Generate Harmonically correct songs using LLMs

Sacha Binder, Julien Boudier, Charles Monte and Constantin Vaillant-Tenzer

March 28, 2025

Table of Contents

What is music

Structural elements:

- Melody;
- Rhythm;
- Harmony.

Interpretative elements:

- Nuance;
- Tempo;
- Pitch (tonality, base octave, base frequency (e.g. A4 at 442Hz).

Music is played by an instrument that has a spectrum signature.

Can LLMs do music ?

Yes ! Using MIDI notations - that is the standard in musical information retrieval (MIR).

Figure: Graphical representation of a MIDI track

There is also a specific format for guitar (GuitarPro).

Table of Contents

Making LLMs understand music

Inspired by the practical work "Teaching LLMs to learn addition", we decided to train a LLM with a custom tokenizer.

Figure: Our architecture

The tokenizer takes in entry a text sequence representing a music, and outputs its tokenized version. The LLM takes the tokenized version, and predicts the end of the music.

We have one problem : How to translate MIDI to a text representation ?

The solution : Simplified MIDI

Inspired by Moakher et al. 2024, we used simplified MIDI notation to tokenize MIDI sequences (Hadjeres and Crestel 2021):

$$(px : vy : dz : tw)$$

Where p means pitch, v velocity, d duration and t the time before the next sounds.

Each sound is now represented by a word. This allows us to translate a graphical representation in a digital interface to text understandable by LLMs. We hence ask the model to generate those MIDI sequences!

Tokenization: Why It Matters

- Language models operate on discrete tokens (integers).
- How we convert MIDI to tokens impacts:
  - Sequence length (memory, context window)
  - Model's grasp of musical structure
  - Vocabulary size (model size, training)
  - Handling of numerical values
- Need to balance representation power, efficiency, and vocabulary size.

Problem 1: Naive Tokenization (e.g., Character-Level)

- Treat `p50:v60:d100:t50` as characters: p, 5, 0, :, v, ...
- **Issues:**
  - ▸ **Very Long Sequences:** Many tokens per note.
  - ▸ **No Structure Awareness:** Model must learn p, 5, 0 means pitch 50. Inefficient.
  - ▸ **Poor Numerical Handling:** Numbers like 100 vs 101 share tokens but differ numerically.

Problem 2: Structured Tokens & Vocabulary Explosion

- **Better Idea:** Tokens for structure (`<note>`) + tokens for values (`<pitch_50>`, `<duration_1234>`).
- **The Catch: Vocabulary Explosion!**
  - Duration (`d`) and Time/Pause (`t`) can have very large values (thousands+).
  - A unique token for every tick value creates an enormous vocabulary (e.g., 20k+ tokens just for pauses).
  - Large vocabularies -> Large embedding layers, slower training, potential sparsity.

Our Solution: Binned Quadruplet Tokenizer

- Addresses vocabulary explosion and leverages note structure.
- **Core Idea 1: Quadruplet Structure**
    - Use a `<note>` token to mark the start of a note event.
    - Expect value tokens in a fixed order: Pitch, Volume, Duration, Time.
- **Core Idea 2: Binning / Quantization**
    - Group large ranges (duration, time) into a smaller, fixed number of bins (e.g., 2048, 4096).
    - Create tokens representing the *bin index* (e.g., `<duration_bin_10>`).
    - Use logarithmic spacing: finer bins for short values, coarser for long values.

Binned Quadruplet Tokenizer: How it Works

- **Special Tokens:** `<pad>`, `<s>`, `</s>`, `<unk>`
- **Structure Token:** `<note>` (marks start of a 4-parameter note)
- **Value Tokens (Pitch/Volume):** Exact value tokens (range 0-127).
  - e.g., `<pitch_0>`... `<pitch_127>`
  - e.g., `<volume_0>`... `<volume_127>`
- **Value Tokens (Duration/Time):** Binned value tokens (fixed number, e.g., 2048-4096).
  - e.g., `<duration_bin_0>`... `<duration_bin_127>`
  - e.g., `<time_bin_0>`... `<time_bin_255>`
- **Fixed Order after** `<note>`**:** Pitch Token → Volume Token → Duration Bin Token → Time Bin Token.

Binned Quadruplet Tokenizer: Advantages

- **Controlled Vocabulary Size:** Binning prevents explosion. Size determined by chosen bins (e.g., 1k tokens).
- **Structure Awareness:** Explicit <note> token + fixed order reinforces the 4-parameter structure.
- **Compact Sequences:** Shorter than character-level and numerical-with-labels approaches. (5 tokens/note).
- **Balances Precision and Efficiency:** Control duration/time precision via the number of bins.

Trade-off: Loss of exact precision for binned values (duration, time). Model learns relationships between bins/ranges.

Table of Contents

Dataset

We used Moakher et al. 2024's translation of the GiantMIDIPiano dataset to Simplified MIDI as a base. It exists in different context length, we adapted the 4000-character context version:

- It contains natural language instructions : we only kept the Simplified MIDI part.
- We limited our training to 40k rows.

Data Preprocessing Steps

1 Filter by Max Pause Time ('t'):
- Define a threshold (e.g., 10k ticks).
- Remove examples containing any 't' value > threshold.

2 Calculate Ranges: Determine min/max duration and time from the filtered dataset to inform tokenizer bin boundaries.

3 Tokenize: Apply the tokenizer to the filtered dataset.

Model Architecture & Training

- **Model:** GPT-2 (Transformer decoder) from scratch using Hugging Face 'transformers'.
  - ▸ `vocab_size`: From tokenizer ~6k4.
  - ▸ `n_embd`: Embedding dimension 384.
  - ▸ `n_layer`: Transformer layers 6.
  - ▸ `n_head`: Attention heads 6.
  - ▸ `n_positions`: Max sequence length 512.
- **Training:** Using 'Trainer' API.
- **Top-p sampling** (p=0.95, T=0.9): Avoid the model getting stuck in a loop at inference.

Model evaluation - Music theory

For the evaluation, we give to each music a grade normalized to be out of 20. The notation for the harmonic criterion is inspired by Lerdahl and Jackendoff 1996 and the weights for the intervals directly taken from the study of Sethares 1993.

We consider harmony and rhythm, but not velocity, on which the LLM does not play much and where style have very different conceptions.

Model evaluation - harmonic theory

For each harmony (characterised by the same time t), the harmony grade is: or a chord C consisting of pitches $p_1, p_2, \ldots, p_n$, the harmonic score is defined as:

$$H(C) = \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} h(p_i, p_j)$$

where $h(p_i, p_j)$ is a pre-defined consonance function of the interval between pitches p_i and p_j. Consonant intervals (e.g., octaves, fifths) have scores close to 1, while dissonant intervals (e.g., seconds, tritones) have scores close to 0.

Model evaluation - rythmic theory

We evaluate duration consistency using a musical reference duration T_{ref}:

$$R_D = e^{-\alpha \cdot \min_j \left| \log_2 \left(\frac{d}{T_{\text{ref}}} \right) - j \right|}$$

Similarly, temporal intervals between notes are evaluated by:

$$R_T = e^{-\beta \cdot \min_j \left| \log_2 \left(\frac{t}{T_{\text{ref}}} \right) - j \right|}$$

where again $j \in \mathbb{Z}$ and $\beta > 0$ controls sensitivity. Time intervals close to rhythmic multiples of two of $T_{\text{ref}} = \text{GCD}\,(d_i, t_i)_{i \in [\![1,N]\!]}$ achieve higher scores.

Music evaluation - Quantitative Results

After only 3 epochs, we already achieve really satisfactory performance with our model.

	Model trained for 3 epochs	Model trained for 3 + 6 epochs
Total score	**18.23**	17.78
Harmonic score	**9.64**	9.59
Duration score	**3.96**	3.79
Time score	**4.63**	4.40

Table: Results of our model for two training lengths

- Re-training the model for 6 epochs after this does not improve the results for our metrics, it even decreases the performance.
- Too much training causes the model to generate a lot more text not following the pattern, making the generation nearly **12 times slower** for only 10 song generations

Music evaluation - Quantitative Results

We decided to focus on the model trained for 3 epochs and analyse how robust the model was regarding our metrics.
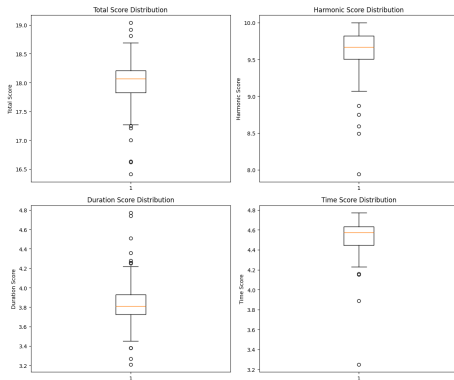
Figure: Boxplot graphs for all the metrics with 100 generations using the model trained for 3 epochs

Music evaluation - Qualitative Results

By listening to the generations having the best scores in a certain category, we can understand the influence of each metric :

- Best total score : 19.04/20 | Play |

- Best harmonic score : 10/10 | Play |

- Best duration score : 4.77/5 | Play |

- Best time score : 4.77/5 | Play |

Music evaluation - Qualitative Results

However, the metric still has problems, it is overall too high as seen before and is a bit far to our qualitative understanding :

- This generation is valued at 18.46 total score $\boxed{\text{Play}}$

- This generation is valued at 18.41 total score $\boxed{\text{Play}}$

They both have really close scores, while we could argue the first sound is one of the best we listened to and the second one of the worst.

Music evaluation - Qualitative Results

There is no distinguishable pattern in the worst generations, as seen with the top 3 worst :

- This generation is valued at 16.63 total score, and sounds like the best one
  Play

- This generation is valued at 16.62 total score   Play

- This generation is valued at 16.41 total score   Play

Table of Contents

Limitations of our evaluation method

- The music harmonic notation relies on perfect consonance and is well suited for pop music, but needs to be adapted to the style (our notation would destroy Ravel!);
- The music notation is only local - need more compute and larger outputs to test something more music wide;
- Try an inferential approach for evaluation (Temperley 2007) - very good for Bach.

Improvements to our naive method : Moakher et al. 2024

The paper on which our idea was based achieved better results for different reasons:

- Using a better LLM (LLaMa 2);
- Using methods to reduce VRAM usage to have easier training : QLoRA, quantization, Unsloth;
- Using more efficient positional embedding (RoPE);
- Improving inference performance with Nucleus Sampling.

Table of Contents

Online resources

- Llama2 Fine-Tuning for Music Generation: `https://github.com/fegounna/LLM-Fine-Tuning-for-Music-Generation/tree/master`;
- Song generation using audiocraft (not training well): `https://github.com/CodeName-Detective/Prompt-to-Song-Generation-using-Large-Language-Models/`;
- SongComposer: A Large Language Model for Lyric and Melody Composition in Song Generation `https://arxiv.org/abs/2402.17645`;
- An Automatic Chord Progression Generator Based On Reinforcement Learning `https://ieeexplore.ieee.org/document/8554901`;
- Make a melody from harmonic input: `https://ezmonyi.github.io/ChatMusician/`.

Our code is available here: `https://github.com/cvt8/song_llm`.

References I

📄 Hadjeres, Gaëtan and Léopold Crestel (2021). "The piano inpainting application". In: *arXiv preprint arXiv:2107.05944.*

📄 Lerdahl, Fred and Ray S Jackendoff (1996). *A Generative Theory of Tonal Music, reissue, with a new preface.* MIT press.

📄 Moakher, Yessin et al. (2024). "Fine-Tuning de Modèles de Langage pour la Génération Musicale". In: URL: https://github.com/fegounna/LLM-Fine-Tuning-for-Music-Generation/.

📄 Sethares, William A (1993). "Local consonance and the relationship between timbre and scale". In: *The Journal of the Acoustical Society of America* 94.3, pp. 1218–1228.

📄 Temperley, David (2007). *Music and probability.* Mit Press.

Questions

Table of Contents

Harmonic weights used for our evaluation

- Octave ($p_i - p_j \equiv 0[12]$): D=0
- Fifth (±7mod 12): D=0.1
- Third major/minor (±3,4,8,9mod 12): D=0.2
- Second, seventh major (±1,2,10,11mod 12): D=0.6
- Triton (±6mod 12): D=0.8