Research project mid-term report
January 2025

Bass guitar tablature conditional generation for guitar accompaniment in western popular music

Student:
Olivier ANOUFA
Data Science Master 2

Supervisors:
Alexandre D'HOOGE
Ken DÉGUERNEL
Algomus team, CRIStAL

Abstract — The field of symbolic music generation has seen great advancements with the rise of transformer-based architectures. Addressing a specific need identified through a user study, we focus on developing AI tools to generate bass guitar tablatures conditioned on scores of other instruments in western popular music. The bass guitar, a vital component of the rhythmic and harmonic sections in music, presents a unique challenge due to its dual role in providing structure and depth.

Building upon the tablature notation, which simplifies the interpretation of music for stringed instruments, this work adopts modern encoding schemes to integrate tablature representation into transformer-based models. To this end, the project involves preprocessing a large dataset of music scores, and fine-tuning state-of-the-art transformer architectures. The generated tablatures will then be evaluated both numerically and qualitatively, with feedback from musicians.

Key-words — music information retrieval, conditional generation, guitar accompaniment;

1 Introduction

Natural language processing methods for the generation of symbolic music have experienced significant advancements in recent years. The transformer architecture, introduced by Vaswani et al. in 2017, has been used to generate scores for various instruments, in diverse styles and genres[1], [2]. An unpublished user study conducted by Bacot et al. showed a potential need for accompaniment generation tools for guitarists. The questionnaire was answered by 31 guitarist-composers, and 7 of them followed up with an interview. During the interviews, several guitarists answered that they would like to be able to generate bass guitar lines and drum parts without requiring familiarity with the instruments. Indeed, guitarists often resort to writing basic bass lines to accompany their compositions, and an AI tool could perform this functional task for them[3].

This need is the starting point for this project, proposed by the Algomus team, part of the CRIStAL laboratory. We focus on the conditional aspect of symbolic music generation, for an instrument that has not been thoroughly studied yet: the bass guitar. Specifically, the goal is to generate bass guitar tablatures given other instruments' scores, in the context of western popular music. Our objective is to try several combinations of conditioning instruments and to evaluate the quality of the generated tablatures both numerically and with the help of musicians.

To better understand what is at stake in this challenge, we will begin by precisely defining the role of the bass guitar in the context of western popular music. Since the human ear perceives low-frequency pulses more distinctly, the bass guitar is considered part of the rhythmic section of the band (together with the drums)[4]. However, the bass guitar also performs a harmonic — and sometimes melodic — role in the music, sustaining the lead instruments and adding depth to the composition. Its adaptability makes the bass guitar a crucial component across various genres of western popular music.

Historically, tablatures have been used since the Renaissance period as a simplified notation system for string instruments like the lute. Originating around the 16th century, tablatures were an intuitive alternative to staff notation, allowing musicians to bypass the complexity of interpreting pitches. This system explicitly linked symbols to physical actions on the instrument, such as pressing specific frets or strings. As a result, tablatures gained popularity among amateur and professional musicians alike. In modern times, tools like GuitarPro have digitalized tablatures, further extending their use for learning, practice, and composition in genres ranging from classical to popular music[5].

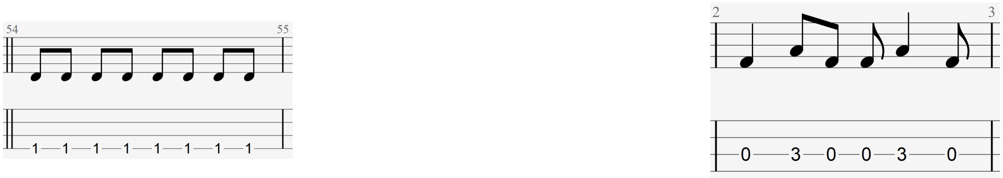


Figure 1: Rhythmic (left) and melodic (right) extracts in Time is Running Out by Muse

Figure 1 shows a tablature and score extract of both a rhythmic and a more melodic bass. Scores display the notes to play, while tablatures show the fingering to use on the instrument. More precisely, each line of the tablature represents a string of the instrument, and the numbers indicate the fret to press on the string. For instance, in the figure on the right, the bassist will start by playing the fourth string with the first fret pressed. Tablatures do not contain rhythmic information, which is why they are generally combined with scores. Otherwise, the musician must know the song to play it correctly.

Generating bass guitar tablatures presents several challenges. At a high level, we first need to scrap and preprocess large datasets of music scores. Then, we need to design a computational representation of music that is adapted to the task of generating bass guitar tablatures. That is, a way to encode music scores in a meaningful way for the transformer architecture we will use. Concerning the generation, we will start by leveraging state-of-the-art models but will adapt and tune them to the task at hand.

2 State of the art

Adapting the transformer architecture — originally developed for text processing — to symbolic music presents many challenges. Symbolic music datasets, unlike text corpus, are limited in both size and diversity, posing challenges for training robust models[2]. Tokenization must be tailored to represent pitch, duration, and dynamics, while attention mechanisms require adaptation to capture the hierarchical and temporal structure of music. Finally, data cleaning and preprocessing steps are critical to standardize music scores and ensure the compatibility with sequence models.

2.1 Data availability

Symbolic music datasets are a cornerstone for training deep learning models, yet their availability and quality significantly vary across domains. In music composition and generation, datasets are often limited in size and diversity, especially when compared to text or image datasets. For bass guitar tablatures, the challenge is even more pronounced due to the niche nature of the instrument and the focus on other instruments in existing datasets.

The MIDI standard has dominated symbolic music datasets for decades, allowing the development of resources like the Lakh MIDI dataset and MAESTRO dataset. However, while these datasets offer general-purpose symbolic music, they lack the specificity required for tasks involving tablatures or instrument-specific representation. The GuitarSet dataset, for example, focuses on acoustic guitar transcription but does not provide sufficient symbolic information for bass guitar. Similarly, the DadaGP dataset addresses the need for multi-instrument symbolic music data in tablature format, but its emphasis is on rock and metal genres, which limits the diversity of bass guitar styles[5].

Bass guitar data, especially in tablature format, suffers from a lack of standardization and availability. Tablature files are often stored in private formats like GuitarPro or as non-standardized text files, making it challenging to preprocess them for machine learning tasks[5]. Moreover, rhythm and dynamics, critical elements in bass guitar playing, are frequently absent in publicly available tablatures, complicating their utility in generative tasks.

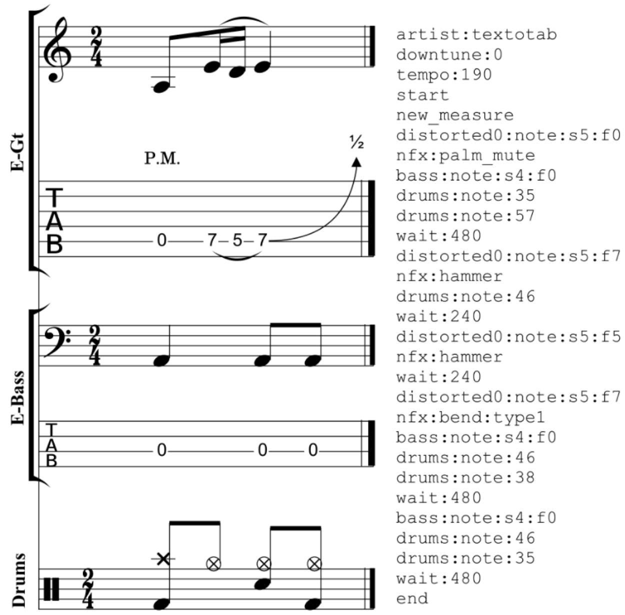
Efforts like DadaGP illustrate the potential of GuitarPro files repositoriess to create symbolic datasets that include bass guitar parts. Moreover DadaGP also provides a tokenized format inspired by MIDI encodings, offering a foundation for training sequence models. On the other hand, pre-trained models on broader datasets like MAESTRO or Lakh MIDI can be fine-tuned on smaller datasets, reducing the dependency on large volumes of task-specific data[5], [6]. For our project, we use the DadaGP dataset. Its very specific tokenization will be discussed in the next section.

2.2 Tokenization

Tokenization in the context of deep learning music generation has been discussed by several previous works[5], [6], [7], [8], [9].

Tokenization in music involves converting complex musical content into a sequence of elements that can be processed computationally. Much like tokenization in natural language processing (NLP), it breaks down music into manageable units, though in music, the focus is on musical features like pitch, duration, and velocity. In symbolic music information retrieval (MIR), tokenization strategies are generally classified into two categories: time-slice-based and event-based. Time-slice-based tokenization divides music into fixed-time intervals, such as 16th notes, which can be represented in formats like piano rolls or multi-hot vectors, capturing simultaneous notes at each time slice. Event-based tokenization, on the other hand, focuses on specific musical events, such as a note being played or a measure starting, often using formats like MIDI, which encode music as a series of events. This approach can involve elementary tokens, which represent individual features like pitch or duration, or composite tokens that aggregate multiple features into one token, providing a more compact representation. Notably, tokenization strategies like REMI (Revamped MIDI-derived events) and MIDI-like tokenization allow for consistent representation of rhythmic and pitch elements while addressing complexities like polyphony and multiple tracks in multi-instrument music[2], [9].

The DadaGP tokenization format adopts an event-based approach, similar to other music generation models, by representing musical events as discrete tokens. It uses a Python script with PyGuitarPro to convert GuitarPro files into a tokenized sequence, beginning with metadata tokens such as artist, downtune, tempo, and start. Pitched instrument notes are encoded with a combination of tokens representing instrument, note, string, and fret, while rests are denoted with a separate token structure. For drumset sounds, a specific tokenization scheme using MIDI percussion maps is employed, where each drum sound is represented by a unique token (e.g., drums:note:36 for a kick drum). The system also uses wait tokens to separate notes and rests, enabling the model to infer note durations without the need for explicit note-off tokens. This approach ensures that new notes automatically silence previous ones, except in cases where ghost notes or let-ring effects are involved. Additionally, the tokenization format records changes in musical structure, such as new measures, tempo shifts, and note effects, with each change being represented as a distinct token. However, the tokenization does not encode dynamics or articulations. These are supposed to be inferred by the user after generation.



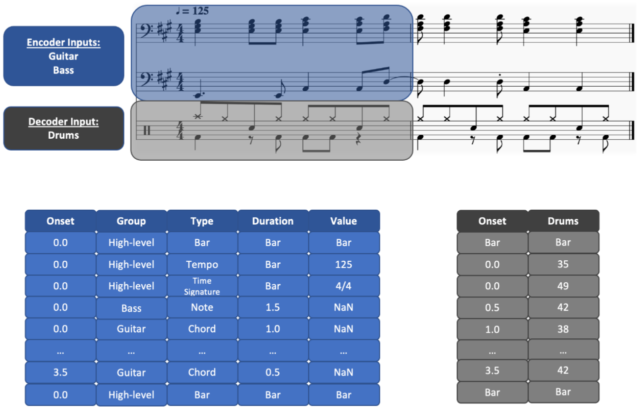
```
artist:textotab
downtune:0
tempo:190
start
new_measure
distorted0:note:s5:f0
nfx:palm_mute
bass:note:s4:f0
drums:note:35
drums:note:57
wait:480
distorted0:note:s5:f7
nfx:hammer
drums:note:46
wait:240
distorted0:note:s5:f5
nfx:hammer
wait:240
distorted0:note:s5:f7
nfx:bend:type1
bass:note:s4:f0
drums:note:46
drums:note:38
wait:480
bass:note:s4:f0
drums:note:46
drums:note:35
wait:480
end
```

Figure 2: Example measure with a distorted guitar, bass and drums in GuitarPro's graphical user interface (left), and its conversion into token format (right). Figure taken from [5]

Figure 2 illustrates the tokenization process for a measure in a GuitarPro file, showing the conversion of a musical event into a tokenized sequence. We notice the three first tokens are the metadata tokens. "start" token marks the beginning of the song. Afterwards, each measure is announced by a "new_measure" token. In between, the instruments' notes are encoded using the tablature notation for the guitars. For instance the first note played here is distorted0:note:s5:f0 (string 5, fret 0). It is followed by a nfx token "nfx:palm_mute", which is a note effect token that applies to the previous note. Finally, the duration of the note is given by the wait token "wait:480" (480 ticks in GuitarPro correspond to an eighth note). If no tokens of a specific instrument are present between two wait tokens, it means that the instrument keeps on playing the same note. This is the case of the bass in the example. Its first quarter note lasts for 960 ticks, therefore we have to look after the three first wait tokens (480 + 240 + 240) before seeing a new token for the bass.

As we plan to use the model built by Makris et al. in 2022, we will also consider the tokenization they used. The proposed tokenization model introduces a compound word (CP) representation, inspired by previous CP-based approaches[8]. Unlike traditional tokenization strategies such as MIDI-like or REMI, which encode music as a linear sequence of tokens, CP groups multiple features of a musical event into a single "super token" or "word". This n-dimensional representation significantly reduces sequence length and improves computational efficiency in Transformer-based architectures[6].

In this model, the CP representation is adapted for an Encoder-Decoder architecture to condition drum generation on accompanying rhythm tracks, such as bass and guitar. The Encoder processes a 5-dimensional CP representation that includes rhythm track features and high-level musical information like tempo and time signature, which are known to influence the complexity and density of drum tracks. Both the Encoder and Decoder adopt bar-based segmentation, consistent with existing CP and REMI representations. By combining rhythm section inputs and high-level parameters, this tokenization approach aligns with the fundamental musical principles of contemporary western music's rhythm section[6]. As said previously, our work will use the DadaGP tokenization. However, the nature of the generation task we want to perform is very similar to the one of Makris et al. in the sense that bass guitar and drums are part of the same rhythm section.



Onset	Group	Type	Duration	Value
0.0	High-level	Bar	Bar	Bar
0.0	High-level	Tempo	Bar	125
0.0	High-level	Time Signature	Bar	4/4
0.0	Bass	Note	1.5	NaN
0.0	Guitar	Chord	1.0	NaN
...	...	...	...	...
3.5	Guitar	Chord	0.5	NaN
0.0	High-level	Bar	Bar	Bar

Onset	Drums
Bar	Bar
0.0	35
0.0	49
0.5	42
1.0	38
...	...
3.5	42
Bar	Bar

Figure 3: Example of a CP representation used for training. Figure taken from [6]

Figure 3 shows an example of a CP representation used for training in the model developed by Makris et al. We notice that the rhythmic section, composed of a rhythmic guitar and a bass (in blue on the figure), constitutes the conditioning part of the model. Their representation as tokens encode the duration and type of the notes played, but not the pitch of the notes as it is not relevant for the drum generation task. In grey, the drum part's encoding excludes all information about velocities and duration. Drums notes and are only represented by their onset in the bar and the drum component played.

2.3 Conditional generation models

Our first idea was to use the GuitarCTRL model developed by Sarmento et al. in 2021[10]. The GTR CTRL model utilizes a Transformer-XL architecture[11], which improves on the vanilla Transformer by introducing recurrence and modified positional encodings, enabling long-term dependency learning. This model includes two key control types: instrumentation (inst-CTRL) and genre (genre-CTRL). For instrumentation, tokens marking the start and end of each instrument are inserted into the header, guiding the model to generate music for specified instruments, such as bass and drums or distorted guitar and drums. Genre control tokens are similarly placed at the beginning of the sequence, with a dataset covering over 200 songs per genre, including Metal, Rock, Punk, Folk, and Classical. The model uses various prompts at inference, ranging from full-prompt (including two measures plus the genre token) to empty-prompt (only the genre token). This model was used to generate a baseline result for our project, conditioning the generation to bass guitar only.

The model we wish to use for our project is the one developed by Makris et al. in 2022[6]. It employs an Encoder-Decoder architecture with a BiLSTM Encoder and a Transformer Decoder utilizing Relative Global Attention. The Encoder is made of several BiLSTM layers. It processes Compound Representation (CP) inputs to generate a latent variable z that encodes high-level features of the input and conditional tracks. BiLSTM (Bidirectional Long Short-Term Memory) were first introduced by Graves et al. in 2005[12]. It is a type of recurrent neural network (RNN) that processes data in both forward and backward directions. This allows the model to consider both past (previous time steps) and future (upcoming time steps) context in the sequence The Decoder consists of self-attention layers and feed-forward layers. It combines z with previous timestep embeddings to predict drum onsets and pitches.

The model uses input sizes that are different from the ones we may use with our tokenization. However we are interested in the ability of the model to generate conditionnally throughout the whole sequence. In comparison, GTR CTRL only conditions the generation at the beginning of the sequence. One of the issue we encountered when generating bass guitar was that other instruments ended up being generated further in the sequence. To avoid this we tried setting the probability of predicting tokens from any other instruments to 0, but even then the bass lines generated were not coherent. This model is simply not suited for our task which explains our motivations to adapt the BiLSTM model.

3 Data preprocessing

4 Workplan

References

[1] A. Vaswani *et al.*, *Attention Is All You Need*, arXiv:1706.03762, Aug. 2023. Accessed: Oct. 15, 2024. [Online]. Available: http://arxiv.org/abs/1706.03762.

[2] D.-V.-T. Le, L. Bigo, M. Keller, and D. Herremans, *Natural Language Processing Methods for Symbolic Music Generation and Information Retrieval: A Survey*, en, arXiv:2402.17467 [cs, eess], Feb. 2024. Accessed: Oct. 6, 2024. [Online]. Available: http://arxiv.org/abs/2402.17467.

[3] B. Bacot, L. Bigo, and B. Navarret, "Tablature software and popular music composition: A user study and perspectives on creative algorithmic tools," 2025.

[4] M. J. Hove, C. Marie, I. C. Bruce, and L. J. Trainor, "Superior time perception for lower musical pitch explains why bass-ranged instruments lay down musical rhythms," *Proceedings of the National Academy of Sciences*, vol. 111, no. 28, pp. 10 383–10 388, Jul. 2014, Publisher: Proceedings of the National Academy of Sciences. DOI: 10.1073/pnas.1402039111. Accessed: Jan. 21, 2025. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1402039111.

[5] P. Sarmento, A. Kumar, C. J. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, *DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models*, en, arXiv:2107.14653 [cs, eess], Jul. 2021. Accessed: Oct. 6, 2024. [Online]. Available: http://arxiv.org/abs/2107.14653.

[6] D. Makris, G. Zixun, M. Kaliakatsos-Papakostas, and D. Herremans, *Conditional Drums Generation using Compound Word Representations*, en, arXiv:2202.04464 [cs, eess], Feb. 2022. Accessed: Oct. 9, 2024. [Online]. Available: http://arxiv.org/abs/2202.04464.

[7] M. Agarwal, C. Wang, and G. Richard, *Structure-informed Positional Encoding for Music Generation*, en, arXiv:2402.13301 [cs, eess], Feb. 2024. Accessed: Oct. 9, 2024. [Online]. Available: http://arxiv.org/abs/2402.13301.

[8] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 178–186, May 2021, Number: 1, ISSN: 2374-3468. DOI: 10.1609/aaai.v35i1.16091. Accessed: Jan. 21, 2025. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16091.

[9] J. Cournut, L. Bigo, M. Giraud, and N. Martin, "Encodages de tablatures pour l'analyse de musique pour guitare," in *Journées d'Informatique Musicale (JIM 2020)*, Strasbourg (en ligne), France, 2020. Accessed: Oct. 6, 2024. [Online]. Available: https://hal.science/hal-02934382.

[10] P. Sarmento, A. Kumar, Y.-H. Chen, C. Carr, Z. Zukowski, and M. Barthet, "GTR-CTRL: Instrument and Genre Conditioning for Guitar-Focused Music Generation with Transformers," en, in *Artificial Intelligence in Music, Sound, Art and Design*, C. Johnson, N. Rodríguez-Fernández, and S. M. Rebelo, Eds., Cham: Springer Nature Switzerland, 2023, pp. 260–275, ISBN: 978-3-031-29956-8. DOI: 10.1007/978-3-031-29956-8_17.

[11] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. DOI: 10.18653/v1/P19-1285. Accessed: Oct. 22, 2024. [Online]. Available: https://aclanthology.org/P19-1285.

[12] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," en, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, Montreal, Que., Canada: IEEE, 2005, pp. 2047–2052, ISBN: 978-0-7803-9048-5. DOI: 10.1109/IJCNN.2005.1556215. Accessed: Oct. 15, 2024. [Online]. Available: http://ieeexplore.ieee.org/document/1556215/.