GENERATING STRUCTURED DRUM PATTERN USING VARIATIONAL AUTOENCODER AND SELF-SIMILARITY MATRIX

I-Chieh Wei[1] **Chih-Wei Wu**[2] **Li Su**[1]
[1]Institute of Information Science, Academia Sinica, Taiwan
[2]Netflix, Inc., USA
`sma1033@iis.sinica.edu.tw, chihweiw@netflix.com, lisu@iis.sinica.edu.tw`

ABSTRACT

Drum pattern generation is a task that focuses on the rhythmic aspect of music and aims at generating percussive sequences. With the advancement of machine learning techniques, several models have been proven useful in producing compelling results. However, one of the main challenges is to generate structurally cohesive sequences. In this study, a drum pattern generation model based on Variational Autoencoders (VAEs) is presented; Specifically, the proposed model is built to generate symbolic drum patterns given an accompaniment that consists of melodic sequences. A self-similarity matrix (SSM) is incorporated in the process for encapsulating structural information. Both the objective evaluation and the subjective listening test highlight the model's capability of creating musically meaningful transitions on structural boundaries.

Index Terms - drum pattern generation, variational autoencoder, self-similarity matrix.

1. INTRODUCTION

Music generation has become an increasingly popular research field as machine learning techniques continue to thrive [3]. Generating symbolic music sequences using variants of deep neural networks (DNNs) has shown promising results with various degrees of success [6, 9, 10, 19, 22]. In the meantime, drum pattern generation, a subtask that mainly concerns the creation of drum sequences, receives relatively less attention. While some models designed for melodic sequences could be applied to drums directly [6, 19], techniques developed specifically for drum patterns are still in need of further exploration.

In Western music genres such as rock, pop, and jazz, drums usually provide the rhythmic support to melodic instruments and reflect the structure of a song. For instance, drum patterns within the same section (e.g., verse) are typically derived from the same rhythmic motif, and new patterns such as drum fills would appear around the structural boundaries (e.g., from verse to chorus). In other words, drum patterns not only enhance the rhythmic progression, but also facilitate the structural segmentation. Additionally, in order to achieve the rhythmic coherence, drum patterns tend to correlate with other instruments (e.g., rhythmic guitar and bass guitar). These particularities suggest that the structural and rhythmic information from other instruments is crucial for designing reasonable drum patterns.

To build a model that accounts for the above considerations, we explore the idea of using a self-similarity matrix (SSM) as an intermediate structural representation for drum pattern generation. Particularly, we utilize a Variational Autoencoder Generative Adversarial Network (VAE-GAN) to predict the corresponding drum SSM given the SSM of polyphonic mixture of melodic instruments. Subsequently, another VAE-based model generates MIDI drum tracks based on the predicted drum SSM. The contributions of this work include:

(i) a new way of incorporating structural information in the context of drum pattern generation,

(ii) a novel bar selection mechanism that encourages self-repetition in similar sections, and

(iii) the insights into the model's capability of handling transitions between structural sections.

2. RELATED WORK

Drum pattern generation involves the creation of rhythmic patterns with all types of drums; it is an important subtask in conditional music generation problems such as multi-track and lead sheet generation [6, 14]. According to the input and output representation, drum pattern generation models can be roughly divided into (i) symbolic-domain and (ii) audio-domain models.

Systems operating in symbolic-domain work exclusively on discretized representations such as MIDI, a format that allows systems to concentrate on essential information at the semantic level. The majority of prior studies falls into this category. Some of the early systems adopt Genetic Algorithms (GA) to create variants of rhythmic patterns [2, 8, 11, 15] and explore new patterns through a simulated evolution process. However, designing an effective fitness function is non-trivial and relies heavily on the

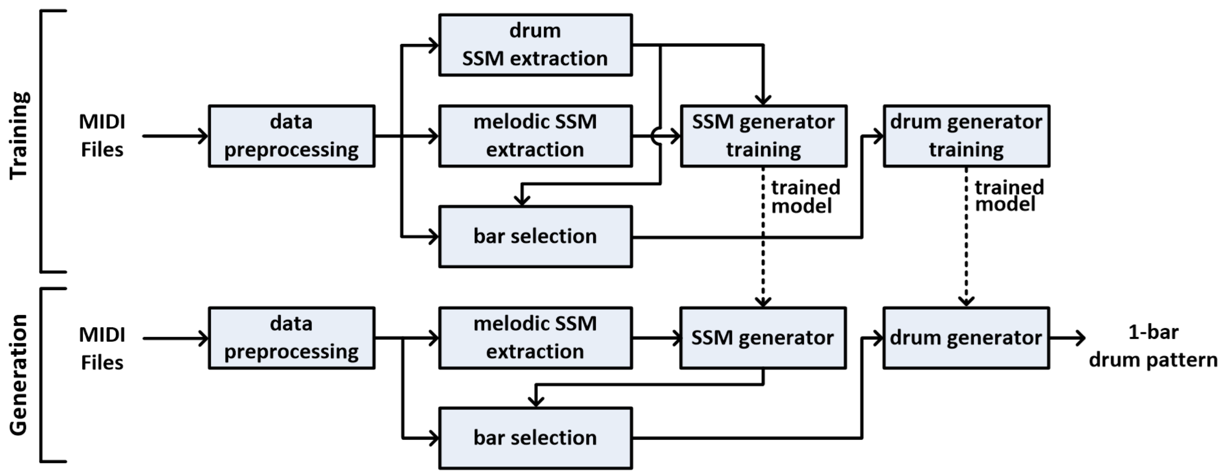

Figure 1. Block diagram of the proposed drum pattern generation system.

domain knowledge. In addition to GA, probabilistic models [16] and deep learning-based models [1,7,14] have also been proposed; these methods generally avoid predefined rules and learn from the data directly, but their performance and generality would vary depending on the data source.

Systems operating in the audio domain directly deal with continuous signals such as waveform. This type of systems usually has a higher degree of freedom in terms of the output. Training such systems requires a set of audio signals with manual annotations. To automate the annotation process, Automatic Drum Transcription (ADT) [21], another on-going research topic, would be a necessary intermediate step. As a result, prior studies in this category are relative scarce. Donahue *et al.* proposed to use generative adversarial network (GAN) for creating audio such as electronic drum beats [5], yet still, the current resulting pieces are short clips without long-term structure.

In this paper, we build our system in both audio and symbolic domains. By utilizing the audio data synthesized from the symbolic representation, we take advantage of a large collection of symbolic data and reserve the possibility of future extension to audio domain. To narrow down the scope of this task, we focus on generating drum patterns in Western music genres with a standard drum kit (e.g., Hihat, Snare Drum, Kick Drum, Toms, etc.). Inspired by previous studies on structural analysis [17] and self-similarity constraint for music generation [13], we use SSM to encapsulate the structural information and investigate its potential impacts on the generated sequences. More details are elaborated in the following sections.

3. METHOD

Figure 1 illustrates the flowchart of the proposed drum pattern generation system. We focus on generating drum patterns given a song as a conditional input. The goal of this conditional generation is to generate drum patterns that are both rhythmically and structurally compatible to the given song. To facilitate the preservation of global structure, we propose a system consisting of two generative models, namely, the SSM generator and the drum pattern generator. In the training phase, the MIDI file is separated into the drum track and the melodic track, followed by the calculation of their corresponding SSMs. The SSM generator is trained to predict drum SSM based on the given melodic SSM, and the drum generator is trained to predict drum patterns based on drum SSM and a bar selection mechanism. In the generation phase, two generators are used sequentially to predict the drum SSM and drum patterns given the melodic tracks.

3.1 Data preprocessing

The input of the proposed model is the audio rendered from all the melodic tracks in the MIDI dataset using a software synthesizer; [1] these audio signals are mono-channel sampled at 44.1 kHz. The advantage of using synthesized audio rather than symbolic data is to enable future adaptation to real-world audio. The tempo of each song is normalized to 120 BPM. Every spectrogram calculated from the audio is denoted as a tensor $\mathbf{Y}$ in shape of (b, f, t), where b is the number of bars, f is the number of frequency bins, and t are the number of time frames in a one-bar spectrogram, respectively. In this study, we set $b = 8$, $f = 84$, and $t = 96$.

For each synthesized audio clip, we first compute the constant-Q transform (CQT) spectrogram using `LibROSA` library. Subsequently, we divide the spectrogram into bars according to the beat and downbeat attributes in the MIDI data, and normalize the time step to 96 frames per bar through linear interpolation. The resulting size of each single-bar spectrogram is therefore 84×96.

The drum patterns are represented in a binary-valued matrix $\mathbf{B} \in \{0,1\}^{i \times t}$, where i and t denote the activated instruments and the number of time steps respectively.

There are two types of SSM used in this work. The first one is melodic SSM and the second one is drum SSM. Both SSMs are 256×256 matrices computed using pairwise Euclidean distance between two sets of bar-level

[1] http://www.fluidsynth.org/, last access 2019/06/28

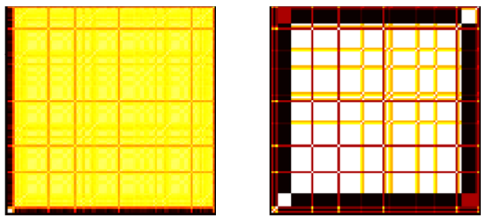

Figure 2. SSM examples of *Can't Buy Me Love* by Beatles. Left: melodic SSM. Right: drum SSM from original MIDI drum track. In the two figures, each pixel represents a Euclidean distance value between two bar-level spectrograms/symbolic drums and brighter color indicates a shorter distance (i.e., higher similarity).

spectorgrams. Since different songs vary in length, we zero-pad all the songs up to a uniform length of 256 bars. It should be noted that melodic SSM is computed using CQT spectrogram converted from synthesized audio domain data, whereas drum SSM is computed using symbolic drum track data directly.

3.2 Drum SSM generator

The motivation of generating a drum SSM from a melodic SSM is shown in Figure 2. It can be observed from Figure 2 that melodic and drum SSM are structurally correlated, and drum SSM seems to provide a relatively clearer view of structural boundaries. This difference could originate from the distinctive roles of percussive versus melodic instruments in Western music. For example, in pop or rock music, drum patterns tend to be homogeneous within a musical section. Sudden changes of drum patterns usually occur before transitioning into a new musical section. As a result, drum SSM could effectively reflect the global structure. Based on these observations, we assume that (i) it is possible to infer drum SSM given melodic SSM because they are highly correlated, and (ii) drum SSM provides more information about song structure information than the melodic SSM does.

To explicitly capture structural information of the entire song prior to drum pattern generation, a model that infers the drum SSM from a melodic SSM is needed. In this work, we propose a drum SSM generator based on the VAE-GAN model in [12]. The VAE-GAN model is a GAN consisting of a VAE-based Generator and a Discriminator. In the training stage, the VAE-based SSM generator is trained to infer the drum SSM based on the input melodic SSM, and the discriminator is trained to distinguish the VAE-generated drum SSM from the original one. The model is optimized by minimizing the total loss function $\mathcal{L}_{ssm}$ consisting of three loss terms, namely the reconstruction loss, KL-divergence loss, and the adversarial loss $\mathcal{L}_D$:

$$\begin{aligned}\mathcal{L}_{ssm} &= -\mathrm{E}_{v\sim q_s(z_s|s_m)}[\log p_s(s_d|z_s)] \\ &\quad + \mathrm{KL}(q_s(z_s|s_m)\|p(z_s)) + \mathcal{L}_D, \qquad (1)\end{aligned}$$

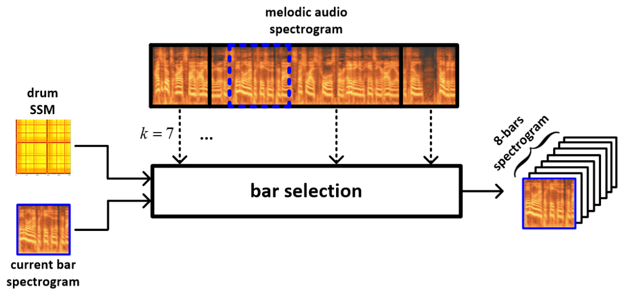

Figure 3. Bar selection mechanism to pick spectrogram data from the current and the other 7 relevant bars.

where s_d represents the generated drum SSM, s_m represents the input melodic SSM and z_s is the latent space representation of s_m.

In the generation stage, we feed the melodic SSM into the pre-trained generator and obtain the predicted drum SSM, which will be used in the following bar selection process.

3.3 Bar selection

To incorporate the structural information into drum pattern generation, the drum SSM is used in a bar selection mechanism. The proposed bar selection mechanism is based on an assumption that musical bars with higher similarities are more likely to provide relevant information for generating compatible drum patterns. To achieve this goal, we first find the k-nearest bars for every bar-level spectrogram according to the drum SSM. These bars are identified by finding the k smallest values in every column of the drum SSM. The process is illustrated in Figure 3. In our study, we set $k = 7$. The corresponding spectrograms of these eight bars are weighted and then stacked into an eight-channel feature representation, which will be used as the input to the subsequent drum pattern generator (see Section 3.4). The weighting coefficient of each channel is $1 - \mathrm{norm}(d(k, k_i))$, where k_i is the ith nearest bar, d is the Euclidean distance, and norm represents normalization over the column of the similarity matrix.

3.4 Drum pattern generator

Figure 4 illustrates the drum pattern generation model. The model is modified from the VAE proposed by Larsen et al. [12] The virtue of VAE is its capability of generating diverse output through simple manipulation in the latent space. For instance, drum patterns in-between two distinctive rhythmic styles can be generated by morphing the latent vector c and z.

To train this VAE model, we feed the encoder E with spectrogram and use symbolic drum track data as ground truth to minimizing the following loss function $\mathcal{L}_{drum}$:

$$\begin{aligned}\mathcal{L}_{drum} &= -\mathrm{E}_{z\sim q(z|y)}[\log p(x|z)] \\ &\quad + \mathrm{KL}(q(z|y)\|p(z)) + r(c;\hat{c}), \qquad (2)\end{aligned}$$

where $r(c;\hat{c}) := |c - \hat{c}|$. x represents the generated drum

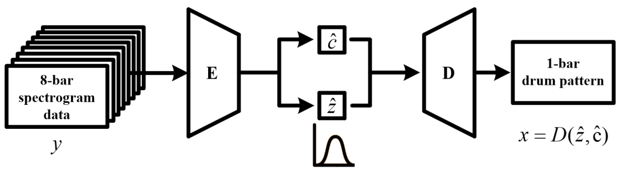

Figure 4. Illustration of the VAE-based drum pattern generator. The 8-bar spectrogram y is obtained from the bar selection mechanism. The encoder maps the spectrogram into two latent spaces, a Gaussian vector $\hat{z}$ and a note density value $\hat{c}$. The values sampled from these two latent spaces are fed to the decoder to generate drum patterns.

patterns, y is the input spectrogram, and c and $\hat{c}$ are the ground-truth and the estimated note density, respectively.

4. IMPLEMENTATION

4.1 Dataset

In this work, we use the Lakh pianoroll dataset (LPD-5) [6], a collection from cleaned MIDI data in the Lakh MIDI Dataset (LMD) [18]. LPD-5 contains 21,425 songs and each song has five tracks (piano, guitar, string, bass, and drums) extracted from the original MIDI data. The dimensionality of each bar in melodic tracks is 128 (pitch) $\times$ 96 (time step). For drum tracks, we process the data with following steps: (i) remove MIDI pitches (representing different percussive instruments) that are relatively inactive (e.g., less than 0.1% of all active drum notes); (ii) reduce time steps from 96 to 16 (see Section 4.2 for more details); (iii) apply binarization on each activated drum note, and (iv) calculate the note count in each single bar as a proxy for note density (rhythmic complexity). This procedure results in a down-sampled drum matrix with a dimensionality of 46×16.

4.2 Data cleaning

Although the LPD-5 dataset has included a series of operations to clean up the original LPD [6], incomplete or noisy examples can still be found. To further improve the data integrity, we proceed with the following steps:

First, we remove songs with inconsistent duration after synthesis. Some tracks contain only a few notes throughout the entire file, and the empty bars are automatically removed during synthesis. By excluding these songs, we ensure the correctness of information regarding song progression (e.g., beat and downbeat locations).

Second, we remove songs with empty or noisy drum tracks. Specifically, we estimate the distribution of note count from drum tracks and exclude the ones that are outside of the two standard deviation range. This outlier removal process reduces the noise and avoids issues that might be induced by data sparsity.

Finally, we apply 16th beat quantization on drum tracks and remove the notes that are largely shifted during the quantization process. In LPD-5 dataset, 95% of activated drum notes are ether on the 16th beat grid or within a tolerance range of 96th beat. Our experiment shows that less than 5% of drum notes are affected by this data cleaning process. Therefore, we believe that 16th beat grid is applicable to provide a compact data representation while retaining a meaningful temporal resolution.

generation method	cosine similarity
OMD	1.0000
ODS	0.9208
PDS	0.9164
NB	0.9056

Table 1. Similarity measure of drum SSMs from different generation method, higher similarity value indicates less deviation from original song structure after generation.

4.3 Experimental setup

After the cleaning process, 9,907 songs remain in the dataset. We randomly split the dataset into 90% and 10% for training and testing, respectively. For each song, drum and melodic parts are extracted accordingly. The melodic tracks are rendered into wave format and transformed into per-bar CQT spectrogram. For drum tracks, we segment the data in bar-level and apply binarization. The training process is done by minimizing the loss function as defined in Section 3.3 and Section 3.4; the selected optimizer is ADAM with a batch size of 64. The models are implemented using Tensorflow.

A similar preprocessing procedure is applied to the data in testing phase. The major difference between the training and the testing is the source of drum SSM. During testing, only the melodic SSM is available, and the drum SSM is predicted using the pre-trained drum SSM generator.

4.4 Model parameters

Both the input and output dimensions of the drum SSM generator (as described in Section 3.2) are 256×256. The encoder is composed of eight convolutional (CONV) layers followed by three fully-connected (FC) layers with skip connections. The FC layer is connected to a bottleneck layer that generates a 32 dimensional latent vector. Similarly, the decoder has the same layers in reverse order. The discriminator is a network similar to the encoder with a sigmoid output; this activation function is chosen for its potential probabilistic interpretation.

For drum pattern generation, the architecture of our VAE (i.e., no discriminator) is similar to the drum SSM generator. The input and output dimensions are changed to $84\times96\times8$ and $46\times16\times1$, respectively. The total numbers of parameters for the SSM generator and the drum pattern generator are around 17M and 62M, respectively. The total training time for the two models is 84 hours on a single 2080 Ti GPU. More implementation details can be found in our Github repository. [2]

[2] https://github.com/Sma1033/drum_generation_with_ssm

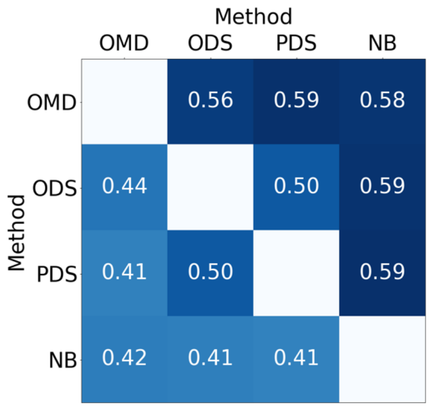

Figure 5. Head-to-head win rate between different models in the listening test.

5. EVALUATION

5.1 Compared methods

To evaluate the quality of generated drum patterns of the proposed model, we conduct both objective and subjective tests on four different derived methods:

- **(OMD) Original MIDI Drums** are the drum patterns predefined in the MIDI files. These drum patterns are directly taken from the drum tracks and serve as the oracle samples among all methods.

- **(ODS) Original Drum SSM** is the model that generates the drum patterns with our pre-trained drum pattern generator. In this case, the drum SSM used for bar selection is computed from the oracle drum tracks (i.e., OMD).

- **(PDS) Predicted Drum SSM** is the model that generates the drum patterns with our pre-trained drum pattern generator. In this case, a drum SSM used for bar selection is predicted from melodic SSM using a pre-trained SSM generator (see Section 3)

- **(NB) Neighboring Bars** is the baseline model. Instead of applying bar selection mechanism, this model simply includes the neighboring bars (i.e., previous 4 bars, current bar, and subsequent 3 bars) to create the 8-bar feature representation; no information from SSM is used. This model ignores global structure and incorporates local structure naively.

5.2 Objective test

In our objective test, we compute the similarity between the oracle drum SSM (i.e., OMD) and the SSMs of the drum patterns generated by ODS, PDS, and NB. This evaluation examines the general quality of drum patterns. Ideally, the SSM with high similarity to oracle implies a better preservation of the structural information. For simplicity, we use a standard cosine similarity as our metric. We randomly collect 100 drum tracks generated from the LPD test set, and calculate the cosine similarity between the original and generated SSMs. The average of results are presented in Table 1. Both drum SSM informed methods (i.e., ODS and PDS) outperformed NB, which suggested the competence for drum SSM to preserve structural information. Interestingly, NB can achieve a relatively high similarity score without any additional information of the global structure. This result implies the need for a better and perceptually relevant evaluation.

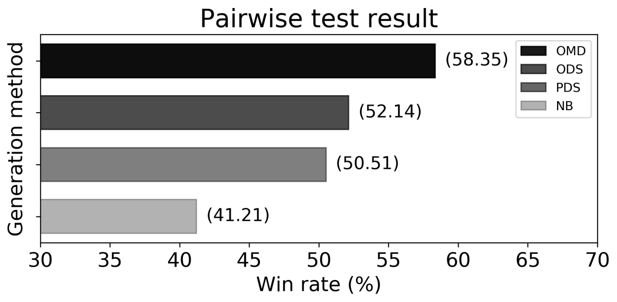

Figure 6. Global win rate of different models in pairwise listening test.

5.3 Subjective evaluation - pairwise test

5.3.1 Experiment settings

In order to evaluate the perceptual quality of different models, we conducted a listening test to compare the above mentioned four methods. In the test, various samples generated by different methods were presented to participants. There are 10 trials per test; each trial consists of two different samples derived from the same melodic track. After listening, the participants were asked to select the sample with higher rhythmic compatibility. 10 different songs are included in the evaluation; 5 songs are randomly selected from LPD-5 test set, and another 5 songs are randomly collected online in order to test the generality of the methods. In order to examine the model's capability of handling transitions, each sample contains a structural boundary. Particularly, we select samples that include one transition from the verse to the chorus, and the duration is roughly 16 seconds. Listening examples are available online. [3]

5.3.2 Results

The evaluation for 1,350 listening pairs are provided by 135 respondents. 67.5% of the subjects have no music composition experience, and 92% of the subjects have never played drums. The listening test results are presented in Figure 6 and Figure 5. Based on the results, the following observations can be made:

First, OMD performs the best among all models. This result is expected since OMD has the most stable and compelling drum patterns compared to the others. However, the margin between OMD and ODS is relatively small. One possible explanation is that OMD tends to be highly

[3] https://sma1033.github.io/drum_generation_with_ssm/

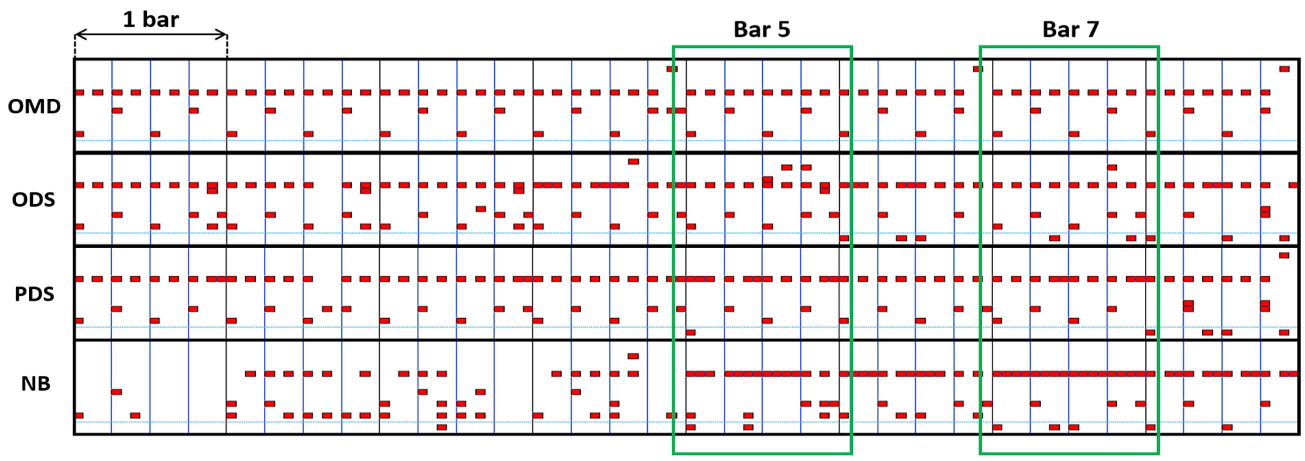

Figure 7. A 8-bar example of drum tracks from four different method in sec 5.1

consistent and predictable, and this can sometimes be regarded as conservative and even boring. On the contrary, the unexpected instrumentation or sequences in other models may accidental attract the audience's attention.

Second, it appears that both ODS and PDS outperform NB. To verify the significance of these comparisons, the t-tests of ODS/NB and PDS/NB pairs are conducted, which produce the p-values of **0.0087** and **0.001**. The results suggest that incorporating SSM in drum pattern generation models is a promising approach.

Third, the performance of PDS is comparable to ODS. The statistics from pairwise listening test suggest that the quality of drum patterns from PDS and ODS are similar for general public. This result not only indicates the effectiveness of the pre-trained SSM generator, but also shows the viability of generating meaningful drum patterns based on melodic SSM.

5.4 Subjective evaluation for professionals

Apart from the aforementioned pairwise test, we also conducted another listening test on professional musicians. The objective of this test is to collect descriptive feedback from subjects with extensive experiences in music composition and drum performance. Two professional drummers and one professional composer (with experience ranging from 3 to 14 years) participated in this test. Each participants was invited to listen to 5 selected songs; each song contains 3 different drum tracks (i.e., ODS, PDS, and NB) presented in random order, resulting in a total number of 15 clips. To further investigate the long-term structure of the generated drum patterns, the duration of each song is extended to 64 seconds. The participants were encouraged to provide detailed comments after each trial. The thematic analysis approach [4, 20] was applied to extract common themes from their comments in a bottom-up manner. The results of thematic analysis are described as follows.

The first theme regards the structural compatibility between the generated drum patterns and the melodic track. From this perspective, ODS seems to receive the best feedback among three competing models. In many occasions, the comments for ODS include "good distinction between sections", whereas PDS and NB are rarely mentioned. Moreover, for transition part between sections, ODS is reported as having active rhythmic changes (e.g., drum-fills). Overall, the professional listeners seem to prefer ODS in terms of its structure.

The second theme regards the stability and variability of the generation result. The comments from professional listeners indicate NB's ability of generating unstructured yet unexpected patterns. ODS and PDS, on the other hand, do not surprise the professional listeners. Figure 7 provides a visual example of all methods. According to Figure 7, ODS and PDS are visually more similar to OMD (e.g., bar 5 and bar 7). However, NB does provide unconventional patterns at several locations, which could be interpreted as "thinking outside of the box".

6. CONCLUSION

We have presented a conditional drum pattern generation model to generate drum patterns based on given melodic tracks. In particular, the model captures the global structure of melodic sequences using SSM and is capable of producing structurally coherent drum sequences. Results from both the objective and subjective evaluation are promising, and the comments from professional listeners also highlight the strength of the model to incorporate drum patterns with the melodic structure. Possible future directions include:

1. Design a user-friendly control interface for general public users. This could potentially encourage more interactions among users and facilitate the music creation process.

2. Generate genre-specific drum patterns according to the input condition. With genre-specific contents, users may customize the styles of outputs.

3. Develop a model that can ultimately perform drum pattern generation in audio domain. This would enable more potential applications in real-world scenarios and increase the expressivity of the model.

7. ACKNOWLEDGEMENT

We thank three professional musicians, Hsin-Ming Lin, Ying-Shu Kuo, and Jian-Hong Chen, for their qualitative assessment on generated drum patterns and our colleague, Yu-Fen Huang, for her generous assistance on the thematic analysis of the questionnaires. We would also like to express our gratitude to every participant of the subjective listening test for providing responses and constructive feedback.

8. REFERENCES

[1] Eric Battenberg and David Wessel. Analyzing Drum Patterns Using Conditional Deep Belief Networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 37–42, 2012.

[2] Gilberto Bernardes, Caros Guedes, and Bruce Pennycook. Style emulation of drum patterns by means of evolutionary methods and statistical analysis. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–4, 2010.

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation - A survey. *CoRR*, abs/1709.01620, 2017.

[4] Kathy Charmaz and Linda Liska Belgrave. Grounded theory. *The Blackwell encyclopedia of sociology*, 2007.

[5] Chris Donahue, Julian Mcauley, and Miller Puckette. Adversarial Audio Synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–16, 2019.

[6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[7] Hamid Eghbal-zadeh, Richard Vogl, Gerhard Widmer, and Peter Knees. A GAN based drum pattern generation UI prototype. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2018.

[8] Damon Horowitz. Generating Rhythms with Genetic Algorithms. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 1994.

[9] Cheng-zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2017.

[10] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, and Douglas Eck. Music transformer: generating music with long-term structure. *ICLR*, 2019.

[11] Maximos A. Kaliakatsos-Papakostas, Andreas Floros, Nikolaos Kanellopoulos, and Michael N Vrahatis. Genetic Evolution of L and FL – systems for the Production of Rhythmic Sequences. In *Proceedings of the Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pages 461–468, 2012.

[12] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, 2016.

[13] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems.*, 2018.

[14] Hao-Min Liu and Yi-Hsuan Yang. Lead sheet generation and arrangement by conditional generative adversarial network. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 722–727, 2018.

[15] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà. Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms. In *Proceedings of the Sound and Music Computing Conference (SMC)*, 2015.

[16] Jean-François Paiement, Yves Grandvalet, Samy Bengio, and Douglas Eck. A Generative Model for Rhythms. In *Neural Information Processing Systems (NIPS), Workshop on Brain, Music and Cognition*, 2007.

[17] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-based music structure analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 625–636, 2010.

[18] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.* PhD thesis, Columbia University, 2016.

[19] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical Variational Autoencoders for Music. In *Neural Information Processing Systems (NIPS)*, volume 256, pages 1–6, 2017.

[20] Anselm Strauss and Juliet M Corbin. *Grounded theory in practice*. Sage, San Jose State University, USA, 1997.

[21] Chih Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Mueller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(9):1457–1483, 2018.

[22] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2017.