MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment

Hao-Wen Dong[*],[1] **Wen-Yi Hsiao**[*],[1,2] **Li-Chia Yang,**[1] **Yi-Hsuan Yang**[1]
[1]Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
[2]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
salu133445@citi.sinica.edu.tw, s105062581@m105.nthu.edu.tw, {richard40148, yang}@citi.sinica.edu.tw

Abstract

Generating music has a few notable differences from generating images and videos. First, music is an art of time, necessitating a temporal model. Second, music is usually composed of multiple instruments/tracks with their own temporal dynamics, but collectively they unfold over time interdependently. Lastly, musical notes are often grouped into chords, arpeggios or melodies in polyphonic music, and thereby introducing a chronological ordering of notes is not naturally suitable. In this paper, we propose three models for symbolic multi-track music generation under the framework of generative adversarial networks (GANs). The three models, which differ in the underlying assumptions and accordingly the network architectures, are referred to as the jamming model, the composer model and the hybrid model. We trained the proposed models on a dataset of over one hundred thousand bars of rock music and applied them to generate piano-rolls of five tracks: bass, drums, guitar, piano and strings. A few intra-track and inter-track objective metrics are also proposed to evaluate the generative results, in addition to a subjective user study. We show that our models can generate coherent music of four bars right from scratch (i.e. without human inputs). We also extend our models to human-AI cooperative music generation: given a specific track composed by human, we can generate four additional tracks to accompany it. All code, the dataset and the rendered audio samples are available at `https://salu133445.github.io/musegan/`.

Introduction

Generating realistic and aesthetic pieces has been considered as one of the most exciting tasks in the field of AI. Recent years have seen major progress in generating images, videos and text, notably using generative adversarial networks (GANs) (Goodfellow et al. 2014; Radford, Metz, and Chintala 2016; Vondrick, Pirsiavash, and Torralba 2016; Saito, Matsumoto, and Saito 2017; Yu et al. 2017). Similar attempts have also been made to generate symbolic music, but the task remains challenging for the following reasons.

First, music is an art of time. As shown in Figure 1, music has a hierarchical structure, with higher-level building blocks (e.g., a phrase) made up of smaller recurrent patterns (e.g., a bar). People pay attention to structural patterns related to coherence, rhythm, tension and the emotion flow while listening to music (Herremans and Chew 2017). Thus, a mechanism to account for the temporal structure is critical.

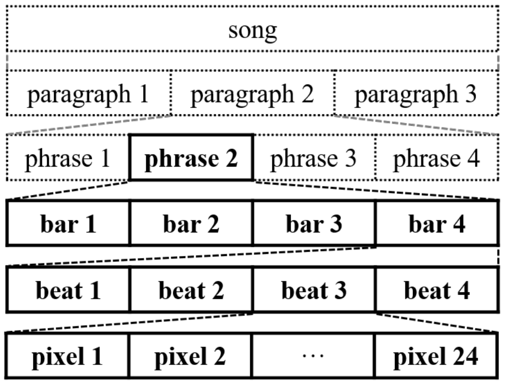


Figure 1: Hierarchical structure of a music piece.

Second, music is usually composed of multiple instruments/tracks. A modern orchestra usually contains four different sections: brass, strings, woodwinds and percussion; a rock band often includes a bass, a drum set, guitars and possibly a vocal. These tracks interact with one another closely and unfold over time interdependently. In music theory, we can also find extensive discussions on composition disciplines for relating sounds, e.g., harmony and counterpoint.

Lastly, musical notes are often grouped into chords, arpeggios or melodies. It is not naturally suitable to introduce a chronological ordering of notes for polyphonic music. Therefore, success in natural language generation and monophonic music generation may not be readily generalizable to polyphonic music generation.

As a result, most prior arts (see the Related Work section for a brief survey) chose to simplify symbolic music generation in certain ways to render the problem manageable. Such simplifications include: generating only single-track monophonic music, introducing a chronological ordering of notes for polyphonic music, generating polyphonic music as a combination of several monophonic melodies, etc.

It is our goal to avoid as much as possible such simplifications. In essence, we aim to generate multi-track polyphonic music with 1) harmonic and rhythmic structure, 2) multi-track interdependency, and 3) temporal structure.

[*]These authors contributed equally to this work.

To incorporate a temporal model, we propose two approaches for different scenarios: one generates music from scratch (i.e. without human inputs) while the other learns to follow the underlying temporal structure of a track given *a priori* by human. To handle the interactions among tracks, we propose three methods based on our understanding of how pop music is composed: one generates tracks independently by their *private* generators (one for each); another generates all tracks jointly with only one generator; the other generates each track by its private generator with additional *shared* inputs among tracks, which is expected to guide the tracks to be collectively harmonious and coordinated. To cope with the grouping of notes, we view bars instead of notes as the basic compositional unit and generate music one bar after another using transposed convolutional neural networks (CNNs), which is known to be good at finding local, translation-invariant patterns.

We further propose a few intra-track and inter-track objective measures and use them to monitor the learning process and to evaluate the generated results of different proposed models quantitatively. We also report a user study involving 144 listeners for a subjective evaluation of the results.

We dub our model as the multi-track sequential generative adversarial network, or MuseGAN for short. Although we focus on music generation in this paper, the design is fairly generic and we hope it will be adapted to generate multi-track sequences in other domains as well.

Our contributions are as follows:

- We propose a novel GAN-based model for multi-track sequence generation.
- We apply the proposed model to generate symbolic music, which represents, to the best of our knowledge, the first model that can generate multi-track, polyphonic music.
- We extend the proposed model to track-conditional generation, which can be applied to human-AI cooperative music generation, or music accompaniment.
- We present the Lakh Pianoroll Dataset (LPD), which contains 173,997 unique multi-track piano-rolls derived from the Lakh Midi Dataset (LMD) (Raffel 2016).
- We propose a few intra-track and inter-track objective metrics for evaluating artificial symbolic music.

All code, the dataset and the rendered audio samples can be found on our project website.[1]

Generative Adversarial Networks

The core concept of GANs is to achieve adversarial learning by constructing two networks: the *generator* and the *discriminator* (Goodfellow et al. 2014). The generator maps a random noise $\mathbf{z}$ sampled from a prior distribution to the data space. The discriminator is trained to distinguish real data from those generated by the generator, whereas the generator is trained to fool the discriminator. The training procedure can be formally modeled as a two-player minimax game between the generator G and the discriminator D:

$$\min_G \max_D \mathbf{E}_{\mathbf{x}\sim p_d}[\log(D(\mathbf{x}))] + \mathbf{E}_{\mathbf{z}\sim p_z}[1-\log(D(G(\mathbf{z})))] \,, \quad (1)$$

[1] `https://salu133445.github.io/musegan/`

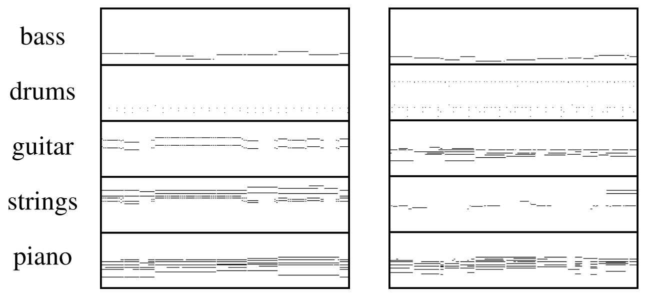


Figure 2: Multi-track piano-roll representations of two music fragments of four bars with five tracks. The horizontal axis represents time, and the vertical axis represents notes (from low-pitched to high-pitched ones). A black pixel indicates that a specific note is played at that time step.

where p_d and p_z represent the distribution of real data and the prior distribution of $\mathbf{z}$, respectively.

In a follow-up research (Arjovsky, Chintala, and Bottou 2017), they argue that using the Wasserstein distance, or the Earth Movers distance, instead of the Jensen-Shannon divergence used in the original formulation, can stabilize the training process and avoid mode collapsing. To enforce a K-Lipschitz constraint, weight clipping is used in Wasserstein GAN, while it is later on found to cause optimization difficulties. An additional *gradient penalty* term for the objective function of the discriminator is then proposed in (Gulrajani et al. 2017). The objective function of D becomes

$$\mathbf{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] - \mathbf{E}_{\mathbf{z}\sim p_z}[D(G(\mathbf{z}))] + \mathbf{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\nabla_{\hat{\mathbf{x}}}\|\hat{\mathbf{x}}\|-1)^2] \,, \quad (2)$$

where $p_{\hat{\mathbf{x}}}$ is defined sampling uniformly along straight lines between pairs of points sampled from p_d and p_g, the model distribution. The resulting WGAN-GP model is found to have faster convergence to better optima and require less parameters tuning. Hence, we resort to the WGAN-GP model as our generative model in this work.

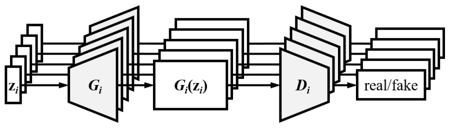
Proposed Model

Following (Yang, Chou, and Yang 2017), we consider bars as the basic compositional unit for the fact that harmonic changes (e.g., chord changes) usually occur at the boundaries of bars and that human beings often use bars as the building blocks when composing songs.

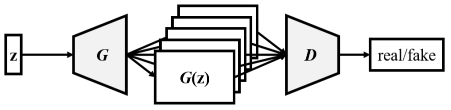
Data Representation

To model multi-track, polyphonic music, we propose to use the *multiple-track piano-roll* representation. As exemplified in Figure 2, a piano-roll representation is a binary-valued, scoresheet-like matrix representing the presence of notes over different time steps, and a multiple-track piano-roll is defined as a set of piano-rolls of different tracks.

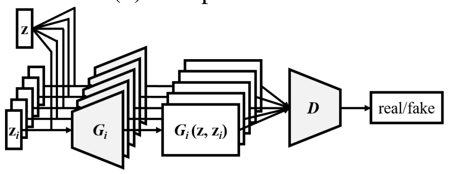
Formally, an M-track piano-roll of one bar is represented as a tensor $\mathbf{x} \in \{0,1\}^{R\times S\times M}$, where R and S denote the number of time steps in a bar and the number of note candidates respectively. An M-track piano-roll of T bars is represented as $\vec{\mathbf{x}} = \{\vec{\mathbf{x}}^{(t)}\}_{t=1}^{T}$, where $\vec{\mathbf{x}}^{(t)} \in \{0,1\}^{R\times S\times M}$ denotes the multi-track piano-roll of bar t.



(a) Jamming model



(b) Composer model



(c) Hybrid model

Figure 3: Three GAN models for generating multi-track data. Note that we do not show the real data $\mathbf{x}$, which will also be fed to the discriminator(s).

Note that the piano-roll of each bar, each track, for both the real and the generated data, is represented as a fixed-size matrix, which makes the use of CNNs feasible.

Modeling the Multi-track Interdependency

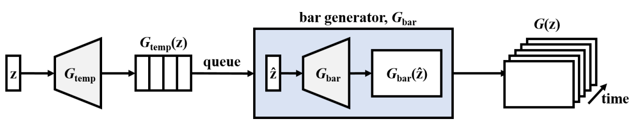
In our experience, there are two common ways to create music. Given a group of musicians playing different instruments, they can create music by improvising music without a predefined arrangement, a.k.a. jamming. Or, we can have a composer who arranges instruments with knowledge of harmonic structure and instrumentation. Musicians will then follow the composition and play the music. We design three models corresponding to these compositional approaches.

Jamming Model Multiple generators work independently and generate music of its own track from a *private* random vector $\mathbf{z}_i$, $i = 1, 2, \ldots, M$, where M denotes the number of generators (or tracks). These generators receive critics (i.e. backpropogated supervisory signals) from different discriminators. As illustrated in Figure 3(a), to generate music of M tracks, we need M generators and M discriminators.

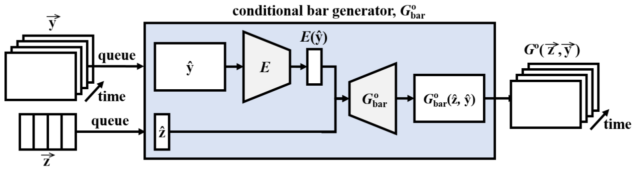
Composer Model One single generator creates a multi-channel piano-roll, with each channel representing a specific track, as shown in Figure 3(b). This model requires only one *shared* random vector $\mathbf{z}$ (which may be viewed as the intention of the composer) and one discriminator, which examines the M tracks collectively to tell whether the input music is real or fake. Regardless of the value of M, we always need only one generator and one discriminator.

Hybrid Model Combining the idea of jamming and composing, we further propose the hybrid model. As illustrated in Figure 3(c), each of the M generators takes as inputs an *inter-track* random vector $\mathbf{z}$ and an *intra-track* random vector $\mathbf{z}_i$. We expect that the inter-track random vector can coordinate the generation of different musicians, namely G_i, just like a composer does. Moreover, we use only one discriminator to evaluate the M tracks collectively. That is to say, we need M generators and only one discriminator.

A major difference between the composer model and the hybrid model lies in the flexibility—in the hybrid model we can use different network architectures (e.g., number of layers, filter size) and different inputs for the M generators. Therefore, we can for example vary the generation of one specific track without losing the inter-track interdependency.



(a) Generation from scratch



conditional bar generator, G°_{bar}
$\vec{y}$
queue
time
$\hat{y}$
E
$E(\hat{y})$
G°_{bar}
$G^{\circ}_{\text{bar}}(\hat{z}, \hat{y})$
$G^{\circ}(\vec{z}, \vec{y})$
queue
$\hat{z}$
$\vec{z}$
time

(b) Track-conditional generation

Figure 4: Two temporal models employed in our work. Note that only the generators are shown.

Modeling the Temporal Structure

The models presented above can only generate multi-track music bar by bar, with possibly no coherence among the bars. We need a temporal model to generate music of a few bars long, such as a musical phrase (see Figure 1). We design two methods to achieve this, as described below.

Generation from Scratch The first method aims to generate fixed-length musical phrases by viewing bar progression as another dimension to grow the generator. The generator consists of two sub networks, the *temporal structure generator* G_{temp} and the *bar generator* G_{bar}, as shown in Figure 4(a). G_{temp} maps a noise vector $\mathbf{z}$ to a sequence of some latent vectors, $\overrightarrow{\mathbf{z}} = \{\overrightarrow{\mathbf{z}}^{(t)}\}_{t=1}^{T}$. The resulting $\overrightarrow{\mathbf{z}}$, which is expected to carry temporal information, is then be used by G_{bar} to generate piano-rolls sequentially (i.e. bar by bar):

$$G(\mathbf{z}) = \left\{ G_{\text{bar}} \left(G_{\text{temp}}(\mathbf{z})^{(t)} \right) \right\}_{t=1}^{T} . \tag{3}$$

We note that a similar idea has been used by (Saito, Matsumoto, and Saito 2017) for video generation.

Track-conditional Generation The second method assumes that the bar sequence $\overrightarrow{\mathbf{y}}$ of one specific track is given by human, and tries to learn the temporal structure underlying that track and to generate the remaining tracks (and complete the song). As shown in Figure 4(b), the track-conditional generator G° generates bars one after another with the *conditional bar generator*, G°_{bar}. The multi-track piano-rolls of the remaining tracks of bar t are then generated by G°_{bar}, which takes two inputs, the condition $\overrightarrow{\mathbf{y}}^{(t)}$ and a *time-dependent* random noise $\overrightarrow{\mathbf{z}}^{(t)}$.

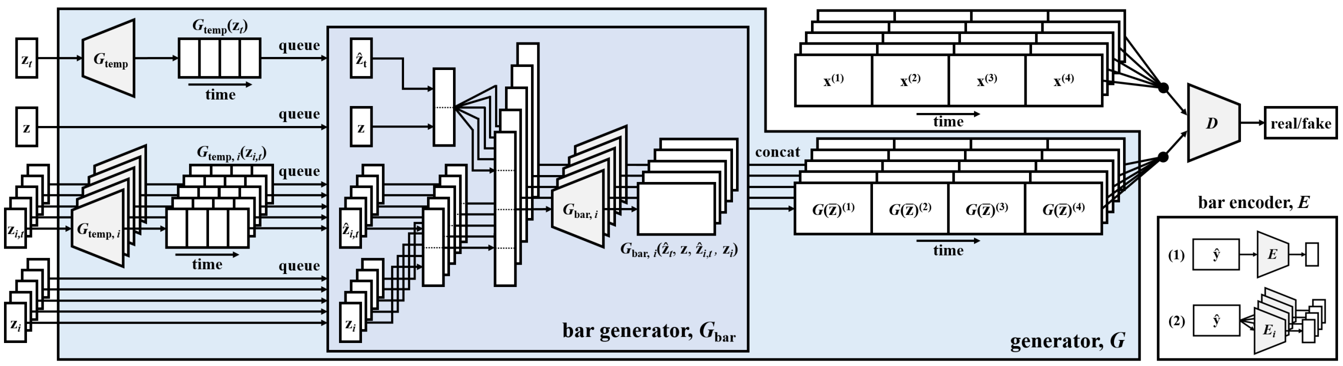


Figure 5: System diagram of the proposed MuseGAN model for multi-track sequential data generation.

In order to achieve such conditional generation with high-dimensional conditions, an additional encoder E is trained to map $\overrightarrow{\mathbf{y}}^{(t)}$ to the space of $\overrightarrow{\mathbf{z}}^{(t)}$. Notably, similar approaches have been adopted by (Yang, Chou, and Yang 2017). The whole procedure can be formulated as

$$G^{\circ}\left(\overrightarrow{\mathbf{z}}, \overrightarrow{\mathbf{y}}\right) = \left\{ G^{\circ}_{\text{bar}}\left(\overrightarrow{\mathbf{z}}^{(t)}, E\left(\overrightarrow{\mathbf{y}}^{(t)}\right)\right)\right\}_{t=1}^{T} . \qquad (4)$$

Note that the encoder is expected to extract inter-track features instead of intra-track features from the given track, since intra-track features are supposed not to be useful for generating the other tracks.

To our knowledge, incorporating a temporal model in this way is new. It can be applied to human-AI cooperative generation, or music accompaniment.

MuseGAN

We now present the MuseGAN, an integration and extension of the proposed multi-track and temporal models. As shown in Figure 5, the input to MuseGAN, denoted as $\bar{\mathbf{z}}$, is composed of four parts: an inter-track time-independent random vectors $\mathbf{z}$, an intra-track time-independent random vectors $\mathbf{z}_i$, an inter-track time-dependent random vectors $\mathbf{z}_t$ and an intra-track time-dependent random vectors $\mathbf{z}_{i,t}$.

For track i ($i = 1, 2, \ldots, M$), the shared temporal structure generator G_{temp}, and the private temporal structure generator $G_{\text{temp},i}$ take the time-dependent random vectors, $\mathbf{z}_t$ and $\mathbf{z}_{i,t}$ respectively, as their inputs, and each of them outputs a series of latent vectors containing inter-track and intra-track, respectively, temporal information. The output series (of latent vectors), together with the time-independent random vectors, $\mathbf{z}$ and $\mathbf{z}_i$, are concatenated[2] and fed to the bar generator G_{bar}, which then generates piano-rolls sequentially. The generation procedure can be formulated as

$$G(\bar{\mathbf{z}}) = \left\{ G_{\text{bar},i}\left(\mathbf{z}, G_{\text{temp}}(\mathbf{z}_t)^{(t)}, \mathbf{z}_i, G_{\text{temp},i}(\mathbf{z}_{i,t})^{(t)}\right)\right\}_{i,t=1}^{M,T} . \qquad (5)$$

For the track-conditional scenario, an additional encoder E is responsible for extracting useful inter-track features from the user-provided track.[3] This can be done analogously so we omit the details due to space limitation.

[2]Other vector operations such as summation are also feasible.

[3]One can otherwise use multiple encoders (see Figure 5).

Implementation

Dataset

The piano-roll dataset we use in this work is derived from the *Lakh MIDI dataset* (LMD) (Raffel 2016),[4] a large collection of 176,581 unique MIDI files. We convert the MIDI files to multi-track piano-rolls. For each bar, we set the height to 128 and the width (time resolution) to 96 for modeling common temporal patterns such as triplets and 16th notes.[5] We use the python library pretty_midi (Raffel and Ellis 2014) to parse and process the MIDI files. We name the resulting dataset the *Lakh Pianoroll Dataset* (LPD). We also present the subset LPD-matched, which is derived from the LMD-matched, a subset of 45,129 MIDIs matched to entries in the Million Song Dataset (MSD) (Bertin-Mahieux et al. 2011). Both datasets, along with the metadata and the conversion utilities, can be found on the project website.[1]

Data Preprocessing

As these MIDI files are scraped from the web and mostly user-generated (Raffel and Ellis 2016), the dataset is quite noisy. Hence, we use LPD-matched in this work and perform three steps for further cleansing (see Figure 6).

First, some tracks tend to play only a few notes in the entire songs. This increases data sparsity and impedes the learning process. We deal with such a data imbalance issue by merging tracks of similar instruments (by summing their piano-rolls). Each multi-track piano-roll is compressed into five tracks: bass, drums, guitar, piano and strings.[6] Doing so introduces noises to our data, but empirically we find it better than having empty bars. After this step, we get the *LPD-5-matched*, which has 30,887 multi-track piano-rolls.

Since there is no clear way to identify which track plays the melody and which plays the accompaniment (Raffel and Ellis 2016), we cannot categorize the tracks into melody, rhythm and drum tracks as some prior works did (Chu, Urtasun, and Fidler 2017; Yang, Chou, and Yang 2017).

[4]`http://colinraffel.com/projects/lmd/`

[5]For tracks other than the drums, we enforce a rest of one time step at the end of each note to distinguish two successive notes of the same pitch from a single long note, and notes shorter than two time steps are dropped. For the drums, only the onsets are encoded.

[6]Instruments out of the list are considered as part of the strings.

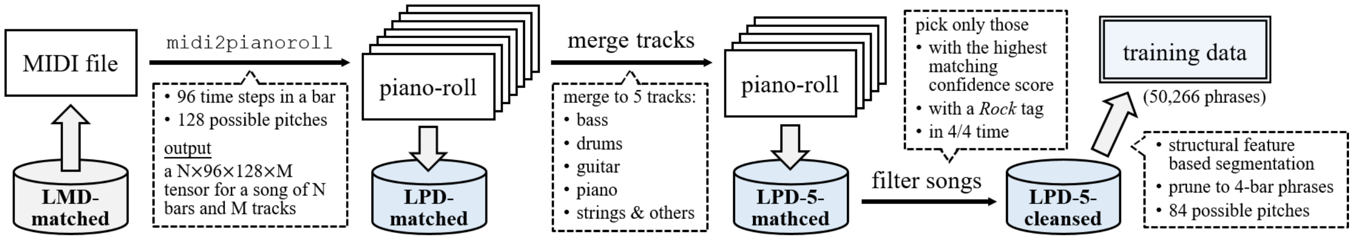


Figure 6: Illustration of the dataset preparation and data preprocessing procedure.

Second, we utilize the metadata provided in the LMD and MSD, and we pick only the piano-rolls that have higher confidence score in matching,[7] that are *Rock* songs and are in 4/4 time. After this step, we get the *LPD-5-cleansed*.

Finally, in order to acquire musically meaningful phrases to train our temporal model, we segment the piano-rolls with a state-of-the-art algorithm, *structural features* (Serrà et al. 2012),[8] and obtain phrases accordingly. In this work, we consider four bars as a phrase and prune longer segments into proper size. We get 50,266 phrases in total for the training data. Notably, although we use our models to generate fixed-length segments only, the track-conditional model is able to generate music of any length according to the input.

Since very low and very high notes are uncommon, we discard notes below `C1` or above `C8`. The size of the target output tensor (i.e. the artificial piano-roll of a segment) is hence 4 (bar) $\times$ 96 (time step) $\times$ 84 (note) $\times$ 5 (track). (See Appendix A for sample piano-rolls in the training data.)

Model Settings

Both G and D are implemented as deep CNNs. G grows the time axis first and then the pitch axis, while D compresses in the opposite way. As suggested by (Gulrajani et al. 2017), we update G once every five updates of D and apply batch normalization only to G. The total length of the input random vector(s) for each generator is fixed to 128.[9] The training time for each model is less than 24 hours with a Tesla K40m GPU. In testing stage, we binarize the output of G, which uses tanh as activation functions in the last layer, by a threshold at zero. (See Appendix B for more details.)

Objective Metrics for Evaluation

To evaluate our models, we design several metrics that can be computed for both the real and the generated data, including four intra-track and one inter-track (the last one) metrics:

- **EB**: ratio of empty bars (in %).
- **UPC**: number of used pitch classes per bar (from 0 to 12).
- **QN**: ratio of "qualified" notes (in %). We consider a note no shorter than three time steps (i.e. a 32th note) as a qualified note. QN shows if the music is overly fragmented.
- **DP**, or drum pattern: ratio of notes in 8- or 16-beat patterns, common ones for Rock songs in 4/4 time (in %).
- **TD**: or tonal distance (Harte, Sandler, and Gasser 2006). It measures the hamornicity between a pair of tracks. Larger TD implies weaker inter-track harmonic relations.

By comparing the values computed from the real and the fake data, we can get an idea of the performance of generators. The concept is similar to the one in GANs—the distributions (and thus the statistics) of the real and the fake data should become closer as the training process proceeds.

Analysis of Training Data

We apply these metrics to the training data to gain a greater understanding of our training data. The result is shown in the first rows of Tables 1 and 2. The values of **EB** show that categorizing the tracks into five families is appropriate. From **UPC**, we find that the bass tends to play the melody, which results in a UPC below 2.0, while the guitar, piano and strings tend to play the chords, which results in a UPC above 3.0. High values of **QN** indicate that the converted piano-rolls are not overly fragmented. From **DP**, we see that over 88 percent of the drum notes are in either 8- or 16-beat patterns. The values of **TD** are around 1.50 when measuring the distance between a melody-like track (mostly the bass) and a chord-like track (mostly one of the piano, guitar or strings), and around 1.00 for two chord-like tracks. Notably, TD will slightly increase if we shuffle the training data by randomly pairing bars of two specific tracks, which shows that TD are indeed capturing inter-track harmonic relations.

Experiment and Results

Example Results

Figure 7 shows the piano-rolls of six phrases generated by the composer and the hybrid model. (See Appendix C for more piano-roll samples.) Some rendered audio samples can be found on our project website.[1]

Some observations:

- The tracks are usually playing in the same music scale.
- Chord-like intervals can be observed in some samples.
- The bass often plays the lowest pitches and it is monophonic at most time (i.e. playing the melody).
- The drums usually have 8- or 16-beat rhythmic patterns.
- The guitar, piano and strings tend to play the chords, and their pitches sometimes overlap (creating the black lines), which indicates nice harmonic relations.

[7]The matching confidence comes with the LMD, which is the confidence of whether the MIDI file match any entry of the MSD.

[8]We use the MSAF toolbox (Nieto and Bello 2016) to run the algorithm: `https://github.com/urinieto/msaf`.

[9]It can be one single vector, two vectors of length 64 or four vectors of length 32, depending on the model employed.

		empty bars (**EB**; %)					used pitch classes (**UPC**)				qualified notes (**QN**; %)				**DP** (%)
		B	**D**	**G**	**P**	**S**	**B**	**G**	**P**	**S**	**B**	**G**	**P**	**S**	**D**
training data		8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
from scratch	jamming	6.59	2.33	18.3	22.6	6.10	1.53	3.69	4.13	4.09	71.5	56.6	62.2	63.1	93.2
	composer	0.01	28.9	1.34	0.02	0.01	2.51	4.20	4.89	5.19	49.5	47.4	49.9	52.5	75.3
	hybrid	2.14	29.7	11.7	17.8	6.04	2.35	4.76	5.45	5.24	44.6	43.2	45.5	52.0	71.3
	ablated	92.4	100	12.5	0.68	0.00	1.00	2.88	2.32	4.72	0.00	22.8	31.1	26.2	0.0
track-conditional	jamming	4.60	3.47	13.3	—	3.44	2.05	3.79	—	4.23	73.9	58.8	—	62.3	91.6
	composer	0.65	20.7	1.97	—	1.49	2.51	4.57	—	5.10	53.5	48.4	—	59.0	84.5
	hybrid	2.09	4.53	10.3	—	4.05	2.86	4.43	—	4.32	43.3	55.6	—	67.1	71.8

Table 1: Intra-track evaluation (**B**: bass, **D**: drums, **G**: guitar, **P**: piano, **S**: strings; values closer to the first row are better)

		tonal distance (**TD**)					
		B-G	B-S	B-P	G-S	G-P	S-P
train.		1.57	1.58	1.51	1.10	1.02	1.04
train. (shuffled)		1.59	1.59	1.56	1.14	1.12	1.13
from scratch	jam.	1.56	1.60	1.54	1.05	0.99	1.05
	comp.	1.37	1.36	**1.30**	0.95	0.98	0.91
	hybrid	**1.34**	**1.35**	1.32	**0.85**	**0.85**	**0.83**
track-conditional	jam.	1.51	1.53	1.50	1.04	0.95	1.00
	comp.	1.41	**1.36**	1.40	**0.96**	1.01	**0.95**
	hybrid	**1.39**	**1.36**	**1.38**	**0.96**	**0.94**	**0.95**

Table 2: Inter-track evaluation (smaller values are better)

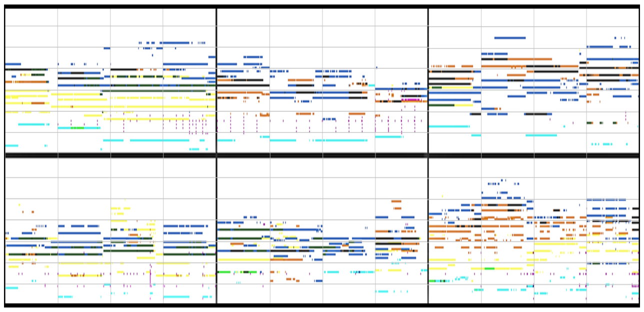


Figure 7: Example generative results for the composer model (top row) and the hybrid model (bottom row), both generating from scratch (best viewed in color—**cyan**: bass, **pink**: drums, **yellow**: guitar, **blue**: strings, **orange**: piano)

Objective Evaluation

To examine our models, we generate 20,000 bars with each model and evaluate them in terms of the proposed objective metrics. The result is shown in Tables 1 and 2. Note that for the conditional generation scenario, we use the piano tracks as conditions and generate the other four tracks. For comparison, we also include the result of an *ablated* version of the composer model, one without batch normalization layers. This ablated model barely learns anything, so its values can be taken as a reference.

For the intra-track metrics, we see that the jamming model tends to perform the best. This is possibly because each generator in the jamming model is designed to focus on its own track only. Except for the ablated one, all models perform well in **DP**, which suggests that the drums do capture some rhythmic patterns in the training data, despite the relatively high **EB** for drums in the composer and the hybrid model. From **UPC** and **QN**, we see that all models tend to use more pitch classes and produce fairly less qualified notes than the training data do. This indicates that some noise might have been produced and that the generated music contains a great amount of overly fragmented notes, which may result from the way we binarize the continuous-valued output of G (to create binary-valued piano-rolls). We do not have a smart solution yet and leave this as a future work.

For the inter-track metric **TD** (Table 2), we see that the values for the composer model and the hybrid model are relatively lower than that of the jamming models. This suggests that the music generated by the jamming model has weaker harmonic relation among tracks and that the composer model and the hybrid model may be more appropriate for multi-track generation in terms of cross-track harmonic relation. Moreover, we see that composer model and the hybrid model perform similarly across different combinations of tracks. This is encouraging for we know that the hybrid model may not have traded performance for its flexibility.

Training Process

To gain insights of the training process, we firstly study the composer model for generation from scratch (other models have similar behaviors). Figure 9(a) shows the training loss of D as a function of training steps. We see that it decreases rapidly in the beginning and then saturates. However, there is a mild growing trend after point B marked on the graph, suggesting that G starts to learn something after that.

We show in Figure 8 the generated piano-rolls at the five points marked on Figure 9(a), from which we can observe how the generated piano-rolls evolve as the training process unfolds. For example, we see that G grasps the pitch range of each track quite early and starts to produce some notes, fragmented but within proper pitch ranges, at point B rather than noises produced at point A. At point B, we can already see cluster of points gathering at the lower part (with lower pitches) of the bass. After point C, we see that the guitar, piano and strings start to learn the duration of notes and begin producing longer notes. These results show that G indeed becomes better as the training process proceeds.

We also show in Figure 9 the values of two objective metrics along the training process. From (b) we see that G can

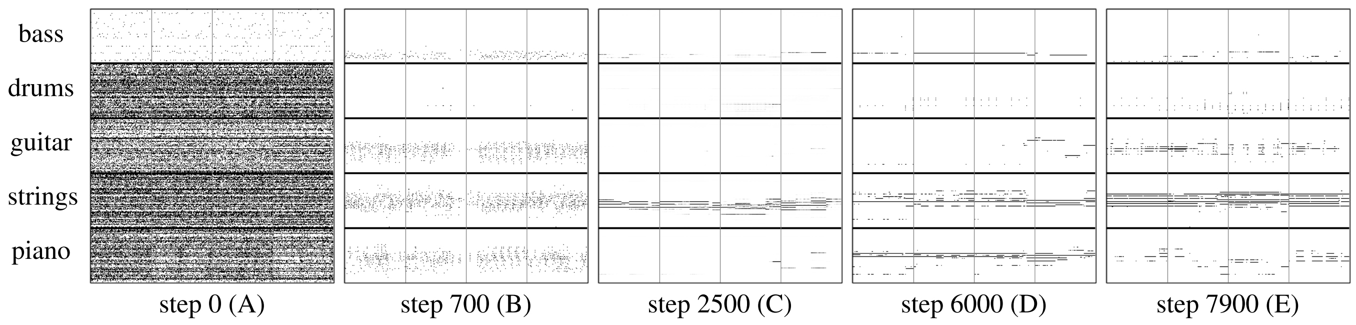


Figure 8: Evolution of the generated piano-rolls as a function of update steps, for the composer model generating from scratch.

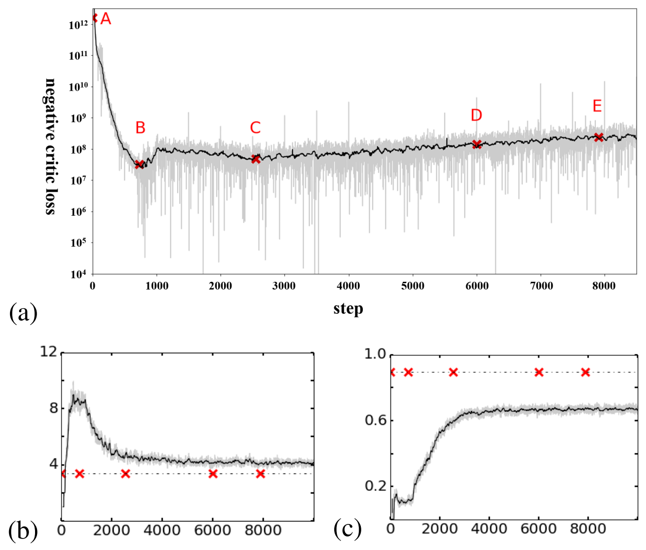


Figure 9: (a) Training loss of the discriminator, (b) the **UPC** and (c) the **QN** of the strings track, for the composer model generating from scratch. The gray and black curves are the raw values and the smoothed ones (by median filters), respectively. The dashed lines in (b) and (c) indicate the values calculated from the training data.

ultimately learn the proper number of pitch classes; from (c) we see that QN stays fairly lower than that of the training data, which suggests room for further improving our G. These show that a researcher can employ these metrics to study the generated result, before launching a subjective test.

User Study

Finally, we conduct a listening test of 144 subjects recruited from the Internet via our social circles. 44 of them are deemed 'pro user,' according to a simple questionnaire probing their musical background. Each subject has to listen to nine music clips in random order. Each clip consists of three four-bar phrases generated by one of the proposed models and quantized by sixteenth notes. The subject rates the clips in terms of whether they 1) have pleasant harmony, 2) have unified rhythm, 3) have clear musical structure, 4) are coherent, and 5) the overall rating, in a 5-point Likert scale.

			H	**R**	**MS**	**C**	**OR**
from scratch	non-pro	jam.	2.83	3.29	2.88	2.84	2.88
		comp.	3.12	**3.36**	2.95	3.13	3.12
		hybrid	**3.15**	3.33	**3.09**	**3.30**	**3.16**
	pro	jam.	2.31	3.05	2.48	2.49	2.42
		comp.	2.66	3.13	2.68	2.63	2.73
		hybrid	**2.92**	**3.25**	**2.81**	**3.00**	**2.93**
track-conditional	non-pro	jam.	**2.89**	**3.44**	2.97	**3.01**	**3.06**
		comp.	2.70	3.29	**2.98**	2.97	2.86
		hybrid	2.78	3.34	2.93	2.98	3.01
	pro	jam.	2.44	**3.32**	2.67	2.72	2.69
		comp.	2.35	3.21	2.59	2.67	2.62
		hybrid	**2.49**	3.29	**2.71**	**2.73**	**2.70**

Table 3: Result of user study (**H**: harmonious, **R**: rhythmic, **MS**: musically structured, **C**: coherent, **OR**: overall rating)

From the result shown in Table 3, the hybrid model is preferred by pros and non-pros for generation from scratch and by pros for conditional generation, while the jamming model is preferred by non-pros for conditional generation. Moreover, the composer and the hybrid models receive higher scores for the criterion Harmonious for generation from scratch than the jamming model does, which suggests that the composer and the hybrid models perform better at handling inter-track interdependency.

Related Work

Video Generation using GANs

Similar to music generation, a temporal model is also needed for video generation. Our model design is inspired by some prior arts that used GANs in video generation. VGAN (Vondrick, Pirsiavash, and Torralba 2016) assumed that a video can be decomposed into a dynamic foreground and a static background. They used 3D and 2D CNNs to generate them respectively in a two-stream architecture and combined the results via a mask generated by the foreground stream. TGAN (Saito, Matsumoto, and Saito 2017) used a temporal generator (using convolutions) to generate a fixed-length series of latent variables, which is then be fed one by one to an image generator to generate the video frame by frame. MoCoGAN (Tulyakov et al. 2017) assumed that a video can be decomposed into content (objects) and motion (of objects) and used RNNs to capture the motion of objects.

Symbolic Music Generation

As reviewed by (Briot, Hadjeres, and Pachet 2017), a surging number of models have been proposed lately for symbolic music generation. Many of them used RNNs to generate music of different formats, including monophonic melodies (Sturm et al. 2016) and four-voice chorales (Hadjeres, Pachet, and Nielsen 2017). Notably, RNN-RBM (Boulanger-Lewandowski, Bengio, and Vincent 2012), a generalization of the recurrent temporal restricted Boltzmann machine (RTRBM), was able to generate polyphonic piano-rolls of a single track. Song from PI (Chu, Urtasun, and Fidler 2017) were able to generate a lead sheet (i.e. a track of melody and a track of chord tags) with an additional monophonic drums track by using hierarchical RNNs to coordinate the three tracks.

Some recent works have also started to explore using GANs for generating music. C-RNN-GAN (Mogren 2016) generated polyphonic music as a series of note events[10] by introducing some ordering of notes and using RNNs in both the generator and the discriminator. SeqGAN (Yu et al. 2017) combined GANs and reinforcement learning to generate sequences of discrete tokens. It has been applied to generate monophonic music, using the note event representation.[10] MidiNet (Yang, Chou, and Yang 2017) used conditional, convolutional GANs to generate melodies that follows a chord sequence given a priori, either from scratch or conditioned on the melody of previous bars.

Conclusion

In this work, we have presented a novel generative model for multi-track sequence generation under the framework of GANs. We have also implemented such a model with deep CNNs for generating multi-track piano-rolls. We designed several objective metrics and showed that we can gain insights into the learning process via these objective metrics. The objective metrics and the subjective user study show that the proposed models can start to learn something about music. Although musically and aesthetically it may still fall behind the level of human musicians, the proposed model has a few desirable properties, and we hope follow-up research can further improve it.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

Bertin-Mahieux, T.; Ellis, D. P.; Whitman, B.; and Lamere, P. 2011. The Million Song Dataset. In *ISMIR*.

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*.

Briot, J.-P.; Hadjeres, G.; and Pachet, F. 2017. Deep learning techniques for music generation: A survey. *arXiv preprint arXiv:1709.01620*.

[10]In the *note event representation*, music is viewed as a series of note event, which is typically denoted as a tuple of onset time, pitch, velocity and duration (or offset time).

Chu, H.; Urtasun, R.; and Fidler, S. 2017. Song from PI: A musically plausible network for pop music generation. In *ICLR Workshop*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: A steerable model for Bach chorales generation. In *ICML*.

Harte, C.; Sandler, M.; and Gasser, M. 2006. Detecting harmonic change in musical audio. In *ACM MM workshop on Audio and music computing multimedia*.

Herremans, D., and Chew, E. 2017. MorpheuS: generating structured music with constrained patterns and tension. *IEEE Trans. Affective Computing*.

Mogren, O. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *NIPS Worshop on Constructive Machine Learning Workshop*.

Nieto, O., and Bello, J. P. 2016. Systematic exploration of computational music structure research. In *ISMIR*.

Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*.

Raffel, C., and Ellis, D. P. W. 2014. Intuitive analysis, creation and manipulation of MIDI data with pretty_midi. In *ISMIR Late Breaking and Demo Papers*.

Raffel, C., and Ellis, D. P. W. 2016. Extracting ground truth information from MIDI files: A MIDIfesto. In *ISMIR*.

Raffel, C. 2016. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. Ph.D. Dissertation, Columbia University.

Saito, M.; Matsumoto, E.; and Saito, S. 2017. Temporal generative adversarial nets with singular value clipping. In *ICCV*.

Serrà, J.; Mller, M.; Grosche, P.; and Arcos, J. L. 2012. Unsupervised detection of music boundaries by time series structure features. In *AAAI*.

Sturm, B. L.; Santos, J. F.; Ben-Tal, O.; and Korshunova, I. 2016. Music transcription modelling and composition using deep learning. In *Conference on Computer Simulation of Musical Creativity*.

Tulyakov, S.; Liu, M.; Yang, X.; and Kautz, J. 2017. MoCoGAN: Decomposing motion and content for video generation. *arXiv preprint arXiv:1707.04993*.

Vondrick, C.; Pirsiavash, H.; and Torralba, A. 2016. Generating videos with scene dynamics. In *NIPS*.

Yang, L.-C.; Chou, S.-Y.; and Yang, Y.-H. 2017. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *ISMIR*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Appendix A Samples of the Training Data

We show in Figure 10 some randomly-chosen sample piano-rolls in the training data.

Appendix B Implementation Details

The network architecture of the proposed model is tabulated in Table 4. Some details can be found below.

Random Vectors

The total length of the input random vector(s) for the whole system is fixed to 128, which can be one single vector, two vectors of length 64 or four vectors of length 32, depending on the model employed. The input random vector of G_{temp} has the same length as its output latent vectors. Thus, the total length of the input vector(s) of G_{bar} is 128 as well.

Network Architectures

Generator G_{temp} consists of two 1-D transposed convolutional layers along the (inter-bar) time axis. G_{bar} is composed of five 1-D transposed convolutional layers along the (intra-bar) time axis and two 1-D transposed convolutional layers along the pitch axis successively. A batch normalization (BN) layer is added before each activation layer.

Discriminator D consists of five 1-D convolutional layers and one fully-connected layer. The negative slope of the leaky rectified linear units (ReLU) is set to 0.2.

Encoder E has a reverse architecture as G, and *skip connections* are applied to the corresponding layers in order to speed up the training process. We constrain the number of filters to 16 for each layer to compress the representation of inter-track interdependency.

Training

We train the whole network end-to-end using the Adam optimizer with $\alpha = 0.001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$. As suggested by (Gulrajani et al. 2017), we update G (and E for the track-conditional generation model) once every five updates of D. The training time for each model is less than 24 hours with a Tesla K40m GPU.

Rendering Audios

First, we quantize the generated piano-rolls by sixteenth notes to avoid overly fragmented notes. After that, we convert the piano-rolls into MIDI files. The tracks are then mixed and rendered to stereo audio files in an external digital audio workstation.

Appendix C Sample Generated Piano-rolls

We provide examples of randomly-chosen piano-rolls generated by our models.

- Figures 11 and 12 show samples generated from scratch by the composer and the hybrid models, respectively.
- Figure 13 shows samples of track-conditional generation for composer model. Note that we use the strings track, instead of the piano track used in the Experiment section, as conditions here to show the flexibility of our models.

Input: $\mathbf{z} \in \Re^{32}$					
reshaped to (1)×32 channels					
transconv	1024	2	2	BN	ReLU
transconv	K_{temp}	3	1	BN	ReLU
Output: $G_{\text{temp}}(\mathbf{z}) \in \Re^{32\times K_{\text{temp}}}$ (K_{temp}-track latent vector)					

(a) the temporal generator G_{temp}

Input: $\mathbf{z} \in \Re^{128}$					
reshaped to (1, 1)×128 channels					
transconv	1024	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	128	3×1	(3, 1)	BN	ReLU
transconv	64	1×7	(1, 7)	BN	ReLU
transconv	K_{bar}	1×12	(1, 12)	BN	tanh
Output: $G_{\text{bar}}(\mathbf{z}) \in \Re^{96\times 84\times K_{\text{bar}}}$ (K_{bar}-track piano-roll)					

(b) the bar generator G_{bar}

Input: $\widetilde{\mathbf{x}} \in \Re^{4\times 96\times 84\times 5}$ (real/fake piano-rolls of 5 tracks)				
reshaped to (4, 96, 84) × 5 channels				
conv	128	2 × 1 × 1	(1, 1, 1)	LReLU
conv	128	3 × 1 × 1	(1, 1, 1)	LReLU
conv	128	1 × 1 × 12	(1, 1, 12)	LReLU
conv	128	1 × 1 × 7	(1, 1, 7)	LReLU
conv	128	1 × 2 × 1	(1, 2, 1)	LReLU
conv	128	1 × 2 × 1	(1, 2, 1)	LReLU
conv	256	1 × 4 × 1	(1, 2, 1)	LReLU
conv	512	1 × 3 × 1	(1, 2, 1)	LReLU
fully-connected	1024			LReLU
fully-connected	1			
Output: $D(\widetilde{\mathbf{x}}) \in \Re$				

(c) the discriminator D

Input: $\mathbf{y} \in \Re^{96\times 84}$ (piano-rolls of the given track)					
conv	16	1 × 12	(1, 12)	BN	LReLU
conv	16	1 × 7	(1, 7)	BN	LReLU
conv	16	3 × 1	(3, 1)	BN	LReLU
conv	16	2 × 1	(2, 1)	BN	LReLU
conv	16	2 × 1	(2, 1)	BN	LReLU
conv	16	2 × 1	(2, 1)	BN	LReLU
Output: $E(\mathbf{y}) \in \Re^{16}$					

(d) the encoder E

Table 4: Network architectures for the (a) temporal generator, (b) bar generator, (c) discriminator and (d) encoder. For convolutional (conv) and transposed convolutional (transconv) layers, the values represent (from left to right): number of filters, kernel size, strides, batch normalization (BN), and activation functions. For fully-connected layers, the values represent (from left to right): number of hidden nodes and activation functions. LReLU stands for leaky ReLU. $(K_{\text{temp}}, K_{\text{bar}}) = (1, 1), (1, 5), (5, 1)$ for the jamming, composer, and hybrid model, respectively.

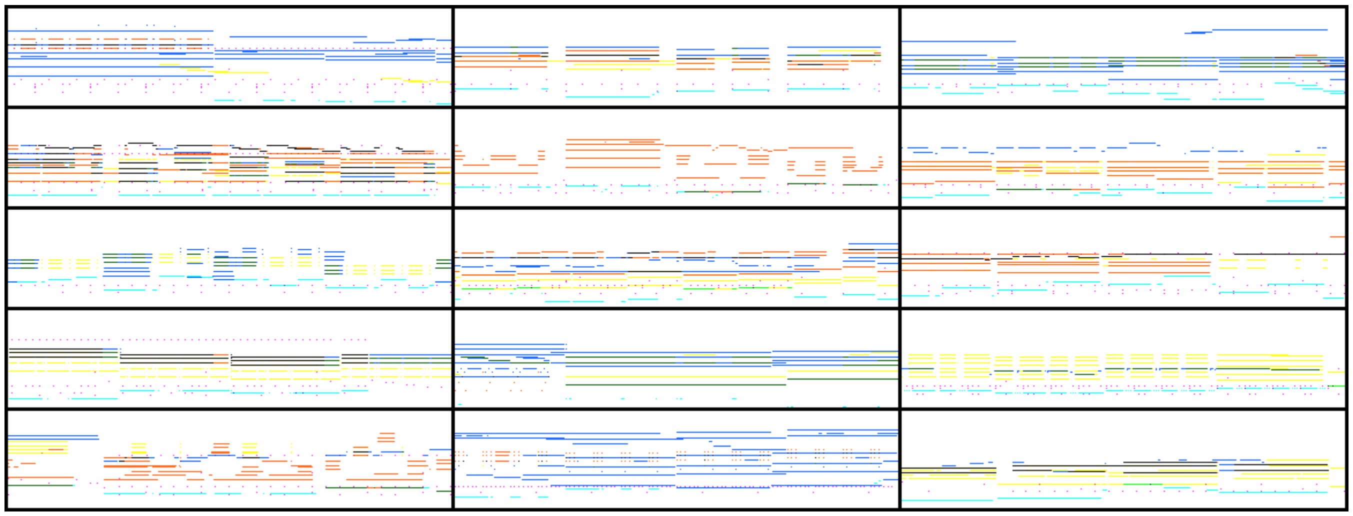
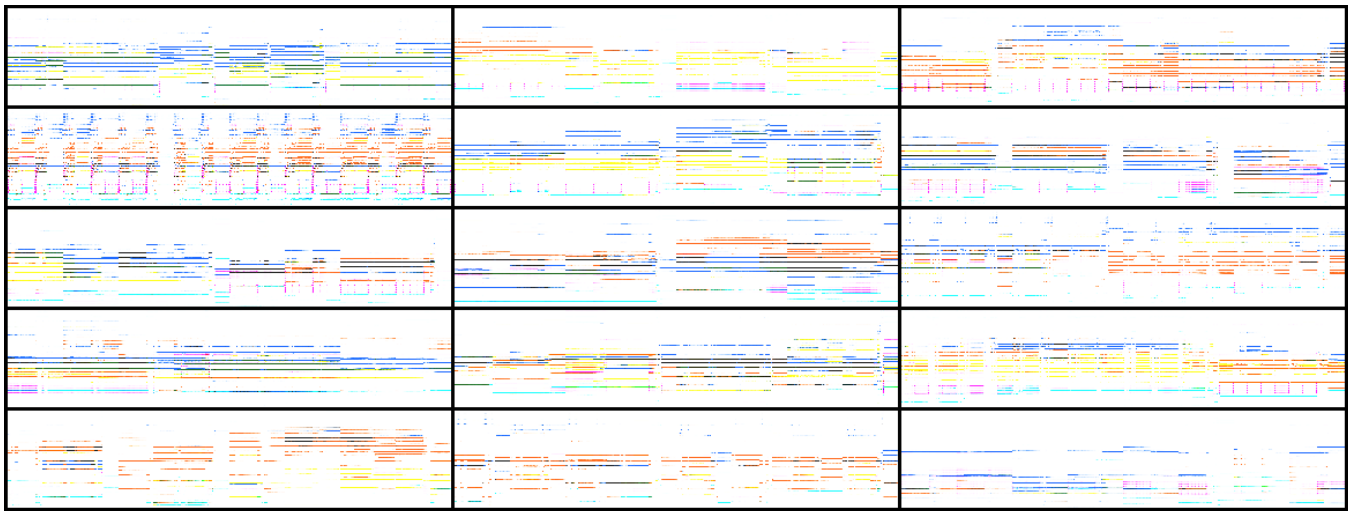
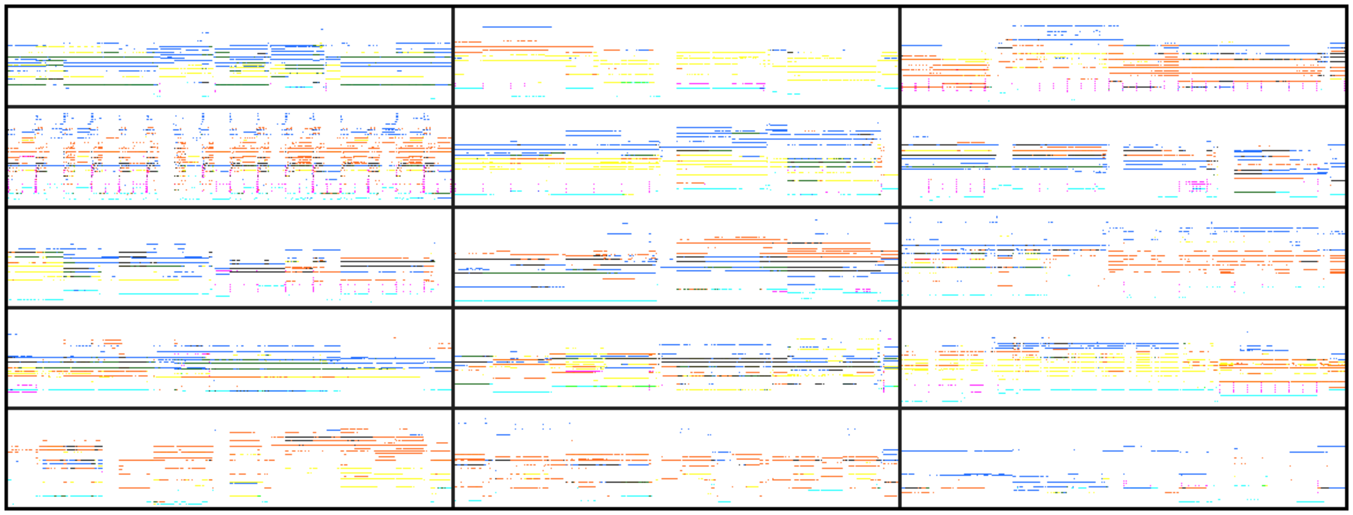


Figure 10: Sample piano-rolls in the training data (best viewed in color—**cyan**: bass, **pink**: drums, **yellow**: guitar, **blue**: strings, **orange**: piano)

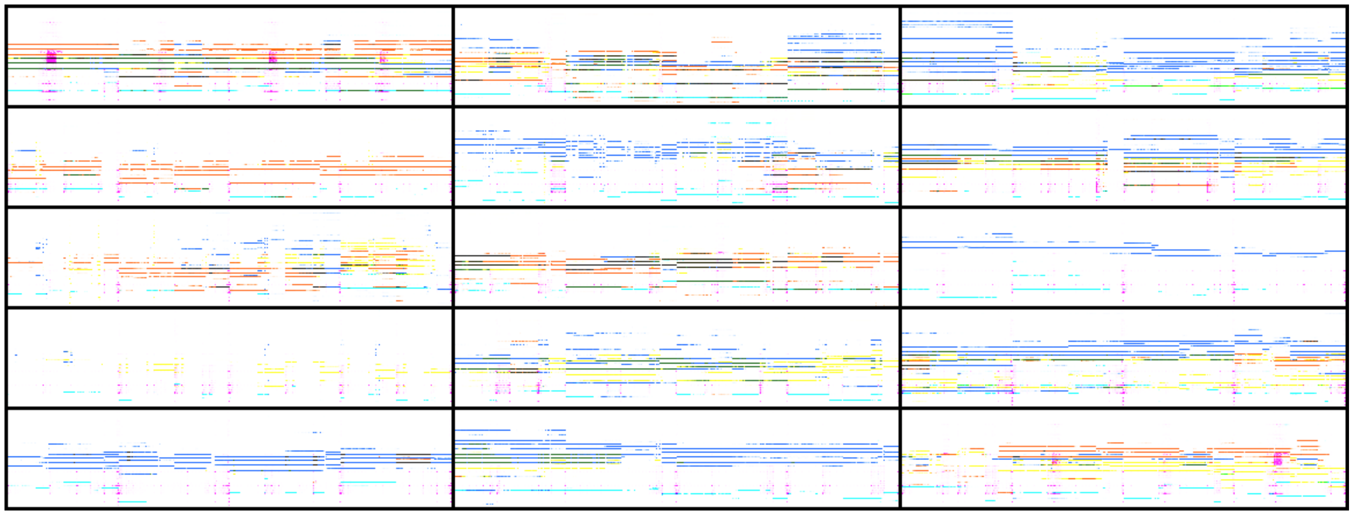


(a) original piano-rolls (before binarization)



(b) binarized piano-rolls

Figure 11: Randomly-chosen piano-rolls generated from scratch by the composer model (best viewed in color—**cyan**: bass, **pink**: drums, **yellow**: guitar, **blue**: strings, **orange**: piano). In (b) we binarize the output of G, which uses tanh as activation functions in the last layer, by a threshold at zero.


(a) original piano-rolls (before binarization)

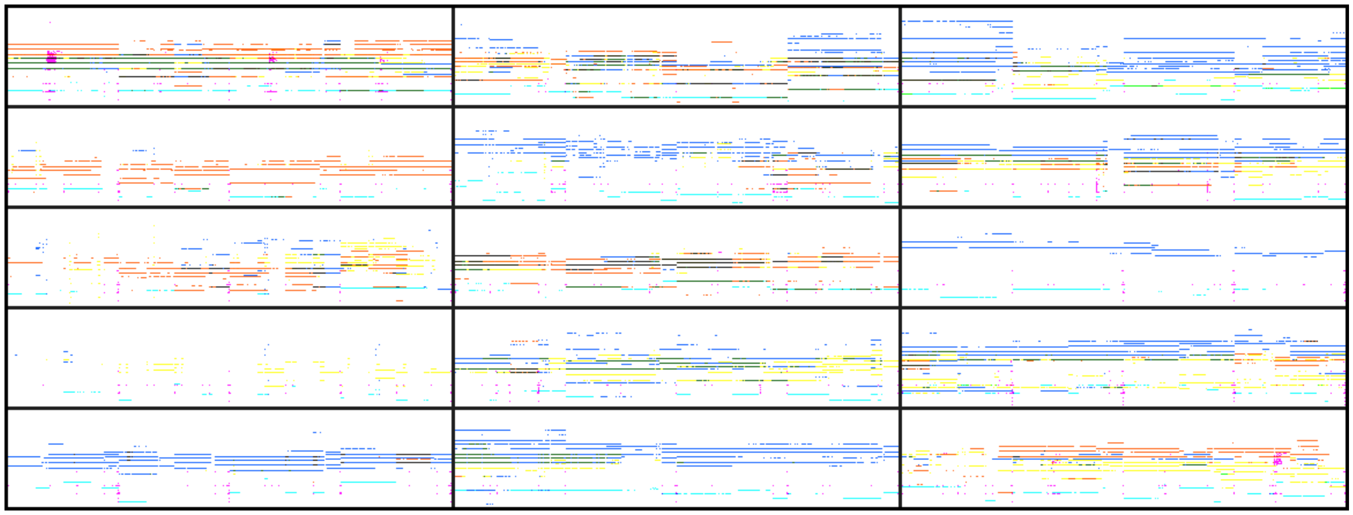
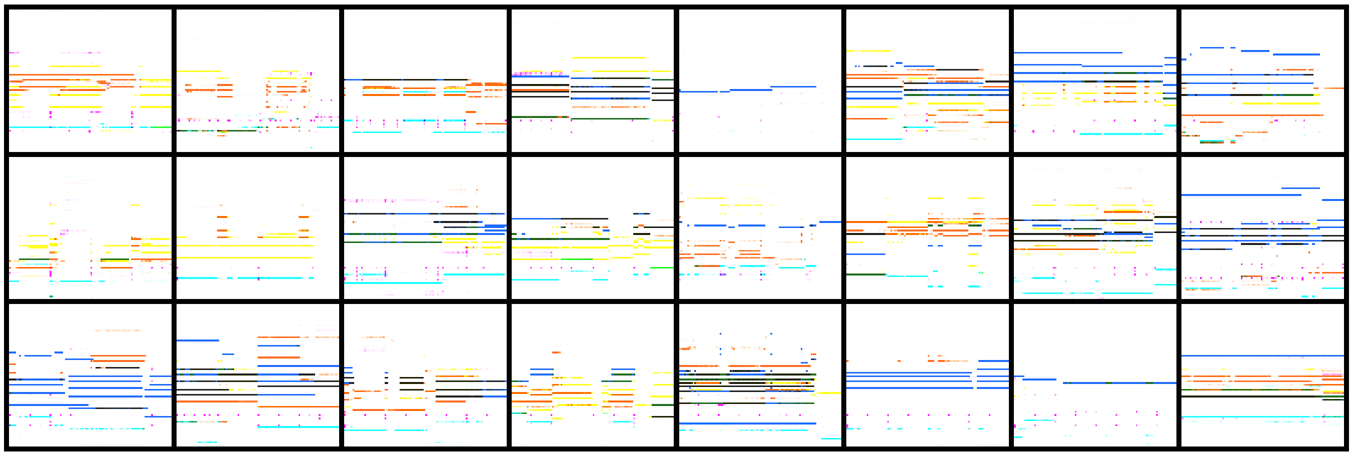

(b) binarized piano-rolls

Figure 12: Randomly-chosen piano-rolls generated from scratch by the hybrid model (best viewed in color—**cyan**: bass, **pink**: drums, **yellow**: guitar, **blue**: strings, **orange**: piano)


(a) original piano-rolls (before binarization)

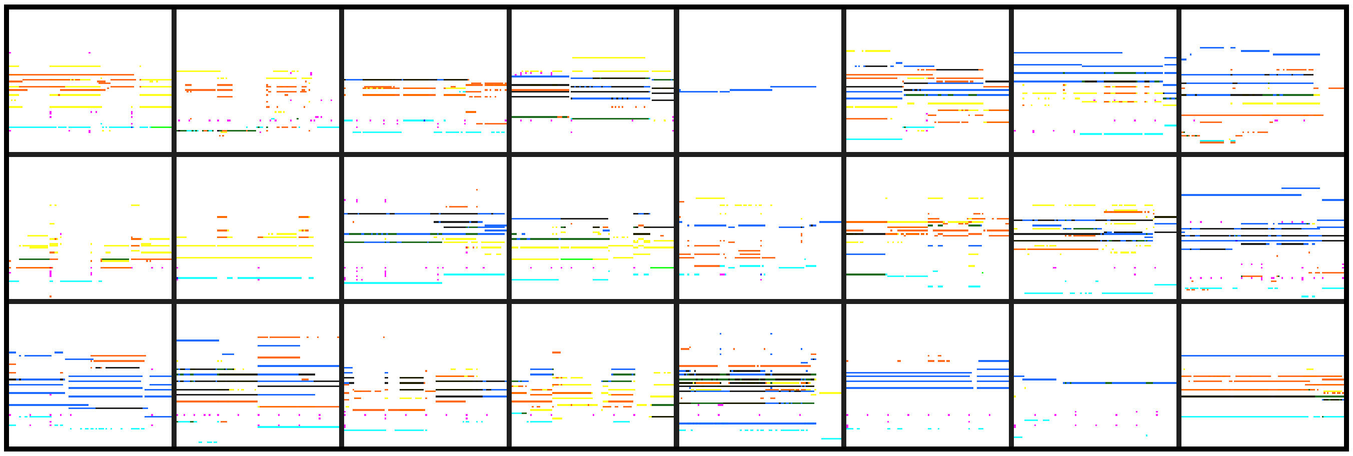

(b) binarized piano-rolls

Figure 13: Randomly-chosen generated piano-rolls for the composer model conditioned on the strings track (best viewed in color—**cyan**: bass, **pink**: drums, **yellow**: guitar, **blue**: strings (conditions), **orange**: piano)