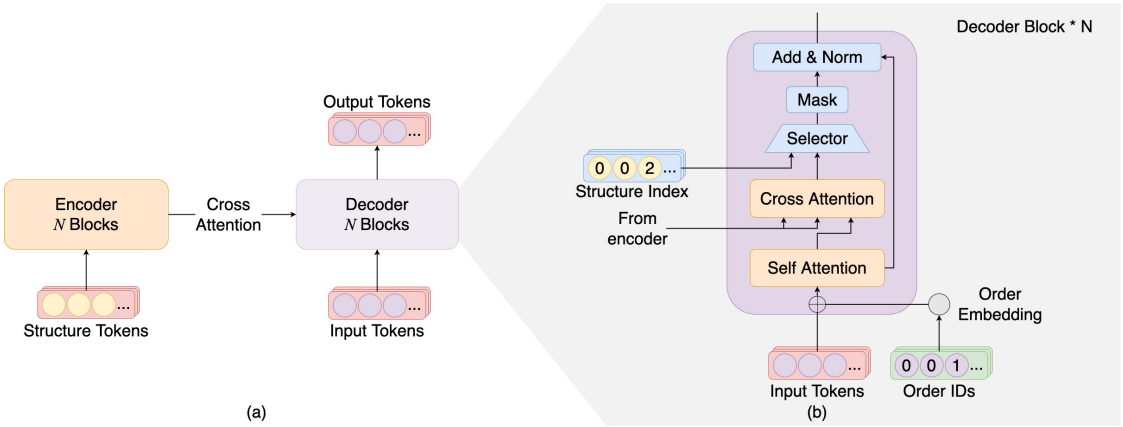

Figure 2: Schematic diagram of (a) the proposed seq2seq Transformer model for structure-aware infilling, and (b) a zoom-in of the decoder highlighting how the decoder utilizes the *order embedding* and *structure index*.

Token type	Voc. size	Values
`Bar`	32	1, 2, ... , 32
`Struct`	16	0, 1, ... , 15
`Tempo`	47	28, 32, ... , 212
`Position`	16	0, 1, ... , 15
`Pitch`	86	22, 23, ... , 107
`Duration`	16	1, 2, ... , 16

Table 1: The vocabulary of the token representation.

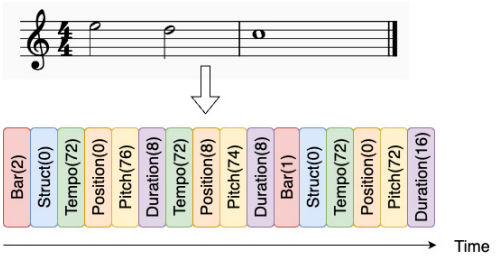

Figure 3: An example of representing a two-bar segment with our adapted REMI-based token representation. With the bar-count-down technique, `Bar(2)` and `Bar(1)` are used for the first and second `Bar` tokens, respectively.

of tokens: `Bar`, `Struct`, `Tempo`, `Position`, `Pitch` and `Duration`. Table 1 lists the vocabulary of our token representation. `Bar` consists of numbers from 1 to 32, standing for the number of remaining bars on the generation process. A music form, e.g., `ABA′B′`, consists of multiple groups of similar phrases or sections (each of multiple bars), e.g., {`A`, `A′`} and {`B`, `B′`}, where each group can be said to be associated with the same *structure label*. We use `Struct` to indicate the structure label for each bar. `Tempo` and `Position` are related to the musical metre. `Tempo` is the current tempo of beats per minute (BPM), and `Position` is the temporal distance between the onset of a musical note and the beginning of its bar, denoted by the number of 16-th notes. `Pitch` and `Duration` are related to musical notes, which are the MIDI pitch number and duration in 16-th notes, respectively. We show an example of how we encode the musical content in Figure 3.

3.2 Bar-Count-Down Technique

The work of Hsu & Chang [21] uses special `BOS` and `EOS` tokens as the "signal" to start or stop the generation process. At inference time, the infilled segment generated by their model comes to an end when the model generates an `EOS` token. While this may work fine in certain cases, doing so cannot give us an explicit control of the number of bars to be generated for the infilled segment. Such a control is preferable when we want to make sure that the previous context and the future context are a certain number of bars apart, which is highly needed for structure-aware infilling. For example, a user may want to specify the music form as `A8B8A′8B′8`, meaning that all the four sections `A`, `B`, `A′`, `B′` are eight-bar long each.

To have such a control, we employ a special token representation technique called "bar-count-down." For each infilled sequence T, we adjust the suffix number of `Bar` tokens to match the length of T. Take Figure 3 as an example: The number in `Bar` tokens are counted down from two because the sequence in Figure 3 is 2-bars long. Once training a model with this setup, the length of T can be controlled effectively by the number of remaining bars associated with the first `Bar` token given on generation.

3.3 Order Embedding

Transformers with causal masking are mainly designed for sequential generation. To apply the model to infilling tasks where the missing part is in the middle of the input sequence, the model proposed by Hsu & Chang [21] attends to bi-directional context from two encoders with cross-attention. As we want to instead use only the self-attention of the decoder to exploit bi-directional information, we reorder the sequence $\{C_{\text{past}}, T, C_{\text{future}}\}$ to $\{C_{\text{past}}, C_{\text{future}}, T\}$. Doing so would however change the original positional relationship among C_{past}, T, and C_{future}.[3] As depicted

[3] The principal idea of the attention mechanism [26] is to use the token embeddings of two tokens to compute their correlation, leading to the so-

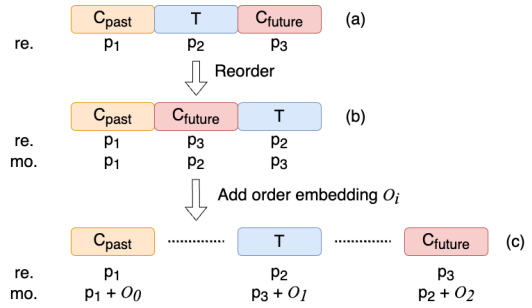

Figure 4: Illustration of the order embedding, where $re.$ and $mo.$ stand for the *real position* and *model-viewed position*. Conceptually, we shift the model-viewed position by O_i to make it consistent with the real position.

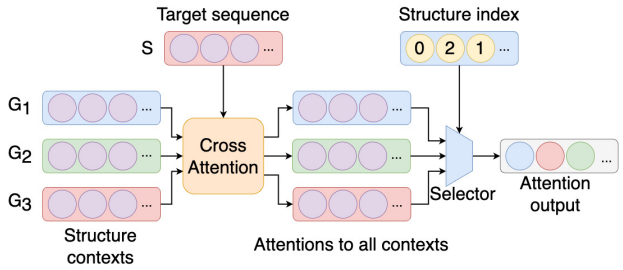

Figure 5: The schematic of attention selecting. Multiple structural contexts G_1, G_2, G_3 are passed to the cross-attention block simultaneously, and the output is filtered by the *selector* with the *structure index*.

in Figure 4(b), the *real* positional relationships among the segments entails associating the tokens in T with positional embeddings corresponding to a set of positions p_2 that signifies the model the segment T is after C_{past} (i.e., $p_1 \prec p_2$) and is before C_{future} ($p_2 \prec p_3$).[4] After reordering, however, using the typical way of computing the positional embeddings from left to right by the Transformers, T would be assigned with positional embeddings corresponding to p_3, meaning that by the *model-viewed* positional relationships, T is after both C_{past} and C_{future}. The real and the model-viewed are mismatched.

We introduce a new position-related *segment embedding* called "order embedding" to tackle this issue. Specifically, for the positional embeddings for all the tokens in C_{past}, we add to them the same additional embedding corresponding to a positional "offset" or "order," denoted as O_0. We similarly incorporate O_1 and O_2 to the positional embeddings for the tokens in T and C_{future}. As long as the "offset" is big enough, we can have $p_1 + O_0 \prec p_3 + O_1 \prec p_2 + O_2$, ensuring that the real and model-viewed positional relationships are matched, as depicted in Figure 4(c).

We note that, as the same order embedding is added to the positional embeddings of all the tokens in a segment, the proposed idea works nicely regardless of whether the segment lengths $|C_{\text{past}}|$, $|T|$, $|C_{\text{future}}|$ are fixed or not.

3.4 Attention-Selecting Module

While the formulation of Eq. (2) considers only one structural context G, in practice, we may want to designate multiple structural contexts $\{G_1, G_2, \ldots, G_N\}$ for infilling, with each segment G_n corresponding to a certain structure label such as `A` and `B`. This is useful, for example, when the first part of the infilled segment T is meant to be similar to phrase `A`, while the latter part similar to phrase `B`. To indicate which G_n a specific token t_k of T should refer to, we define the *structure index* $y_k \in \{0, \ldots, N\}$, for $k = \{1, \ldots, |T|\}$. The structure indices are also given by the user when the user specifies the intended musical form. With all these, we extend Eq. (2) to:

$$\prod_{0<k\leq|T|} P(t_k|t_{<k}, C_{\text{past}}, C_{\text{future}}; G_1, ..., G_N, y_k)\,. \quad (3)$$

We use $y_k = 0$ to indicate the case where t_k is supposed to follow none of the structural contexts (e.g., when t_k is part of the bridge). When $y_k \neq 0$, t_k follows only one of the structural context G_{y_k}. In our implementation, for tokens whose $y_k = 0$, we only use the self-attention to attend to the prompt and $t_{<k}$, leading to a formulation akin to Eq. (1). When $y_k \neq 0$, we have a "selector" that picks by y_k the structural context G_{y_k} to be attended to via cross-attention, resulting in a formulation akin to Eq. (2). This is depicted in Figure 2(b) and Figure 5.

Figure 5 also shows that, in our implementation, instead of computing the cross attention between the target sequence with a specific structural context G_{y_k}, we actually compute the cross attention between the target sequence with every structural contexts $\{G_1, G_2, \ldots, G_N\}$ and let the selector pick the right one. While this seems a waste of computing, we find doing so faster when GPUs are used.[5]

4. EXPERIMENTAL SETUP

We collect the melody data[6] from the POP909 dataset compiled by Wang *et al.* [30]. POP909 contains 909 MIDI files of pop piano performances, though we discard 8 of them due to errors encountered in the preprocessing stage. We split all remaining songs into 811 (90%) songs for training and 91 (10%) songs for testing. The 16-th note is set as the minimal temporal resolution to quantize the tempo, beat, and duration for reducing the vocabulary size.

Dai *et al.* [31] publicly share the *structural information* of songs in POP909 with letters and integers to indicate the structure labels and their lengths in bars, such as

`i4 A8 B8 x4 A8 B8 B8 X2 c4 c4 X2 B9 o2`, (4)

called attention score. However, using the token embeddings alone fails to consider the position-related relations of the two tokens (e.g., whether they are neighbors or distant apart). Accordingly, in practice, people add token-wise positional embeddings to the token embeddings before computing their attention [35–37].

[4] We use $p \prec q$ to denote that any elements in the set p is smaller, or "temporally before," any elements in the set q.

[5] There is another implementation detail: actually, not only the tokens in T but also those in C_{past} and C_{future} would go through the Transformer's self- and cross-attention blocks. This is to get the latent vectors for the tokens in C_{past} and C_{future}. In doing so, we calculate and employ the structure indices for C_{past} and C_{future} as well.

[6] Please note that the "melody" data in this paper is not exact monophonic music. We merge two midi tracks, "MELODY" and "BRIDGE" of the MIDI files of POP909 [30] to generate the melody data, which is monophonic most of the time but not always.

where the music phrases labeled with the same letter are considered to share the same structural context.[7] [8] Besides, the musical phrases with the labels `i`, `x`, and `o` are respectively the introduction, bridge, and ending, which do not have structural context in our setting. For each song, we use the phrases corresponding to the first occurrence of the structure labels such as `A` and `B` as the structural contexts G_1, G_2, etc. For example, we choose bars 5–12 (i.e., the `A8` phrase after `i4`) and bars 13–20 (i.e., the `B8` phrase before `x4`) in the example shown in Eq. (4) as the structural contexts for `A` and `B` since they are the first music phrases in the song with the corresponding labels.

The training data is generated with the following steps: (i) iterating through all labels except for the first and last ones in the structural information, (ii) choosing their corresponding music phrases as the infilled sequences T, and (iii) concatenating T with their preceding and following 6-bars music segments to get $\{C_{\text{past}}, T, C_{\text{future}}\}$, yielding 8,607 data in total. Before feeding the training data into the model, we reorder the input sequences and insert additional special tokens, `BOS`, `SEP`, and `EOS`, to change them into $\{\texttt{BOS}, C_{\text{past}}, \texttt{SEP}, C_{\text{future}}, \texttt{SEP}, T, \texttt{EOS}\}$. The reordered input sequence and structural contexts are transformed to the embeddings with size 512. The model consists of an encoder and a decoder, where both of them consist of six 8-head self-attention layers with intermediate layers of dimension 2,048. Each layer of the encoder and decoder is connected with an 8-head cross-attention, as shown in Figure 2. The output from the model is transformed back to a probability distribution of the vocabulary with the softmax function. At inference time, we use nucleus sampling [38] to sample the output tokens with the threshold value of 0.9.

For evaluation, we create the testing data by: (i) searching from the testing songs all the 4-bar phrases that correspond to only a structure label, (ii) keeping only the phrases that share the same structure label with one of its two neighboring phrases but a different structure label with the other neighbor, and (iii) setting the 4-bar phrase as the target T and concatenate them with their preceding and succeeding 6-bar music segments, which are set as the past context C_{past} and future context C_{future}, respectively. We get 156 test cases of $\{C_{\text{past}}, T, C_{\text{future}}\}$, each with 16 bars (i.e., $6+4+6$). All the cases have the form of, e.g., `AA'B` or `ABB'`, and the target lengths are **4 bars** with arbitrary number of notes (hence *variable* sequence lengths). In this setting, the attention selection mechanism is used only for training, since the target sequence T in our testing data only have one structural context to refer to.

We consider VLI [20] and the model from Hsu & Chang [21] as the baselines. By design, only our model has access to an external structural context. However, we consider the comparison as valid, since C_{past} and C_{future} are presumably long enough to provide sufficient context, and at least one of them has the same structure label as T. Besides, in our implementation, we found the model Hsu & Chang [21] rarely generates infilled segments with the desirable number of bars. Therefore, we slightly improve their model by incorporating the bar-count-down technique.

[7] The lower-case and capital letters indicate respectively *non-melodic* and *melodic* phrases (i.e., where a clear melody is present, mostly a vocal line or an instrument solo), but we do not use this information.

[8] Depending on the underlying musical form of a song, different songs may have different numbers of phrase groups. For example, a song with a simpler form may only have phrase groups `A` and `B`, while other songs have much more. In the POP909 dataset, a song can contain up to 9 unique phrase groups, which are labeled as `A`, `B`, `C`, `D`, etc.

	$H \downarrow$	$GS \uparrow$	$D \downarrow$
Ours	**2.75**$\pm$0.80	**0.70**$\pm$0.08	**25.73**$\pm$19.45
VLI [20]	3.47$\pm$1.57	0.67$\pm$0.09	49.40$\pm$25.12
Hsu [21]	9.87$\pm$4.64	0.64$\pm$0.09	65.41$\pm$38.00
Original	2.78$\pm$0.89	0.70$\pm$0.09	0.00$\pm$ 0.00

Table 2: Objective evaluation results. H: pitch class histogram cross entropy, GS: grooving pattern similarity, D: melody distance (↑/↓: the higher/lower the better).

5. OBJECTIVE EVALUATION RESULTS

We propose three new metrics for objective evaluation, all of which have not been used in the literature of music infilling. The first two metrics, **pitch class histogram cross entropy** (H) and **grooving pattern similarity** (GS), are extensions of the ones proposed by Wu & Yang for sequential generation [10] to our infilling task, evaluating respectively the consistency in terms of pitch class distribution (which is related to tonality) and rhythmic pattern. For H, we compute per test case the pitch class histogram of T, and that histogram of the concatenation of C_{past} and C_{future}, and then report the cross entropy between these two histograms. For GS, we use per bar a 16-dim binary vector indicating where there is at least a note onset for every position in a bar, calculate one minus the normalized XOR difference between every pair of bars [10], one from T and the other from either C_{past} and C_{future}, and then report the average per test case. The third metric, **melody distance** (D) measures the melody distance (dissimilarity) between the infilled segment T and the ground truth one (denoted as $T\#$) using the algorithm proposed by Hu *et al.* [39]. Lower H and higher GS may imply that T connects C_{past} and C_{future} smoothly, while lower D indicates that the generation result is similar to a human-made one.

Table 2 shows that our model achieves the best result in all the three metrics, followed by VLI and then the model of Hsu & Chang. Besides being consistent with the contexts, the infilling result of our model is closest to the ground truth one $T\#$, demonstrating the effectiveness of exploiting the structural context. Figure 6 exemplifies how the infilled bars by our model fit the desired musical form.

6. SUBJECTIVE EVALUATION RESULTS

We conduct additionally an online user study for subjective evaluation. We have 91 anonymous volunteers, where 12 of them are marked as professionals according to the question about their musical background. Each subject is pre-

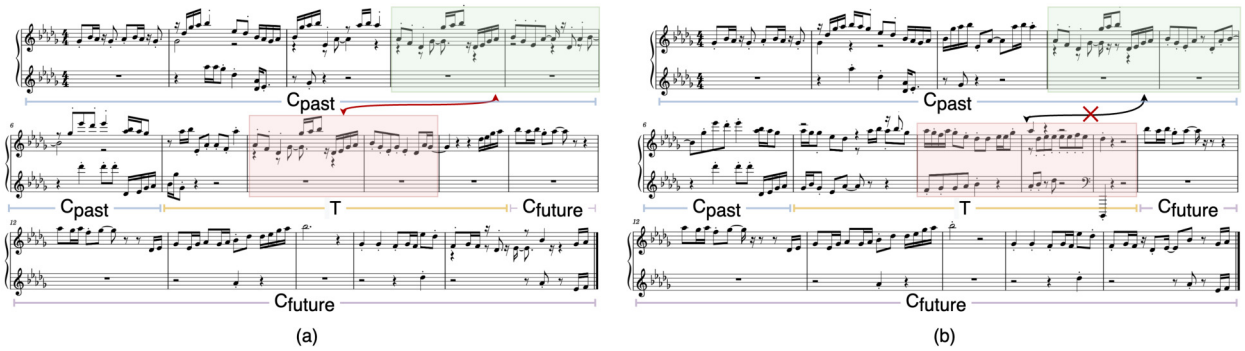

Figure 6: Results generated by (a) our model and (b) VLI. The structural relations of bars 4-5 (green area) and bars 8-9 (red area) are preserved well in (a) but not in (b).

		M	R	S	O
all	Ours	**3.46**	**3.51**	**3.40**	**3.42**
	VLI [20]	2.96	3.14	3.12	2.97
	Hsu [21]	2.60	2.95	2.75	2.64
	Real	**3.77**	**3.77**	**3.62**	**3.66**
pro	Ours	**3.58**	**3.28**	**3.28**	**3.42**
	VLI [20]	2.67	2.86	2.78	2.72
	Hsu [21]	2.36	2.75	2.39	2.44
	Real	**3.61**	**3.56**	**3.42**	**3.42**

Table 3: Results of the user study: mean opinion scores in 1–5 in **M** (melodic fluency), **R** (rhythmic fluency), **S** (Structureness), and **O** (Overall), from 'all' the participants or only the music 'pro'-fessionals.

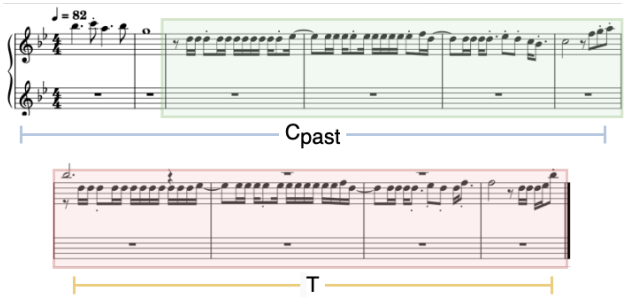

Figure 7: A result generated by our model, where C_{future} is not shown in this figure. Bars 3–6 (green area) and bars 7–10 (red area) look identical since the model overly imitates the given structural context. (Best viewed in color.)

sented with 3 out of 15 sets of music segments randomly sampled from the testing data. We inform them that the first and last 6 bars are the given prompts, and the middle 4 bars are the music generated by a model. Each set of music contains in random order 4 music including 3 generated by the models and 1 from the real data. The subjects rate each of the 4 music in a 5-point Likert scale (the higher the better) according to their (i) **melodic fluency**: do the pitches of notes go in the right tonality and connect the contexts fluently? (ii) **rhythmic fluency**: are the notes played on the right beats? (iii) **structureness**: how is the generated part of the music similar to its contexts? (ix) **overall**: how much do they like the music?

Table 3 shows the mean opinion scores (MOS) of the user study. Echoing the result of the objective evaluation, the proposed model outscores the baselines by a large margin in all the four subjective metrics, and is close to the real music with a small gap. The same observations can be made from either the average result of all the subjects, or only that from the 12 professionals.

However, we notice that our model may overly imitate the given structural contexts in some cases, as exemplified in Figure 7. When this happens, the generated music sounds rigid and non-creative. We conjecture that this can be attributed to the limited diversity of our training data—the melodies corresponding to the same structure label in POP909 appear to be too similar to each other, which may not be uncommon for pop songs. To study this, we implement additionally a 'Copy' baseline that simply copies the structural contexts as the result, and include its infilling result to the demo website. Our own subjective listening of its result confirms that the proposed method still outperforms the 'Copy' baseline most of the time, as the connections between the targets and their contexts are considered by our model, but not by the 'Copy' baseline.

7. CONCLUSION

In this paper, we have proposed a new structure-aware conditioning approach for music score infilling. To help Transformers exploit bi-directional contexts, we employ the order embedding to shift the position viewed by the model. Besides, we introduce a new attention-selecting mechanism to account for multiple structural contexts. Evaluations on 4-bar melody infilling validate the superiority of the proposed model over two existing Transformer-based structure-agnostic infilling methods [20, 21].

We can extend this work in three ways. First, instead of relying on user inputs, we may build models that generate the musical form automatically and predict the structural context for a specific infilled segment to refer to. Second, we like to expand our work to other music genres and polyphonic music. Finally, we like to study how the attention-selecting mechanism can be applied to sequential generative tasks such as theme-based generation [29].

8. ACKNOWLEDGEMENT

We are grateful to Shih-Lun Wu for sharing the unpublished idea of the bar-count-down technique, Ping-Yi Chen and Chin-Jui Chang for the assistance with experiments and discussion, Shan Lee for the help of figure drawing, and Chun-Wei Lai and Sophia Lin for the help in finding volunteers for the user study. We also thank the anonymous reviewers for their valuable feedbacks. Our research is funded by grants NSTC 109-2628-E-001-002-MY2 and NSTC 110-2221-E-006-137-MY3 from the National Science and Technology Council of Taiwan.

9. REFERENCES

[1] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. International Conference on Machine Learning*, 2012, pp. 1881–1888.

[2] F. Colombo, "Algorithmic composition of melodies with deep recurrent neural networks," *Poster Session of International Conference on Artificial Intelligence and Statistics*, 2016.

[3] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. Conference on Computer Simulation of Musical Creativity*, 2016.

[4] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. International Conference on Machine Learning*, 2018, pp. 4364–4373.

[5] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. International Conference on Music Information Retrieval*, 2017.

[6] 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," in *Proc. International Conference on Learning Representations*, 2019.

[7] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proc. International Conference on Music Information Retrieval*, 2019.

[8] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proc. ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[9] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proc. ACM Multimedia*, 2020.

[10] S.-L. Wu and Y.-H. Yang, "The Jazz Transformerm on the front line: Exploring the shortcomings of ai-composed music through quantitative measures," in *Proc. International Conference on Music Information Retrieval*, 2020.

[11] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. International Conference on Music Information Retrieval*, 2021.

[12] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with one Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[13] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "Symphony generation with permutation invariant language model," *arXiv preprint arXiv:2205.05448*, 2022.

[14] H.-W. Dong, K. Chen, S. Dubnov, J. McAuley, and T. Berg-Kirkpatrick, "Multitrack Music Transformer: Learning long-term dependencies in music with diverse instruments," *arXiv preprint arXiv:2207.06983*, 2022.

[15] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proc. International Conference on Music Information Retrieval*, 2019.

[16] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. International Society for Music Information Retrieval*, 2017.

[17] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proc. International Conference on Machine Learning*, 2017, pp. 1362–1371.

[18] D. Ippolito, A. Huang, C. Hawthorne, and D. Eck, "Infilling piano performances," in *Proc. NIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[19] T. Bazin and G. Hadjeres, "Nonoto: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. International Conference on Computational Creativity*, 2019.

[20] C.-J. Chang, C.-Y. Lee, and Y.-H. Yang, "Variable-length music score infilling via XLNet and musically specialized positional encoding," in *Proc. International Conference on Music Information Retrieval*, 2021.

[21] J.-L. Hsu and S.-J. Chang, "Generating music transition by using a Transformer-based model," *Electronics*, vol. 10, no. 18, 2021.

[22] C.-P. Tan, C.-J. Chang, A. W. Y. Su, and Y.-H. Yang, "Music score expansion with variable-length infilling," in *Proc. International Conference on Music Information Retrieval*, 2021, late-breaking and demo paper.

[23] S. Wei, G. Xia, Y. Zhang, L. Lin, and W. Gao, "Music phrase inpainting using long-term representation and contrastive loss," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 186–190.

[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," *Proc. Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, Reissue, with a New Preface*. MIT press, 1996.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. Annual Meeting of the Association for Computational*, 2019.

[28] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[29] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Muller, and Y.-H. Yang, "Theme Transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, 2022.

[30] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proc. International Conference on Music Information Retrieval*, 2020.

[31] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," in *Proc. Conference on AI Music Creativity*, 2020.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations*, 2014.

[33] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. International conference on machine learning*, 2020, pp. 1597–1607.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[35] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics*, 2018.

[36] G. Ke, D. He, and T.-Y. Liu, "Rethinking positional encoding in language pre-training," in *Proc. International Conference on Learning Representations*, 2021.

[37] A. Liutkus, O. Cífka, S.-L. Wu, U. Simsekli, Y.-H. Yang, and G. Richard, "Relative positional encoding for Transformers with linear complexity," in *Proc. International Conference on Machine Learning*, 2021.

[38] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. International Conference on Learning Representations*, 2020.

[39] N. Hu, R. B. Dannenberg, and A. L. Lewis, "A probabilistic model of melodic similarity," in *Proc. International Computer Music Conference*, 2002.

ROBUST MELODY TRACK IDENTIFICATION IN SYMBOLIC MUSIC

Xichu Ma **Xiao Liu** **Bowen Zhang** **Ye Wang**
National University of Singapore
{ma_xichu, liuxiao, zbowen}@u.nus.edu, wangye@comp.nus.edu.sg

ABSTRACT

Melody tracks are worthy of special attention in symbolic music information retrieval (MIR) because they contribute more towards music perception than many other musical components. However, many existing symbolic MIR systems neglect the preprocessing of melody track identification (MTI). Moreover, existing MTI methods often deal with MIDIs of the same genres and follow specific track configuration. This limits their generalization and robustness to unseen data. To address these challenges, we propose a CNN-Transformer-based MTI model designed to identify a single melody track for arbitrary MIDI files robustly. To accommodate longer inputs, We also employ a sparse Transformer, speeding up attention computation. Our experiments show that our proposed model outperforms state-of-the-art (SOTA) algorithms in accuracy and benefits downstream MIR tasks.

1. INTRODUCTION

Listeners' perception of musical works is greatly shaped by the melodic lines of those works [1]. Human listeners can usually easily distinguish the melody from other musical components such as harmonic lines [2]. It is, however, not as easy for machines. Melody is also a major focus of many music information retrieval (MIR) tasks. These tasks fall into three categories: 1) Melody as a target, where the system seeks to output a suitable melody. Examples of this task include melody segregation [3, 4] and melody generation [5, 6]. 2) Melody as a requirement, where the model takes in an explicitly-identified melodic line as an input. Such tasks include lyrics generation from music [7, 8] and singing synthesis [9, 10]. 3) Melody as a dependency, where the melody is not explicitly provided as input but still heavily influences the results. Examples include genre classification [11] and music similarity measurement [12].

However, for melody-as-a-target tasks, the data of annotated melody labels can be difficult to obtain. Thus, an automatic MTI model can create more data for their training. The melodic line is the decisive factor for these melody-as-a-dependency tasks [13, 14]. However, current studies on melody-as-a-dependency tasks often neglect to intelligently identify the melodic lines of music. For example, a music embedding learning model, PiRhDy [15], simply regards the track with the highest average pitch as the melody. Existing genre classifiers treat all tracks equally rather than give special emphasis to the melodic line [11]. On the one hand, melody plays fundamental role in music perception [16]; studies have shown the success of attention and self-attention mechanisms in sequence modeling [17, 18]. On this basis, we hypothesize that melody-as-a-dependency tasks' performance would likely improve if the data are available with identified melodic lines. This preprocessing of MTI would also help melody-as-a-requirement applications. For instance, some stylistic music generation systems can only accommodate specifically formatted melody note sequences as input, demanding musicians' manual annotation [19]. Equipped with a MTI module, it could eliminate the dependence on manual processing.

There are two main challenges in developing a robust Melody Track Identification (MTI) model for symbolic music. First, many existing methods either rely on normative composition theories [20]. They assume that the input MIDIs are highly-formatted (i.e., one channel has only one track) and usually in limited genres [21, 22]. This assumption limits their generalization and robustness in dealing with more diverse or "dirtier" samples. Second, while local and global musical features are both significant for MTI, architectures such as CNNs and RNNs are not ideal in capturing short-and-long-term features simultaneously.

To address the above challenges, we propose a CNN-Transformer model to identify one single track as the melody for a given input MIDI file. This architecture effectively captures local musical features at the level of a measure (or a frame) while also making predictions over the whole piece based on global aggregations of those local features. We also utilize sparse attention in the Transformer [23] to speed up computation and reduce memory requirements so longer musical works can be processed.

We conduct the MTI experiment on a manually annotated dataset of 11,625 MIDIs. The results show that the proposed model makes a breakthrough in the accuracy of MTI. We also evaluate our proposed model as a preprocessing plug-in in the three types of melody-sensitive tasks. The results indicate that our proposed model is easy to incorporate and effectively improves their performance. This work includes a dataset with 13,100 widely used MIDIs whose melody tracks have been labeled. This dataset will be an asset to other MIR studies in the future.

© F. Author, S. Author, and T. Author. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Author, S. Author, and T. Author, "Robust Melody Track Identification in Symbolic Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. RELATED WORK

2.1 Melody-Sensitive Tasks

Many studies have been done on MIR tasks which are heavily influenced by melody. Whether any specific such task has melody as a target, melody as a requirement, or melody as a dependency, accurate identification and knowledge of melody can help solve that task.

2.1.1 Melody as a Target

Tasks which set the melody as a target seek to output a melody. These tasks include generating melodies, extracting melodies from existing music, and converting one melody into another. Most existing systems for these tasks require annotated melodic lines as labels. For instance, MSNet is an encoder-decoder network designed for melody extraction from audio [4]. It takes an audio spectrum as input, extracts a salience map via UNet-like networks, and then estimates the audio's melodic line. Another system treating melody as a target developed a LSTM-GAN model conditioned on lyrics to generate symbolic melody sequences [5].

2.1.2 Melody as a Requirement

Tasks which take melody as a requirement, by definition, require an explicitly labeled melodic line as an input. The aforementioned PiRhDy is a model designed for one such task: it needs an explicitly labeled melodic line so that it can predict a MIDI file's melodic notes by the contexts and thus learning the representation of the music [15]. Similarly, Melody2Vec [24] can only utilize MIDIs with labeled track names of "melody" or "vocal" as training data. Another example is AI-Lyrics, a system which automatically generates lyrics to parallel an input musical piece's style and syllable alignment [8]. This generator requires a syllable template which can only be extracted from the melody track of the input MIDI files. Other lyric generators and singing synthesis systems also require vocal melody to be explicitly identified as inputs [7, 9, 10]. Currently, the melodic lines in MIDI files must be manually labeled for those files to work with these systems, which hinders the systems' practical applications.

2.1.3 Melody as a Dependency

Tasks which use melody as a dependency do not require MIDIs with explicitly labeled melodic lines, but they are dependent on melody information nonetheless. For instance, MuSeReNet is a genre classifier for symbolic music which extracts latent features of MIDI files with CNNs and then feeds them into a Multilayer Perceptron (MLP) to identify their genre [11]. Genre is heavily influenced by melody, so better knowledge of melody would no doubt help improve this system's accuracy. Another example is MusicBERT, which is a large-scale pre-trained model for music understanding, resembling the BERT from the field of Natural Language Processing (NLP) [25, 26]. It pre-trains the Transformer encoder [18] by solving a pretext task where it learns a music embedding by reconstructing notes' eight music elements such as tempo, pitch, velocity, and many others. Considering these elements are influenced by melody, we believe the knowledge of melody would likely help the learning.

2.2 Melody Identification

Melody extraction and identification problems can also be subdivided based on the different source and target data forms. The aforementioned MSNet is designed to extract melodic lines from acoustic audio, but though it achieves SOTA performance, it is sensitive to pre-defined settings such as input frame length and is not robust to inputs which do not match with those settings. Other models attempt to perform melodic line extraction from symbolic music [2, 3]. Namely, they output a note sequence representing the melodic line of an input MIDI. Hsiao et al., for instance, uses 1D-CNN networks to extract contextual information and further predict the probability of every pair of notes belonging to the same track. However, this model requires a lot of computation and thus have difficulty handling scaled up data size in a short time.

Rather than generating a note sequence as the melodic line, MTI systems aim to simply label the track which contains the melody. The Skyline algorithm, for instance, picks the track with the highest average pitch as the melody track [27]. There also exist Bayesian approaches which estimate a track to be a melody track if that track maximizes an accumulated probability derived from features such as note velocity, pitch values, and so on [28]. There are few deep learning approaches seeking to identify a single track of a MIDI file as the melody track. However, the emergence of self-attention mechanism makes the deep-learning-based models more powerful to cope with short term and long term musical structures simultaneously. And this motivates us to develop our CNN-Transformer model for this problem.

3. METHOD

In this section, we formulate the MTI problem, then present the data representation and the architecture design.

3.1 Problem Formulation

First, we define robustness of a MTI model for symbolic music in four aspects. A robust MTI model should be capable of identifying the melody tracks from MIDI files that 1) do not follow pre-defined track configurations (i.e., the first channel contains the melody track, the melody track is named as "Melody", etc.), 2) have several melodic and non-melodic tracks within one channel, 3) consist of various musical parameters (instruments, tempos, velocities and time signatures), and 4) diverse in genres, styles, and sentiments.

An arbitrary piece of symbolic music in MIDI format is denoted as $\mathcal{S} = \{Tr_N\}$, where N denotes its number of tracks, each track $tr_i \in Tr$ is a sequence of MIDI events, which trigger control demands or music notes. Namely,