DADAGP: A DATASET OF TOKENIZED GUITARPRO SONGS FOR SEQUENCE MODELS

Pedro Sarmento[1] **Adarsh Kumar**[2,3] **CJ Carr**[4]
Zack Zukowski[4] **Mathieu Barthet**[1] **Yi-Hsuan Yang**[2,5]
[1] Queen Mary University of London , UK [2] Academia Sinica, Taiwan
[3] Indian Institute of Technology Kharagpur, India [4] Dadabots [5] Taiwan AI Labs
{p.p.sarmento, m.barthet}@qmul.ac.uk, emperorcj@gmail.com, yhyang@ailabs.tw

ABSTRACT

Originating in the Renaissance and burgeoning in the digital era, tablatures are a commonly used music notation system which provides explicit representations of instrument fingerings rather than pitches. GuitarPro has established itself as a widely used tablature format and software enabling musicians to edit and share songs for musical practice, learning, and composition. In this work, we present DadaGP, a new symbolic music dataset comprising 26,181 song scores in the GuitarPro format covering 739 musical genres, along with an accompanying tokenized format well-suited for generative sequence models such as the Transformer. The tokenized format is inspired by event-based MIDI encodings, often used in symbolic music generation models. The dataset is released with an encoder/decoder which converts GuitarPro files to tokens and back. We present results of a use case in which DadaGP is used to train a Transformer-based model to generate new songs in GuitarPro format. We discuss other relevant use cases for the dataset (guitar-bass transcription, music style transfer and artist/genre classification) as well as ethical implications. DadaGP opens up the possibility to train GuitarPro score generators, fine-tune models on custom data, create new styles of music, AI-powered songwriting apps, and human-AI improvisation.

1. INTRODUCTION

Historically, tablatures' proliferation is closely linked to the lute repertoire, compositions that roughly span from the 16th century onwards, and are still available today [1]. In opposition to standard notational practices (usually referred to as staff notation), in a tablature system for string instruments each staff line on the score represents a string of the instrument, substituting a representation of pitch by a given location on said instrument (i.e. a fingering) [2]. Tablatures are a *prescriptive* type of notation, where the emphasis is on the action (symbol-to-action), contrary to *descriptive* forms of notation, which establishes a symbol-to-pitch relationship. This characteristic makes tablatures an intuitive and inclusive device for music reading and learning, which can explain their large prevalence for music score sharing over the Internet [3,4]. Often represented as non-standardised text files that require no specific software to read or write, tablatures' online dissemination has surpassed more sophisticated music notation formats, such as Music XML or MIDI [3]. However, tablature representations that rely solely on text have limitations from a user perspective. For example, it is common that rhythm indications are discarded, preventing a comprehensive transcription of the music and automatic playback. Tablature edition software (e.g. GuitarPro [1] , PowerTab [2] , TuxGuitar [3]) can be regarded as a solution for this problem, keeping the *prescriptive* approach, and supporting rhythm notations and playback. By supporting the annotation of multiple instruments, as observable in Figure 1, these tools account for an interactive music experience, either for songwriting or music learning purposes.

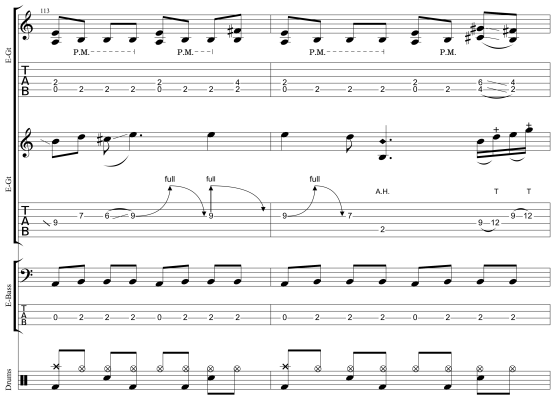

Figure 1. An excerpt from a GuitarPro song notation using tablatures and score for two guitars, bass and drums.

The release of this dataset intends to leverage the GuitarPro format used by the before-mentioned software to support guitar and bands/ensembles' related research within the MIR community, focusing specifically on the

[1] https://www.guitar-pro.com/
[2] http://www.power-tab.net/guitar.php
[3] https://sourceforge.net/projects/tuxguitar/

task of symbolic music generation. The contributions of this paper are: (1) a dataset of over 25,000 songs in GuitarPro and token format, together with statistics on its features and metadata, (2) an algorithm and Python software to convert between any GuitarPro file and a dedicated token format suitable for sequence models [4] , (3) results from its main use case, the task of symbolic music generation, and (4) a discussion about further applications for DadaGP and its ethical implications.

In this paper, we first present some relevant background concerning previously released music datasets in symbolic format. In Section 3, we discuss advantages of tab-based datasets for MIR research. We then describe, in Section 4, the details of the DadaGP dataset, its encoder/decoder support tool, the features it encompasses and the ones it lacks. Within Section 5 we present a use case of symbolic music generation using our proposed dataset, supported by previous approaches concerning databases of symbolic music. Section 6 proposes additional applications for the dataset. Finally, in Section 7 we explain the steps needed in order to acquire the dataset, further pointing out some ethical considerations in Section 8.

2. BACKGROUND

Since its release in 1983, the MIDI (Music Instrument Digital Inferfaces) standard has remained highly ubiquitous. Unsurprisingly, MIDI has been the most recurrent option in terms of musical notation formats, concerning datasets released within the MIR community, either targeting music generation purposes, that lately have boomed by leveraging deep learning approaches, or aiming for musical analysis, musicology or purely information retrieval ends. A comprehensive overview of previously released datasets in symbolic format is presented in [5]. The authors present MusPy, a toolkit for symbolic music generation, that natively supports a total of eleven datasets. Considering cumulative song duration, the top five datasets are the Lakh MIDI dataset [6], the MAESTRO dataset [7], the Wikifonia Lead Sheet dataset [5] , the Essen Folk Song database [8], and the NES Music database [9]. With respect to music notation formats, these datasets employ MIDI, MusicXML and ABC. Recently, the GiantMIDI-Piano dataset [10], comprising 10,854 unique piano solo pieces, the POP909 dataset [11] and the Ailabs.tw Pop1K7 dataset [12], containing respectively piano arrangements of 909 and 1,748 popular songs, were also released, all relying on MIDI format. This standardisation around MIDI is useful for there are several Python libraries to work with this format, such as music21 [13], mido [14], pretty_midi [15], and jSymbolic [16].

Regarding guitar-oriented research, previous dataset releases have not particularly targeted automatic music generation goals, instead focusing on guitar transcription or playing technique detection. The GuitarSet consists of 360 excerpts of acoustic guitar along with annotations for string and fret positions, chords and beats [17]. Furthermore, the Guitar Playing Techniques dataset [18] contains 6,580 clips of notes together with playing technique annotations. Likewise, the IDMT-SMT-Guitar dataset [19] also comprises short excerpts that include annotations of single notes, playing techniques, note clusters, and chords. Lately, Chen et al. compiled a dataset of 333 tablatures of fingerstyle guitar, created specifically for the purpose of music generation [20].

To the authors best knowledge, there exists no multi-instrument dataset that is able to combine the ease of use of symbolic formats whilst providing guitar (and bass) playing technique information. Such expressive information is lacking in other formats, and GuitarPro appears as a viable resource for music analysis and generation.

3. MOTIVATIONS: WHY GUITARPRO?

GuitarPro is both a software and a file format, widely used by guitar and bass players, but also by bands. It is mostly utilized for tasks such as music learning and practicing, where musicians simply read or play along a given song, and for music notation, in which composers/bands use the software to either support the songwriting process, or simply as a means for ease of distribution once compositions are done. As an example of the software's widespread dissemination, the tablature site Ultimate Guitar [6] hosts a catalogue of over 200,000 user-submitted GuitarPro files, containing notations of commercial music, mostly from the genres of rock and metal. One of the main motivations for the creation of DadaGP is to engage the MIR community into research that leverages the expressive information, instrumental parts and song diversity in formats such as GuitarPro. Although GuitarPro is a paid software, free alternatives such as TuxGuitar are capable of editing/exporting into GuitarPro format. Moreover, GuitarPro files can be easily imported into MuseScore [7] , a free software notoriously known for music notation, which also possesses tablature features. However, using MuseScore might present some occasional incompatibilities, specifically those regarding the selection of instruments (e.g. drums are often imported as piano, and the corresponding MIDI instruments need to be manually switched). Another important motivation for the release of this dataset is that it is possible to make conversions between GuitarPro and MIDI files. This can be done inside any of the aforementioned software, by simply exporting into MIDI, or by scripting. Thus, by converting the dataset's GuitarPro files into MIDI, MIDI-based music feature extraction functions available (e.g. Python libraries referenced in Section 2) can be applied. Finally, we believe that our dataset is able to provide researchers with the information present in standard MIDI datasets, while including at the same time prescriptive information useful for guitar-oriented research.

[4] Available at: `https://github.com/dada-bots/dadaGP`
[5] No longer available.
[6] `https://www.ultimate-guitar.com/`
[7] `https://musescore.com/`

4. DADAGP DATASET

Leveraging the proliferation of music transcriptions available online as GuitarPro files, we compiled DadaGP, a dataset containing 26,181 songs. We also devised an encoding/decoding tool to convert GuitarPro files into tokens, which is described in Section 4.1. In total, it contains 116M tokens, which is about the size of WikiText-103 [21]. In terms of duration, the dataset amounts to over than 1,200 hours (average song length of 2:45 minutes).

4.1 Encoding/Decoding Tool

4.1.1 Feature Extraction with PyGuitarPro

PyGuitarPro [22] is a Python library which reads, writes and manipulates GuitarPro files [8] . Our encoding/decoding tool explores its feature extraction functions, in order to convert much of the information into a tokenized text format. With PyGuitarPro it is possible to acquire information regarding music-theoretic features (e.g. pitch, rhythm, measure, instrument) and playing technique information.

4.1.2 Tokenization

The token format takes inspiration from event-based MIDI encodings used in previous music generation works, such as MuseNet [23], REMI [24] and CP [12]. The tool consists of a Python script that utilizes PyGuitarPro to process GuitarPro files into/from token format. Syntactically, every song begins with `artist`, `downtune`, `tempo` and `start` tokens. A depiction of the conversion process can be seen in Figure 2. Notes from pitched instruments are represented by a combination of tokens in the format of `instrument:note:string:fret` and rests by `instrument:note:rest`. For the drumset, the representation is done by `drums:note:type`, leveraging GuitarPro 5 percussion MIDI maps (e.g. `drums:note:36` for a kick drum, `drums:note:40` for a snare). Every note or rest is separated in time by `wait` tokens. This is sufficient for the decoder to figure out note durations. There is no need to use note-off tokens, because new notes silence old notes, unless a *ghost note* or *let ring* effect is used. Every new measure, note effect, beat effect, and tempo change is registered as a token. Effect tokens are applied to the preceding note token. A histogram containing the most common tokens in DadaGP is available in Figure 4(g).

Furthermore, the DadaGP token format is resilient to syntax errors, such that random token sequences will still produce decodable music. We believe this is helpful when creatively pushing generators to make out-of-distribution sequences using high temperatures, early epochs, extreme latent dimension values, interpolated style conditioning, and other experimental practices.

4.2 Repertoire

Each song is labelled with artist and genre information, although genre tags are absent within original GuitarPro files. To this end, we compiled a genre list, with information acquired from the Spotify Web API [9] , querying by artist and song title, resulting in genre metadata for each composition. It is worth mentioning that a given song can have more than one genre attached to it. Information about the most prevalent genres within DadaGP can be seen in Figure 3. While its emphasis is on genres and sub-genres from rock and metal, its corpus is diverse, also including stylistically distinct genres such as jazz, classical, pop and EDM. From Figure 4(a) we observe that most of the songs in DadaGP contain four instrumental parts, usually two guitars, a bass and drums.

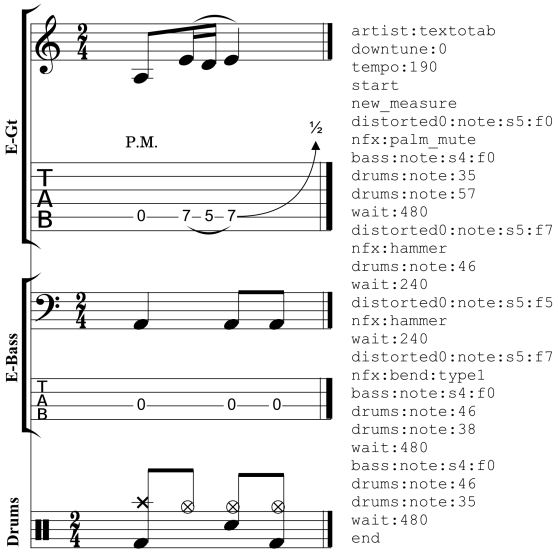

Figure 2. A measure with a distorted guitar, bass and drums in GuitarPro's graphical user interface (left), and its conversion into token format (right).

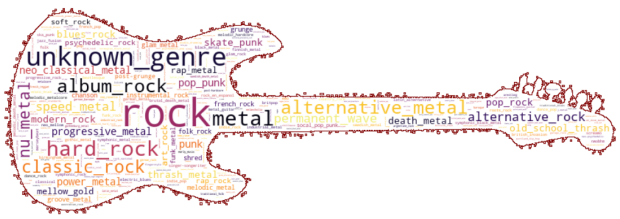

Figure 3. Word cloud representation of the musical genres in DadaGP. Tag size increases with amount of songs.

4.3 Instruments

Regarding instrumentation, for DadaGP a maximum of nine instruments were chosen: three distorted or overdriven guitars, two clean or acoustic guitars, one bass, one drumset, one lead (for instruments with sharp attacks, e.g. piano), and one pad (for instruments used more ambiently, like a choir or a string ensemble). Multiple drum tracks are combined into one. Rare instruments are combined into

[8] Currently, it supports GP3, GP4 and GP5 files.

[9] Available at: `https://developer.spotify.com/documentation/web-api/`

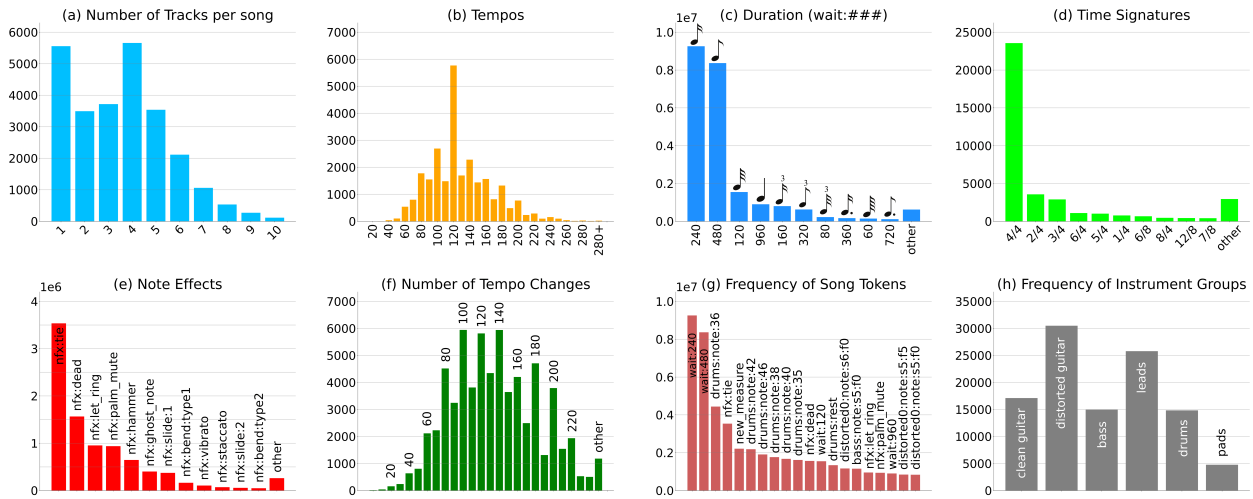

Figure 4. Statistical information about the DadaGP dataset. Histograms of tracks per song (a), initial tempos (b), most common note durations in token and staff notation format (c), time signatures (d), note effects (e), amount of tempo changes (f), most frequent tokens (g) and instruments (h).

the lead and pad tracks. In Figure 4(h) we can notice a predominance of distorted guitars in the dataset. Intuitively this is justified by the presence of two distorted guitars (often one rhythmic and one lead) on most of the songs in DadaGP, due to the predominance of the rock/metal genre. Concerning guitar and bass, 7 string guitars are supported, as are 5 and 6 string basses. Downtuning is supported only if all instruments downtune the same amount, and common tunings such as *Drop D* [10] and *Drop AD* are also included. Rare tunings were dropped from the dataset as the encoder does not support them.

Guitar playing technique notations are represented by note effect tokens (`nfx`), although this family of tokens also holds information about other instruments (e.g. `nfx:tie`, which acts as a link between two adjacent notes). On Figure 4(e) we present a histogram of the most frequent occurrences of these in our dataset, namely *palm mute* (a technique often used with distortion guitars where the guitar player dampens the strings with the right hand palm), bends and vibratos, slides, hammer-ons and pull-offs (both under `nfx:hammer`).

4.4 Meter

As clarified before, each note/rest event is followed by a `wait` token which specifies the number of ticks between it and the succeeding event. In DadaGP, tick resolution uniformly corresponds to 960 ticks per quarter note. For a tempo of 100 bpm, a tick corresponds to $60/(100*960) = 0.000625$ seconds. Referring to the excerpt in Figure 2, eighth note events are separated by `wait:480` tokens, and sixteenth note ones by `wait:240`. A histogram with the most common durations in DadaGP is presented in Figure 4(c), in both token and standard staff notation formats, to ease visualization.

[10] A tuning in which only the lowest string is downtuned by one whole step, usually from E to D.

Usually, in a GuitarPro file a default tempo is specified for the entire song, although it supports the inclusion of tempo changes throughout the piece. This is addressed by our encoder/decoder with the tokens `tempo` and `bfx:tempo_change` respectively, which affects note/rest duration. In Figure 4(b) and Figure 4(f) are presented plots corresponding to the most frequent tempos and tempo changes.

The encoder/decoder also supports the representation of measure repetitions with the `measure:repeat` token. Although time signatures are not tokenized, they are inferred by summing the `wait` tokens between the occurrences of `new_measure`. However, this method is insufficient to distinguish between $3/4$ and $6/8$ measures, for example. To circumvent this, for the plot presented in Figure 4(d) we leveraged PyGuitarPro functions to extract accurate information about the most prevalent time signatures for each measure in our dataset.

4.5 What is Missing?

Information regarding key signature is not provided as part of the dataset. Although key signature can be represented in GuitarPro format, it is rarely present within files. Similarly to the results presented in [6] for the Lakh MIDI dataset, 93.7% of the songs in DadaGP were automatically assigned the key of C Major, rendering these statistics inaccurate.

GuitarPro does not include note velocity information as in MIDI. However, in GuitarPro loudness between notes and musical phrases is notated using traditional dynamic instructions (e.g. *forte*, *pianissimo*, *mezzo-forte*). In its token format, DadaGP does not yet support this, but there is a possibility of accentuating notes at two levels with `nfx:accentuated_note` and `nfx:heavy_accentuated_note`.

Concerning vocals, a common practice with GuitarPro files is to select MIDI wind instruments to notate singing

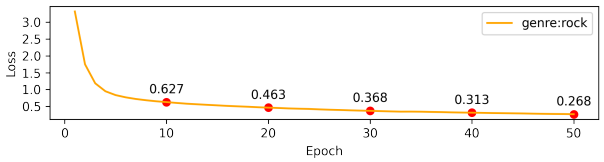

Figure 5. Training loss of the rock subset model, per epoch.

melodies. Currently, our dataset is not well-suited to handle vocals, for these get converted into the `leads` instrument, which may also contain information about other instruments, such as the piano. Lyrics are also possible to include in GuitarPro, but that feature is currently not supported by our encoder/decoder tool.

5. USE CASE: SYMBOLIC MUSIC GENERATION

Recently, the field of symbolic music generation has witnessed consistent progress. Considering works that target symbolic music generation with Transformer-based models, MusicTransformer [25] is a MIDI generator trained on piano performances with improved long-term coherence over vanilla RNNs due to the use of the Transformer [26]. Similarly, MuseNet [23] is a generative Sparse Transformer [27] trained on a larger dataset of MIDI including over 600 styles. An API for the model was launched by OpenAI, which powers the songwriting app MuseTree [28]. However, the model was not released, so it cannot be fine-tuned on custom data. In [29] the author trained a charRNN generator on dozens of GuitarPro songs encoded as a sequence of ASCII characters. It only supported one instrument, and its verbose character-sequence format opened up the possibility for syntax errors.

We tested the DadaGP dataset for a symbolic music generation use case by using the Pop Music Transformer model [24], in which the authors devised a Transformer-XL [30] architecture to generate pop piano compositions in MIDI format. The reason for the choice of this architecture is because this work considers metrical structure in the input data, allowing for an increased awareness in terms of beat-measure structure. We chose the Transformer-XL model as it is able to learn dependencies that are 450% longer than vanilla Transformers, thus well-suited for our task. As per the settings, similarly to the original paper, we used $M = 8$ attention heads and $N = 12$ self-attention layers.

As a proof-of-concept, we collected a subset from our dataset, retrieving 6,910 songs labelled as `genre:rock`. We generated a list of all the unique tokens in this subset, creating a vocabulary with 2,104 entries.

Training was set to run for 50 epochs. With around 43M parameters, this model took around 10 days to perform this task on a Tesla K80 GPU. We consider this to be impractical in terms of reproducibility, so we intend to release pretrained models from epochs 40 and 50, for which losses can be seen in Figure 5.

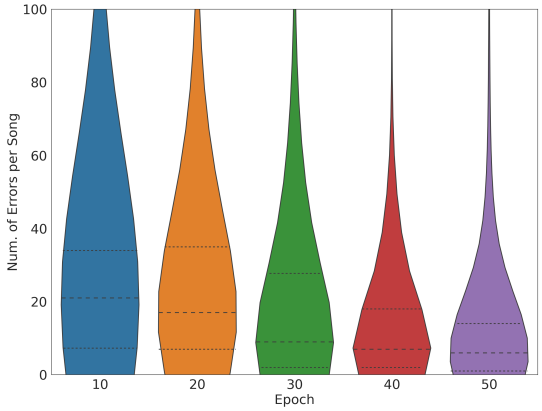

Figure 6. Violin plot of number of errors per song at different epochs.

Regarding inference, we conditioned the model by prompting it with an initial list of tokens comprising `artist`, `downtune`, `tempo` and `start`, necessary for the DadaGP decoder. Furthermore, in an attempt to guide the model towards the generation of music comprising specific instruments, we included tokens for a single note of a distorted guitar, bass guitar and drums. Through experimentation, we set on a limit of 1,024 tokens for each generated song, using 1.2 as temperature parameter. Finally, we manually appended an `end` token in order for the decoder to be able to convert it to GuitarPro format, as this is the instruction which tells the decoder when the song finishes.

As a simple evaluation metric, we focused on the notion of *grammar errors*, namely repetitions of the tokens that should only occur once (`artist:`, `downtune:`, `tempo:`, `start` and `end`), or adjacent repetitions of the same token. Using this, we estimated the number of errors per song, for a corpus of 1,000 generated songs from the model at epochs 10, 20, 30, 40 and 50. As observable in Figure 6, not only the median of the number of errors per song is smaller in later epochs, but also the occurrence of outliers is diminished, as expected.

Despite the limitations of the current evaluation, it allowed us to notice a predominance of a specific error, namely the repetition of the token `end`. This is problematic, because the decoder immediately stops the conversion when an `end` token appears, ultimately shortening songs when in GuitarPro format. To counter this effect, we devised a condition that, during inference, would force the model to sample a different token in the event that an `end` token is selected. Results of generated songs without any curation or post-processing have been made available [11] .

6. PROSPECTIVE APPLICATIONS

Although primarily tailored for symbolic music generation, below we describe further applications for DadaGP.

[11] Available at: `https://drive.google.com/drive/folders/1USNH8olG9uy6vodslM3iXInBT725zult?usp=sharing`

6.1 Guitar-Bass Transcription

The task of guitar-bass transcription from audio recordings is still mostly done manually by musicians, requiring expertise and being both effort and time consuming. In order to automate this, previous research has focused on both solo bass guitar [31, 32] and solo guitar [33–35] transcription. As a contribution to solve this problem, we anticipate that DadaGP can be used to create a synthetic dataset for training guitar-bass transcription models, by rendering its corpus from tablatures into audio, using a DAW and appropriate sound fonts. Such a synthetic dataset can be used to pre-train a model, which can then be fine-tuned afterwards using realistic sounds with aligned scores. This argument is supported by the promising results shown by the Slakh dataset, a synthesized version of the Lakh MIDI dataset, on the task of music source separation [36].

6.2 Music Style Transfer

Recently, the task of style transfer, the process of changing the style of an image, video, audio clip or musical piece so as to match the style of a given example, has been the subject of much attention. First investigated in applications that target computer vision, music style transfer has recently shown promising results in both the audio [37] and symbolic domains [38–40]. As a prospective application of DadaGP, we envisage that genre information can be leveraged in segregating the dataset across different genres, rendering it suitable for the task of musical genre style transfer, as proposed in [41] for the specific morphing between Bach chorales and Western folk tunes. Furthermore, besides musical genre, artistic information can also be used towards the task of composer style transfer, once again by filtering DadaGP across distinct artists.

6.3 Artist/Genre Classification

Another task for which artistic and musical genre information present in DadaGP is useful is artist/genre classification. We hypothesize that these features can be used to train classification models, in order to predict composer style and genre related information from the symbolic representation of the songs itself, similarly to what has been implemented in [42–44]. A thorough survey of the most important approaches regarding music genre classification in the symbolic domain can be consulted in [45]. Furthermore, there is a symbiosis between this task and the one present in the previous subsection, since the models trained for artist/genre classification can be prospectively used in composer style-based feature extractions, which can be further utilized in tasks like composer style conditioned generation and music style transfer.

7. DISTRIBUTION

To ensure reproducibility and facilitate the usage of the dataset, we allow researchers to access DadaGP from a Zenodo repository [12], on application by request. Here we include the token format versions of the songs, the encoder/decoder Python script in order to convert them into/from GuitarPro format, and the statistical data presented on this paper.

[12] `https://zenodo.org`

8. ETHICAL CONSIDERATIONS

Training large models has a carbon footprint. Some cloud services are carbon neutral, others are not. This should be considered when training large models on this data. Releasing pre-trained models reduces impact, and we intend to do so with the models present in this paper.

Many questions regarding production and consumption of music created with AI are still unanswered. For example: Is it wrong to train machine learning models on copyrighted music? Should this be protected by fair use for artists and scientists? What about commercial use? How to acknowledge, reward and remunerate artists whose music has been used to train models? What if an artist does not want to be part of a dataset? Should creators have a monopoly on their style and exclude others from using their style? Or is style communal? Some of these questions were also raised upon the release of Jukebox [46], an audio model trained on more than 7,000 artists. However, OpenAI made the case that "*Under current law, training AI systems constitutes fair use (...)*" and that "*Legal uncertainty on the copyright implications of training AI systems imposes substantial costs on AI developers and so should be authoritatively resolved*" [47].

9. CONCLUSION AND FUTURE WORK

In this paper we presented DadaGP, a dataset of songs in GuitarPro and token formats, together with its encoding/decoding tool. We discussed the features, strengths and weaknesses of the dataset. Moreover, we presented a symbolic music generation use case entailing a novel approach for multi-instrument music generation in tablature format. Finally, we pointed out additional research applications for DadaGP and discussed some ethical implications. We intend to improve the DadaGP dataset, namely the possibility of removing `measure:repeat` tokens. During generation, we discovered that these tokens were often hard for the model to interpret, sometimes leading to disproportionate measure repetitions. Also, we plan to include note and phrase dynamics information, and the support for vocal instrumental parts. Regarding music generation, we envision to (1) release a pre-trained model which can be fine-tuned on new music, (2) collaborate with artists that use GuitarPro, (3) explore genre/style transfer, (4) and attempt to play the generated songs in social performances.

10. ACKNOWLEDGMENTS

This work is supported by the EPSRC UKRI Centre for Doctoral Training in Artificial Intelligence and Music (Grant no. EP/S022694/1). Thanks to Colin Raffel, Brian McFee, and Sviatoslav Abakumov for discussions and advice.

11. REFERENCES

[1] R. De Valk, R. Ahmed, and T. Crawford, "JosquIntab: A Dataset for Content-based Computational Analysis of Music in Lute Tablature," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, 2019.

[2] T. Magnusson, *Sonic Writing: Technologies of Material, Symbolic & Signal Inscriptions*. Bloomsbury Academic, 2019.

[3] R. Macrae and S. Dixon, "Guitar Tab Mining, Analysis and Ranking," in *Proc. of the 12th International Society for Music Information Retrieval Conference*, 2011.

[4] M. Barthet, A. Anglade, G. Fazekas, S. Kolozali, and R. Macrae, "Music Recommendation for Music Learning: Hotttabs, a Multimedia Guitar Tutor," in *Workshop on Music Recommendation and Discovery*, 2011, pp. 7–13.

[5] H. W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "MusPY: A Toolkit for Symbolic Music Generation," in *Proc. of the 21st International Society for Music Information Retrieval, ISMIR*, 2020.

[6] C. Raffel and D. P. W. Ellis, "Extracting Ground Truth Information from MIDI Files: A MIDIfesto," in *Proc. of the 17th International Society for Music Information Retrieval Conference*, 2016.

[7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. Anna Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck Google Brain, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," 2019.

[8] "Essen Folk Song Database." [Online]. Available: http://www.esac-data.org/

[9] C. Donahue, H. H. Mao, and J. Mcauley, "The NES music database: A Multi-Instrumental Dataset with Expressive Performance Attributes," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, 2018.

[10] Q. Kong, B. Li, J. Chen, and Y. Wang, "GiantMIDI-Piano: A Large-Scale MIDI Dataset for Classical Piano music," in *Transactions of the International Society for Music Information Retrieval*, 2020.

[11] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "POP909: A Pop-Song Dataset for Music Arrangement Generation," in *Proc. of 21st International Conference on Music Information Retrieval*, 2020.

[12] 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," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2021.

[13] M. S. Cuthbert and C. Ariza, "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," in *Proc. of the 11th International Society for Music Information Retrieval Conference*, 2010.

[14] O. M. Bjørndalen, "Mido: Midi objects for python." [Online]. Available: https://github.com/mido/mido

[15] C. Raffel and D. P. W. Ellis, "Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi," in *Late-Breaking Demos of the 15th International Society for Music Information Retrieval Conference*, 2014.

[16] C. Mckay and I. Fujinaga, "jSymbolic: A Feature Extractor for MIDI Files," in *Proc. of the International Computer Music Conference*, 2006.

[17] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A Dataset for Guitar Transcription," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, 2018.

[18] L. Su, L.-F. Yu, and Y.-H. Yang, "Sparse Cepstral, Phase Codes for Guitar Playing Technique Classification." in *Proc. of the 15th International Society for Music Information Retrieval Conference*, 2014.

[19] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score and Instrument-related Parameters," in *Proc. of the 17th Int. Conference on Digital Audio Effects*, 2014.

[20] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic Composition of Guitar Tabs by Transformers and Groove Modelling," in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020.

[21] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer Sentinel Mixture Models," *Proc. of the 5th International Conference on Learning Representations*, 2016.

[22] S. Abalumov, "PyGuitarPro." [Online]. Available: https://github.com/Perlence/PyGuitarPro

[23] C. Payne, "Musenet," 2019. [Online]. Available: openai.com/blog/musenet

[24] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proc. of the 28th ACM International Conference on Multimedia*, 2020.

[25] C.-Z. Anna Huang and A. M. Vaswani Jakob Uszkoreit Noam Shazeer Ian Simon Curtis Hawthorne Andrew Dai Matthew D Hoffman Monica Dinculescu Douglas Eck Google Brain, "Music Transformer: Generating Music with Long-term Structure," in *Proc. of the 7th International Conference on Learning Representations*, 2019.

[26] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. of the 31st Conference on Neural Information Processing Systems*, 2017.

[27] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[28] S. Waterman, "Musetree," 2019. [Online]. Available: https://stevenwaterman.uk/musetree/

[29] M. Moocarme, "Deep learning metallica with recurrent neural networks," 2016. [Online]. Available: https://www.mattmoocar.me/blog/tabPlayer/

[30] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. 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*, 2019.

[31] J. Abeßer and G. Schuller, "Instrument-centered music transcription of solo bass guitar recordings," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, 2017, pp. 1741–1750.

[32] J. Abeßer, S. Balke, K. Frieler, M. Pfleiderer, and M. Müller, "Deep Learning for Jazz Walking Bass Transcription," in *Proc. of the AES International Conference on Semantic Audio*, 2017.

[33] A. Wiggins and Y. E. Kim, "Guitar Tablature Estimation with a Convolutional Neural Network," in *Proc. International Conference on Music Information Retrieval*, 2019, pp. 284–291.

[34] S. Rodríguez, E. Gómez, and H. Cuesta, "Automatic transcription of Flamenco guitar falsetas," in *Proc. International Workshop on Folk Music Analysis*, 2018.

[35] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang, "TENT: Technique-embedded Note Tracking for Real-World Guitar Solo Recordings," in *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, 2019, p. 15–28.

[36] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and Quantity," in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2019.

[37] Y.-N. Hung, I.-T. Chiang, Y.-A. Chen, and Y.-H. Yang, "Musical Composition Style Transfer via Disentangled Timbre Representations," in *Proc. of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4697–4703.

[38] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, "Symbolic Music Genre Transfer with CycleGAN," in *Proc. of the IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 786–793.

[39] O. Cífka, U. Şimşekli, and G. Richard, "Groove2Groove: One-Shot Music Style Transfer With Supervision From Synthetic Data," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, 2020, pp. 2638–2650.

[40] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[41] Y.-Q. Lim, C. S. Chan, and F. Y. Loo, "Style-Conditioned Music Generation," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[42] T. J. Tsai and K. Ji, "Composer Style Classification of Piano Sheet Music Images Using Language Model Pretraining," in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020.

[43] S. Kim, H. Lee, S. Park, J. Lee, and K. Choi, "Deep Composer Classification Using Symbolic Representation," in *Late-Breaking Demo Session of the 21st International Society for Music Information Retrieval Conference*, 2020.

[44] A. Kotsifakos, E. E. Kotsifakos, P. Papapetrou, and V. Athitsos, "Genre Classification of Symbolic Music with SMBGT," in *Proc. of the 6th International Conference on PErvasive Technologies Related to Assistive Environments*. New York, NY, USA: Association for Computing Machinery, 2013.

[45] D. C. Corrêa and F. A. Rodrigues, "A Survey on Symbolic Data-based Music Genre Classification," *Expert Systems with Applications*, vol. 60, pp. 190–210, 2016.

[46] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A Generative Model for Music," 2020. [Online]. Available: https://github.com/openai/jukebox.

[47] OpenAI, "USPTO Comment Regarding Request for Comments on Intellectual Property Protection for Artificial Intelligence Innovation," 2019. [Online]. Available: https://www.uspto.gov/sites/default/files/documents/OpenAI_RFC-84-FR-58141.pdf