

DOSU: DIFFICULTY-CONDITIONED ON-GRID SEQUENCE UNMASKING FOR OSU!TAIKO CHART GENERATION

Junyoung Koh^{♫#} Hao-Wen Dong[#]

[♫]Department of Artificial Intelligence, Yonsei University

[#]Department of Performing Art Technology, University of Michigan

solbon1212@yonsei.ac.kr

ABSTRACT

We present **DOSU**, a **D**ifficulty-conditioned **O**n-grid **S**equences **U**nmasking model, our submission to the MIREX 2026 Rhythm Game Chart Generation task for osu!taiko. Because the task provides ground-truth timing points, we formulate chart generation as sequence prediction over a fixed beat-synchronous grid rather than direct onset-time prediction. DOSU generates chart labels through masked iterative refinement with cross-attention difficulty conditioning, followed by calibration of chart-level properties using statistics from human-authored charts. Evaluation on held-out songs shows monotonic control of star rating across difficulty levels and few violations under automated playability checks. Playtesting further revealed discrepancies in local density, color patterns, and note spacing that were not captured by the automatic metrics.

1. INTRODUCTION

Rhythm game chart generation maps musical audio to a sequence of playable actions aligned with the music. Prior work commonly decomposes this task into *step placement*, which determines when notes occur, and *step selection*, which determines their types [1–5]. The MIREX 2026 Rhythm Game Chart Generation task provides ground-truth timing points, shifting the problem from estimating note timing toward generating coherent note sequences over a known rhythmic structure.

This setting raises two practical challenges. First, chart decisions depend on surrounding notes. Note density, spacing, and patterns should remain coherent over time rather than being determined independently at each position. Second, these characteristics should vary with the requested difficulty. We propose **DOSU**, a **D**ifficulty-conditioned **O**n-grid **S**equences **U**nmasking model (Figure 1), which formulates chart generation as difficulty-conditioned sequence completion on a beat-synchronous

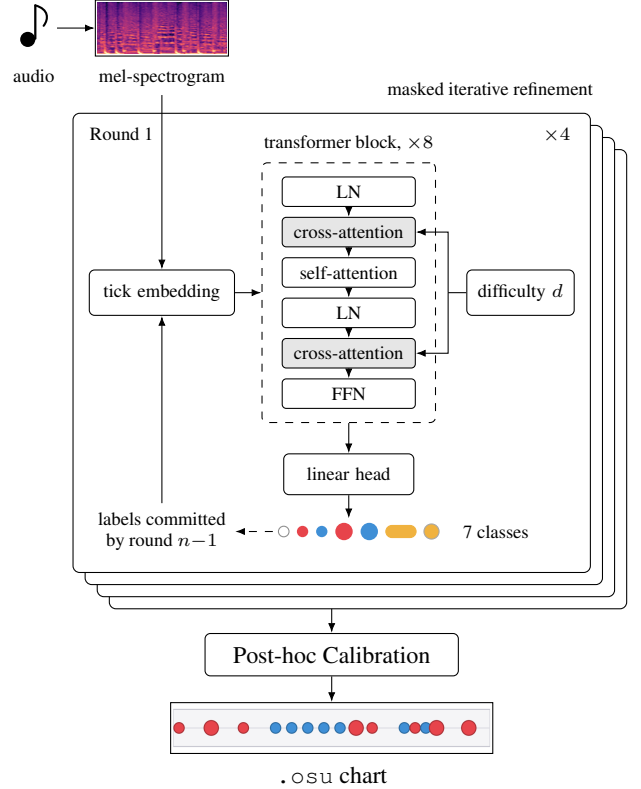


Figure 1. DOSU. The boxed network runs four times, each round conditioned on the labels the previous one committed.

grid. DOSU iteratively refines masked chart positions using the generated context, with cross-attention over a difficulty embedding applied at each block. The grid restricts generated objects to valid rhythmic positions, and post-hoc calibration adjusts chart-level statistics to match human-authored charts.

2. METHOD

2.1 Dataset

Following the official MIREX filtering and difficulty-tier criteria, we obtain 727 ranked beatmapsets using osu!dl. Ranked charts have undergone community review and are used as training data, with held-out charts serving as human references for evaluation. Among the 727 beatmapsets, 499 contain a standard-named osu!taiko dif-



Conditioning	d_{model}	Layers	Context	Params	Note F1	s/song	Star MAE
Additive	384	8	1024	14.8M	0.342	0.88	0.309
FiLM [13]	384	8	1024	19.5M	0.348	0.94	0.295
AdaLN-Zero [12]	384	8	1024	21.9M	0.335	1.01	0.304
Cross-attention [10], 8 tokens	384	8	1024	24.3M	0.344	1.31	0.273
FiLM [13]	512	12	2048	51.9M	0.346	2.11	0.299
FiLM [13]	768	16	2048	153.5M	0.343	3.93	0.295
Cross-attention [10] (ours)	384	8	1024	24.3M	0.349	1.30	0.304

Table 1. Conditioning and capacity ablation. *Context* is the window length in ticks. *s/song* times one song at all five tiers on one GPU.

ficulty, yielding 1,805 charts. Only 138 contain all five difficulty tiers, limiting paired supervision across difficulty levels.

2.2 Beat-synchronous representation

The uninherited timing points give a grid of 12 subdivisions per beat, which captures 99.38% of human hit objects within 12 ms. Each tick carries one of seven labels: *none*, *don*, *kat*, *large don*, *large kat*, *drumroll*, or *denden*. Drumroll and *denden* intervals are represented by assigning their labels to all spanned ticks, preserving an invertible representation. Audio is represented by an 80-band log-mel spectrogram and pooled within each tick together with onset statistics, yielding 162 features per tick.

2.3 Model architecture

Each tick embedding sums the projected audio features with learned subdivision, beat-position, previous-label and positional embeddings, and a depthwise-separable convolution mixes neighboring ticks before the stack. The network is eight pre-norm blocks of width 384 over a 1024-tick window. Difficulty conditioning is applied at each block through cross-attention over a learned difficulty embedding. The resulting vector serves as the key and value for cross-attention from the tick representations. The attention output projection is zero-initialized, so training starts from an unconditioned network. We train it to reconstruct randomly masked tick labels from the unmasked ones, and at inference fill all positions over four cosine-scheduled rounds [11].

2.4 Training

Models were trained for 30 epochs with AdamW [9] and one-cycle on a single GPU, keeping the best-validation checkpoint (epoch 10). The loss is a class-weighted cross-entropy taken only on the masked positions $\mathcal{M}$,

$$\mathcal{L} = -\frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} w_{y_t} \log p(y_t | z_t, d, y_{\mathcal{M}}) \quad (1)$$

Beatmapsets (official filter)	727
Sets with a standard-named taiko chart	499
Number of charts	1,805

Table 2. Dataset statistics. Splits are by *set*, 450 train and 49 validate.

with $w_c = \min(\sqrt{n_0/n_c}, 40)$, n_c the corpus count of label c and $c=0$ the empty label, carried by 89.8% of ticks. With probability 0.15, all positions are masked to expose the model during training to the fully masked state used in the first inference round.

2.5 Inference

All positions are initially masked and filled over four refinement rounds. To compensate for the calibration shift introduced by class-weighted training, label biases are adjusted according to chart-level statistics from the training corpus. Target note density is predicted from tempo and audio features rather than fixed for each tier, avoiding a direct dependence of note rate on BPM. Corpus-derived priors are also applied to rhythmic spacing and local patterns. Drumroll and *denden* durations are determined separately because the model predicts primarily one-tick runs, whereas these objects typically span multiple beats. Their positions and classes are obtained from the model, while occurrence rates and durations are determined from tier-specific corpus statistics, with durations capped at four beats.

2.6 Evaluation

Note F1 is the macro-average of the per-class F1 scores for the four hit classes on masked validation positions. The empty label is excluded because it accounts for 89.8% of ticks, while drumroll and *denden* are excluded because their durations are determined separately during inference. *Star MAE* is computed by applying the osu! star-rating calculator [7] to each generated chart and measuring the absolute difference from the ranked chart of the corresponding tier. We also record whether a song’s five ratings increase monotonically. *Playability* uses our reimplement of MapsetVerifier’s [8] taiko rules. The implementation is validated against ranked charts to identify false-positive violations. Charts are generated for 15 held-out in-corpus songs and, to probe generalization beyond the training period, for six songs ranked before the training window and eight ranked after its cutoff.

3. RESULTS

Star rating [7] increases monotonically with target difficulty for all 29 held-out songs, with MAEs of 0.304, 0.257, and 0.470 for the in-corpus, earlier, and later subsets, respectively. The higher error on later songs is partly associ-

ated with two 260-BPM songs. Table 1 shows no clear advantage among conditioning methods or larger model capacities, with Note F1 differences within the 0.020 variation across three seeds. We therefore use single-token cross-attention in the final system. DOSU produces no smallest-snap, finisher-spacing or concurrency violations, but raises six pattern-length warnings at Futsuu, where the ranked charts raise none. Against Mapperatorinator [6], DOSU has higher star-rating MAE (0.304 vs. 0.173) but no spacing violations, whereas Mapperatorinator violates the minimum Oni spacing on 3 of 15 songs.

Playtest analysis. Playtesting revealed three systematic differences between the generated and ranked charts that were not captured by any check we applied. Local note density was more variable, with 11.4% of 8-beat Oni windows exceeding $1.6\times$ the song-level mean density, compared with 2.7% for ranked charts. Generated color sequences were also less concentrated around common four-note patterns and exhibited a wider range of inter-note gaps. A color-pattern prior and gap-transition model were subsequently introduced to reduce these differences.

4. CONCLUSION

We presented DOSU, which generates osu!taiko charts through difficulty-conditioned masked refinement on a beat-synchronous grid. Across 29 held-out songs, star rating increases monotonically with the requested tier, and the grid makes unsnapped objects inexpressible rather than merely rare. Reference agreement reaches a macro-F1 of 0.35 over the four hit classes, with higher star-rating error for songs above 250 BPM. Playtesting further reveals discrepancies in note density, color patterns, and spacing that are not captured by any check we applied, indicating the importance of corpus-derived chart statistics beyond model capacity and playability constraints.

5. REFERENCES

- [1] C. Donahue, Z. C. Lipton, and J. McAuley, “Dance dance convolution,” in *Proc. ICML*, 2017.
- [2] E. Halina and M. Guzdial, “TaikoNation: Patterning-focused chart generation for rhythm action games,” in *Proc. FDG*, 2021.
- [3] J. Hanzen, E. Halina, and M. Guzdial, “Time-based chart partitioning: Improving local coherency in rhythm game chart generation,” in *Proc. AIIDE*, 2025.
- [4] J. Yi, S. Lee, and K. Lee, “Beat-aligned spectrogram-to-sequence generation of rhythm-game charts,” *ISMIR LBD*, 2023.
- [5] S. Lee and D. Jeong, “AutoOsu: Audio-aware action generation for rhythm games,” *ISMIR LBD*, 2023.
- [6] OliBomby, “Mapperatorinator,” 2025. [Online]. Available: <https://github.com/OliBomby/Mapperatorinator>
- [7] MaxOhn, “rosu-pp: osu! difficulty and performance calculation,” 2025. [Online]. Available: <https://github.com/MaxOhn/rosu-pp>
- [8] Naxess, “MapsetVerifier,” 2024. [Online]. Available: <https://github.com/Naxesss/MapsetVerifier>
- [9] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proc. ICLR*, 2019.
- [10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. NeurIPS*, 2017.
- [11] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, “MaskGIT: Masked generative image transformer,” in *Proc. CVPR*, 2022.
- [12] W. Peebles and S. Xie, “Scalable diffusion models with transformers,” in *Proc. ICCV*, 2023.
- [13] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, “FiLM: Visual reasoning with a general conditioning layer,” in *Proc. AAAI*, 2018.