat standardized. As a result, newly composed music may be exposed to plagiarism and copyright infringement even when there is no purpose to do so.

To solve the above problems, we propose a conditional music generation model. We borrowed musegan [1] model to implement the music generation model. This model's drawback is that the user cannot select the genre of the music when we generate it. Therefore, we altered musegan model to generate music following genre. Experiments on performance differences were conducted by applying CGAN [3], BIGGAN [4], and ACGAN [5] to musgan.

Section 2 introduces the studies conducted to provide condition information to music generation and GAN [2]. Section 3 explains the proposed method . Section 4 shows the dataset and metrics for the experiment and the results of the experiment, and Section 5 concludes.

II. RELATED WORKS

Musegan [1] generates a midi file with five multi-track. Based on GAN [2], the model can train high level features like music style. However, there is a limitation that music cannot be created by genre.

Many studies have been conducted to improve the performance and control the output according to the desired condition by giving the condition to the GAN as additional information. CGAN [3] can artificially generate data of a desired class and uses labels to train generators and discriminators. BIGGAN [4] generates high-resolution image of 512x512. However, BIGGAN has a disadvantage that class leakage [4] occurs, and it cannot cleanly create an image of a label related to human. ACGAN [5] creates a conditional image by making the discriminator determine the class of the input image. In previous studies, quality decreased when the number of classes increased, but ACGAN has the advantage of being stable and complying with quality because it can learn Generator and Discriminator for each subset after dividing large data sets by class.

III. METHODS

A. Model

In this paper, we propose three methods: CGAN-based, BIGGAN-based, and ACGAN-based.. The overall structures are shown in Figure 1. In all three methods, the structure of the generator and discriminator is the same as that of musegan [1].

The CGAN-based model is the first model devised among the three methods. CGAN]3] has a simple structure. Therefore, there is no significant difference in structure when musegan and CGAN are combined. So, we tried to make a model with similar performance to musegan.. The musegan was modified as follows. First, by concatenating the genre condition C transformed into an embedding vector and noise vector Z, it enters the input of the generator to generate fake music X_{fake}. Second, the embedding vector Z and real music X_{real} or X_{fake} are concatenated, and this information is input to the discriminator, which then outputs the results of the real and fake discrimination.

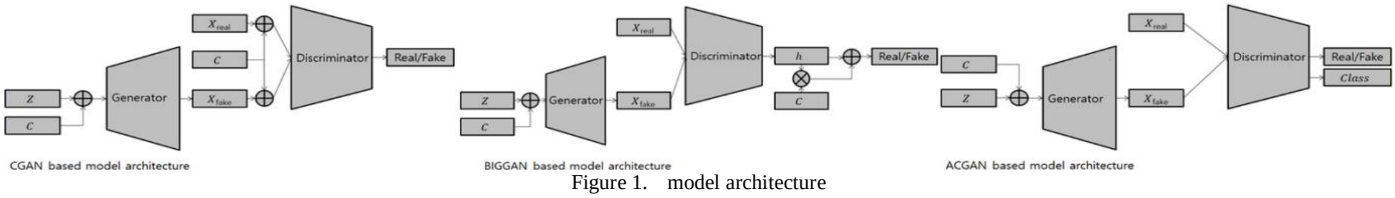

Figure 1. model architecture

BIGGAN [4] was successful in producing high-resolution images. So when musegan and BIGGAN are combined, it would be possible to generate better quality music than the CGAN-based model. This is why the BIGGAN-based model was devised. We implemented the BIGGAN-based model by changing the CGAN-based model as follows. This model projects C transformed into an embedding vector and h. After concatenating the projected with h, it is used as a result of real and fake discrimination. h denotes the output of the discriminator.

The ACGAN-based model additionally learns the classifier. Since ACGAN [5] showed good performance in the existing class condition image generative model, we applied ACGAN to musegan. In doing so, we aimed to develop a model that distinguishes between genres more effectively than models based on CGAN and BIGGAN. In contrast to the CGAN and BIGGAN-based models, we added a softmax layer at the end of the discriminator to get the class prediction results. And we didn't give the discriminator any condition information,. When X_{fake} or X_{real} enters the discriminator input, the output is the result of class prediction and the distinction between real and fake.

B. Model Training

The loss used to train the model was used through modification in WGAN-GP [6], the same as that used in musegan [1]. The WGAN-GP has been modified so that G (generator) and D (discriminator) receive additional conditional inputs called C (condition).

$$GP = \lambda E[(\left|\nabla D\left(\alpha X - \left(1-\alpha G(Z|C)\right)|C\right)\right| - 1)^2] \quad (1)$$

GP is the gradient penalty. λ=10, α is a random value between 0 and 1. *X* is real data and *Z* is a latent vector.

$$L_D^{WGAN-GP} = E[D(X|C)] - E[D(G(Z|C)|C)] + GP \quad (2)$$

$$L_G^{WGAN-GP} = E[G(Z|C)] \quad (3)$$

$L_G^{WGAN-GP}$ and $L_D^{WGAN-GP}$ are the losses used for training generators and discriminators of CGAN and BIGGAN-based models.

In the case of the ACGAN-based model, the loss is different because the condition input does not enter the discriminator and the classifier needs to be additionally learned.

$$L_D^{AC} = E[P(C|X)] + E[P(C|G(Z|C))] \quad (4)$$

$$L_G^{AC} = E[P(C|G(Z|C))]. \quad (5)$$

L_D^{AC} and L_G^{AC} are the loss of generator and discriminator for classifier learning. So, the loss used in the ACGAN-based model can be expressed mathematically as follows.

$$L_D^{ACGAN\ based} = E[D(X)] - E\left[D\left(G(Z|C)\right)\right] + GP + L_D^{AC} \quad (6)$$

$$L_G^{ACGAN\ based} = L_G^{WGAN-GP} + L_G^{AC} \quad (7)$$

IV. Experiments

A. Metric

We used EB, UP, and PP [1] for quantitative analysis, and the three metrics are the same as those used by musegan [1].

Empty Bar rate (EB) indicates the ratio of the used bars among all the bars. The closer EB is to 1, the more frequently the track is used. EB was used to check the difference in frequency of the instrument use between the dataset and the generated samples.

Used Pitch (UP) indicates the number of notes used. The higher UP indicates that the track uses more notes. UP was used to check the pitch range of the instrument between the dataset and the sample.

Polyphonic rate (PP) refers to the ratio of time steps in which several pitches are played among the total time steps. Closer to 1 means that multiple pitches are played. To make sure the instrument was playing correctly, PP was used. For example, for drum and bass, the PP value should be low, as it is not often the case that several notes are played at once. On the other hand, guitar, piano, and string play multiple notes using chord than drum and bass, so the PP value should be higher than that of both instruments.

Table 1 shows EB, UP, and the number of songs by dividing the dataset by genre and Table 2 shows EB, UP and, PP of generated samples. PP was not used in Table 1. This is because PP better represents the performance characteristics of the instrument than the genre characteristics and the instruments of the songs in the dataset play properly. On the other hand, there is no guarantee that the instrument will perform properly in the generated music. Therefore, PP was used only in Table 2 to evaluate whether the instrument is playing properly. Experimental statistics were produced using 1000 samples, and the values were expressed in the form of average values (standard deviation values).

Table 1. EB, UP, and Number of songs of dataset

Genre	EB(%)					UP					Number of songs
	drum	piano	guitar	bass	string	drum	piano	guitar	bass	string	
blues	0.501 (0.185)	0.734 (0.328)	0.618 (0.267)	0.897 (0.052)	0.789 (0.234)	12.543 (4.943)	27.500 (13.253)	19.800 (6.823)	13.100 (3.208)	29.200 (10.078)	22
contry	0.527 (0.185)	0.796 (0.236)	0.790 (0.257)	0.899 (0.097)	0.786 (0.240)	10.307 (4.229)	23.980 (9.258)	22.056 (8.180)	12.968 (5.541)	23.473 (9.653)	512
electronic	0.622 (0.202)	0.546 (0.300)	0.514 (0.324)	0.742 (0.191)	0.796 (0.215)	11.600 (4.699)	17.798 (9.906)	15.898 (9.061)	11.111 (5.667)	25.450 (8.990)	889
folk	0.572 (0.217)	0.696 (0.308)	0.731 (0.260)	0.898 (0.107)	0.872 (0.116)	11.522 (5.055)	21.652 (9.915)	20.379 (7.356)	12.957 (5.401)	26.478 (9.833)	45
international	0.547 (0.210)	0.677 (0.300)	0.709 (0.263)	0.817 (0.158)	0.815 (0.189)	12.368 (4.887)	21.952 (12.259)	20.852 (8.357)	13.689 (6.595)	26.995 (10.739)	206
jazz	0.586 (0.203)	0.745 (0.259)	0.623 (0.291)	0.830 (0.165)	0.759 (0.245)	13.097 (5.450)	29.581 (13.124)	21.081 (8.854)	15.145 (5.875)	29.306 (10.973)	157
latin	0.565 (0.212)	0.680 (0.293)	0.714 (0.291)	0.840 (0.132)	0.849 (0.171)	13.277 (5.334)	22.798 (10.076)	19.832 (7.889)	13.225 (5.220)	26.762 (10.200)	360
newage	0.562 (0.196)	0.717 (0.294)	0.709 (0.324)	0.809 (0.159)	0.894 (0.121)	11.364 (3.489)	21.727 (9.989)	21.394 (6.619)	12.697 (5.665)	32.545 (14.418)	67
poprock	0.551 (0.203)	0.659 (0.300)	0.707 (0.284)	0.825 (0.149)	0.829 (0.195)	11.578 (4.610)	21.134 (10.787)	20.239 (8.090)	12.767 (5.394)	25.907 (10.142)	4345
rap	0.649 (0.238)	0.571 (0.277)	0.605 (0.299)	0.730 (0.183)	0.765 (0.234)	11.763 (4.544)	19.880 (16.256)	15.117 (6.808)	11.312 (4.528)	24.991 (10.808)	164
raggae	0.647 (0.211)	0.666 (0.305)	0.584 (0.295)	0.811 (0.122)	0.838 (0.160)	11.941 (4.249)	17.706 (9.344)	15.451 (8.015)	11.745 (7.401)	25.745 (9.258)	45
RnB	0.620 (0.209)	0.719 (0.277)	0.621 (0.285)	0.830 (0.136)	0.848 (0.153)	11.436 (4.603)	23.531 (12.108)	17.755 (8.355)	13.811 (5.567)	28.139 (10.990)	397
vocal	0.518 (0.226)	0.726 (0.307)	0.662 (0.306)	0.836 (0.168)	0.848 (0.170)	10.600 (5.157)	27.867 (15.177)	21.444 (8.900)	13.933 (5.454)	30.400 (11.966)	114

B. Dataset

For the data set used in the experiment, we used the LPD-5-cleansed dataset which was utilized in musegan [1]. LPD-5-cleansed offers an id list for information on 13 genre labels.

We pre-processed LPD-5-cleansed to train three models. In Table 1, the data imbalance of the number of songs by genre is very severe. A model cannot learn in moderation if there are not enough songs. So we excluded genres with fewer than 100 songs. In Table 1, there is no part where the EB differs by more than 0.3 for each instrument according to the genre. Aside from the difference between the piano UP values for the jazz and electronic genres, there are no other notable differences in UP value. In this respect, it is difficult to distinguish between genres from a numerical point of view, so we decided to use only genres that can be easily distinguished when listening. As a result, songs were generated in country, electronic, jazz, and poprock genres. The number of songs in other genres except jazz was randomly selected to be 200 and the experiment was conducted

C. Results

Table 2 shows EB, UP, and PP of samples generated by CGAN, BIGGAN, and ACGAN-based models in 4 genres. Looking at the PP values of the CGAN-based model, the PP values for drum and bass are less than 10%, and PP values for piano, guitar, and strings are around 50%, which is higher than drum and basses. This shows that the instruments are generating music properly because the PP values of drums and basses is lower than the PP values of guitars, pianos, and strings. In the CGAN-based model, the piano EB and UP of the electronic genre are higher than the dataset values by 27% and 9 pitches, respectively. And EB of guitar is higher than that of the dataset by about 27%. Except for this, the differences in EB and UP for all genres are similar to the values in the dataset, at about 15% and 5 pitches, respectively. Through this, it can be seen that the CGAN-based model can generate music similar to the distribution of the dataset. When we listened to the samples generated by the CGAN-based model, the quality of the music was good, but the music genres were not clearly distinguished. The result values of generated samples and those of the dataset were similar since the CGAN-based model just concatenates the conditions. But when we listened to the samples, the genre distinction was not properly done.

Looking at the PP value of the BIGGAN-based model, the PP value is low in the drum, piano, and guitar, and high in bass and string. This means that instruments do not generate music properly. The EB values for piano and guitar in the BIGGAN-based model are extremely low in comparison to the dataset. And the EB value of the bass and string is too high at 1. Also, the UP value of all instruments is significantly lower than that of the dataset. This means that the BIGGAN-based model cannot generate music according to the distribution of the dataset. The music was of very poor quality when we listened to the sample produced by the BIGGAN-based model. The music was poorly generated on both a quantitative and qualitative level. The BIGGAN-based model projects and concatenates the discriminator output. Unlike the other two models, the difference is that the output value of the discriminator changes. As a result, the model did not train and could not generate music properly.

In the ACGAN-based model, the PP value of piano, guitar, and string is close to 50%, which is higher than the PP value of drum and bass. The PP value of drum and bass is less than 12%. This shows that the instruments are generating the music properly. In the ACGAN-based model, EB of the guitar in jazz genre is higher than that of the dataset by about 20%. And the EB of drums in the poprock genre is lower than that of the dataset by about 23%. Except for this, EB and UP are 18% for

Table 1. EB, UP and, PP of generated samples

		EB(%)					UP					PP(%)				
		drum	piano	guitar	bass	string	drum	piano	guitar	bass	string	drum	piano	guitar	bass	string
CGAN-based model	country	0.521 (0.073)	0.828 (0.165)	0.772 (0.162)	0.896 (0.079)	0.751 (0.148)	15.168 (4.675)	26.872 (6.054)	20.188 (4.814)	10.832 (2.409)	20.052 (4.900)	0.088 (0.049)	0.689 (0.185)	0.477 (0.196)	0.054 (0.052)	0.443 (0.200)
	electronic	0.495 (0.070)	0.818 (0.163)	0.789 (0.153)	0.879 (0.077)	0.869 (0.120)	14.740 (4.206)	27.380 (5.601)	21.792 (4.247)	11.832 (2.554)	24.196 (4.974)	0.086 (0.048)	0.655 (0.195)	0.512 (0.178)	0.091 (0.083)	0.535 (0.202)
	jazz	0.512 (0.080)	0.811 (0.153)	0.786 (0.173)	0.885 (0.079)	0.870 (0.102)	14.156 (4.872)	26.768 (5.979)	20.620 (4.864)	11.132 (2.387)	23.596 (5.223)	0.076 (0.052)	0.656 (0.181)	0.505 (0.212)	0.070 (0.058)	0.539 (0.208)
	poprock	0.556 (0.087)	0.792 (0.157)	0.798 (0.167)	0.901 (0.080)	0.819 (0.148)	13.024 (4.231)	23.960 (5.363)	19.056 (4.251)	10.936 (2.359)	21.540 (4.856)	0.080 (0.044)	0.616 (0.183)	0.514 (0.209)	0.059 (0.047)	0.534 (0.212)
BIGGAN-based model	country	0.371 (0.009)	0.156 (0.035)	0.209 (0.040)	1 (0)	1 (0.001)	5.024 (0.153)	1.604 (0.794)	2.256 (0.758)	4 (0)	9.228 (0.465)	0.125 (0)	0 (0.001)	0.001 (0.004)	1 (0.002)	0.996 (0.009)
	electronic	0.394 (0.023)	0.283 (0.093)	0.303 (0.096)	1 (0)	1 (0)	6.388 (0.556)	3.164 (0.587)	6.868 (1.810)	4 (0)	12.472 (1.630)	0.132 (0.007)	0.019 (0.013)	0.098 (0.057)	0.940 (0.041)	0.969 (0.018)
	jazz	0.449 (0.027)	0.476 (0.088)	0.491 (0.089)	1 (0)	1 (0)	6.984 (0.464)	3.220 (0.451)	8.788 (1.065)	4 (0)	14.056 (1.079)	0.148 (0.010)	0.029 (0.012)	0.224 (0.060)	0.849 (0.050)	0.944 (0.018)
	poprock	0.391 (0.025)	0.277 (0.100)	0.299 (0.103)	1 (0)	1 (0)	6.240 (0.571)	3.108 (0.721)	6.164 (2.024)	4 (0)	12.060 (1.659)	0.131 (0.007)	0.019 (0.014)	0.092 (0.059)	0.945 (0.042)	0.971 (0.019)
ACGAN-based model	country	0.516 (0.053)	0.783 (0.174)	0.858 (0.129)	0.931 (0.061)	0.861 (0.131)	17.068 (2.471)	23.896 (5)	26.940 (4.265)	10.928 (2.197)	23.512 (4.211)	0.130 (0.050)	0.580 (0.170)	0.594 (0.182)	0.018 (0.014)	0.656 (0.174)
	electronic	0.448 (0.093)	0.699 (0.216)	0.635 (0.204)	0.928 (0.045)	0.766 (0.175)	14.514 (3.412)	22.932 (6.053)	20.149 (6.665)	11.566 (2.343)	19.550 (4.985)	0.061 (0.040)	0.590 (0.212)	0.400 (0.223)	0.052 (0.052)	0.511 (0.197)
	jazz	0.481 (0.068)	0.708 (0.197)	0.824 (0.161)	0.877 (0.115)	0.937 (0.082)	15.576 (2.826)	24.244 (5.984)	26.276 (5.232)	11.408 (2.107)	26.604 (3.383)	0.107 (0.053)	0.512 (0.195)	0.571 (0.174)	0.022 (0.016)	0.747 (0.141)
	poprock	0.318 (0.112)	0.779 (0.174)	0.915 (0.099)	0.76 (0.172)	0.697 (0.194)	9.944 (2.828)	24.676 (5.899)	25.828 (5.008)	9.512 (2.602)	20.188 (5.435)	0.039 (0.044)	0.558 (0.169)	0.679 (0.172)	0.040 (0.031)	0.489 (0.201)

almost all genres and 6 pitches for almost all genres, respectively. And Both are similar to dataset values. This is not as much as the CGAN-based model, but it is a part that shows that the ACGAN-based model can generate music similar to the distribution of the dataset. The atmosphere between the genres was formed differently when listening to the sample produced by the ACGAN-based model, but it was not enough to distinguish the genre after listening. It is presumed that this is because the condition is only the genre when creating a song, and there are cases where the genre sounds ambiguous in some datasets. The ACGAN-based model additionally learns the classifier. Therefore, that is more effective at discriminating conditions. As a result, there was a relatively greater difference between genres than the other two models when listening.

V. CONCLUSION

In this paper, we proposed a music generation model by genre using condition. We confirmed the performance difference of music generation according to the method of giving condition. The EB, UP, and PP values for the CGAN-based and ACGAN-based models were similar to the values from the experimental dataset, and the generated music was of good quality. When considering both qualitative and quantitative aspects, the performance of the ACGAN-based model was the best. The method of generating high resolution images like BIGGAN [4] did not have a good effect on the midi music generation, and the method of concatenating conditions simply like CGAN [3] did not bring out the optimal performance. However, when training by adding a classifier as in ACGAN [5], optimal performance could be obtained. Through this, we plan to study and find a way to create better music using classifiers.

REFERENCES

[1] 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. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014

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

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096, 2018.

[5] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In International conference on machine learning, pages 2642–2651. PMLR, 2017.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. Advances in neural information processing systems, 30.