CONTEXT-AWARE GENERATION OF MELODIC MIDI LOOPS

Nikolay Glazyrin
Yandex Music
`nglazyrin@gmail.com`

ABSTRACT

In this paper we describe a method of generation of symbolic melody loops to fit in an existing set of melodic, harmonic and bass loops. Two main contributions are the way of representing MIDI data in textual form and the post-processing procedure for selecting best candidates among a potentially infinite set of generated melodies. Due to the proposed representation a generative model can be conditioned on music style and music intensity. Overall performance of our method has been evaluated with the help of a professional music producer.

1. INTRODUCTION

The vast field of computer-assisted music generation is getting increasing amount of attention in recent years. An extensive overview of tasks and approaches emerging within that field can be found in [1]. In this paper we focus on the task of generating 8-bar symbolic melody loops to enhance an existing collection of loops used for assembling a potentially endless stream of music in real time.

2. PROBLEM

Existing generative music services, such as Mubert [1] or Endel [2], are focused on creating a stream of music that does not try to really please the listener, but rather to put them in a desired state or mood. Such music must have a certain degree of overall quality and diversity so as to not distract the listener and not being too monotonic within a listening session and from session to session.

Building musical streams from loops constitutes a compromise between the amount of human labor and the quality and diversity of the result. Creation of a number of quality loops requires less effort from a professional musician than production of several complete tracks. A music generation system can then arrange and mix those loops in various ways thus providing the diversity.

From our preliminary experiments we found that listeners tend to memorize the melodies they've heard while keeping less attention to other parts of music, such as bass and harmony. So given the existing collection of human generated loops, where each loop is available in both MIDI and audio formats, we want to add more melodies while maintaining their quality and stylistic coherency.

In the considered collection loops are grouped into projects. Each loop is associated with an "instrument" (e.g. *bass*, *drums*, *harmony*, *melody*), and within a project any combination of loops of different instruments is allowed. A project has associated key, style (one of *hiphop*, *rock*, *pop*, *electronic*) and intensity (an integer value from 1 to 4). Projects with higher intensity have faster tempo, more notes in loops and more aggressive sound. All this information needs to considered when generating new melodies. We only want to generate MIDI notes, and leave the choice of timbre up to a professional musician.

3. METHOD

Transformer-based models became a de facto standard for working with various types of data, primarily texts. We have experimented with a number of text generation models from HuggingFace Transformers [2] library [3]. The best results were obtained using Reformer [3].

For training we use triplets (bass loop, harmony loop, melody loop) encoded in textual form as described in section 3.1, the order of instruments is always the same. Each triplet's loops are chosen from one project. The trained model is then able to generate new texts or continue existing ones. During the inference an encoded representation of bass and harmony loops and the header token for melody are given to the input. To generate a new melody, the model is asked to continue the input sequence.

The representation is designed so that each word is fixed, independent of adjacent words, and the vocabulary is small. Therefore we have used a word-level tokenizer, so that each word in the encoded representation is directly translated to a single token and vice versa.

3.1 Music Encoding

The encoding format was partially inspired by [4] and [5]. It is designed to minimize the possibility of incorrect token sequences, e.g. start token without corresponding stop token. See Figure 1 for an example of encoding. An input has the header token `piece` and two tokens that represent music style (e.g. `electronic`) and intensity (e.g. `intensity2`) of the project that hosts given loops. These

[1] `https://mubert.com/`
[2] `https://endel.io/`
[3] `https://github.com/huggingface/transformers`

```
piece electronic intensity2 loop bass
bar q0 p4 o4 d2 v64 bar q8 p4 o4 d1 v56
bar q0 p4 o4 d1 v60 ... end_loop loop ...
```

Figure 1. Example encoding.

two kinds of tokens are always given to the input and act as a condition on model's output. Each loop is then encoded between `loop` and `end_loop` tokens.

Within each loop first token encodes the instrument (`bass` here), following tokens encode its notes separated into bars using `bar` tokens.

Each note is encoded as a tuple of 5 tokens: (position, pitch, octave, duration, velocity). Position tokens are in range `q0`–`q15` and encode the start position of the note within a bar quantized to 1/16. Duration tokens are in range `d0`–`d24` and encode the quantized duration of the note from 1/32 to 8 bars with step size doubled at 1/16, 8/16, 1, 2 and 4 bars, so that the resolution is higher for shorter notes. Velocity values are quantized to multiples of 4, and the corresponding tokens are chosen from {`v0`, `v4`, `v8`, ..., `v128`}.

Pitches of notes belonging to the scale of current project's key are represented as `p1` (tonic) to `p7`. Off-scale notes are then encoded as `p8` to `p12` starting from the one closest to tonic. This way the encoded representation becomes invariant to key and mode (major or minor). The octave for each note is calculated as if tonic of the current key was the first note in its MIDI octave instead of C. The corresponding token is chosen from the range `o0`–`o8`.

The resulting model vocabulary has size of 116 tokens, including special ones required only for training.

3.2 Training

Compared to original Reformer, we have decreased the number of hidden layers to 6 and the number of attention heads to 8. The resulting model has 3.6M trainable parameters. It was trained for 200 epochs using a relatively small dataset of nearly 14000 encoded triplets consisting of MIDI loops created by professional music producers.

3.3 Postprocessing

A number of heuristics were adopted to select best candidates among the model outputs.

First of all, we have noticed that the model tends to occasionally shift the whole generated melody by 1/16 forward or backward relatively to downbeats. This becomes especially noticeable with longer notes. To compensate for that, we shift the notes with duration of at least 1/4 starting 1/16 apart from beat positions to those corresponding beat positions. Then we clip the generated melodies to 8 bars and discard melodies shorter than 3 bars.

A number of metrics are then calculated over resulting melodies. They originate from the MusPy [6] library [4], except for the one called max_notes_in_a_row, which is calculated as the number of notes having the same pitch and starting one after another with no breaks in between. Table 1 summarizes these metrics and corresponding threshold values.

Metric	Desired values
n_pitch_classes_used	≥ 2
pitch_in_scale_rate	1
pitch_entropy	> 1
polyphony_rate	≤ 0.5
max_notes_in_a_row	≤ 2

Table 1. Metrics used to filter unsuitable melodies.

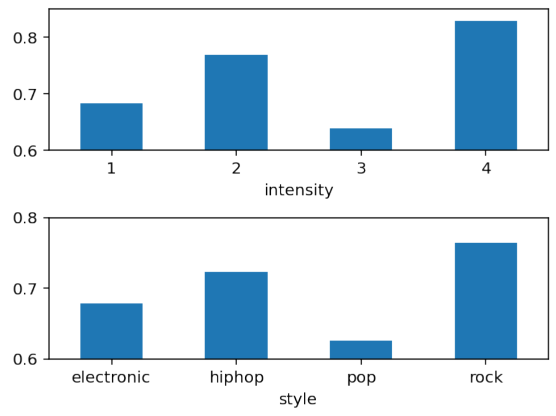

Figure 2. Fraction of unchanged notes by intensity (above) and style (below)

For each input we sample 20 candidate melodies that have all metrics within the desired range.

4. RESULTS AND DISCUSSION

Using the model described above, additional melodies in MIDI format have been generated for 63 projects having different styles and intensities. These melodies have been assessed in context of the corresponding project by a professional music producer, who was allowed to accept a melody as is, accept after modifying some notes or discard.

In total 70% of generated loops have been accepted. Within the accepted loops 71% of notes have been left unchanged. As can be seen from Figure 2, the fraction of unchanged notes is quite high for all different intensities and styles, which means that the model is able to control its output given the tokens for style and intensity in the beginning of the input sequence.

Desired metric values listed in Table 1 are selected to filter out some the cases unwanted in our scenario, such as when the model gets stuck on a particular note or a pair of notes, or when the generated melody is made up of multiple notes playing in parallel. But the final judgment on generated melodies is intentionally left to a human expert.

The encoding method presented here can easily be modified to generate scores for other types of instruments, e.g. bass. But the set of metrics and their thresholds need to be modified accordingly.

[4] `https://github.com/salu133445/muspy`

5. REFERENCES

[1] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," *CoRR*, vol. abs/2011.06801, 2020. [Online]. Available: https://arxiv.org/abs/2011.06801

[2] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[3] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgNKkHtvB

[4] J. Ens and P. Pasquier, "Mmm : Exploring conditional multi-track music generation with the transformer," 2020.

[5] Y.-S. Huang and y.-h. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *MM '20: Proceedings of the 28th ACM International Conference on Multimedia*, 10 2020, pp. 1180–1188.

[6] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "Muspy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.