Flexible Generation with the Multi-Track Music Machine

Jeff Ens
Simon Fraser University
jeffe@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

ABSTRACT

We propose the Multi-Track Music Machine (MMM), a generative system based on the Transformer architecture that is capable of generating multi-track music. In contrast to previous work, which represents musical material as a single time-ordered sequence, where the musical events corresponding to different tracks are interleaved, we create a time-ordered sequence of musical events for each track and concatenate several tracks into a single sequence. This takes advantage of the Transformer's attention-mechanism, which can adeptly handle long-term dependencies. We explore how various representations can offer the user a high degree of control at generation time, providing an interactive demo that accommodates track-level and bar-level inpainting, and offers control over track instrumentation and note density.

1. INTRODUCTION

Research involving generative music systems has focused on modelling musical material as an end-goal, rather than on the affordances of such systems in practical scenarios. As a result, there has been a focus on developing novel architectures and demonstrating that music generated with these architectures is of comparable quality to human-composed music, often via a listening test. Although this is a necessary first step, as systems must be capable of generating compelling material before they can be useful in a practical context, given the impressive capabilities of the Transformer-based models in the music domain [1,2], we shift our focus to increasing the affordances of a Transformer-based system. To achieve this goal, we develop a novel representation for multi-track musical material that enables a variety of generation methods.

Our work is most similar to LahkNES [1], MusicVAE [3] and MuseNet [4], which all employ token-based representations to model multi-track music. However, there are several key differences. First of all, we decouple track information from `NOTE_ON` and `NOTE_OFF` tokens, allowing the use of the same `NOTE_ON` and `NOTE_OFF` tokens in each track. Secondly, using our representation, we are able to accommodate a wide variety of instruments, including all 128 general midi instruments, and handle an arbitrary set of tracks. In contrast, LahkNES (resp. MusicVAE) have a fixed-schema of 4 (resp. 3) tracks, and MuseNet only supports 10 distinct instruments. Third, our system allows for specific attribute control over the instrument for each track, with the guarantee that a particular instrument will be used. While MuseNet offers a similar type of control, user selected instruments are only treated as a suggestion, and the generated output is not guaranteed to feature the selected instruments. Fourth, we offer the user control over the note-density of each track, which is not accomodated with LahkNES, MusicVAE or MuseNet. Finally, we allow for track-level and bar-level inpainting, which is not possible using LahkNES, MusicVAE and MuseNet. Collectively, these improvements afford the user a high-degree of control when generating.

2. PROPOSED REPRESENTATION

To provide a comprehensive overview of the proposed representation, we first describe how a single bar of musical material is represented. We represent musical material using 128 `NOTE_ON` tokens, 128 `NOTE_OFF` tokens, and 48 `TIME_SHIFT` tokens. Since musical events are quantized using 12 subdivisions per beat, 48 `TIME_SHIFT` tokens allow for the representation of any rhythmic unit from sixteenth note triplets to a full 4-beat bar of silence. Each bar begins with a `BAR_START` token, and ends with a `BAR_END` token. Tracks are simply a sequence of bars delimited by `TRACK_START` and `TRACK_END` tokens. At the start of each track, immediately following the `TRACK_START` token, an `INSTRUMENT` token is used to specify the MIDI program which is to be used to play the notes on this particular track. Since there are 128 possible MIDI programs, we have 128 distinct `INSTRUMENT` tokens. A `DENSITY_LEVEL` token follows the `INSTRUMENT` token, and indicates the note density of the current track. A piece is simply a sequence of tracks, however, all tracks sound simultaneously rather than being played one after the other. A piece begins with the `PIECE_START` token. Notably, we do not use a `PIECE_END` token, as we can simply sample until we reach the n^{th} `TRACK_END` token if we wish to generate n tracks. We refer to this representation as the MultiTrack representation.

In order to accommodate bar-level inpainting, we must guarantee that the bars on which we want to condition precede the bars we wish to predict, in the sequence of tokens that is passed to the model. To do this, we remove all the bars which are to be predicted from the piece, and replace each bar with a `FILL_PLACEHOLDER` token. Then, at the end of the piece (i.e. immediately after the last `TRACK_END` token), we insert each bar, delimiting each bar with `FILL_START` and `FILL_END` tokens instead of `BAR_START` and `BAR_END` tokens. Note that these bars must appear in the same order as the they appeared in the original MultiTrack representation. We refer to this representation as the BarFill representation. Note that the MultiTrack representation is simply a special case of the BarFill representation, where no bars are selected for inpainting.

3. INTERACTIVE DEMO

We use the Lahk MIDI Dataset (LMD) [5], which is comprised of 176,581 MIDI files. For each of the 128 general MIDI instruments, we calculate the number of note onsets for each bar in the dataset, and use the quantiles of the resulting distributions to define distinct note-density bins for each MIDI instrument. Note that using the same note-density bins for all instrument types would be problematic as note-density varies significantly between instruments. We use 10 different note-density bins, where the i^{th} bin is bounded by the $10i$ (lower) and $10(i+1)$ (upper) quantiles. We train a GPT2 [6] model using the HuggingFace Transformers library [7] with 8 attention heads, 6 layers, an embedding size of 512, and an attention window of 2048. We train two types of models: MMMBar, which is trained using the BarFill representation; MMMTrack, which is trained using the MultiTrack representation. We train 4-bar and 8-bar versions of MMMBar and MMMTrack. For 4-bar (resp. 8-bar) models we provide the model with at most 12 (resp. 6) tracks. Each time we select a n-bar segment, we randomly order the tracks so that the model learns each possible conditional between different types of tracks. When training the MMMBar models, we also select a random subset of bars for inpainting.

In order to illustrate the flexibility of MMM, we make available [1] examples generated by the system, and an interactive demo. The demo was developed in Google Colab, making it accessible to all users with a compatible internet browser. The interface automatically selects the appropriate model, either MMMBar or MMMTrack, based on the bars or tracks that are selected for generation. We briefly outline the various ways that one can interact with MMM when generating musical material.

1. Track Inpainting : Given a possibly empty set of tracks $\mathbf{t} = \{t^1, ..., t^k\}$, we can generate n additional tracks. To do this, we condition the model with the tokens representing k tracks and then sample until the n^{th} `TRACK_END` token is reached.

2. Bar Inpainting : Given a set of tracks $\mathbf{t} = \{t^1, ..., t^k\}$ and a set of bars $\mathbf{b} = \{b_1, ..., b_n\}$, we can resample each bar in $\mathbf{b}$. For this method, we condition the model with the tokens representing all the tracks, replacing each b_i in $\mathbf{b}$ with the `FILL_PLACEHOLDER`. Then we sample until the n^{th} `FILL_END` token is reached.

3. Attribute Control for Instruments: We can specify a set of MIDI instruments for each generated track, from which the model will choose. Practically, this is accomplished by masking the MIDI instruments we wish to avoid when sampling.

4. Attribute Control for Note Density : We can specify the note density level for each generated track.

5. Iterative Generation : The user can chain together various generation methods to iteratively compose a piece of music. Iterative generation also affords the user the opportunity to iteratively explore variations on generated material, or gradually refine a piece by progressively resampling bars.

We have introduced a novel approach to representing musical material that offers increased control over the generated output. This offers a new and exciting avenue for future work, harnessing the strengths of the Transformer architecture to provide fine-grained control for the user.

[1] `https://jeffreyjohnens.github.io/MMM/`

4. REFERENCES

[1] 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. of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 685–692.

[2] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations*, 2019.

[3] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4361–4370.

[4] C. Payne, "Musenet," *OpenAI*, April 2019, openai.com/blog/musenet.

[5] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.