AUTOMATIC COMPOSITION OF GUITAR TABS BY TRANSFORMERS AND GROOVE MODELING

Yu-Hua Chen[1,2,3], **Yu-Hsiang Huang**[1], **Wen-Yi Hsiao**[1], **and Yi-Hsuan Yang**[1,2]
[1] Taiwan AI Labs, Taiwan, [2] Academia Sinica, Taiwan, [3] National Taiwan University, Taiwan
r08946011@ntu.edu.tw, {yshuang,wayne391,yhyang}@ailabs.tw

ABSTRACT

Deep learning algorithms are increasingly developed for learning to compose music in the form of MIDI files. However, whether such algorithms work well for composing guitar tabs, which are quite different from MIDIs, remain relatively unexplored. To address this, we build a model for composing fingerstyle guitar tabs with Transformer-XL, a neural sequence model architecture. With this model, we investigate the following research questions. First, whether the neural net generates note sequences with meaningful note-string combinations, which is important for the guitar but not other instruments such as the piano. Second, whether it generates compositions with coherent rhythmic groove, crucial for fingerstyle guitar music. And, finally, how pleasant the composed music is in comparison to real, human-made compositions. Our work provides preliminary empirical evidence of the promise of deep learning for tab composition, and suggests areas for future study.

1. INTRODUCTION

Thanks to the cumulative efforts in the community, in recent years we have seen great progress in using deep learning models for automatic music composition [8]. An important body of research has been invested on creating piano compositions, or more generally keyboard style music. For instance, the "Music Transformer" presented by Huang *et al.* [19] employs 172 hours of piano performances to learn to compose classical piano music. Another group of researchers extends that model to generate pop piano compositions from 48 hours of human-performed piano covers [20]. They both use a MIDI-derived representation of music and describe music as a sequence of event tokens such as `NOTE-ON` and `NOTE-VELOCITY`. While the MIDI format works the best for representing keyboard instruments and less for other instruments (for reasons described below), Donahue *et al.* [14] and Payne [31] show respectively that it is possible for machines to learn from a set of MIDI files to compose multi-instrument music.

There are, however, many other forms of musical notation that are quite different from the staff notation assumed by keyboard music. For example, the tabulature, or "tab" for short, is a notation format that indicates instrument fingering rather than musical pitches. It is common for fretted stringed instruments such as the guitar and ukulele, and free reed aerophones such as the harmonica. It makes more sense for people playing such instruments to read the tabs, as they suggest how to move the fingers.

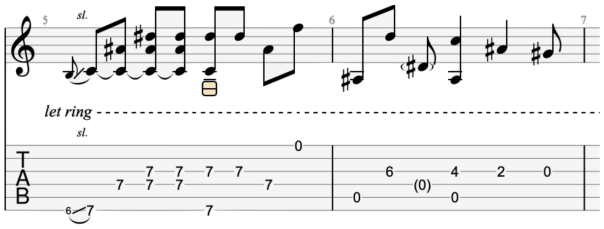


Figure 1. An example of fingerstyle guitar tab composed by human, along with the corresponding staff notation.

As shown in Figure 1, a tab contains information such as the fingering configuration on the fretboard (six strings for the case of the guitar) as well as usage of the left-hand or right-hand playing techniques. Such information is usually missing in the corresponding staff notation and MIDI files. Learning to automatically compose guitar music directly from MIDI files, though possible, has the limitation of ignoring the way people play these instruments. However, to our best knowledge, little has been done to use tabs to train a deep generative model.

To investigate the applicability of modern deep learning architectures for composing tabs, we compile a new dataset of 333 TAB files of "fingerstyle guitar" (including originally fingerstyle guitar music and fingerstyle adaptation) [3], and modify the data representation of the Music Transformer [19] to make the extended model learn to compose guitar tabs. With this model, we aim to answer three research questions (RQs):

- Whether the neural network learns to generate not only the note sequences but also the fingering of the notes to be played on a fretboard, from reading only the tabs (instead of, for example, watching videos demonstrating how people play the guitar)?
- Whether the neural network generates compositions with coherent "groove," or the use of rhythmic patterns over time [13, 32, 39]? It is generally assumed that the layers of a neural network learn abstractions of data on their own to perform the intended

task, e.g., to predict the next events given the history. However, in music, groove is usually indirectly implied according to the arrangement of notes along the time axis, instead of explicitly specified in either a MIDI or TAB file. Therefore, it remains to be studied whether the model can do better if it has to explicitly handle bar-level `GROOVING` events, inserted into the training data as a high-level information in some way, or if such a modification is not needed. This is in particular relevant in the context of fingerstyle composition, as in fingerstyle a guitarist has to take care of the melody, chord comping, bass line and rhythm simultaneously [3].

- Finally, how the compositions generated by the neural network compare with human-composed guitar tabs, when both rendered into audio waveforms and presented to human listeners? This gives us a direct evaluation of the effectiveness of the neural network in modeling guitar music.

We provide audio rendition of examples of the generated tabs (using a guitar synthesizer of a DAW called Ample Sound [1]) at `https://ss12f32v.github.io/Guitar-Transformer-Demo/`, along with a video recording of a guitarist playing a generated tab.

In what follows, we review some related work in Section 2, and then present the tab dataset in Section 3. After that, we describe in Section 4 the methodology for modeling and learning to compose guitar tabs. We present the result of objective and subjective evaluations addressing the aforementioned research questions in Section 5.

2. RELATED WORK

2.1 Guitar-related Research in MIR

In the music information retrieval (MIR) community, research concerning guitar is often related to automatic guitar transcription [5, 7, 9, 16, 18, 21, 22, 27, 34, 46] and playing technique detection [4, 10, 37, 38]. For example, Su *et al.* [38] built a convolution neural network (CNN) model for detecting the playing techniques associated with the string-pressing hand, and incorporated that for transcribing audio recordings of unaccompanied electric guitar performances. Rodríguez *et al.* [34] presented a model for transcribing Flamenco guitar falsetas, and Abeßer and Schuller [5] dealt with the transcription of solo bass guitar recordings. We note that, while automatic transcription concerns with recovering the tab underlying an audio guitar performance, our work deals with automatic composition of original guitar tabs in the symbolic domain, and therefore does not consider audio signals.

As there are multiple fret positions to play the same note on a guitar, it may not be easy for a novice guitar learner to play a guitar song without the corresponding tab. Automatic suggestion of the fingering given a human-made "lead sheet," a symbolic format that specifies the melody and chord sequence but not their fingering, has therefore been a subject of research. Existing work has explored the use of hidden Markov models, genetic algorithm, and neural networks to predict the fingering by examining its playing difficulty for a guitarist, viewing the task as an optimal path finding problem [6, 28, 35, 40]. While such prior arts can be considered as performing a MIDI-to-TAB conversion, our work aims to model TABs directly.

Xi *et al.* developed the GuitarSet [45], a set of 360 audio recordings of a guitar equipped with the hexaphonic pickup. The special pickup is able to capture the sound from each string individually, making it possible for a model to learn to perform multipitch estimation and tabulature fingering arrangement at the same time. Using the dataset, Wiggins and Kim [43] built such a model with CNN, achieving 0.83 F-score (i.e., the harmonic average of precision and recall) for multipitch estimation, and 0.90 for identifying the string-fret combinations of the notes. While the dataset is relevant for guitar transcription, its recordings are all around 12–16 bars in length only, which seems to be too short for deep generative modeling.

McVicar *et al.* [24–26] used to build sophisticated probabilistic systems to algorithmically compose rhythm and lead guitar tabs from an input chord and key sequence. Our work differs from theirs in that we aim to build a general-purpose tab composition model using modern deep generative networks. An extra complexity of our work is that we experiment with fingerstyle guitar, a type of performance that can be accomplished by a single guitarist.

2.2 Transformer Models for Automatic Composition

The Transformer [41] is a deep learning model that is designed to handle ordered sequences of data, such as natural language. It models a word sequence $(w_1, w_2, \dots w_T)$ seen in the training data by factorizing the joint probability into a product of conditionals, namely, $P(w_1) \cdot P(w_2|w_1) \cdot \dots \cdot P(w_T|w_1, \dots, w_{T-1})$. During the training process, the model optimizes its parameters so as to correctly predict the next word w_t given its preceding history $(w_1, w_2, \dots w_{t-1})$, for each position t in a sequence.

Following some recent work on recurrent neural network (RNN)-based automatic music composition [29, 42], Huang *et al.* [19] viewed music as a language and for the first time employed the Transformer architecture for modeling music. Given a collection of MIDI performances, they converted each MIDI file to a time-ordered sequence of musical "events," so as to model the joint probability of *events* as if they are *words* in natural language (see Section 4.1 for details of such events). The Transformer with relative attention was shown to greatly outperform an RNN-based model, called PerformanceRNN [29], in a subjective listening test [19], inspiring the use of Transformer-like architectures, such as Transformer or Transformer-XL [12], in follow-up research [11, 14, 20, 31, 44]. [1]

There are lots of approaches to automatic music composition, deep learning- and non-deep learning based included [8, 15, 30]. We choose to consider only the Transformer architecture here, to study whether we can translate its strong result in modeling MIDIs to modeling TABs.

[1] We note that it is debatable whether music and language are related. We therefore envision that some other new architectures people will come up with in the future might do a much better job than Transformers in modeling music. This is, however, beyond the scope of the current work.

	# tabs	# bars	# bars per tab	# events per tab
training	303	24,381	80±41	5,394±3,116
validation	30	2,593	74±35	5,244±3,183

Table 1. Statistics of the dataset; the last two columns show the mean and standard deviation values across each set. Please see Table 2 for definitions of the events.

3. FINGERSTYLE GUITAR TAB DATASET

There have been some large-scale MIDI datasets out there, such as the Lakh MIDI dataset [33] and BitMidi [2]. The former, for example, contains 176,581 unique MIDI files of *full songs*. In contrast, existing datasets of tabs are usually smaller and shorter, as they are mainly designed for learning the mapping between tabs and audio (i.e., for transcription research), rather than for generative modeling of the structure of tabs. The tabs in the GuitarSet [45], for example, are performances of *short excerpts of songs*, typically 12–16 bars in length, which are not long.

For the purpose of this research, we compile a guitar tab dataset on our own, focusing on the specific genre of fingerstyle guitar. Specifically, we collect digital TABs of *full songs*, to facilitate language modeling of guitar tabs. We go through all the collected TABs one-by-one and filter out those that are of low quality (e.g., with wrong fingering, obvious annotation errors), or are not fingerstyle (e.g., have more than one tracks). We also discard TABs that are not in standard tuning, to avoid inconsistent mapping between notes and fingering. As shown in Table 1, this leads to a collection of 333 TABs, each with around 80 bars. This includes TABs of famous professional fingerstyle players such as Tommy Emmanuel and Sungha Jung. All the TABs are in 4/4 time signature, and they can be in various keys. We reserve 30 TABs for validation and performance evaluation, and use the rest for training.

Please note that, similar to the MIDI files available in Lakh MIDI [33], the TAB files we collect do not contain *performance* information such as expressive variations in dynamics (i.e., note velocity) and micro-timing [23, 29]. To increase velocity variation, we use Ample Sound [1] to add velocity to each note by its humanization feature. We do not deal with micro-timing in this work.

3.1 Fingerstyle

It is interesting to focus on only fingerstyle guitar in the context of this work, as we opt for validating the effectiveness of Transformers for single-track TABs first, before moving to modeling multi-track performances that involve at least a guitar (e.g., a rock song). We give a brief introduction of fingerstyle guitar below.

Fingerstyle [3] is at first a term that describes using fingertips or fingernails to pluck the strings to play the guitar. Nowadays, the term is often used to describe an arrangement method to blend multiple parts of musical elements or tracks, which are initially played by several instruments, into the composition of one guitar track. Therefore, a guitarist playing fingerstyle has to simultaneously take care of the melody line, bass line, chord comping and the rhythmic groove. Groove, in particular, is important in fingerstyle, as it is now only possible to work on the rhythmic flow of music with a single guitar and the use of the two hands. We hence pay special attention to groove modeling in this work (see Section 4.3).

category/type	description
`NOTE-ON`	45 different pitches (E2–C6)
`NOTE-DURATION`	multiples of the 32th note (1–64)
`NOTE-VELOCITY`	note velocity as 32 levels (1–32)
`POSITION`	temporal position within a bar; multiples of the 16th note (1–16)
`BAR`	marker of the transition of bars
`STRING`	6 strings on a tab
`FRET`	20 fret positions per string
`TECHNIQUE`	5 playing techniques: slap, press upstroke, downstroke, and hit-top
`GROOVING`	32 grooving patterns

Table 2. The list of events adopted for representing a tab as an event sequence. The first five are adapted from [19, 20], whereas the last four are tab-specific and are new. We have in total 45+64+32+16+1+6+20+5+32=231 unique events.

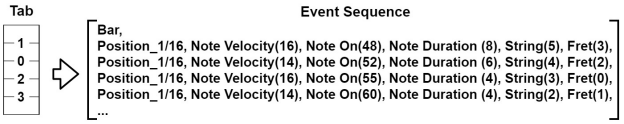


Figure 2. An example of the result of "TAB-to-event" conversion needed for modeling a tab as a sequence. Here, we show the resultant event representation of a C chord.

4. MODELING GUITAR TABS

In this section, we elaborate how we design an event representation for modeling guitar tabs, or more generally tabs of instruments played by string strumming.

4.1 Event Representation for MIDIs: A Quick Recap

In representing MIDIs as a sequence of "events," Huang *et al.* [20] considered, amongst others, the following event tokens. Each note is represented by a triplet of `NOTE-ON`, `NOTE-DURATION`, and `NOTE-VELOCITY` events, representing the MIDI note number, quantized duration as an integer multiple of a *minimum duration*, and discrete level of note dynamics, respectively. The minimum duration is set to the 32th note. The onset time of the notes, on the other hand, is marked (again after quantization) on a time grid with a specific *resolution*, which is set to the 16th note as in [19]. Specifically, to place the notes over the 16-th note time grid, they use a combination of `POSITION` and `BAR` events, indicating respectively the position of a note onset within a bar, among the 16 possible locations, and the beginning of a new bar as the music unfolds over time. This

event representation has been shown effective in modeling pop piano [20]. We note that the time grid outlined with this combination of POSITION and BAR events can also contribute to modeling the rhythm of fingerstyle guitar.

4.2 Event Representation for Tabs

To represent TABs, we propose to add, on top of the aforementioned five types of events for MIDIs,[2] the following three new types of fingering-related events: STRING, FRET, TECHNIQUE, and a type of rhythm-related events: GROOVING. We introduce the first three below, and the last in the next subsection. Table 2 lists all the events considered, whereas Figure 2 gives an example of how we represent a C chord with such an event representation.

We use the first 20 frets of the 6 strings in the collected TABs, i.e., each string can play 20 notes. The pitch range of the strings overlaps, so a guitarist can play the same pitch on different strings, with moderate but non-negligible difference in timbre. The fingering of the notes also affects playability [46]. In standard tuning, the strings can play 45 different pitches, from E2 to C6.

In our implementation, we adopt the straightforward approach to account for the various possible playing positions of the notes—to add STRING and FRET tokens right after the NOTE-ON tokens in the event sequence representing a tab. We note that the FRET tokens are actually *redundant*, in that the combination of NOTE-ON and STRING alone is sufficient to determine the fret position to use. However, in pilot studies we found the inclusion of FRET makes the model converges faster at the training time.

Specifically, instead of a 3-tuple representation of a note as the case in MIDIs, we use a 5-tuple note representation that consists of successive tokens of NOTE-VELOCITY, NOTE-ON, NOTE-DURATION, STRING and FRET for TABs. As such five tokens always occur one after another in the training sequences, it is easy for a Transformer not to miss any of them when generating a new NOTE-ON event at the inference time, according to our empirical observation of the behavior of the Transformers.

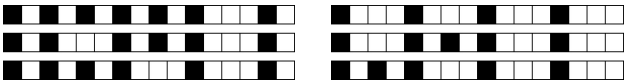
However, as we do not impose constraints on the association between NOTE-ON and STRING, it remains to be studied whether a Transformer can learn to compose tabs with reasonable note-string combinations. This is the subject of the **1st RQ** outlined in Section 1.

As for the TECHNIQUEs, we consider the following five right-hand techniques: slap, press, upstroke, downstroke, and hit-top, which account for $\sim$1% of the events in our training set. The inclusion of other techniques, such as sliding and bending, is left as a future work.

Similar to [19, 20], we consider the 16th note as the resolution of onset times, which is okay for 4/4 time signature. Increasing the resolution further to avoid quantization errors and to enhance expressivity is also left to the future.

4.3 Groove Modeling

Groove can be in general considered as a rhythmic feeling of a changing or repeated pattern, or "humans' pleasurable urge to move their bodies rhythmically in response to music" [36]. Unlike the note-related or time-related events, groove is usually *implicitly* implied as a result of the arrangement of note onsets over time, instead of be explicitly specified in either a MIDI or TAB file. Hence, it might be possible for a Transformer to learn to compose music with reasonable groove, without we *explicitly* inform it what groove is. We refer to this baseline variant of our Transformer as the **no grooving** version, which considers all the events listed in Table 2 but GROOVING.



(a) (b)

Figure 3. Samples of 16-dim hard grooving patterns assigned to 2 different clusters (a), (b) by $k$means clustering.

However, as a tab is now represented as a sequence of events, it is possible to add groove-related events to help the model make sense of this phenomenon. Since our event representation has the BAR events to mark the bar lines, we can ask the model to learn to generate a "bar-level" GROOVING event right after a BAR event, before proceeding to generate the actual content of the bar. Whether such a groove-aware approach benefits the quality of the generated tabs is the subject of our **2nd RQ**.

To implement such an approach, we need to come up with 1) a bar-level grooving representation of symbolic music, and 2) a method to convert the grooving representation, which might be a vector, to a discrete event token.

In this work, we represent groove by the *occurrence of note onset* over the 16-th time grid, leading to the following four grooving representations of music.

- **Hard grooving**: A 16-dim binary vector marking the presence of (at least one) onset per each 16 positions of a bar. A popular pattern in our dataset, for example, is $[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]$, meaning onsets on beats only.
- **Soft grooving**: A soft version that considers the number of onsets (but disregarding the velocity values) for each position, normalized by the maximum in the bar, leading to a 16-dim real-valued vector.
- **Multi-resolution hard (or soft) grooving**: Variants of the last two that additionally consider corresponding down-sampled 8-dim and 4-dim vectors to emphasize the beats (e.g., counting only the onsets on beats), and then concatenate the vectors together, yielding a 28-dim vector (i.e., 16+8+4).

To convert the aforementioned grooving patterns to events, a discretization is needed. Among various possible approaches, we experiment with the simplest idea of grouping the grooving patterns seen in the training set into a number of clusters. We can then use the ID of the cluster a grooving pattern is associated with for the GROOVING event of that grooving pattern. For simplicity, we employ the classic $k$means algorithm [17] here, setting k to 32. Please see Figure 3 for an example of the clustering result.

[2] Huang *et al.* [20] actually considered the Chord and Tempo events additionally; we found these two types of event less useful in modeling tabs, according to preliminary experiments.

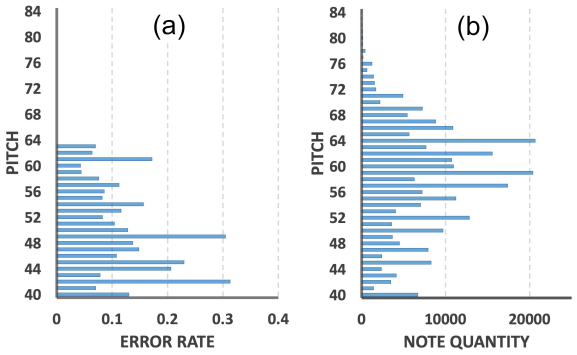


Figure 4. Distributions of (a) the error rate for each note in the string arrangement prediction of our model, and (b) the counts of each note in the training set.

	string (high-pitched ↔ low-pitched)					
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**
(a) accuracy	100%	99%	97%	94%	91%	90%
(b) pitch 42	∼0%	∼0%	10%	∼0%	27%	63%
(c) pitch 57	∼0%	6%	65%	26%	∼0%	∼0%
(d) pitch 69	85%	14%	∼0%	∼0%	∼0%	∼0%

Table 3. (a) The average accuracy of our model in associating each `STRING` with a `NOTE-ON`, broken down by string; (b–d) The string-relevant output probability estimated by our model for three different pitches.

4.4 Transformer-XL-based Architecture

Following [14, 20], we use the Transformer-XL [12] for the architecture of our model. Unlike the Transformer used in [19], the Transformer-XL gains a longer receptive filed with a segment-level recurrence mechanism, thereby seeing further into the history and benefiting from the extended memory. We base our implementation on the open source code of [20], adopting many of their settings. For example, we also set the sequence length and recurrence length to 512 events, and use 12 self-attention layers and 8 attention heads. The model has in total ∼41M trainable parameters. The training process converges within 12 hours on a single NVIDIA V100 GPU, with batch size 32.

5. EVALUATION

5.1 Experiment 1: On Fingering

The 1st RQ explores how a Transformer learns the association between notes and fingering, without human-assigned prior knowledge/constraints on the association. For simplicity, we use the **no grooving** variant of our model here.

A straightforward approach to address this RQ is to let the model generates randomly a large number of event sequences (i.e., compositions) and examine how often it generates a plausible `STRING` event after a `NOTE-ON` event. Table 3(a) shows the average note-string association accuracy calculated from 50 generated 16-bar tabs, broken down into six values according to `STRING`. To our mild disappointment, the accuracy, though generally high, is not perfect. This indicates that some post-processing is still needed to ensure the note-string association is correct.

As Table 3(a) shows larger errors toward the 6th string, we also examine how the errors distribute over the pitches. Interestingly, Figure 4(a) shows that the model makes mistakes only in the low end; the fingering prediction is good for pitches (i.e., MIDI numbers) from 64 to 84.

It is hard to find out why exactly this is the case, but we present two more observations here. First, we plot in Figure 4(b) the popularity of these pitches in the training set. The Pearson correlation coefficient between the note quantity and the error rate is weak, at 0.299, suggesting that this may not be due to the sparseness of the low-pitched notes. Second, we show in Table 3(b)–(d) the note-string association output probability estimated by our model for three different pitches. Interestingly, it seems the model has the tendency to use neighboring strings for each pitch. For example, pitch 42 is actually a bass note playable on the 6th string, and it erroneously "leaks" mostly to the 5th string.

	Hard accuracy ↑		**Soft distance** ↓	
	mean	max	mean	min
hard grooving	76.2%	82.4%	56.3	44.6
soft grooving	76.9%	83.0%	**56.2**	**43.7**
multi-hard	**79.0%**	**85.7%**	57.8	44.3
multi-soft	74.6%	81.1%	64.7	52.9
no grooving	70.0%	80.1%	58.6	47.7
training data	**82.1%**	**89.5%**	**43.8**	**28.6**
random	64.9%	71.3%	70.6	59.6

Table 4. Objective evaluation on groove coherence.

5.2 Experiment 2: On Groove

Figure 5 gives two examples of tabs generated by the hard grooving model. It seems the grooving is consistent across time in each tab. But, how good it is?

The 2nd RQ tests whether the added `GROOVING` events help a Transformer compose tabs with better rhythmic coherence. We therefore intend to compare the performance of models trained with or without `GROOVING` for generating "continuations" of a given "prompt."

We consider both objective and subjective evaluations here. For the former, we compare the models trained with `GROOVING` events obtained with each of the four vector-quantized grooving representations described in Section 4.3. We ask the models to generate 16-bar continuations following the first 4 bars of the 30 tabs in the validation set. The performance of the models is compared against that of the 'no-grooving' baseline, the 'real' continuations (of these 30 tabs), and a 'random' baseline that picks the next 16 bars from another tab at random from the validation set. The last two are meant to set the high-end and low-end performances, respectively. For fair comparison, we also project the note onsets of the validation data onto the 16th-note grid underlying our training data.

We consider the following two simple objective metrics:

- **Hard accuracy**: Given the hard grooving patterns $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ of the prompt, and those of the

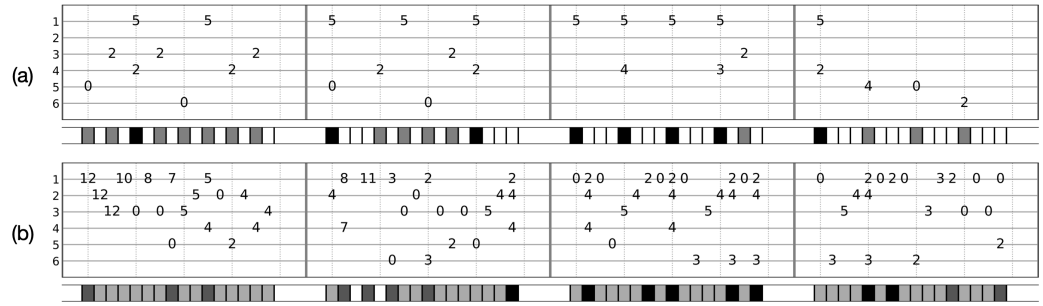


Figure 5. Segments of 2 tabs randomly generated by the hard grooving model; below each tab—the soft grooving patterns.

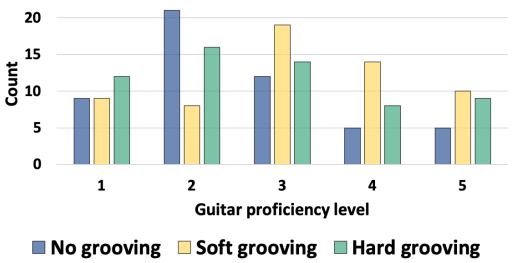


Figure 6. Result of the first user study asking subjects to choose the best among the three continuations generated by different models, with or without GROOVING, given a man-made prompt. The result is broken down according to the self-report guitar proficiency level of the subjects.

continuation $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$, where both $\mathbf{x}_i$ and $\mathbf{y}_j$ are in $\{0,1\}^K$, $N = 4$, $M = 16$, $K = 16$, we compare the similarity between $\mathbf{X}$ and $\mathbf{Y}$ by

$$\text{mean}_{i=(1,\ldots,N)} \frac{1}{MK} \sum_{j=1}^{M} \sum_{k=1}^{K} \text{XNOR}(x_i^{(k)}, y_j^{(k)}) \,, \tag{1}$$

where $\text{XNOR}(\cdot,\cdot)$ returns, element-wisely, whether the k-th element of $\mathbf{x}_i$ and $\mathbf{y}_j$ are the same. Alternatively, we replace the mean aggregator by $\max$, to say it is good enough for $\mathbf{y}_j$ to be close to any $\mathbf{x}_i$.

- **Soft distance**: We consider instead the soft grooving patterns $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_j$, and compute the distance between them as $\text{mean}_{i=(1,\ldots,N)} \frac{1}{M} \sum_{j=1}^{M} \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_j\|_2^2$. We can similarly replace mean by the $\min$ function.

Table 4 shows that, consistently across different metrics, groove-aware models outperform the no-grooving model. Moreover, the scores of the groove-aware models are closer to the high end than to the low end. It is also important to note that, there is still a moderate gap between the best model's composition and the real data, which has to be further addressed in the future work.

Figure 6 shows the result of the subjective evaluation, where we present the audio rendition (using a guitar synthesizer) of the aforementioned 16-bar continuations to human listeners, and ask them to choose the one they like the most, among those generated by the 'no-grooving,' 'soft-grooving,' and 'hard-grooving' models. We divide the response from 57 participants by their self-report proficiency level in guitar. Figure 6 shows that professionals are aware of the difference between groove-aware and no-grooving models. According to their optional verbal response, groove-aware models continue the prompts better, and generate more pleasant melody lines.

	Real	**No grooving**	**Hard grooving**
MOS	3.48±1.16	2.80±1.03	3.43±1.12

Table 5. Result of the second user study (in mean opinion score, from 1 to 5) comparing audio renditions of real tabs and machine-composed tabs by two variants of our model.

5.3 Experiment 3: On Comparison with Real Tabs

Finally, our last RQ involves another user study where we ask participants to rate, on a Likert five-point scale how they like the audio rendition of the continuations, this time including the result of real continuations. For groove-aware models, we consider hard-grooving only, for its simplicity and also for reducing the load on the subjects. Much to our surprise, the average result from 23 participants (see Table 5) suggests that hard-grooving compositions are actually on par with real compositions. We believe this result has to be taken with a grain of salt, as it concerns with only fairly short pieces (i.e., 16 bars) that do not contain performance-level variations. Yet, it provides evidence showing the promise of deep learning for tab composition.

6. CONCLUSION

In this paper, we have presented a series of evaluations supporting the effectiveness of a modern neural sequence model, called Transformer-XL, for automatic composition of fingerstyle guitar tabs. The model still has troubles in ensuring the note-string association and the rhythmic coherence of the generated tabs. How well the model generates tabs of plausible long-term structure is not yet studied. And, much of the expression in guitar music is left unaddressed. Much work are yet to be done to possibly redesign the network architecture and the tab representation. Yet, we hope this work shows promises that inspire more research on this intriguing area of research.

7. REFERENCES

[1] Ample sound. https://www.amplesound.net/en/index.asp.

[2] The BitMidi dataset. https://github.com/feross/bitmidi.com.

[3] Fingerstyle guitar. https://en.wikipedia.org/wiki/Fingerstyle_guitar/.

[4] J. Abeßer, H. Lukashevich, and G. Schuller. Feature-based extraction of plucking and expression styles of the electric bass guitar. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 2290–2293, 2010.

[5] J. Abeßer and G. Schuller. Instrument-centered music transcription of solo bass guitar recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(9):1741–1750, 2017.

[6] S. Ariga, S. Fukayama, and M. Goto. Song2Guitar: A difficulty-aware arrangement system for generating guitar solo covers from polyphonic audio of popular music. In *Proc. International Society for Music Information Retrieval Conference*, pages 568–574, 2017.

[7] A. M. Barbancho, A. Klapuri, L. J. Tardon, and I. Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE Trans. Audio, Speech and Language Processing*, 20(3):915–921, 2012.

[8] J.-P. Briot, G. Hadjeres, and F. Pachet. *Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems*. Springer, 2019.

[9] G. Burlet and I. Fujinaga. Robotaba guitar tablature transcription framework. In *Proc. International Society for Music Information Retrieval Conference*, 2013.

[10] Y.-P. Chen, L. Su, and Y.-H. Yang. Electric guitar playing technique detection in real-world recordings based on F0 sequence pattern recognition. In *Proc. International Society for Music Information Retrieval*, 2015.

[11] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel. Encoding musical style with transformer autoencoders. *arXiv preprint arXiv:1912.05537*, 2019.

[12] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[13] S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *Proc. International Society for Music Information Retrieval*, pages 509–516, 2004.

[14] 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. International Society for Music Information Retrieval*, pages 685–692, 2019.

[15] J. D. Fernández and F. Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48(1):513–582, 2013.

[16] S. Gorlow, M. Ramona, and F. Pachet. Decision-based transcription of Jazz guitar solos using a harmonic bident analysis filter bank and spectral distribution weighting. *arXiv preprint arXiv:1611.06505*, 2016.

[17] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[18] A. Hrybyk and Y. E. Kim. Combined audio and video analysis for guitar chord identification. In *Proc. International Society for Music Information Retrieval Conference*, 2010.

[19] C.-Z. 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 *Proc. International Conference on Learning Representations*, 2019.

[20] Y.-S. Huang and Y.-H. Yang. Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions. In *Proc. ACM International Conference on Multimedia*, 2020.

[21] E. J. Humphrey and J. P. Bello. From music audio to chord tablature: Teaching deep convolutional networks toplay guitar. In *Proc. IEEE International Conference on Acoustics, Speech & Signal Processing*, pages 6974–6978, 2014.

[22] 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. International Conference on Digital Audio Effects*, 2014.

[23] A. Lerch, C. Arthur, A. Pati, and S. Gururani. Music performance analysis: A survey. In *Proc. International Society for Music Information Retrieval Conference*, pages 33–43, 2019.

[24] M. McVicar, S. Fukayama, and M. Goto. AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space. In *Proc. International Conference on Signal Processing*, pages 599–604, 2014.

[25] M. McVicar, S. Fukayama, and M. Goto. AutoRhythmGuitar: Computer-aided composition for rhythm guitar in the tab space. In *Proc. International Computer Music Conference*, 2014.

[26] M. McVicar, S. Fukayama, and M. Goto. AutoGuitarTab: Computer-aided composition of rhythm and lead guitar parts in the tablature space. *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, 23(7):1105–1117, 2015.

[27] J. Michelson, R. M. Stern, and T. M. Sullivan. Automatic guitar tablature transcription from audio using inharmonicity regression and bayesian classification. *Journal of The Audio Engineering Society*, 2018.

[28] E. Mistler. Generating guitar tablatures with neural networks. *Master of Science Dissertation, The University of Edinburgh*, 2017.

[29] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 2018.

[30] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *Proc. AISB Symposium on Musical Creativity*, pages 110–117, 1999.

[31] C. Payne. MuseNet. `https://openai.com/blog/musenet/`, 2019.

[32] G. Peeters. Rhythm classification using spectral rhythm patterns. In *Proc. International Society for Music Information Retrieval*, pages 644–647, 09 2005.

[33] C. Raffel and D. P. W. Ellis. Extracting ground truth information from MIDI files: A MIDIfesto. In *Proc. International Society for Music Information Retrieval*, pages 796–802, 2016. [Online] `https://colinraffel.com/projects/lmd/`.

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

[35] S. I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[36] O. Senn, L. Kilchenmann, T. Bechtold, and F. Hoesl. Groove in drum patterns as a function of both rhythmic properties and listeners' attitudes. *PLOS ONE*, 13:1–33, 06 2018.

[37] L. Su, L.-F. Yu, and Y.-H. Yang. Sparse cepstral and phase codes for guitar playing technique classification. In *Proc. International Society for Music Information Retrieval*, 2014.

[38] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang. TENT: Technique-embedded note tracking for real-world guitar solo recordings. *International Society for Music Information Retrieval*, 2(1):15–28, 2019.

[39] L. Thompson, S. Dixon, and M. Mauch. Drum transcription via classification of bar-level rhythmic patterns. In *Proc. International Society for Music Information Retrieval*, pages 187–192, 2014.

[40] D. R. Tuohy and W. D. Potter. Guitar tablature creation with neural networks and distributed genetic search. In *Proc. International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2006.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[42] E. Waite, D. Eck, A. Roberts, and D. Abolafia. Project Magenta: Generating long-term structure in songs and stories, 2016. `https://magenta.tensorflow.org/blog/2016/07/15/lookback-rnn-attention-rnn/`.

[43] A. Wiggins and Y. E. Kim. Guitar tablature estimation with a convolutional neural network. In *Proc. International Conference on Music Information Retrieval*, pages 284–291, 2019.

[44] S.-L. Wu and Y.-H. Yang. The Jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. In *Proc. International Society for Music Information Retrieval Conference*, 2020.

[45] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello. GuitarSet: A dataset for guitar transcription. In *Proc. International Conference on Music Information Retrieval*, pages 453–460, 2018. [Online] `https://github.com/marl/guitarset/`.

[46] K. Yazawa, D. Sakaue, K. Nagira, K. Itoyama, and H. Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In *Proc. IEEE International Conference on Acoustics, Speech & Signal Processing*, pages 196–200, 2013.