NEURAL DRUM ACCOMPANIMENT GENERATION

Rishabh Dahale[1] **Vaibhav Talwadker**[1] **Prateek Verma**[2] **Preeti Rao**[1]
[1] Department of Electrical Engineering, Indian Institute of Technology Bombay, India
[2] Stanford University
dahalerishabh1@iitb.ac.in, talwadkerv@gmail.com, prateekv@stanford.edu, prao@ee.iitb.ac.in

ABSTRACT

Transformer and its variants have demonstrated state-of-the-art performance on various tasks, including music generation, and yet existing works have not explored this model for drum accompaniment generation. In this work, a transformer model is designed to generate an accompanying symbolic drum pattern conditioned on an input melodic track. We propose a data representation scheme for symbolic music, in which silences are also considered. It can be easily scaled to an arbitrary number of instruments, unlike the popular serialized grid, thus eliminating the need of sampling. We propose a loss function accounting for imbalance in the occurrence of the different percussion instruments and report performance on the Lakh Pianoroll dataset via two musically relevant evaluation metrics.

1. INTRODUCTION

Learning-based deep neural networks have been prominent in recent music generation research [1–8]. However, drum pattern generation, a subtask of music generation, has attracted relatively less attention with its challenges of incorporating aspects of the melody, such as its verse transitions and repetitions in a setting that combines coherence and diversity. A number of deep learning methods like RNNs [1,2], GANs [3], Variational Autoencoder (VAE) [4,6] and even transformers [5, 9–11] have been applied for the task of music generation. Although these methods have shown good results, they use a serialized grid representation to circumvent the problem of polyphony estimation and sampling. In this work, we have trained a transformer encoder model, similar to the one trained by [12], for drumbeat generation in the symbolic domain and proposed a data representation that can be easily scaled to an arbitrary number of instruments. [5] showed that transformers could achieve compelling results for generating long-duration piano music, which we are trying to leverage for drum pattern generation.

2. DATASET

In this work, we have used the Lakh pianoroll dataset (LPD-5 cleansed) [3] which is derived from Lakh MIDI dataset [7]. LPD-5 cleansed contains 21,425 songs consisting of 5 tracks: Piano, Guitar, Strings, Bass, and Drums. Among these five instruments, we choose Piano, Guitar, Strings, and Bass as the melody line (input) and have generated drums patterns conditioned on this input. All the songs are of `4/4` time signature, and each beat is divided into 24 parts in the dataset.

3. IMPLEMENTATION

3.1 Data Processing

Each beat is downsampled from 24 parts to 8 parts to compress the inputs while retaining 98.6% of the drum beats. Among the 21,425 multitrack songs, 16,832 were used for training, while 4,603 were used for validation. Non-overlapping contiguous 12 bar samples were extracted from all the songs to generate 133,842 12 bar training samples and 22,794 12 bar testing samples.
Each instrument in melodic track is represented with a 64-dimensional vector at every time step, of which the first dimension represents the silence state. This binary value denotes whether the instrument is playing some note or is silent for that time step. The remaining 63 dimensions capture the velocity of `MIDI 21` to `MIDI 83` pitches at each timestep as shown in figure 2a. For percussion instrument representation, we have used a 17-dimensional vector similar to the melody vector, i.e., first dimension representing silence state while other 16 dimensions capture the following instruments: snare drum, open hi-hat, close hi-hat, kick drum, ride cymbal, crash cymbal, low-floor tom, high-floor tom, high tom, hi-mid tom, low tom, cowbell, pedal hi-hat, tambourine, cabasa, and maracas. As each percussive instrument is not equally used while playing drums, the distribution of the number of samples is skewed, as shown in figure 1. These 16 instruments capture `85.3%` of all the percussion strokes.

3.2 Proposed Model

A semantic overview of the proposed architecture is shown in figure 2b. As the input representation of pianoroll is highly sparse, it is first passed through a series of embedding layers. A layer of positional encoding follows this to preserve the sub-beat position of each individual vector. This layer concatenates the fixed sinusoidal positional representation proposed by [13] followed by a dense layer

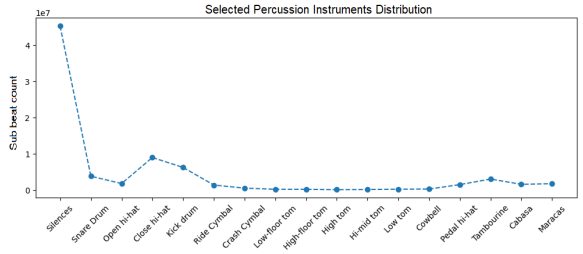

Figure 1. Number of occurrences of each of the selected Drum instruments

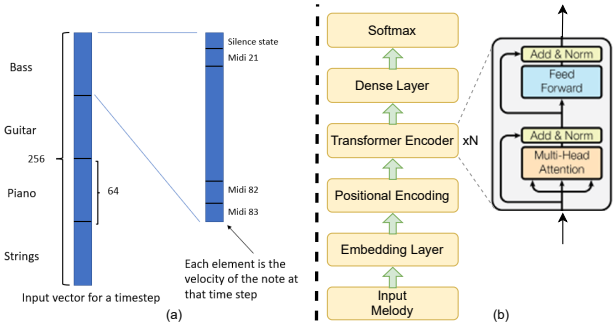

Figure 2. (a) Input representation for our model. (b) Our model for Drum beat generation from given melody input using the multi head self-attention module

to project the resulting matrix to the original dimension. These positional embedded vectors are passed through N layers of transformer encoder. It is finally passed through a dense layer with softmax activation. For evaluation, we have used a simple thresholding mechanism for generating drum beats, i.e., instruments with probability (softmax output) above a threshold are considered active. Another method for decoding the output is by sampling from the distribution as proposed by [14], but such kind of approach does not ensure the continuity in the predicted drum pattern.

3.3 Loss Function

The output of our model is a matrix indicating the ON/OFF state of different drum instruments. We used weighted categorical cross-entropy loss. The ground truth values are normalized by multiplying with a factor of $\frac{1}{k}$ [15], where $k \geq 1$ is the number of active instruments in that timestep. This resulted in significantly improved performance to handle multi-label predictions over sigmoid + euclidean loss. As every output state does not have an equal representation in the dataset (figure 1), the outputs are weighted inversely to the occurrence of the instruments. Loss is calculated as the cross-entropy of the modified ground truth ($\frac{1}{k}$ scaling) and weighted softmax activations.

4. EVALUATION METRICS

Our evaluation metrics are designed to compare certain musically relevant properties of original drum patterns with the generated ones. We propose following 2 metrics:

Polyphony Correlation (PC): Polyphony is defined as the number of independent voices being played at a given moment of time. PC is the Pearson correlation of the polyphony of the generated track and real drummer across

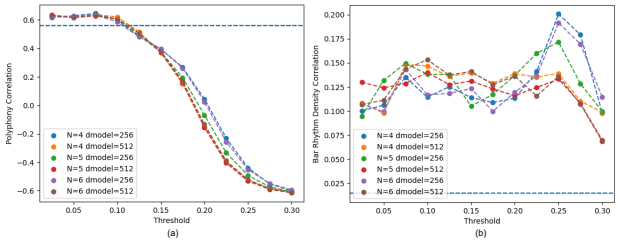

Figure 3. (a) PC plot for different thresholds (b) BRDC plot for different thresholds. Benchmark values of PC=0.56 and BRDC=0.0143 are shown in both the plots by the horizontal dashed blue line

time-steps. This metric will help us understand if the model could learn the correct collective onsets and offsets of different drum instruments.

Bar Rhythm Density Correlation (BRDC): Rhythm density (RD) is defined as the fraction of non-silent timesteps in a given sequence to the total number of timesteps. RD is calculated for 12 bars individually, and Pearson correlation of the original drum RD and predicted drum RD is taken. This metric captures the model learning at fills and improvisation positions of the real drummer.

Benchmark values of the above two metrics are calculated by pairing the original drum pattern with another randomly chosen drum pattern of another song from the dataset. With this, we get the PC as 0.56 and BRDC as 0.0143. The PC is expected to be higher as all the songs belong to the Rock genre, which emphasizes alternate beats.

5. RESULTS AND FUTURE WORK

We have tuned this model on following parameters $N = 4, 5, 6$ and $d_{model} = 256, 512$, where N is the number of transformer encoder layers used and d_{model} is the dimension of the output of the multi-head attention module. To understand the effect of thresholding of output probabilities on generated drum patterns, we tested our model with the following thresholds: 0.025, 0.05, 0.075, ..., 0.275, 0.3. Variation of the PC and BRDC can be seen in figure 3.

The blue horizontal line in figure 3 shows the benchmark values of the evaluation metrics as explained above. All the trained models exhibit a better PC than benchmark value at a lower threshold. This indicates that our model could partially learn the drum pattern (collective note onsets and offsets) from a given melody. It can also be verified from the BRDC plot (figure 3b) that all the models were above the benchmark value by a significant amount, indicating that the model was able to partially learn the fills and improvisations done by a real drummer. Sample results can be found at this link.

In future work, we plan to explore the different sampling methods from the output distribution instead of using a simple thresholding-based approach. Moreover, we plan to train another network to grade the quality of the predicted drum pattern for the melody and incorporate this grading in the loss function to help our model learn variations. We also plan to explore RL algorithms to train the model to help it generate better and realistic drum accompaniments.

6. REFERENCES

[1] D. Makris, M. Kaliakatsos-Papakostas, I. Karydis, and K. L. Kermanidis, "Combining lstm and feed forward neural networks for conditional rhythm composition," in *International conference on engineering applications of neural networks*. Springer, 2017, pp. 570–582.

[2] ——, "Conditional neural sequence learners for generating drums' rhythms," *Neural Computing and Applications*, vol. 31, no. 6, pp. 1793–1804, 2019.

[3] 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.

[4] I.-C. Wei, C.-W. Wu, and L. Su, "Generating structured drum pattern using variational autoencoder and self-similarity matrix." in *ISMIR*, 2019, pp. 847–854.

[5] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[6] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 35–39.

[7] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[8] N. Jiang, S. Jin, Z. Duan, and C. Zhang, "Rl-duet: Online music accompaniment generation using deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 710–718.

[9] T. Nuttall, B. Haki, and S. Jorda, "Transformer neural networks for automated rhythm generation," 2021.

[10] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[11] O. Thörn, "Ai drummer-using learning to enhancearti cial drummer creativity," 2020.

[12] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[15] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 181–196.