AUTOOSU: AUDIO-AWARE ACTION GENERATION FOR RHYTHM GAMES

Sihun Lee **Dasaem Jeong**
Department of Art & Technology, Sogang University
Seoul, South Korea

ABSTRACT

Rhythm-based video games challenge players to match their actions with musical cues, turning songs into interactive experiences. The design of the game charts, which dictate the timing and placement of on-screen notes, are manually crafted by players and developers. With AutoOsu, we introduce a CRNN-based model for generating rhythm game charts for a given audio track, conditioned on an intended difficulty level. In previous studies, this task is often divided into two: onset detection, which determines timing points for notes; and action generation, where notes are distributed among a set of available keys. These two sub-tasks are typically handled with two separately trained models, and audio information is only given to the onset detection model. We instead jointly train the two recurrent layers who both receive audio information, which streamlines the training process and helps better utilize musical features.

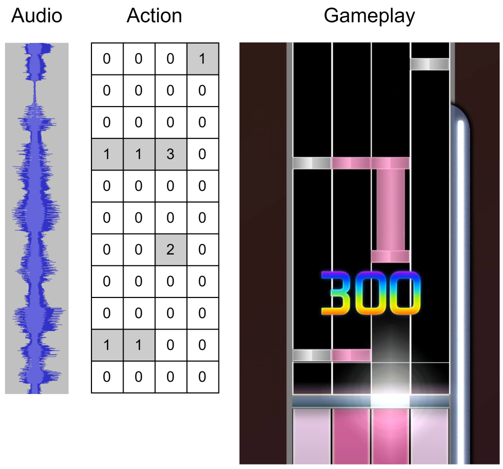

Figure 1. *osu! mania* gameplay. Notes fall down onto the line at the bottom, and the player has to hit the corresponding keys at the correct time.

1. INTRODUCTION

Rhythm games are a popular genre of modern video games. The gameplay of most rhythm games involves the player hitting specific keys at precise timings according to the notes that appear on the screen, the sequence of which is often called a *chart*. The charts are manually created by game developers or community members to follow the rhythmic and melodic structures of a song. By invoking a sense of moving along to the music, rhythm games provide players with a new and entertaining way of experiencing songs they like.

As a machine learning task, generating charts for rhythm games from a given audio input of music can be regarded as similar to music onset detection [1] and automatic music transcription [2]. However, a key difference is that there is no definitive answer for how one should chart for a given music track; a model needs to learn a wide range of idiosyncratic patterns in charts that are rooted in the physicality of how the games are played, the conventional note combinations used by the community, and how chart creators tend to interpret different musical features of a song. In this sense, the task can also be regarded as conditional symbolic music generation.

In previous studies, this task is typically decomposed into two and handled with separate models. Donahue et al. [3] name the two sub-tasks *step placement* and *step selection*. In the former, the precise timing points of notes are determined—a process analogous to onset detection. In the latter, the notes are then distributed among a set of keys the player can hit. Liang et al. [4] adopt a similar 2-step approach composed of *timestep generation* and *action type generation*. For clarity, we define these two problems as *onset detection* and *action generation*.

In both studies mentioned above, the audio information is utilized by the onset detection model alone, and the action generation model is only conditioned on the time difference between the previous and the current note. In this research, we instead jointly train the two models and provide audio context to both modules, simplifying the training pipeline and fully utilizing the musical features extracted from audio tracks.

2. DATA

We focus on *osu! mania*, one of the game modes of the popular rhythm game *"osu!"*, presented in Figure 1. Content for *osu!* is mainly produced by community members,

Number of songs	400 (16.4 hrs)
Avr. song length	148 secs
Number of charts	1,126
Notes / chart	676.54

Table 1. Dataset statistics

who create charts for songs and upload them to the game's database. Among the publicly available charts, we collected 400 songs to compose the dataset, handpicking them to maintain balance in genre and difficulty. Statistics of the dataset are provided on Table 1.

3. METHOD

3.1 Feature Extraction

We extract raw audio tracks from charts in the dataset. To preserve a wider range of low and high-level musical features, we perform multiple timescale short-time Fourier transforms in window lengths of 23ms, 46ms, and 93ms [5]. We use a stride of 10ms, creating a grid of time frames to which the inputs and outputs of all of the model's components are aligned.

Following [3], we compute rhythmic information for each time frame: 1) *Beat number*, an integer that denotes the beat in a measure that contains the time step; and 2) *Beat phase*, representing the fraction of a beat at which the time step occurs. For this, we assume that all audio tracks are of consistent tempo and in a time signature of 4/4.

For action generation, we focus on the 4-key mode of *osu! mania*. Each of the four keys can be assigned one of the following actions at any time: *no note, normal note, hold start*, and *hold end*. This results in a total of $4^4 = 256$ possible *action tokens* for each time step.

To condition chart generation on an intended difficulty level, we collect the *star rating* of each chart, which is an objective difficulty measure determined by the game's internal logic. We limit the dataset to only contain charts of difficulty levels lower than 4.0.

3.2 Model Architecture

As presented in Figure 2, the model comprises a stack of convolution layers for processing audio, a bidirectional Gated Recurrent Unit (GRU) [6] for onset detection, and an auto-regressive unidirectional GRU for action generation, which resembles an auto-regressive model for piano music transcription [7]. The arriving spectrogram is forwarded through the convolution stack with a gradual increase in the channel dimension and then flattened along the channel and frequency dimensions. While preserving the temporal dimension, we concatenate the following tensors to the audio representation: beat number embeddings, beat phase embeddings, and difficulty projection, which is produced by feeding a difficulty scalar into a multilayer perceptron. The resulting concatenated tensor is used as input for both GRUs.

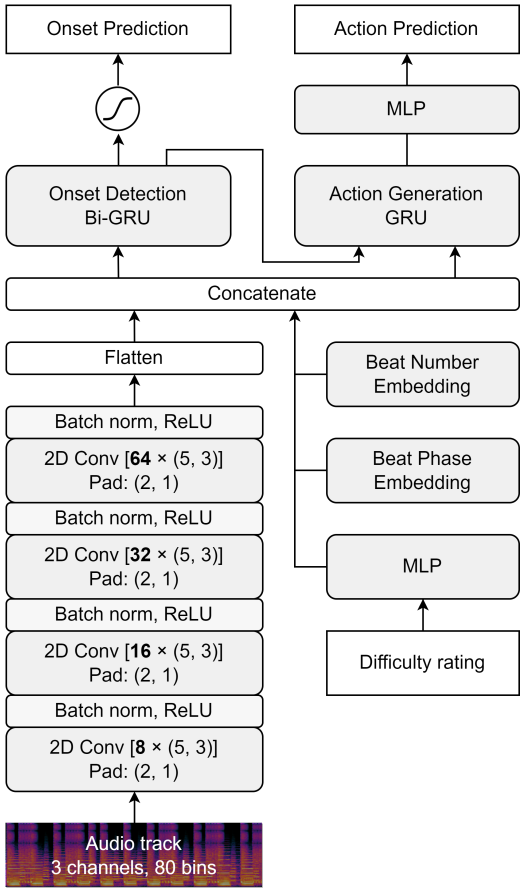

Figure 2. Overall model architecture

Additionally, the action generation GRU receives the output of the onset detection GRU as part of its input. It also receives the predicted action token from the previous time step, making it the only autoregressive layer in the model. Since the vast majority of ground-truth time steps contain no notes, we utilize binary and multi-class focal loss [8] for onset detection and action generation, respectively, to mitigate class imbalance.

4. RESULTS

We compare the proposed model against a control model, which only utilizes audio context during onset detection. While the two models show no significant difference in quantitative metrics, such as perplexity on the validation set, we found that the generated charts by the proposed model excel in aligning special patterns (hold notes, compound notes) with salient musical events.

By inputting an audio track along with manual annotation on tempo and the offset for the first downbeat, the model can be used to perform inference with arbitrary songs. For annotation, we use the tap-tempo feature included in *osu!*, since it also calculates offset along with tempo. Further examples and demos are provided in the link [1]. We also share the dataset, source code, and model weights [2].

[1] https://issyun.github.io/autoosu

[2] https://github.com/issyun/AutoOsu

5. REFERENCES

[1] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2014, pp. 6979–6983.

[2] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[3] C. Donahue, Z. C. Lipton, and J. McAuley, "Dance dance convolution," in *International conference on machine learning*. PMLR, 2017, pp. 1039–1048.

[4] Y. Liang, W. Li, and K. Ikeda, "Procedural content generation of rhythm games using deep learning methods," in *Entertainment Computing and Serious Games: First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11–15, 2019, Proceedings 1*. Springer, 2019, pp. 134–145.

[5] P. Hamel, Y. Bengio, and D. Eck, "Building musically-relevant audio features through multiple timescale representations," 2012.

[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, p. 1724.

[7] T. Kwon, D. Jeong, and J. Nam, "Polyphonic piano transcription using autoregressive multi-state note model," in *Proc. of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.