Can GAN Generate Creative Music? A Study on using GAN to Generate Novel Styles of Music

Ricky Qiao
Master of Data science
The University of Auckland
Auckland, NZ
Email: tqia361@aucklanduni.ac.nz

Walter Wang
Master of Information Technology
The University of Auckland
Auckland, NZ
Email: wwan999@aucklanduni.ac.nz

Phillis Ye
Master of Data science
The University of Auckland
Auckland, NZ
Email: zye615@aucklanduni.ac.nz

Carmen Wen
Master of Data science
The University of Auckland
Auckland, NZ
Email: jwen758@aucklanduni.ac.nz

Qiaoyi Huang
Master of Data science
The University of Auckland
Auckland, NZ
Email: qhua658@aucklanduni.ac.nz

***Abstract*—Since the introduction of Generative Adversarial Network (GAN), researchers have proposed various music generation models based on GAN. These models are capable of generating music that emulates the style of training data. However, music composition is a creative process. Composers often create their own styles of music based on the styles they are familiar with. In this paper, we investigate the ability of GAN to generate novel styles of music based on existing ones. To achieve this, we proposed a GAN model that can take into account the melody, harmony and rhythmic components of piano music for learning and combining styles. We also implement an extension of this GAN model, a Creative Adversarial Network (CAN) which encourages the music generated to diverge from the styles given in the training data. We trained the GAN model with pop and jazz piano music separately and compared it with the GAN model trained with both styles through user study. We also trained the CAN model with both styles and compared it to the GAN model trained with both styles. We find that GAN trained with multiple styles of music is able to generate more likable and novel styles of music than GAN trained with a single style. CAN did not show an improvement from GAN in generating novel styles of music.**

Keywords—GAN, CAN, deep learning, music generation, computer-based music composition

I. Introduction

Since the dawn of the computer age, people have wondered if computers can create music. The first musical piece ever written by computer was in the 1950s. It was a string quartet called Illiac Suite, composed by using a stochastic process that rejects or accepts musical notes based on screening rules defined by the authors [1]. This selection process requires expertise on musical and might not be accurate at capturing musical structures [1]. The author thus concluded that computers can not generate music without human assistance [2].

However, with the recent advancement in deep learning, algorithmic music composition has been completely revolutionised. Many deep learning music generation models have been proposed in the past ten years [1]. Deep learning music generation models can work with two domains: either in audio domain or symbolic domain in the form of Musical Instrument Digital Interface (MIDI), a protocol that allows communication between electronic instruments, musical software and computer devices [1]. In contrast to previous music generation models, deep models are autonomous, in which the models learn the distribution of corpus of music and generate similar pieces without human assistance [1].

There are two types of deep models: predictive models and generative models [3]. Prediction models generate music by performing next-token predictions based on previous MIDI events or audio waves [4–7]. A few exceptional examples of prediction models are MuseNet [4] and MusicAVE [5] that work with MIDI; JukeBox [6] and WaveNet [7] that work with audio. The generative models were started by Mirza et al in their work C-RNN-GAN [8], which takes the form of Generative Adversarial Networks [9]. GAN was proposed in 2014 [9] and it is predominantly used for image generation. To use GAN for music generation, we can map the music pieces into one channel images with MIDI information [3, 10–14] or directly use MIDI information as features [8, 15]. GAN utilises a generative model and a discriminative model. The generator generates new data from random noises and discriminator classifies it into real data or fake data. The two models are trained together with a Minimax function in which two models compete with each other in a zero sum game [9]. The difference between generative and predictive models is that generative models are able to generate entire phrases of music at once rather than predicting future musical notes based on existing ones.

Although existing models such as MidiNet [10] and LeadSheetGAN [11] are both capable of generating promising results, there is still a problem with current GAN models. The idea of GAN is emulation, to generate music that is in the same distribution as the training data. However, as a form of art, music composition is not just emulation of existing pieces, but also to create new styles of music. Successful musicians tend to learn from existing styles and derive their own style in writing music. This is not the case with existing generative models [3, 8, 10–15], as they are focused on emulating only one style of music and generate similar pieces in that style.

To generate music with novel styles, we need a model that can learn more than one style of music and derive new styles from existing styles of music. However, current models are not designed for learning multiple styles. Monophonic models such as C-RNN-GAN, MidiNet and JazzGAN [15] can only learn the lead melodies of music pieces. Music styles can not be represented with just the melody information, we would also require harmonic and rhythmic components of music. On the other side, polyphonic models such as MuseGAN [12], BinaryMuseGAN [13], LeadSheetGAN [11] and INCO-GAN [3] are multi instruments that generate music in a pop band ensemble of instruments, which overcomplicate the issue. Another problem with the monophonic and polyphonic models mentioned above is that besides C-RNN-GAN which provide almost no harmonic information, all other GANs need to condition on chord progressions to work well. This creates a problem with mixing styles as different styles have different chord progressions and these are not learnt by the models. Furthermore, besides C-RNN-GAN, these models do not take into account the volume information of music in the training data, thus creating music with the same volume throughout the entire generated pieces. This lack of dynamics also results in loss of information for learning styles and less quality of music generated.

To solve this problem, we propose a new Deep Convolutional Generative Adversarial Network (DCGAN) [16] music generation model. This model is designed for polyphonic piano music. Piano, as the king of instruments, can capture the melodic, harmonic and rhythmic nature of the music pieces and it is able to play various styles of music. This model also takes into account the volume information and can generate dynamic music. As a result, it is able to learn various styles of piano music at the same time and learns the melodic, harmonic and rhythmic of different styles with no conditions on chord progressions. We will also implement a variant of this DCGAN model, CAN [17], to encourage generating more creative music. CAN is based on the extension proposed by [17, 18], which adds an additional classifier to the discriminator, for style classification. As a result, the generator not only needs to confuse the discriminator on whether the music generated is real or fake, it also needs to confuse the discriminator on the style of music it generated. In this way, the generator aims to generate music that sounds like real music, but is not any of the existing styles in the training data, thus creating novel styles that deviate from the style norms [17]. From now one, this proposed DCGAN model and CAN model will be referred to as PianoGAN and PianoCAN for the ease of readers to differentiate from other DCGAN and CAN models in related works.

II. Related Work

GAN in music generation is a new research area, there are not many existing models. The first attempt was C-RNN-GAN which utilised LSTM for both the generator and discriminator. Another model that uses LSTM is JazzGAN, which is based on SeqGAN [19], with LSTM as the generator and CNN as the discriminator. After Yang et al proposed MidiNet in 2017, which utilises the DCGAN architecture, and the later development of MuseGAN by Dong et al, it was recognised that GAN with CNN components can generate better music than those using RNN [3].

In MidiNet, the authors tried to generate more create music by placing conditions in every transposed convolution layer of the generator [10]. This conditions the music generated on the melodies from previous bars [10]. The result is that the more sequential melody can sound more creative than the melody that is primarily based on chord conditions. However, the generated music stays in the same style of training data. This also comes at the price of generating melody that violates the chord condition, which can sound unpleasant at times [10].

MuseGAN has a Jamming mode in which there is no multi-track interdependency, which means each instrument generator will generate music independently [12]. However, this can lead to different instruments playing in different keys and results in less coherent music. Again, the music generated would stay within the same style of training data. Its variant BinaryMuseGAN also has the same implementation of a Jamming mode [13].

Tokui investigated in his work RhythmCAN the ability of CAN to generate novel rhythmic patterns [14]. However, this is only one of the three important aspects of music. Melody and harmony are excluded.

C-RNN-GAN, JazzGAN, LeadSheetGAN and INCO-GAN did not attempt to generate more creative music but rather focused on better emulation of training data. A comparison of all existing GAN music generation models and proposed models are shown below. 10 models are compared by considering the architecture, type of neural networks, loss function, volume information, chord conditions and styles of music (Table. I).

III. Methodology

A. Data Representation

Pypianoroll packages released by dong et al provided the function to automatically turn MIDI files into pianoroll matrices [20]. We can see a example of pianoroll matrix in Figure 1 below, the y-axis represents the pitch of the note played, the x-axis represents the timesteps used to divide the sample MIDI file, which record the length of notes. The values stored in the matrix record how loud each note is played. Figure 1 is a 1780×128 matrix, the pitch range is 128 keys ranging from C-2 to slightly higher than C8, and there are 1780 timesteps (Fig.1).

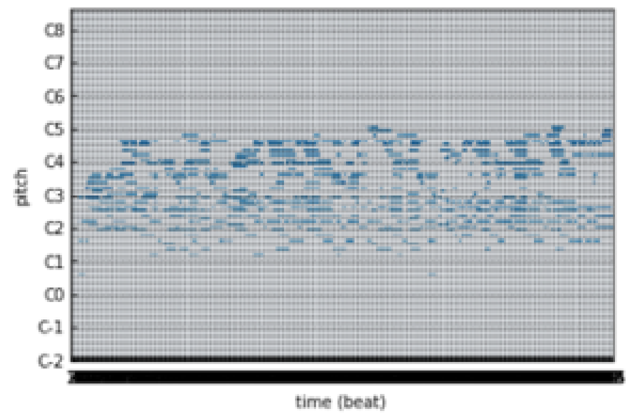

Fig. 1: Pianoroll representation

The 1780 time steps result from selecting a resolution of 4 for timesteps, which means each beat is represented by 4 timesteps. Below in Fig.2 we can see that in common time music, a bar of music consists of 4 beats in total. 1780 timesteps therefore represent 111 bars of music and an extra pick-up bar of 1 beat. This matrix is too big for DCGAN to learn. Since common time music often consists of 4 bar phrases of music, we will cut each piece of music into consecutive 4 bars music phrases, which result in 64 timesteps. The pitch range is also too wide for piano music as a full piano keyboard only has 88 keys. We will cut the pitch range to 61 keys keyboard range which is more than enough to play most contemporary piano music. We add extra 3 keys to make it the same number as the number of timesteps for convenience of training square images with GAN.

The 64×64 matrices are then transposed to 12 keys. This is done by shifting the values in the matrices up or down. For example, shifting the values up a row will result in a shift of key for a semitone higher, if the original music is in G key, the result will be in Ab key. We shift each music phrase up for up to 5 rows and down for up to 6 rows, this results in 12 keys including the original key.

[1]https://www.pianotheoryexercises.com/reading-music/measures-bars/

TABLE I: Summary of GAN music generation models

Models	GAN Architectures	Type of neural network	Loss Functions	Volume Information	Conditioned on chords	Styles of Music
C-RNN-GAN	Continuous recurrent networks	LSTM	MSE (G), Cross entropy (D)	Yes	No	Classical Melody
JazzGAN	Sequence Generative Adversarial Nets	LSTM (G), CNN (D)	Reinforce-algorithm (G), Cross entropy (D)	No	Yes	Jazz Melody
MidiNet	DCGAN, Conditional GAN	CNN	Feature matching with one-sided label smoothing (G), Cross entropy (D)	No	Yes	Pop Melody
MuseGAN	DCGAN, Conditional GAN	CNN	Gradient penalty Wasserstein (G,D)	No	Yes	Pop Band
BinaryMuseGAN	DCGAN, Conditional GAN	CNN	Gradient penalty Wasserstein (G,D)	No	Yes	Pop Band
LeadSheetGAN	Recurrent Convolutional GAN	CNN, RNN	Gradient penalty Wasserstein (G,D)	No	Yes	Pop Band
INCO-GAN	DCGAN, Conditional GAN	CNN (Inception Model)	MSE (G,D)	No	Yes	Pop Band
RhythmCAN	CAN	LSTM	Cross entropy (G,D)	No	No	Electronic Dance Rhythm Patterns
PianoGAN	DCGAN	CNN	BCE loss with logits (G,D)	Yes	No	Multi-Style Piano Music
PianoCAN	CAN	CNN	BCE loss with logits (G,D), Cross entropy (G)	Yes	No	Multi-Style Piano Music

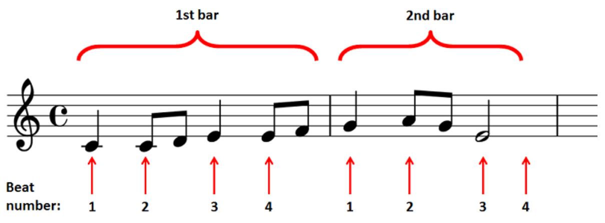

Fig. 2: Musical bars and beats [1]

All other DCGAN music generation models mentioned above [3, 10–13] choose to binarize the matrices so every note that appear in the matrix will become a value of 1. We decide to keep the volume information inside the pianoroll matrices so we can generate dynamic music. The volume is first capped at 90 by setting all volume louder than 90 back to 90, this is to minimise the effect from some music pieces being a lot louder than others. We then normalise the volumes into a range of between -1 and 1, this corresponds to the Tanh activation function on the last layer of Generator in PianoGAN and PianoCAN, which generate values between -1 and 1. The generated matrices can be re-normalise back into the range of 0-90 and then remove quiet noises by setting values lower than 30 back to 0.

B. *PianoGAN* [2]

1) Overview:

PianoGAN is based on the DCGAN architecture. The model contains a generator that aims to generate similar music to training data, and a discriminator that classifies the generated music into real or fake. The discriminator's role is to provide feedback to the generator which facilitates its learning. The generator and discriminator compete with each other in a zero sum game formulated by the Minimax function (Equation 1):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

- D represent the discriminator
- G represents the generator
- $x \sim p_{data}$ represents the data from real music
- $z \sim p_z$ represents the random input noise
- $G(z)$ represents the generated sample from random noise

[2]**Colab link** for PianoGAN: https://colab.research.google.com/drive/1IpQkxoomZ8BuwNJrlC7t5Q-oRmRuXf1_?usp=sharing

The first part of the function represents how accurately the discriminator could identify real music. When the $D(X)$ is close to 1, the discriminator predicts nearly all real music data as real. The second part of the function is how accurately the discriminator could identify generated music. When $(1 - D(G(z)))$ is close to 1, the discriminator predicts nearly all generated music data as not real. The generator aims to produce music that confuses the discriminator, by minimizing $log(1 - D(G(z)))$. In the meantime, the discriminator aims to be able to correctly classify real and generated music by maximizing both $log(D(X))$ and $log(1 - D(G(z)))$. The end goal of training is for both models to reach a Nash equilibrium [9]. The optimal generator can generate music that result in random guess with discriminator as generated music is too similar to training data [9].

Both the generator and discriminator can be trained through backpropagations. However, the Minimax function is usually modified in training, the loss function of minimizing $log(1–D(G(z)))$ for G can lead to saturation, that is when $D(G(z))$ is close to zero, the gradient vanishes for G and results in ineffective training [21]. A non-saturating strategy can be adopted, a popular way is to use cross-entropy minimization for the generator, as shown in Equation (2) [21]. Here $y = 1$ for generated samples.

$$\min_G\{-y\log(D(G(z)))\} = \underset{G}{min}\{-\log(D(G(z)))\} \tag{2}$$

Similarly, binary cross entropy minimization is used for the discriminator, as shown in Equation (3). Here $y = 1$ for real data and 0 for generated samples.

$$\min_D\{-y\log D(x) - (1-y)\log(1 - D(x))\} \tag{3}$$

In our PianoGAN model, we uses binary cross entropy loss with logits, which utilises the log-sum-exp trick for numerical stability, by combining a Sigmoid layer and log-loss into one class [22]. As a result, we don't need to use a Sigmoid final layer in the discriminator like the original DCGAN architecture [16]. The learning rate is set at 0.0002, which is also the same as [16]. The choice of optimiser is AdamW instead Adam in [16]. As suggested by Loshchilov et al, AdamW improved performance of image classification by decoupling the weight decay from the learning rate, which solves the issue of Adam incorrectly equating weight decay regularization with L2 regularization [23]. Therefore AdamW should be able to help our discriminator in classifying music. We also initialise weights of both models before training with 1-centered normal distribution, and standard deviation of 0.02, as suggested by [16].

2) Generator:

The underlying structure of the generator is a convolution neural network (CNN). It takes a Gaussian random noise vector with length of 100 as input to generate samples. The noise vector will be turned into 512 channels of 4×4 matrices. For each layer except for the final layer, we apply transposed convolution layers to upsample the matrices. We also use batch normalization except for the last layer to prevent the vanishing gradient and exploding gradient problems [16]. We use ReLU as our activation function for all layers except for the last layer, which uses the Tanh function to generate output from range -1 to 1 (Fig. 3).

3) Discriminator:

The discriminator is also a CNN. The input is a 64×64 matrix that represents a music phrase. Like the generator, there are also 5 layers. We use spectral normalization [24] for each layer other than the final layer and batch normalisation on each layer except for the first and last layer. Miyato et al find that spectral normalization for model weights outperforms gradient penalty for weight normalization used in WGAN-GP, with the extra benefit of a lower computational cost [24]. We use LeakyReLU activation functions for all layers with negative slope set at 0.2. We also apply dropout as regularization

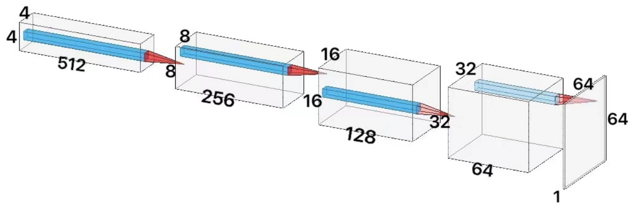

Fig. 3: Generator structure of GAN

technique on all layers except for the last layer, with the a drop out rate of 0.3 to prevent overfitting. We also added Gaussian instance noises to the input of discriminator to stabilise training, this is suggested by Sonderby et al [25] and validated by Mescheder et al [26]. It slightly minimises the differences between real data and generated sample to stop discriminator overpowering the generator [25]. This is implemented by using 10% of the input values as the standard deviation of Gaussian noises added to the input values (Fig. 4).

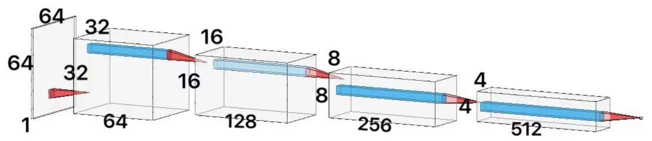

Fig. 4: Discriminator structure of GAN

C. PianoCAN[3]

1) Overview:

To further encourage the generation of novel styles of music, we add an extension of a second classifier in the discriminator to turn PianoGAN into PianoCAN. The second classifier can determine which style the given music phrase is. Fig.5 shows a generalised architecture of the CAN model. The generator is the same as the GAN model. For the discriminator, We need to provide at least two styles of music and label which style the music belongs to. During the training process, the style classifier will predict the probability of a musical phrase belonging to each style of music. The results will then be used along with the real style labels to update the model. The Minimax function will become (Equation 4):

$$\begin{aligned}\underset{G}{min}\,\underset{D}{max}\,V(D,G) &= \mathbb{E}_{x,\hat{c}\sim p_{data}}[\log D_r(x) + \\ &\log D_c(c=\hat{c}|x)] + \mathbb{E}_{z,p_z}[\log(1 - D_r(G(z))) - \\ &\sum_{k=1}^{K}(\frac{1}{K}\log(D_c(c_k|G(z)) + \\ &(1-\frac{1}{K})\log(1 - D_c(c_k|G(z)))]\end{aligned} \tag{4}$$

- D_r represents the classifier that distinguish between real and generated music
- D_c represents the style classifier
- $x, \hat{c} \sim p_{data}$ represents real music and their style labels.
- K represents the total number of music styles
- $D_c(c_k|G(z))$ represents the posterior probability of each phrase of music belonging to each style

The first half of this function consists of how accurate the real/fake classifier makes predictions and how accurate the style

[3]**Colab Link** for PianoCAN: https://colab.research.google.com/drive/1XsZ6x0Sa4lYL9P7obv6ObextcOzY-IXL?usp=sharing

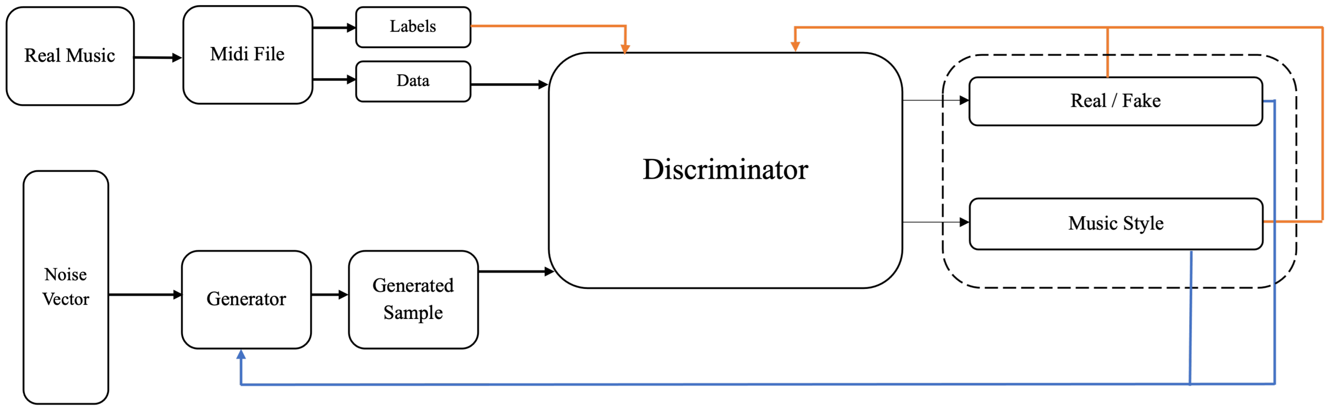

Fig. 5: CAN model

classifier distinguish between styles on real music x. The second half of the function focus on the generated music $G(z)$. The intuition behind that is if we have K styles of music, then we want the discriminator to predict 1/K probability for each type for generated music, which means the generated music confuses the discriminator about which style the music is, thus achieving style ambiguity. Style ambiguity can signifies a style that is novel and different to existing ones.

2) *Discriminator:*

The discriminator for the CAN model has an extra classifier to distinguish the style. Both the classifiers inside the discriminator share the same five convolution neural network layers. We add three fully connected layers at the end for the style classifier. Those 3 layers all use LeakyReLU as activation function with a negative slop of 0.2. (Fig.6).

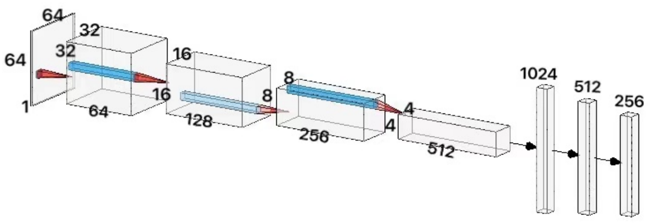

Fig. 6: Discriminator structure of CAN

3) *Training:*

The training process of CAN model is similar to the GAN model, we generate a set of random noise vectors and pass it to the generator to get a set of generated music. The generated music are passed into the discriminator to get the real/fake output and the style predictions output. The discriminator updates itself by comparing the predictions with real labels. The generator will also update its weights by receiving signals from both of these two outputs. The loss function is similar to PianoGAN, we use BCE loss with logits, the difference is that the for the second signal from class style classification, the generator will minimize the cross entropy with uniform distribution for style predictions.

4) *Turning off the Discriminator:*

CAN is more unstable than GAN. During the training process, we noticed that the discriminator may sometimes become too strong in real/fake and style classification, which might overpower and stop the learning process of the generator. To fix this problem, we turn off the training of discriminators temporarily when we detect the generator loss is greater than the previous epoch, or when the ratio of generator loss and discriminator loss exceed 10:1.

D. *Quantitative Validation*

1) *Dataset:*

To investigate GAN's ability of generating novel styles from existing ones. We built two datasets, one dataset is a collection of Chinese pop love songs played on piano, the other is jazz piano music. Both data are collected as audio recording from YouTube and transformed into MIDI format by GiantMIDI-Piano [27]. After preprocessing the data into pianoroll matrices and performed data augmentations, we have 90,528 pianoroll matrices of pop music and 22,592 pianoroll matrices of jazz music, which are 362,328 and 90,368 bars of music. With these two datasets in hand, we can train 4 versions of models: PianoGAN with pop dataset, PianoGAN with Jazz dataset, PianoGAN with both datasets and PianoCAN with both datasets.

2) *Experimental Design:*

To investigate whether GAN can generate novel styles of music, we need to compare PianoGAN trained with both styles with PianoGAN trained with single styles. To validate whether CAN can improve the creativity of GAN, We also need to compare PianoCAN trained with both styles with PianoGAN trained with both styles. Since creativity of music is objective, we need to conduct a user study to evaluate the results. To account for the participants' varying degrees of musical knowledge, we seleted five questions from the Goldsmiths Musical Sophistication Index [4] to categorize participants into two groups by their rough level of knowledge in music. To evaluate the performance of each version of models, we adopted three questions from the second user study conducted in the original CAN paper [17] to see if participants can tell the difference between generated music and real music, how the generated music are liked by the participants, and how novel the participants think the music is. For each version of the models, we generated 50 phrases of music, and randomly selected 5 phrases, each phrase is 10 seconds long. We also randomly select 5 phrases of music from our training data as the ground truth. The result is 25 phrases: 5 phrases of real music, 5 phrases of music generated from PianoGAN trained with pop dataset, 5 phrases of music generated from PianoGAN trained with jazz dataset, 5 phrases of music generated from PianoGAN trained with both datasets, 5 phrases of music generated from PianoCAN

[4]https://www.gold.ac.uk/music-mind-brain/gold-msi

trained with both datasets. To avoid experimental biases such as confirmation bias, the order of music phrases are randomly shuffled, and participants were unaware of the how many number of music phrases are real or generated.

For our experiment, we surveyed 38 volunteers and requested that they choose a quiet space to complete the questionnaire.

Part 1:
The five questions from The Goldsmiths Musical Sophistication Index are shown below, for each question there are 7 options to choose from ranging from completely disagree (-3), neutral (0), to completely agree (3):

- **Q1**: Does the participant spend a significant portion of time participating in music-related activities? (for instance, listening to music or playing an instrument)
- **Q2**: If someone begins singing a song that the participant is unfamiliar with, is it common for the participant to join in?
- **Q3**: Is the participant capable of determining whether or not someone is a good singer?
- **Q4**: Does the volunteer have the ability to sing or play music from memory?
- **Q5**: Is the participant interested in learning more about musical genres that he or she is unfamiliar with?

Part 2:
We request the participants to listen to each phrase of music and answer the following questions:

- **Q1**: Do the participants believe the music was composed by a musician or was it produced by a computer? The user has to choose one of three answers: Musician, Computer, or Can not distinguish.
- **Q2**: Rate the music on a scale from 1 (extremely dislike) to 5 (extremely like).
- **Q3**: Rate the novelty of the music on a scale from 1(no novelty) to 5(extremely high novelty).

IV. Results

A. Training Results

We trained each version of models for 50 epochs. We printed out the generated music and real music to see whether the generator is able to learn the patterns from the data. By comparing the generated music from each epoch, we could see that the distribution of patterns of the music is looking better and better, so we know that the generator is actually learning (Fig. 7).

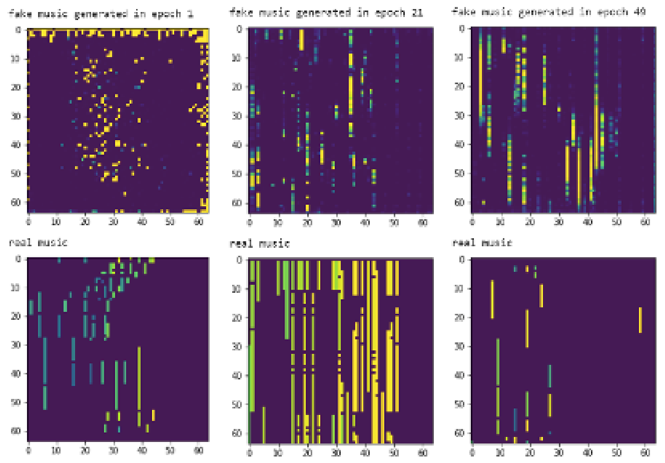

Fig. 7: Music for the 1st, 21st and 49th Epoch

At the end of the training process, we plotted the average generator loss and discriminator loss from each epoch. We can see from Fig.8,9,10, the training of PianoGAN has been successful in which the models converged with Nash equilibrium.

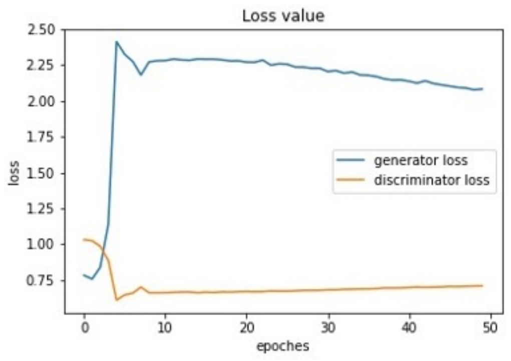

Fig. 8: Loss Graph for training PianoGAN with pop dataset

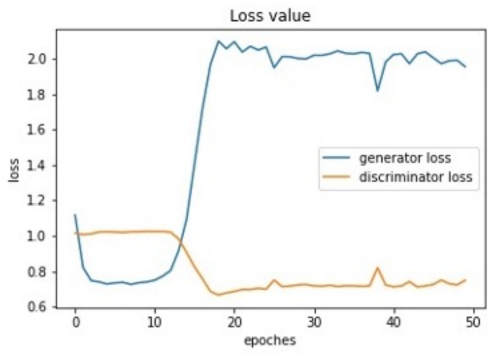

Fig. 9: Loss Graph for training PianoGAN with jazz dataset

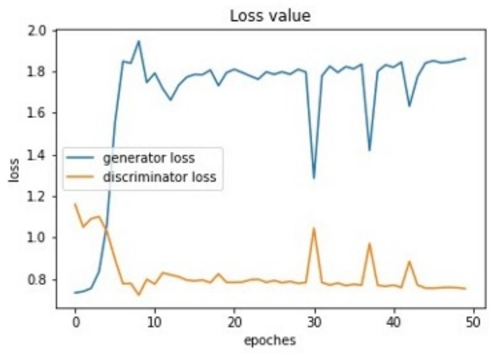

Fig. 10: Loss Graph for training PianoGAN with both datasets

From Fig.11, we could see that when we train the PianoCAN model without controlling the discriminator, the discriminator will become too strong and overpowers the generator, resulting in diminished gradient for the generator and non-convergence. From Fig.12, we can see that after adopting the strategy of controlling the discriminator by temporarily pause the training of discriminator, PianoCAN is able to converge.

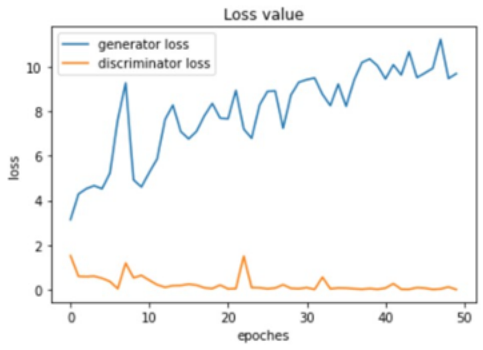

Fig. 11: Training PianoCAN without controlling the discriminator

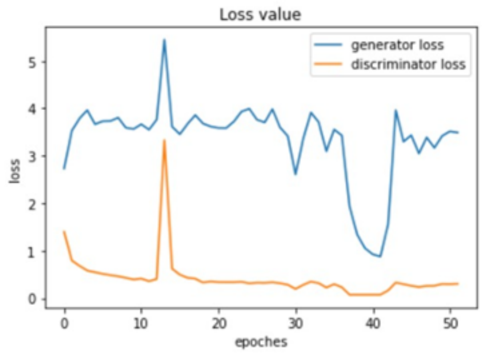

Fig. 12: Training PianoCAN with discriminator control

B. Survey Evaluations[5]

In this section, we will evaluate the five music sets mentioned in Quantitative Validation by performing a user study. We divide the 38 participants into two groups: participants who are more sophisticated in music and participants who are less sophisticated in music, based on Part 1 of the survey. The distribution of Part 1 scores of all the participants is shown in Fig. 13, and we choose the median score to split the participants into 20 participants who are considered more sophisticated in music and 18 participants who are considered less sophisticated in music.

1) Evaluation Metrics:

We used four metrics to evaluate the performance of each music set with different groups of participants. The metrics are: Likeness, Novelty, Real Music Rate and Cannot Distinguish Rate. For Likeness and Novelty records, we computed the mean and standard deviation scores. Real Music Rate and Cannot Distinguish Rate are calculated from the first question in the second part of the survey, which has three choices: Musicians, Computer, Can not distinguish. Real Music Rate is calculated by the following formula: $\frac{|Number\ of\ participants\ that\ chose\ Musicians|}{|Total\ number\ of\ participants|}$,which is the rate that participants believe the music pieces they heard in a music set are created by musicians. Cannot Distinguish Rate is calculated by $\frac{|Number\ of\ participants\ that\ choose\ Can\ Not\ Distinguish|}{|Total\ number\ of\ participants|}$, which is the rate that participants can not decide whether the music pieces they heard in a music set are created by a musician or generated by computer. The higher the Cannot Distinguish Rate, the more likely that the generated music confuse the listener about whether it is real or fake. To be more precise, $(1 - Real\ Music\ Rate - Cannot\ Distinguish\ Rate)$ means the rate that participants think the music pieces they heard is generated by computer.

[5]**Colab Link** for survey evaluation: https://colab.research.google.com/drive/1QrGhJ48BRAr1fjSAqfxnzkFUGTPGd6ru?usp=sharing

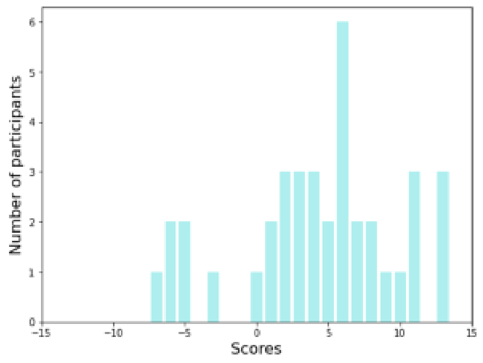

Fig. 13: Distribution of Part 1 scores

2) Music Sets and Participants Groups Comparison:

The evaluation results by five music sets and three groups of participants are shown in Table.II, Table.III . We observe that standard deviations in the brackets are around 1, which are relatively high. To be more clear, we plotted the results from the table to compare the scores given by different groups of participants to different music sets. As shown in Fig.16 and Fig.17, the blue bars represent all participants groups; the yellow bars represent the more sophisticated participants group, and the grey bars represent the less sophisticated participants group. From now on, music generated from PianoGAN trained with pop dataset, music generated from PianoGAN trained with jazz dataset, music generated from PianoGAN trained with both datasets and music generated from PianoCAN trained with both datasets in user study will be denoted as *GAN_Pop*, *GAN_Jazz*, *GAN_Both* and *CAN* respectively.

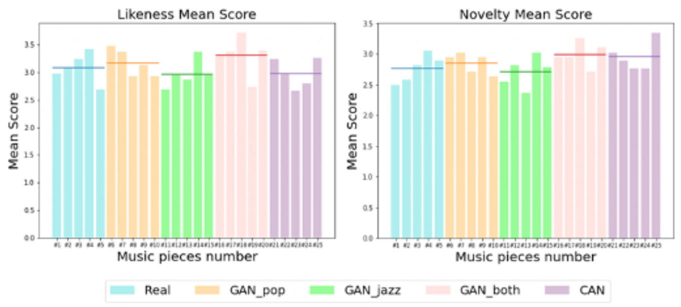

Fig. 14: Mean Scores of Likeness & Novelty of each of the 25 music pieces vs 5 music sets

Fig.16 shows the mean Likeness scores (left) and mean Novelty scores (right), where we observe that:

- *GAN_Both* has the highest mean likeness scores in all the three groups.
- More Sophisticated Participants tend to give higher Likeness scores than the Less Sophisticated Participants in all the five music sets.
- *GAN_Both* and *CAN* have very similar mean novelty scores and are higher than the other three music sets.
- *GAN_Pop* performed better than *GAN_Jazz* in both mean Likeness scores and mean Novelty scores.

TABLE II: Scores by music sets and participants (a)

Music Set	Likeness (std)			Novelty (std)		
	All	More Sophisticated	Less Sophisticated	All	More Sophisticated	Less Sophisticated
Real	3.08 (0.99)	3.08 (1.11)	3.08 (0.83)	2.77 (0.98)	2.71 (1.09)	2.83 (0.83)
GAN pop	3.16 (0.93)	3.24 (0.99)	3.08 (0.85)	2.85 (1.03)	2.79 (1.12)	2.92 (0.92)
GAN Jazz	2.96 (0.9)	3.05 (1.01)	2.87 (0.75)	2.71 (0.97)	2.81 (1.12)	2.6 (0.74)
GAN Both	3.31 (0.94)	3.42 (0.98)	3.18 (0.86)	2.99 (0.94)	3.03 (0.97)	2.96 (0.89)
CAN	2.98 (0.89)	3.03 (0.96)	2.93 (0.79)	2.96 (1.0)	2.93 (1.11)	2.99 (0.88)

TABLE III: Scores by music sets and participants (b)

Music Set	Real Music Rate			Cannot Distinguish Rate		
	All	More Sophisticated	Less Sophisticated	All	More Sophisticated	Less Sophisticated
Real	0.45	0.49	0.41	0.14	0.08	0.2
GAN pop	0.41	0.43	0.38	0.18	0.12	0.24
GAN Jazz	0.31	0.32	0.29	0.19	0.14	0.26
GAN Both	0.43	0.44	0.41	0.18	0.12	0.24
CAN	0.35	0.38	0.32	0.2	0.16	0.24

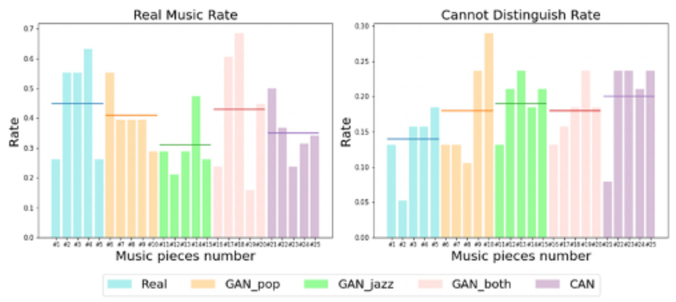

Fig. 15: Real Music Rate & Cannot Distinguish Rate of each of the 25 music pieces vs 5 music sets

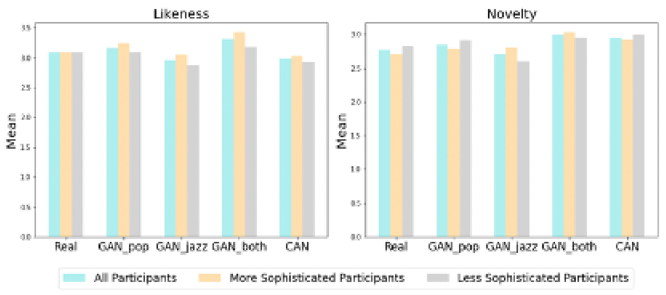

Fig. 16: Mean scores of Likeness & Novelty by music sets and participants

Fig.17 shows the Real Music Rate (left) and Cannot Distinguish Rate (right), where we observe that:

- *GAN_Both* has the second-highest Real Music Rate besides the Real music set that participants thought the music phrases they heard from that music set were created by musicians.
- *GAN_Jazz* has the lowest Real Music Rate suggests that the music pieces in that set are more likely to be recognised as computer generated.

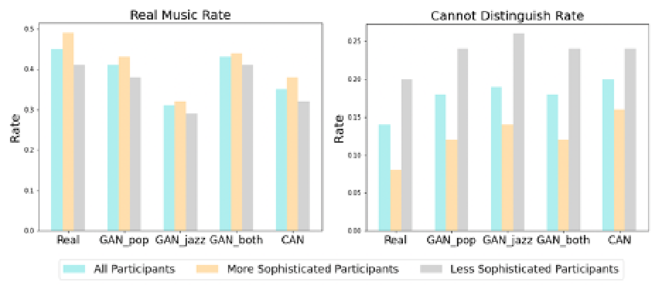

Fig. 17: Real Music Rate & Cannot Distinguish Rate by music sets and participants

- *GAN_Both* and *GAN_pop* have similar second-lowest Cannot Distinguish Rate beside the Real music set which suggest these two music sets are slightly less likely to confuse the listener about whether it is created by musicians or generated by computer than other sets.
- The Less Sophisticated Participants have significantly higher Cannot Distinguish Rates than the other groups which implies that the way we divide the participants into two groups is reasonable.
- *CAN* music set doesn't perform well in Real Music Rate but it has highest Cannot Distinguish Rate.

3) Music Pieces and Music Sets Comparison:

We will also compare the mean scores and rates of each 25 music pieces in our survey to see how they compare with the values over the five different music sets. As shown in Fig.14 and Fig.15, music pieces NO.1 to NO.5 are the selected five music pieces in the Real music set with blue bars, NO.6 to NO.10 are music pieces in the *GAN_pop* music set represented by orange bars, NO.11 to NO.15 are the music pieces in the *GAN_Jazz* music set with green bars, NO.16 to NO.20 are *GAN_Both* music with pink bars, and NO.21 to NO.25 are *CAN* music with purple bars. The lines are the mean scores and rates for each of the five music sets from all participants. We can see that likeness and novelty scores are quite similar but Real Music Rate and Cannot Distinguish rate have major differences between pieces.

From those results, we can see that PianoGAN trained with 2 styles can generate more novel and likable music than PianoGAN trained with single styles. PianoCAN doesn't show a improvement in terms of novelty from PianoGAN trained with both styles, and it result in less liked music than PianoGAN trained with both styles. PianoGAN trained with both styles can also generate more realistic music and PianoCAN is more likely to confuse the listener about whether it is real or generated.

V. Limitations

A. Models

Our PianoGAN and PianoCAN models are based on the DCGAN architecture. DCGAN is one of the earliest models of GAN and has since become the foundation of GAN in image generation. However, DCGAN's results cannot compete with newer GANs such as Progressive GAN [28] and BigGAN [29] in terms of quality of image generated. DCGAN can only generate images of low to medium qualities. The consequence of this in music generation is that DCGAN is unable to learn complex styles of playing. Beside the pop dataset and jazz dataset used in user study, we also compiled a Chopin dataset with 3 hours of Chopin piano music. However, PianoGAN can not converge with the Chopin dataset and can only generate music of low quality, and it is therefore not used in our experiment. As shown below in fig.18, we can see that the loss of generators still has an upward trend after 50 epochs. Note that we did not have the computational power to train each model for more than 50 epochs.

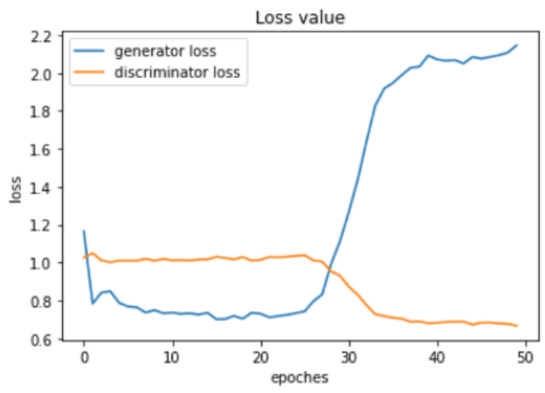

Fig. 18: Loss Graph when training PianoGAN with Chopin Dataset

The same thing happened when we tried to train PianoGAN with the Emopia dataset [30], which has many different kinds of pop music played on piano, including Japanese anime music, game music, American pop music played virtuosically on piano. After carefully inspecting the data, we concluded that the music when transferring into one channel images will result in complex distributions if the playing styles are too complex or too diverse. We can see in Fig.19 and Fig.20, these are two examples of Chopin's playing shown as MIDI pianoroll format in Logic Pro, the first passage is in normal playing style and has a relatively simple pattern whereas the second passage looks a lot more complex with cluttered patterns. These complex distributions of patterns resulted in non convergence for DCGAN.

Stability remains an issue for GAN, with the introduction of another classifier and the modified loss function, CAN becomes even more unstable than GAN. Although we have implemented a method

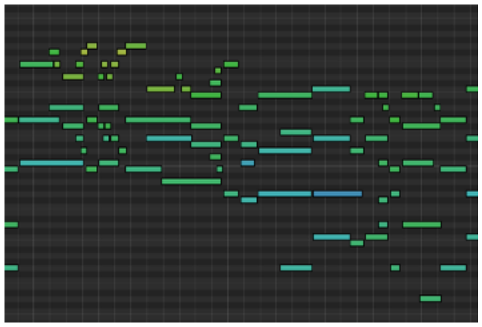

Fig. 19: Chopin's music example shown in Logic Pro [6]

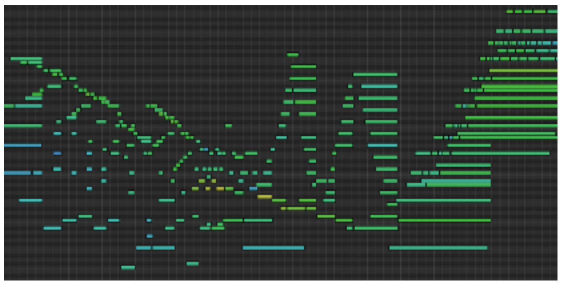

Fig. 20: Chopin's virtuoso playing shown in Logic Pro

to stop the discriminator overpowering the generator, this might not be the optimal solution for CAN. The author of CAN only specified using batch normalisation to stabilize training [17], which we already implemented. Since there are no other literature regarding training of CAN, a better method to train CAN is yet to be investigated, CAN's potential is therefore still unknown at this stage.

B. User Study

Table.II and Table.III show all the standard deviations are around 1, which results in no statistical significance and less reliable results. Since perception of music are subjective, the 38 participants might not be a representative sample of population who listen to piano music. Furthermore, due to the pandemic, we can not expand the user study in person. This leads to a problem: we can not guarantee that all the participants were listening to the music pieces in the same quiet environment. We only random selected once from generated music by each version of models for the 25 music phrases used in the survey, this will increase the bias of our survey evaluation. As we have observed in the music pieces and music sets comparison, different phrases can vary to a large degree in terms of Real Music Rate and Cannot Distinguish Rate. Lastly, the amount of data used to train each model is not fully controlled. Although PianoGAN trained with both styles and PianoCAN is trained with the same data, PianoGAN trained with single styles is trained with less amount of data, especially with jazz dataset. This might give a disadvantage for *GAN_Jazz* music set.

VI. Conclusion

A. Summary[7]

In this paper, we proposed two models PianoGAN and PianoCAN, to address the problem of the lack of novelty of generated music. PianoGAN is based on DCGAN to handle the generation of polyphonic piano music, and PianoCAN is a variant of PianoGAN to generate novel styles of music based on the idea of the CAN

[6]https://www.apple.com/nz/logic-pro

[7]Samples of generated music: https://drive.google.com/drive/folders/1oLx6pg9eLXCXbOWpesO9VBoLW64GTs0M?usp=sharing

model used in generating novel styles of paintings in the CAN paper [17]. We trained PianoGAN on two single music styles and combination of these two music styles and compare those results. To validate if CAN can improve creativity, we also trained PianoCAN on the same combination of these two music styles and compare it with PianoGAN trained with both styles. From the survey evaluation results, we concluded that PianoCAN generated music is not more creative than PianoGAN, and we can generate more creative and more liked music by training the PianoGAN model with more music styles.

B. Future Works

Due to time limit and computational requirement of using GiantPianoMIDI to convert mp3 into MIDI files, our jazz dataset has less amount of music than the pop dataset, about 1:4. We can investigate if the different amount of music used in training can have an impact on the music generated. We can also controll all version of models to be trained with the same amount of data to minimize bias introduced by varying amount of data. Furthermore, for simplicity, this study only investigated combinations of 2 styles of music. With a collection of more genres of music, we can try different combinations as well as different numbers of styles to further investigate how GAN generates novel styles from existing ones.

CAN as a new architecture is quite unstable in training. We should further investigate how to stabilize GAN, drawing from literature on GAN training. Furthermore, the music generated by PianoCAN can sound uncanny and therefore received lower likeness and real music rate than music generated by PianoGAN trained with both styles. This is caused by the generator trying to maximise style ambiguity. We could try to give weights to loss from real/fake classification and loss from style classification to make the generator prioritise generating real music, then generating music that deviates from style norms.

Music is not like images, it has a sequential nature, a bit like languages. This means that we can borrow ideas from natural language processing such as n-grams. Instead of taking consecutive four bars like bars 1-4, 5-8, 9-12 etc, we can take overlapping bars like bar 1-4, 2-5, 3-6, 4-7, 5-8 etc. This in theory should be able to model the sequential nature of music better.

INCO-GAN has used the inception model for its generator to better learn the patterns of music. We can also try newer architectures such as Progressive GAN or BigGAN to see if they can learn more complex styles of music playing.

In order to evaluate the performance of CAN in painting generation, Elgammal et al have conducted in total 4 user studies for quantitative validation [17]. Our user study is a shortened one which imitates their second experiment [17]. To better evaluate our result, we will also need more user studies with a more diverse range of questions. Furthermore, we can mitigate the bias caused by random selection of music samples by using two approaches. The first one would be to select more samples from each music set, for example, 10 pieces instead of 5 pieces. We can also program the survey in which the random selection happens for each individual participant instead of just one selection for all participants. Lastly, we can conduct user study specifically designed for music lecturers at university and music professionals so the music generated can be analysed more thoroughly.

VII. Contributions

Ricky Qiao:

- Report: Wrote Introduction, Related Works, Limitation-A, Conclusion-B; proof read and edited the report.
- Project: Implemented data pre-processing, modified and trained DCGAN for piano music (PianoGAN).

Walter Wang:

- Report: Wrote Methodology-B, Methodology-C, Results-A.
- Project: Implementation and training of CAN (PianoCAN).

Phillis Ye:

- Report: Wrote Results-B, Limitation-A, Conclusion-A.
- Project: Implemented data post-processing, performed survey evaluation.

Carmen Wen:

- Report: Wrote Methodology-A, Methodology-D-1.
- Project: Data collection, data transformation, logistics (video editing, coding for report & survey on Overleaf etc).

Qiaoyi Huang:

- Report: Wrote Abstract, Methodology-D-2.
- Project: Survey design.

References

[1] J.-P. Briot, "From artificial neural networks to deep learning for music generation: History, concepts and trends," *Neural Computing and Applications*, vol. 33, no. 1, pp. 39–65, 2021.

[2] O. Sandred, M. Laurson, and M. Kuuskankare, "Revisiting the illiac suite–a rule-based approach to stochastic processes," *Sonic Ideas/Ideas Sonicas*, vol. 2, pp. 42–46, 2009.

[3] S. Li and Y. Sung, "Inco-gan: Variable-length music generation method based on inception model-based conditional gan," *Mathematics*, vol. 9, no. 4, p. 387, 2021.

[4] S. Ilya, *Musenet*, https://openai.com/blog/musenet/, December 11, 2015.

[5] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*, PMLR, 2018, pp. 4364–4373.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[7] A. v. d. Oord *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[8] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[9] I. Goodfellow *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[10] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[11] H.-M. Liu and Y.-H. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2018, pp. 722–727.

[12] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] H.-W. Dong and Y.-H. Yang, “Convolutional generative adversarial networks with binary neurons for polyphonic music generation,” *arXiv preprint arXiv:1804.09399*, 2018.

[14] N. Tokui, “Can gan originate new electronic dance music genres?–generating novel rhythm patterns using gan with genre ambiguity loss,” *arXiv preprint arXiv:2011.13062*, 2020.

[15] N. Trieu and R. Keller, “Jazzgan: Improvising with generative adversarial networks,” in *MUME workshop*, 2018.

[16] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

[17] A. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone, “Can: Creative adversarial networks, generating” art” by learning about styles and deviating from style norms,” *arXiv preprint arXiv:1706.07068*, 2017.

[18] J.-P. Briot and F. Pachet, “Deep learning for music generation: Challenges and directions,” *Neural Computing and Applications*, vol. 32, no. 4, pp. 981–993, 2020.

[19] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[20] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, “Pypianoroll: Open source python package for handling multitrack pianoroll,” *Proc. ISMIR. Late-breaking paper;[Online] https://github.com/salu133445/pypianoroll*, 2018.

[21] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” *arXiv preprint arXiv:1701.04862*, 2017.

[22] *Bcewithlogitsloss — pytorch 1.9.1 documentation*, https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html, Accessed September 28, 2021.

[23] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[24] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” *arXiv preprint arXiv:1802.05957*, 2018.

[25] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, “Amortised map inference for image super-resolution,” *arXiv preprint arXiv:1610.04490*, 2016.

[26] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for gans do actually converge?” In *International conference on machine learning*, PMLR, 2018, pp. 3481–3490.

[27] Q. Kong, B. Li, J. Chen, and Y. Wang, “Giantmidi-piano: A large-scale midi dataset for classical piano music,” *arXiv preprint arXiv:2010.07061*, 2020.

[28] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.

[29] J. D. Curtó, I. C. Zarza, F. De La Torre, I. King, and M. R. Lyu, “High-resolution deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1711.06491*, 2017.

[30] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y.-H. Yang, “Emopia: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation,” *arXiv preprint arXiv:2108.01374*, 2021.