Algorithm 1 Out-of-note beat prediction

Input: List of in-note beats $\mathcal{B}^i$
Output: List of all beats $\mathcal{B}$ after adding out-of-note beats

$n \leftarrow 1$
for $K = 0, 1, 2, 3$ **do**
    Initialise objective function $\mathcal{O}_K \leftarrow 0$
    Initialise beat sequence $\mathcal{B}_K \leftarrow \{b_1\}$
end for
for $n = 1, 2, \ldots, N^i - 2$ **do**
    for $K_{cur} = 0, 1, 2, 3$ **do**
        Get out-of-note beats for current step by Eq. (9)
        if Tempo is beyond tempo range limits **then**
            Go to next K_{cur}
        end if
        for $K_{prev} = 0, 1, 2, 3$ **do**
            Update objective function by Eq. (11)
        end for
        Select the minimum objective among all K_{prev} values
        Add out-of-beats for current step to the beat sequence mapped to the selected K_{prev}
    end for
    for $K_{cur} = 0, 1, 2, 3$ **do**
        Update $\mathcal{O}_K$, $\mathcal{B}_K$ mapped with K_{cur}
    end for
end for
Return the beat sequence in $\mathcal{B}_K$ with the minimum objective function $\mathcal{O}_K$

and $n \in \{1, 2, \ldots, N^i\}$. We insert out-of-note beats b^o from candidates in:

$$b^o_{n,K} = \{b^i_n + \frac{k}{K+1}(b^i_{n+1} - b^i_n)\}^K_{k=1} \tag{9}$$

where $K \in \{0, 1, 2, 3\}$ is the number of out-of-note beats to insert for the current interval. We try to find a way that minimises the tempo change after adding out-of-note beats to the beat sequence. To describe the level of tempo change for a list of beats, we define an objective function as follows:

$$\mathcal{O}_1 = \sum_{n=1}^{N-2} \left| log\left(\frac{b_{n+2} - b_{n+1}}{b_{n+1} - b_n}\right) \right| \tag{10}$$

where N is the number of beats in the final beat sequence and b_n is the n-th beat. However, this function does not take into account adding too many out-of-note beats. We thus add a penalty associated with the number of out-of-note beats to the objective function, resulting in:

$$\mathcal{O} = \mathcal{O}_1 + \lambda \times N^o \tag{11}$$

where λ is the penalty coefficient applied to avoid adding too many out-of-note beats which is tuned based on experiments and N^o is the number of out-of-note beats added. In this way, we obtain an objective function $\mathcal{O}$ to be minimised in the out-of-note beat prediction process that encourages both a low level of tempo change and adds fewer out-of-note beats.

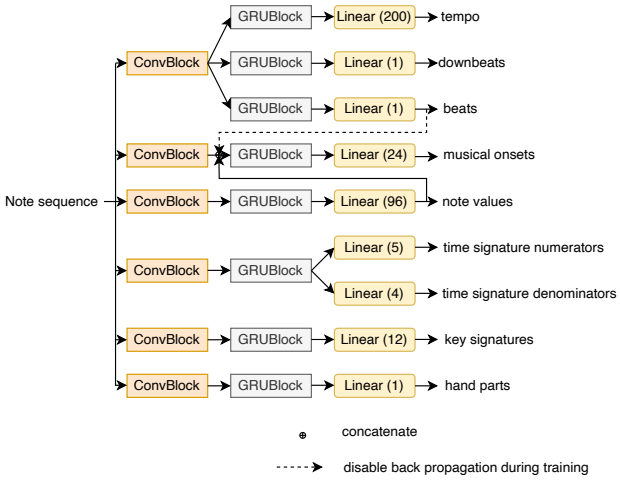

Figure 2. Model architecture for performance MIDI-to-score conversion.

We use a dynamic programming algorithm (defined in Algorithm 1) to find the optimal out-of-note beats while minimizing $\mathcal{O}$.

2.4 Performance MIDI-to-Score

Based on the problem definition in Section 2.1, we make some modifications to the prediction of time signature and key signature changes. In our proposed model, we define $\mathbf{Y_t}$ and $\mathbf{Y_k}$ in a way that the time signature and key signature values are mapped with each note by the note onsets. We also add beat b_n, downbeat db_n, and tempo tem_n prediction in note level. Beat probabilities are defined as in Eq. (7); similar definitions are used for downbeats. As a result, we define our model as $\mathbf{X} \rightarrow \mathbf{Y}'$ where:

$$\mathbf{Y}' = \{(mo_n, nv_n, h_n, tn_n, td_n, k_n, b_n, db_n, tem_n)\}^N_{n=1} \tag{12}$$

We then define a CRNN-based model that maps the input note sequence $\mathbf{X}$ with $\mathbf{Y}'$. Among the output elements, hand part h_n, beats b_n and downbeats db_n are defined as binary classification tasks; the others are defined as multi-class classification tasks, where

- Musical onset mo_n is defined as in Eq. (6), whose value is quantised by beat subdivisions;
- Note values nv_n are quantised by beat subdivisions;
- Time signature is defined to be $tn_n \in \{0, 2, 3, 4, 6\}$ and $td_n \in \{0, 2, 4, 8\}$, 0s indicate other values;
- Key signature k_n is defined to be in {C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B};
- Tempo is quantised following a minimum inter-beat-interval of 0.01s.

Our proposed model structure is shown in Figure 2. The Convolutional blocks (ConvBlock) and GRU blocks (GRUBlock) have similar structure to the ones in Figure 1. Links between branches allow the output elements to learn from each other. The model is trained jointly using binary cross-entropy loss for binary classification output elements and negative log-likelihood loss for multi-task classification output elements.

A MIDI score can then be generated by combining the information given in the performance MIDI and the output

Statistics	Train	Valid	Test	Total
Distinct pieces	426	49	29	504
Performances	1324	157	29	1510
Duration (hour)	108.3	12.7	2.2	123.2
Notes (10^3)	3984.0	517.6	73.2	4574.7

Table 1. Dataset Statistics. Performances from the MAPS dataset and the CPM database for the same piece are not counted as different performances, since MAPS pieces are originally obtained from the CPM database.

predictions from our proposed model.

3. EXPERIMENTS

3.1 Data

3.1.1 Datasets

To better demonstrate the ability of our proposed method on quantising expressive performance, we use a collection of classical piano music pieces from 1) The MAPS dataset [22] and corresponding metrical annotations from the A-MAPS dataset [27]; 2) The Classical Piano-Midi (CPM) database [28]; and 3) The ASAP dataset [23].

In order to avoid piece overlaps between train, validation and test splits, we use the distinct music piece labels for all piano performances given in the ACPAS dataset [29]. We use a real recording subset in MAPS (the ENSTDkCl subset) as the test set as in [5, 20]. Other music pieces not included in the test set are randomly split into training and validation based on their distinct music piece labels. In this way, we get a train/validation/test setup with no overlapping music pieces among splits. Table 1 shows the dataset statistics.

3.1.2 Annotation

Annotations are provided in different formats in the datasets we use. The A-MAPS dataset and the CPM database provide fully annotated MIDI scores with tempo and metrical information. Thus, we extract the annotations we need from the MIDI scores directly. Performances from the ASAP dataset come with two sets of annotations, MIDI scores and annotations in .tsv files. However, the MIDI scores were written by non-professionals and are not a good source of ground truth. Thus, we follow the authors' suggestion to use the provided annotations in the .tsv files, in which beat, downbeat, time signature and key signature annotations are provided. Due to the fact that the .tsv annotations do not cover precise fine-grained metrical and hand part annotations, we mask the ASAP data on musical onset time, note value, and hand part prediction during training. Moreover, when matching note onset times to beats or non-beats in our proposed model, we set a tolerance of $\pm$50ms to beat matching to comprise short-time variances introduced by human performance.

3.1.3 Data augmentation

Given the note sequence information defined in Section 2, we consider the following data augmentation methods:

- *Pitch shift*: Shift MIDI pitch values up or down for the whole music performance. The shifted pitch is defined as: $p_s = p_0 + p_{shift}$ where p_0 is the original pitch value and pitch shift $p_{shift} \in \{0, \pm1, \pm2, ..., \pm12\}$.
- *Tempo change*: Change the tempo to a ratio of the original tempo. The new tempo $tem_c = r_t * tem_0$ where tem_0 is the original tempo and $r_t \in [0.8, 1.2]$ is the ratio.
- *Note removal*: For polyphonic music, there should be little influence to the metrical structure of the music piece when removing some concurrent notes. Thus, for each group of M concurrent notes, we randomly remove 0 to M-1 of them from the MIDI performance.
- *Note introduction*: Contrary to note removal, we randomly select 0-100% of notes from the MIDI performance, and add new notes that are concurrent with the selected ones. We keep the velocity and duration the same, so as to preserve the original music structure as much as possible. The new note pitches are ±12 semitones apart from the original note pitches.

3.2 Evaluation metrics

3.2.1 Beat tracking evaluation

We define a *note-level F-measure* for evaluating in-note beat tracking, and a *beat-level F-measure* which follows the benchmark F-measure for beat and downbeat tracking [30] with a time tolerance of $\pm$70ms. In both cases, a true positive means a predicted beat is in the ground truth; a false positive means a predicted beat is not in the ground truth; and a false negative means a ground truth beat is missing in prediction.

For both F-measures, we report the precision p, recall r and F-score f. Similar definitions are used for downbeats.

3.2.2 Performance MIDI-to-Score evaluation

We use the MV2H metric [26] to evaluate the system performance on PM2S conversion. The metric covers five sub-metrics including multi-pitch detection (F_p), voice separation (F_{vo}), metrical alignment (F_{me}), note value detection (F_{va}), and harmonic analysis (F_{ha}). The final accuracy F is the average of all the sub-metrics.

3.3 Comparative experiments

Among the subtasks in PM2S, rhythm quantisation is a crucial and difficult part. It is also highly related to the estimation of time signatures, musical onsets, and note values. In our proposed rhythm quantisation method which combines neural beat tracking and musical onset time estimation, we consider beat tracking as a more important step since it works on drawing the skeleton of the metrical grid.

Thus, for the first part of our experiments, we investigate different input data configurations and data augmentation methods for the beat tracking part of our proposed method. To get rid of influences from out-of-note beat prediction, we use the note-level F-measure in our comparison. After that, we validate our beat tracking method on

beat-level F-measure combining out-of-note beat prediction in comparison with a baseline beat tracking model.

3.3.1 Input data encoding

Given the input data in note sequences, we consider the following ways of encoding the input elements:

- Pitches:
  - M: 128-dimensional one-hot vectors in MIDI pitch numbers;
  - C: 12-dimensional one-hot vectors in octave values.
- Onset times:
  - AR: the raw value in seconds;
  - AO: one-hot vectors quantised by 10ms resolution;
  - SR: onset time shift in seconds compared to the previous note onset ($o_i^{shift} = o_i - o_{i-1}$ for $i > 0$, $o_0^{shift} = 0$);
  - SO: one-hot onset time-shift quantised by 10ms resolution (with a maximum onset time shift of 4s, larger values are trimmed to 4s).
- Durations:
  - R: the raw values in seconds;
  - O: one-hot vectors quantised by 10ms resolution (similar to onset shift, large values are trimmed to 4s).
- Velocities are always normalised to 0-1.

For all possible input encoding combinations, we train and evaluate the in-note beat prediction part of our proposed model (excluding downbeat). The model learning rate is set to be 0.001. Since different input encodings can cause changes in the model convergence speed, we do not add learning rate decay in this comparison. The model is trained using a batch size of 32 over 4 GPUs with a dropout rate of 0.15. For each combination, we take the best model checkpoint on the validation set during training, and evaluate it over the test set. The performances of the model on all different input data encoding combinations are reported in Table 2.

From the results, we can see that in terms of pitch encoding, using the MIDI pitches tends to outperform using the chroma groups most of the time. For onsets, onset shift leads to better results than absolute onset across all encoding combinations. Using one-hot encoding for onset is better than using the raw values for most cases. 6 out of 8 encoding combinations result in better model performance with onsets encoded in one-hot format. However, an opposite preference is discovered for duration encoding, where duration in raw values ends up with better results for more cases.

Among all the 16 input data encoding combinations tested, the one with MIDI pitch, one-hot onset shift, and duration in raw value shows the best model performance. We use this input data encoding combination in our subsequent experiments.

Input encodings			Note-level F-measure		
Pitch	Onset	Duration	p	r	f
M	AR	R	86.7	77.7	79.9
M	AR	O	89.9	57.2	65.2
M	AO	R	82.1	78.3	79.3
M	AO	O	83.3	72.2	76.0
M	SR	R	89.2	89.4	87.8
M	SR	O	88.8	91.9	89.5
M	SO	R	91.2	94.3	**91.3**
M	SO	O	91.1	90.3	89.1
C	AR	R	86.7	68.7	73.6
C	AR	O	80.7	64.4	67.2
C	AO	R	82.8	76.2	78.4
C	AO	O	82.7	71.9	76.0
C	SR	R	90.4	88.0	87.3
C	SR	O	90.2	88.9	87.5
C	SO	R	89.9	91.3	89.1
C	SO	O	88.8	90.6	88.0

Table 2. Model performance on different input data encoding combinations.

Input feature omitted	p	r	f
Pitch	90.4	93.7	90.6
Onset	84.6	72.8	76.4
Duration	89.5	93.1	90.1
Velocity	89.5	93.9	90.6
Use all features	91.2	94.3	**91.3**
Augmentation method omitted	p	r	f
Pitch shift	92.0	92.0	90.6
Tempo change	91.7	89.5	89.7
Note removal	91.5	90.9	90.4
Extra note	91.2	94.3	91.3
Use all methods	92.9	93.7	**92.2**
No data augmentation	89.7	95.2	90.9

Table 3. Note-level F-measure results on the ablation studies.

3.3.2 Ablation studies

Using the best input encoding combination observed in the previous comparison, we run an ablation study on the input features and different data augmentation techniques. We use all four input features when exploring different data augmentation methods.

Table 3 shows the ablation study results. From the table, we see that all four input features are helpful in beat tracking, among which the onset feature is the most beneficial one. This is consistent with the fact that onsets can be the feature to carry most metrical information in music performances. People can usually infer beat times from drum beats without knowing other information such as pitch or duration. Pitch and velocity are less important but still make a difference. That may be because pitch carries some harmony information that helps beat prediction. Velocity can provide useful information as well since beats are more probable to be concurrent with heavy notes. Finally, duration is of certain importance, suggesting it carries more metrical information than pitch and velocity.

Models (output data)	f_b	f_{db}
Baseline (b, db)	66.9	57.6
Proposed (b, db)	85.7	63.3
Proposed (d, db, t)	**86.2**	**69.8**

Table 4. Beat-level F-measure results on the baseline and proposed models. b: beat, db: downbeat, t: tempo.

Methods	F_p	F_{vo}	F_{me}	F_{va}	F_{ha}	F
Finale	82.2	54.6	9.9	92.2	86.2	65.0
MuseScore	10.0	65.0	15.3	95.0	84.5	54.0
Proposed	**99.8**	**87.0**	**61.7**	**99.9**	**91.1**	**87.9**

Table 5. MV2H evaluation on performance MIDI-to-score conversion.

Results on different data augmentation methods suggest that all four proposed data augmentation methods improve performance, among which tempo change is the most beneficial one. Not performing data augmentation does not result in the lowest note-level F-score. However, its low precision rate indicates a limitation on predicting more false positives, which is discouraged since we will be adding out-of-note beats based on the predicted in-note beats. It is possible that we can add the false negatives back when predicting out-of-note beats, but we cannot remove the false positives.

3.3.3 Proposed model vs. baseline model

Using the best configurations in previous experiment results, we combine the out-of-note beat prediction step and evaluate our proposed model performance on beat tracking using beat-level F-measure. We compare our method with a baseline model that is similar to a state-of-the-art audio beat tracking model [31]. We retrain the baseline model using a pianoroll input replacing the original audio spectrogram input, where the pianoroll is calculated from the performance MIDI, and let the model predict beat and downbeat probabilities. The probabilities are then passed to a dynamic Bayesian network [32, 33] to get beats and downbeats in seconds.

Table 4 shows the comparative results between the baseline and our proposed model. Results suggest our proposed model largely outperforms the baseline when jointly trained with beats and downbeats. This is possibly because that the baseline model is a general purpose system designed to operate on richer content than piano music alone, and that our proposed model can better handle tempo changes. By adding tempo to the output data, the performance of our proposed model can be further improved, which suggests a benefit of joint learning.

3.4 System performance evaluation

Using the best configurations found in the comparative experiments, we train and evaluate our proposed CRNN model defined in Section 2.4 on PM2S conversion. We report the model performance on the MV2H metric described in Section 3.2.2.

When looking for other methods for comparison, we realise that there are some difficulties in comparing our system with existing academic works. Cogliati et al. [14] evaluated their method in a subjective way by inviting five music theory students to rate the transcribed musical scores. Works such as [5, 19, 34] do provide a combination of methods to achieve PM2S conversion, however, the system performance was reported on audio-to-score transcription. Thus, we compared our proposed method with two commercial software products Finale v27 [25] and MuseScore v3 [24] that can do PM2S conversion.

Results in Table 5 suggest that our proposed model outperforms MuseScore and Finale across all MV2H submetrics. A significantly better performance can be observed in the metrical alignment sub-metric F_{me} which is highly related to the rhythm quantisation step. By checking outputs generated from MuseScore, we found that its low performance on F_p is caused by time shifts introduced when quantising notes according to a constant tempo estimated over the whole music piece. Constant tempo estimation also caused its low performance reported on F_{me}. A similar limitation can be found in output scores from Finale. On the contrary, our proposed method tracked tempo changes during rhythm quantisation and preserved the expressiveness of music performance as much as possible. This not only benefits metrical alignment, but also results in high accuracies on F_p and F_{va}. Still, the rhythm quantisation performance (F_{me}) is far from satisfactory. Some typical errors include double/half tempo error and errors introduced by missing/extra beat predictions.

To provide a better understanding of the performance of our proposed method, we provide some example outputs from our model together with their performance MIDI recordings [1] .

4. CONCLUSION

We described our proposed method for rhythm quantisation by using a CRNN beat tracking model that predicts whether notes are at a beat position or not. We explored different input data encoding and data augmentation methods on the beat tracking model. We found that note onset time is the most important input feature in beat prediction and it is best to encode it into a one-hot onset-shift vector. Tempo change benefits most among the data augmentation methods explored. We validated our model's beat tracking ability in comparison with a pianoroll-input baseline model. In the end, we report the performance of our proposed system on PM2S conversion in comparison with MuseScore and Finale.

Possible next steps include investigating more powerful model architectures such as the Transformer [35], expanding the output data to generate a machine-readable score [21], probing our system's ability in dealing with more genre and instrumentation [19, 36, 37], and exploring our method's potential in automatic music transcription by combining multi-pitch detection [38, 39].

[1] Example outputs: `https://cheriell.github.io/research/PM2S`

5. ACKNOWLEDGEMENTS

L. Liu is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported jointly by the China Scholarship Council and Queen Mary University of London. The work of L. Liu is supported by The Alan Turing Institute through an Enrichment Scheme.

We would like to thank Huan Zhang, Changhong Wang and the reviewers for their valuable feedback to improve our work.

6. REFERENCES

[1] M. Good, "MusicXML: An internet-friendly format for sheet music," *XML Conference and Expo*, pp. 1–12, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.5431&rep=rep1&type=pdf

[2] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a System for Automated Music Engraving," in *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, 2003, pp. 167–171.

[3] S. D. Reeves and T. T. P. Walsh, *Creative jazz improvisation*. Taylor & Francis, 2022.

[4] R. G. C. Carvalho and P. Smaragdis, "Towards End-to-End Polyphonic Music Transcription: Transforming Music Audio Directly to A Score," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, vol. 2017-Octob, 2017, pp. 151–155.

[5] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon, "Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2018-April. IEEE, 2018, pp. 101–105.

[6] L. Liu and E. Benetos, "From Audio to Music Notation," in *Handbook of Artificial Intelligence for Music*, E. R. Miranda, Ed. Springer, 2021, pp. 693–714.

[7] L. Liu, V. Morfi, and E. Benetos, "Joint Multi-pitch Detection and Score Transcription for Polyphonic Piano Music," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[8] J. Huang, E. Benetos, and S. Ewert, "Improving Lyrics Alignment through Joint Pitch Detection," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers (IEEE), 2 2022, pp. 451–455. [Online]. Available: https://arxiv.org/abs/2202.01646v1

[9] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2019.

[10] P. Desain and H. Honing, "The Quantization of Musical Time: A Connectionist Approach," *Computer Music Journal*, vol. 13, no. 3, pp. 56–66, 1989.

[11] D. Temperley and D. Sleator, "Modeling meter and harmony: A preference-rule approach," *Computer Music Journal*, vol. 23, no. 1, pp. 10–27, 1999.

[12] C. Raphael, "Automated Rhythm Transcription," in *ISMIR*, 2001, pp. 99–107.

[13] H. Takeda, N. Saito, T. Otsuki, M. Nakai, H. Shimodaira, and S. Sagayama, "Hidden Markov model for automatic transcription of MIDI signals," *Proceedings of 2002 IEEE Workshop on Multimedia Signal Processing, MMSP 2002*, pp. 428–431, 2002.

[14] A. Cogliati, D. Temperley, and Z. Duan, "Transcribing Human Piano Performance into Music Notation," in *ISMIR*, 2016.

[15] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009.

[16] E. Nakamura, K. Yoshii, and S. Sagayama, "Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 4, pp. 794–806, 2017.

[17] E. Nakamura, K. Yoshii, and S. Dixon, "Note Value Recognition for Piano Transcription Using Markov Random Fields," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 9, pp. 1542–1554, 2017.

[18] A. McLeod and M. Steedman, "Meter detection and alignment of MIDI performance," *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 113–119, 2018.

[19] K. Shibata, E. Nakamura, and K. Yoshii, "Non-Local Musical Statistics as Guides for Audio-to-Score Piano Transcription," *arXiv preprint arXiv:2008.12710*, 2020. [Online]. Available: http://arxiv.org/abs/2008.12710

[20] Y. Hiramatsu, E. Nakamura, and K. Yoshii, "Joint Estimation of Note Values and Voices for Audio-to-Score Piano Transcription," in *ISMIR*. International Society for Music Information Retrieval Conference, 2021, pp. 278–284.

[21] M. Suzuki, "Score Transformer: Generating Musical Score from Note-level Representation," in *ACM International Conference Proceeding Series*, vol. 1, 2021.

[22] V. Emiya, R. Badeau, and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[23] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: A Dataset of Aligned Scores and Performances for Piano Transcription," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020.

[24] "MuseScore." [Online]. Available: https://musescore.org

[25] "Finale music notation software." [Online]. Available: https://www.finalemusic.com

[26] A. Mcleod and M. Steedman, "Evaluating Automatic Polyphonic Music Transcription," in *ISMIR, International Society for Music Information Retrieval Conference*, 2018, pp. 42–49. [Online]. Available: https://www.github.com/apmcleod/MV2H.

[27] A. Ycart and E. Benetos, "A-MAPS: Augmented MAPS Dataset with Rhythm and Key Annotations," in *ISMIR, International Society for Music Information Retrieval Conference, Late-Breaking Demo*, 2018.

[28] "Classical Piano-Midi Database." [Online]. Available: http://www.piano-midi.de

[29] L. Liu, V. Morfi, and E. Benetos, "ACPAS: A Dataset of Aligned Classical Piano Audio And scores for Audio-To-Score Transcription," in *ISMIR Late-Breaking Demo*, 2021, pp. 2–4.

[30] M. E. P. Davies, N. Degara, and M. D. Plumbley, "Evaluation Methods for Musical Audio Beat Tracking Algorithms," *Centre for Digital Music, Queen Mary University of London, Tech. Rep.*, vol. C4DM-TR-09, no. October, p. 17, 2009.

[31] S. Böck and M. E. P. Davies, "Deconstruct, Analyse, Reconstruct: How To Improve Tempo, Beat, and Downbeat Estimation," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020.

[32] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pp. 603–608, 2014.

[33] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pp. 72–78, 2015.

[34] A. Mcleod, "Evaluating Non-Aligned Musical Score Transcriptions with MV2H," in *ISMIR, International Society for Music Information Retrieval Conference, Late-Breaking Demo*, 2019.

[35] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding," *arXiv preprint arXiv: 2107.05223*, 7 2021. [Online]. Available: https://arxiv.org/abs/2107.05223v1

[36] K. Tanaka, T. Nakatsuka, R. Nishikimi, K. Yoshii, and S. Morishima, "Multi-instrument Music Transcription Based on Deep Spherical Clustering of Spectrograms and Pitchgrams," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020, pp. 327–334.

[37] Y. T. Wu, B. Chen, and L. Su, "Multi-Instrument Automatic Music Transcription with Self-Attention-Based Instance Segmentation," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 2796–2809, 2020.

[38] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution Piano Transcription with Pedals by Regressing Onsets and Offsets Times," *arXiv preprint arXiv:2010.01815*, pp. 1–10, 2020. [Online]. Available: http://arxiv.org/abs/2010.01815

[39] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation," in *ICASSP*, 2022, pp. 781–785.

SYMBOLIC MUSIC LOOP GENERATION WITH NEURAL DISCRETE REPRESENTATIONS

Sangjun Han[1], **Hyeongrae Ihm**[1], **Moontae Lee**[1,2], **Woohyung Lim**[1]
[1] LG AI Research, [2] University of Illinois at Chicago
{sj.han, hrim, moontae.lee, w.lim}@lgresearch.ai

ABSTRACT

Since most of music has repetitive structures from motifs to phrases, repeating musical ideas can be a basic operation for music composition. The basic block that we focus on is conceptualized as loops which are essential ingredients of music. Furthermore, meaningful note patterns can be formed in a finite space, so it is sufficient to represent them with combinations of discrete symbols as done in other domains. In this work, we propose symbolic music loop generation via learning discrete representations. We first extract loops from MIDI datasets using a loop detector and then learn an autoregressive model trained by discrete latent codes of the extracted loops. We show that our model outperforms well-known music generative models in terms of both fidelity and diversity, evaluating on random space. Our code and supplementary materials are available at https://github.com/sjhan91/Loop_VQVAE_Official.

1. INTRODUCTION

With the advance of generative models, many studies are trying to model sequential data such as language and speech. Music can also benefit from their previous works since it consists of a sequence of multiple notes to represent the composer's intention. Through the advancement, individuals can imitate the musical inspiration of artists without musical expertise.

Several works related to music generation have focused on generating long sequences by utilizing the expressive power of Transformer [1-4]. It is a promising approach because the Transformer with hierarchical layers can learn various types of repetitive structures on its self-attentions. However, it still has limitations, derived from the error accumulation and rhythmic irregularity, to achieve the ultimate goal which is to compose full-length music [1, 2]. To tackle that problem, we explicitly utilize recurrence properties in music, generating short and fixed-length music phrases that can be used as basic patterns of music.

Repeating musical ideas is a basic operation for music composition. This operation conceptualizes the loop, an essential ingredient for creating remixes or mash-ups [5]. With the concept, we can simplify the generation task into generating one distinctive pattern which consists only of a few bars. If assuming 8 bars as a minimal and meaningful phrase for the loop, we can describe several patterns as follows; (A - B - A - B - A - B - C - D or A - B - C - D - A - B - C - D or A - B - A - B - A - B - A - B and so on). For loop extraction, previous works have attempted to detect autocorrelated peaks (it relies on an assumption that it is sufficient to detect the starting point "A" of phrases) [5, 6], but we apply the knowledge of overall loop structures, obtained from the audio domain, to the MIDI domain.

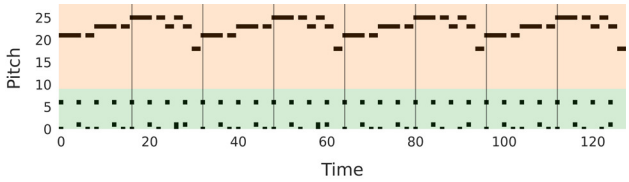

Figure 1. An example of 8 bars loop pianoroll representation of the bass (orange area) and drum (green area).

Another intuition is that the musical ideas can be formed in combinations of finite symbols. It is known that learning discrete latent codes is sufficient to represent the continuous world since many modalities consist of sequences of symbols [7]. For example, objects in vision, words in language, and phonemes in speech may be candidates of symbols. Also, compressing raw data into discrete semantic units makes an autoregressive model easy to train by capturing long-range data dependency [8]. If we regard consecutive notes as symbols, it is natural to adopt discrete representation as the basis of our autoregressive generator. This process can emphasize pattern to pattern structures sacrificing the minimal loss of note details.

In contrast to using Inception Score (IS) and Fréchet Inception Distance (FID) for computer vision [9, 10], no consensus has been made for evaluating generated music. This is because the absence of pre-trained feature extractors prohibits the comparison of true and generated samples on feature space. Recently, Naeem has shown that random embedding can also be effective when the target distribution is far from pre-trained model statistics [11]. Using the random initialized networks, we evaluate our generative models on sample fidelity and diversity as firstly suggested in [12].

In this work, we propose symbolic music loop generation via learning discrete representations. It involves two main processes; 1) *loop extraction* from MIDI datasets using a loop detector and 2) *loop generation* from an autoregressive model trained by discrete latent codes of the extracted loops. The outputs of the generative model are loops consisting of 8 bars, which can be repeated seamlessly (Figure 1). Since we aim to generate polyphony and multitrack sequences, the bass and drum are chosen for our experiments, which are fundamental components of mel-

ody and drum. Additionally, we adopt an evaluation protocol from [11] to measure two-dimensional score which stands for fidelity and diversity. Our contributions are summarized as follows;

- We propose the framework of symbolic music loop generation, which involves loop extraction, loop generation, and its evaluations.
- For loop extraction, we design a structure-aware loop detector trained by external audio sources to extract loops of 8 bars from MIDI.
- For loop generation, we verify that an autoregressive model combined with discrete representations can generate plausible loop phrases which can be repeated.
- With randomly initialized networks for embedding, we evaluate sample quality in terms of fidelity and diversity.

2. RELATED WORK

We introduce several works related to the history of loop extraction and music generation, the effectiveness of discrete representations, and the development of evaluating generative models.

2.1 Loop Extraction

For symbolic music generation, some researchers have prepared their dataset by sliding a window with a stride of 1 bar [13, 14]. Although they have achieved good generative performance, this process does not consider relative positions within music, generating ambiguous phrases.

Some works have imposed structural constraints on music generation models [15, 16], or directly detect novel segments which are repetitive in time series [17, 18]. In the audio domain, there have been attempts to extract loops explicitly by capturing repeated phrases [5, 6]. They extract harmonic features such as chroma vectors or mel-frequency cepstrum and catch autocorrelation peaks to determine the starting point of loops. They also require a heuristic process to decide which features should be more weighted. In contrast, our loop detector works on a data-driven approach, so it does not require a manual process such as feature extraction and weighting strategies.

A recent work for drum loop generation has informed the availability of a human-created loop dataset from Looperman (https://www.looperman.com/) [19]. Although it consists of audio sources with various instruments and genres, we suggest a promising approach to combine them with Lakh MIDI Dataset (LMD) [20]. Concretely, we extract domain-invariant loop structures from Looperman and train a loop detector using them to extract loops from LMD.

2.2 Symbolic Music Generation

To make MIDI available in machine learning, two representation methods are prevalent; event-based representation and time-grid based representation [21]. Although the former can represent high time-resolution with a few event vectors, we choose the latter one to benefit from fixed-length and repetitive structures for music. There have been several works to deal with polyphony multitrack representation [1, 13, 22]. Especially for time-grid based representation, MuseGAN [22] has stacked five instruments each of which consists of 4 bars and 84 pitches. We follow their method while restricting to two instruments, the bass and drum.

2.3 Discrete Representations

As opposed to VAE [23] with continuous prior, Oord has proved that a finite set of latent codes is sufficient to reconstruct while it prevents posterior collapse [7]. Powerful autoregressive priors with the latent codes have shown promising performance not only in image generation [8] but also text2image [24], and video generation [25]. Loop generation can also benefit from discrete data compression by expressing long-range structural patterns.

2.4 Evaluation of Generative Models

Proper evaluation metrics for generative models are important to reduce human evaluation labor. Previous works of music generation have inspected model metrics (*e.g.,* log-likelihood) or musical metrics on data space to evaluate how much true and generated samples are similar [3, 22, 26]. Comparison on data space, however, is vulnerable to pixel by pixel phase difference, ignoring data semantics. Without available pre-trained networks, it has been reported that random embeddings are more robust for evaluation rather than using models trained by other data domains [11]. In this respect, we take randomly initialized networks as our feature extractor and evaluate our music samples on feature space.

The commonly used metrics in computer vision are IS and FID which compute a one-dimensional score. To distinguish fidelity and diversity from the score, Sajjadi has suggested precision and recall evaluating overlapped ratio between true and generated distribution [12]. After that, some works have proposed different ways of constructing data distribution using k-nearest neighbors (KNN) [11, 27]. We adopt KNN based evaluation protocol since it is not affected by its initialization and is robust to outliers.

3. PROPOSED METHOD

Our work starts with collecting MIDI loops using a structure-aware loop detector. Since we design the detector to take not the music itself but bar-to-bar structures, we can train it with a loop-labeled dataset from audio domain. After obtaining the MIDI loop set, we design a loop generator in two-stage; compressing data into discrete space and building an autoregressive model with them. Lastly, generated samples are evaluated on qualitative and quantitative metrics.

3.1 Data Preparation

3.1.1 Looperman Dataset

Looperman Dataset from the audio domain is collected and transformed for training our loop detector. We collect 1,000 loops of 8 bars from Looperman, a website allowing to upload and download free music loops. Formally, we