Three-modal guidance for symbolic music generation: melody, structure, texture

Daniel Alexander Lucht
Bielefeld University
daniel.lucht@uni-bielefeld.de

David Leins
Bielefeld University
dleins@techfak.de

Dimitri von Rütte
ETH Zürich
dvruette@ethz.ch

Alexandra Moringen
University of Greifswald
alexandra.moringen@uni-greifswald.de

Abstract

The vision of this work is a flexible co-creation of music between a human and a trained model that can be used with or without domain knowledge. Building upon previous work, the transformer-based FIGARO framework, we propose a symbolic music generation that takes up three separate guiding modalities: a melody, structural piece description termed expert description, and music texture. Our approach aims to enable a composer to try out combinations of different melodies, expert descriptions, and textures.

FIGARO is capable of generating music based on a structural expert description generated with domain knowledge, and a learned representation of a music piece. The description part of the input is generated for each bar and provides a multitude of features, such as mean pitch, chords, note density, etc. The learned representation is generated for each bar as a whole. The main contribution of this work is a more extensive modularisation of the input to the model, i.e. the concept of explicit separation of the input into three above-mentioned distinct modalities commonly used in music composition, and symbolic description of the musical works: melody, domain knowledge-driven description of the piece, and texture guiding the feel of the music. We demonstrate our preliminary results with a novel model-based implementation of a piece, provided a melody, a bar-wise description, and a multi-track accompaniment.

1 Introduction

The vision of this work is a flexible co-creation of music between a human and a trained model that can be performed with different input levels of domain knowledge, modularized and multifaceted guidance. FIGARO [1], the foundation framework for this work, is a transformer-based symbolic music generator that enables fine-grained control of the output. The main feature of the model is its bar-wise and multifaceted control of the output. Previously developed advanced models for music generation often offer global control that conditions the overall output, e.g. style of the generated piece or text prompting (see also [2]). FIGARO implements fine-grained control by adding an extra representation for the bar-wise structure (expert description) to a traditional input embedding of the music piece into the generators' input. FIGARO can be used without domain knowledge, as the expert description can be generated from an input in MIDI format. Its other strengths are multi-track support and multi-signature capabilities.

In order to generate a music sequence, FIGARO performs two main steps: embedding of the input data, and the music token generation with the transformer. In the embedding step, FIGARO performs a mapping of the input into a latent space using a trained VQ-VAE model [6] resulting in a high-fidelity representation of the input. Complementary to this, it embeds a bar-wise expert description of the piece derived with domain knowledge, to gain the second low-fidelity part of the embedding vector. Such descriptions consist of specifications of e.g. time signature, chords, and instruments. Its embedding is trained with reconstruction loss for the original sequence of notes. The main contribution of this work is a more extensive modularisation of the input to the model, i.e. the concept of explicit separation of the input into three above-mentioned distinct modalities commonly used in music composition, and symbolic description of the musical works: melody, domain knowledge-driven expert description of the piece, and texture guiding the feel of the music. We demonstrate our preliminary results with a novel model-based implementation of a piece, provided a melody, a bar-wise description, and a multi-track accompaniment.

2 Methods

The input to the model is three-fold, the melody, the expert description (generated or hand-crafted), and the accompaniment. An important step that precedes the training of the model is the separation of the melody and the accompaniment for a given piece of music. In the further steps, integration of the input, and generation of a new sequence, the main components of the method are: tokenization of the input MIDI melody and accompaniment, learning to represent the token sequences through appropriate embeddings, generation, and embedding for the expert description, and finally, the training of the generative model itself.

2.1 Tokenization: MIDI to REMI+

MIDI (Musical Instrument Digital Interface) is the input format of our data. MIDI files consist of a sequence of messages, with the most crucial ones being Note-On and Note-Off events, along with note numbers, velocity, and program change events. Note-On and Note-Off events mark the exact moments when a note begins and ends, respectively. Pitch is represented by note numbers ranging from 0 to 127, with 60 corresponding to Middle C. Velocity indicates the intensity at which a note is played, such as the force used to press a piano key or blow a trumpet. This value also ranges from 0 to 127. Instruments are selected with Program Change events, where the program number is an integer between 0 and 127. A key feature of the MIDI format is its ability to encode messages across 16 independent channels, which is particularly useful for representing polyphonic music. To convert MIDI in a format that is more suitable for processing with language models, MIDI is tokenized into REMI+ format [1]. It is capable of representing multi-instrument and multi-signature pieces as a sequence of tokens. REMI+ encodes notes as a tuple of their position, pitch, velocity, and duration. Figure 2 shows an exemplary REMI+ sequence.

2.2 Input Melody and Accompaniment

In order to separate input MIDI files into melody and accompaniment we use MELODIA [3]. The MELODIA algorithm is based on the creation and characterization of pitch contours, time continuous sequences of pitch candidates grouped using auditory streaming cues [4]. After extracting the pitch values with the MELODIA algorithm we use pitch contour segmentation [7] to retrieve each MIDI note's onset time, duration, and pitch value. The most important information that is lost throughout this process however is each note's instrument. To recover the complete note information for the melody notes, we compare them with the original MIDI data. We use a threshold for start, end, and pitch because the MELODIA algorithm and pitch contour segmentation produce approximations rather than exact matches. A melody note can be played by multiple instruments at the same time thus the comparison is done for each of the original's instruments to preserve the polyphonic character of the melody. The notes that are not part of the melody get selected to form the accompaniment. Figure 1 demonstrates an example of the separation of the input into the melody and accompaniment. Each box resembles a MIDI note with the box's width equalling the note's length and the number denoting the pitch. The colors indicate where each input note is placed in the resulting melody and accompaniment. For this simple demonstration, we assume a pitch threshold of 4 and no time threshold meaning that the notes' start and end times have to match exactly the pitches are allowed to have a deviation of 4. The second bar displays the information we retrieve from the MELODIA algorithm and pitch contour segmentation which are pitch values and times,

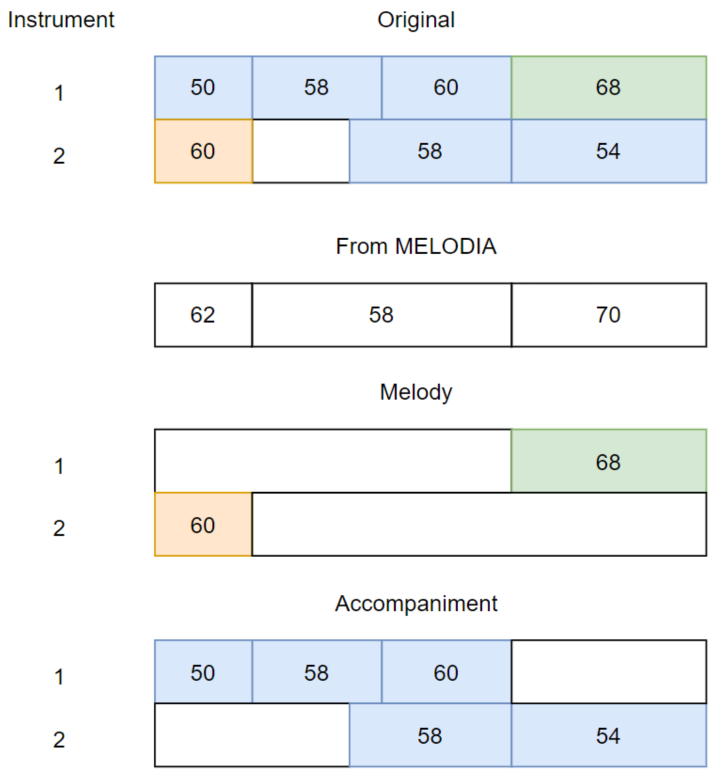

Figure 1: An example for the computation of melody and accompaniment from multi-track MIDI input with MELODIA.

however no instrument data. For the training we use existing combinations of melody and accompaniment, however, during generation, we are interested in using new combinations of both input modalities.

2.3 Representation for Melody, Expert Description and Accompaniment

In this section, we describe how we convert the three input modalities into a format that can be processed by the transformer.

Melody and Expert description Building upon the previous work [1], we use expert descriptions to define the structure of the piece. An expert description can be either generated from a MIDI file or written by hand. It describes the structure of a song bar-wise[1]. The *low-fidelity* features that the expert description captures are time signature, note density, instruments, chords, and mean values for pitch, velocity, and duration of the notes.

The expert description is designed as a front-end interaction channel between the artist and the model. In this work, we extend the expert description used in [1] by adding the tokens corresponding to the melody to it (see Figure 3). The resulting expert description does not only contain the structural piece information but also the melodic component. The melody is only present in this input and is excluded from all other inputs. Note that capturing every combination of a note's values (s. section 2.1) would significantly increase the vocabulary size as valid values for pitch and velocity reach from 0 to 127 according to the MIDI format specification and the duration has no clear boundary. Therefore we decided to follow the present approach used to represent MIDI notes in the

[1]see an example of an expert description under this link `https://colab.research.google.com/drive/1UAKFkbPQTfkYMq1GxXfGZOJXOXU_svo6#scrollTo=lYwuwTlezIDc`

REMI+ Tokens

```
<bos>
Bar 1 TimeSignature 3 /4
Pos 0 Tempo 120
Pos 0 Chord C : min
Pos 0 Instrument Drums Pitch 36 Vel 90 Dur 0
Pos 0 Instrument Piano Pitch 64 Vel 85 Dur 4
Pos 4 Instrument Piano Pitch 66 Vel 85 Dur 4
Bar 2 TimeSignature 3 /4
Pos 0 Tempo 120
. . .
<eos>
```

Figure 2: An example of a music piece in REMI+ format (example taken from [1]).

Expert description with notes as single tokens

```
<Bar_1> <Time Signature_4/4> <Note Density_1> <Mean Pitch_15>
<Mean Velocity_19> <Mean Duration_32>
<Instrument_Acoustic Grand Piano> <Chord_C:maj> <Chord_G:maj>
<Melody Instrument_Acoustic Grand Piano> <Melody Instrument_Clarinet>
<Melody Note_53;56;384> <Melody Note_55;56;168>
<Melody Note_54;56;192> <Melody Note_53;48;408>
```

Figure 3: An example of an extended expert description for one bar where the numbers at each melody note token correspond to pitch, velocity and duration, resp.

REMI+ format and used quantization for velocity and duration and kept the whole range of possible pitch values. This results in a total vocabulary size increase from 960 to 246080 tokens.

Despite this large increase in vocabulary size the concise representation of notes as a single token enables the model to keep a lower amount of tokens in each bar which is beneficial for the model's small context size of 256 tokens.

An alternative modeling approach that we used in our experiments represents the notes' pitches, velocities, durations, and instruments using REMI+ tokens with position tokens before each note, denoting the notes' order with ascending integers (see Figure4). This reduces the total vocabulary size to 1456 tokens but increases the number of tokens per bar respectively. However, it also enables the representation of polyphonic melodies because instruments are assigned to notes individually instead of being declared for the entire melody beforehand.

Accompaniment The music piece without the melody forms the accompaniment. It is currently used to represent the texture of the piece, complementing the structural representation provided in the expert description, and by the melody. Like in the previous work [1], the embedding for the accompaniment tokens is trained/generated with the VQ-VAE (Vector Quantized-Variational AutoEncoder) [6]. Other than in previous work, only the accompaniment and not the whole piece is encoded in this component. This part of the embedding is termed *the learned description* and represents *high fidelity* features of the input.

Both embeddings The accompaniment and the extended expert description are concatenated for training and generation.

2.4 Dataset

For our preliminary experiments, we used a subset of LakhMIDI set - "Clean MIDI" [8] - which contains 15887 files. The full LakhMIDI set contains a small amount of invalid or corrupt files

Expert description with notes as multiple tokens

```
<Bar_1> <Time Signature_4/4> <Note Density_1> <Mean Pitch_15>
<Mean Velocity_19> <Mean Duration_32>
<Instrument_Acoustic Grand Piano> <Chord_C:maj> <Chord_G:maj>
<Pos_0> <Instrument_Clarinet> <Pitch_53> <Vel_56> <Dur_384>
<Pos_1> <Instrument_Clarinet> <Pitch_55> <Vel_56> <Dur_168>
<Pos_2> <Instrument_Acoustic Grand Piano> <Pitch_54> <Vel_56> <Dur_192>
```

Figure 4: An example of an extended expert description for one bar with each melody note's pitch, velocity, and duration being encoded as separate tokens.

impacting the quality of the extracted melodies [9]. Thus we decided to use the subset instead of the full set as in [1].

3 Experimental Results

For our preliminary experiments, we trained the VQ-VAE and the transformer-based generator on the above-mentioned dataset. In our first experiment (see Table 1) we used songs from the dataset as a source for input descriptions and measured the difference between input and output pieces using the metrics from previous work [1].

Model	I_F1	C_F1	TS_Acc	ND	P_MOA	V_MOA	D_MOA	CHR	GRO
Original	0.952	0.598	0.996	0.210	0.840	0.771	0.755	0.803	0.860
Tuple Notes	0.877	0.394	0.986	0.413	0.712	0.644	0.555	0.679	0.683
REMI+ Notes	0.936	0.448	0.994	0.311	0.779	0.769	0.687	0.751	0.680

Table 1: Comparison of the models using the generated results conditioned on input descriptions using the metrics described in [1]. Models are "Original" (the previous FIGARO), "Tuple Notes" representing notes as single tokens) and "REMI+ Notes" (representing a single note as multiple tokens, very similar to REMI+). Metric names (from left to right) represent F1 scores for instruments and chords, accuracy of time signature, note density NRMSE, MOA of pitch, velocity and duration, and cosine similarity for chroma and groove.

The results clearly show that our second approach outperforms the first in every metric except GRO. Overall its scores lie below the original FIGARO model.

In a second experiment, we assigned the melodies and accompaniments computed from the dataset arbitrarily to each other to evaluate how the model variants perform on data they have not been trained on. This experiment's results (see Table 2) underline the first experiment's findings.

Model	I_F1	C_F1	TS_Acc	ND	P_MOA	V_MOA	D_MOA	CHR	GRO
Tuple Notes	0.669	0.176	0.981	1.06	0.519	0.420	0.317	0.425	0.336
REMI+ Notes	0.736	0.229	0.993	0.823	0.596	0.533	0.454	0.523	0.340

Table 2: Comparison of combinations of melodies and accompaniments.

We provide examples of generated MIDI files in the supplementary material here.

4 Conclusion

In this work, we have performed a preliminary test of the pipeline that envisions a three-fold guidance of music generation with melody, structural description, and accompaniment. To this end, we have tested two representations of melody as tokens containing a combination of pitch, velocity, and duration. Building upon this baseline, in our future work we will explore other representations of the melody and test alternative methods of melody extraction. In the long run, we are interested in exploring, how to integrate the resulting framework as a building block into raw audio generation.

References

[1] FIGARO: Controllable Music Generation using Learned and Expert Features. Dimitri von Rütte, Luca Biggio, Yannic Kilcher, Thomas Hofmann, ICML, 2023

[2] Motifs, Phrases, and Beyond: The Modelling of Structure in Symbolic Music Generation. Keshav Bhandari, Simon Colton, Artificial Intelligence in Music, Sound, Art and Design, 2024

[3] MELODIA project: `https://github.com/justinsalamon/audio_to_midi_melodia`

[4] Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics. J. Salamon and E. Gómez, IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770, Aug. 2012.

[5] https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html

[6] Neural Discrete Representation Learning. Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu, Neurips 2017

[7] Signal processing for melody transcription. Rodger J. Mcnab, Lloyd A. Smith, Ian Witten in Proc. Proc. 19th Australasian Computer Science Conf., 1996

[8] Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. Colin Raffel. PhD Thesis, 2016.

[9] `https://colinraffel.com/projects/lmd/#get`

[10] `https://github.com/mdeff/fma`