TOWARDS AUTOMATIC TRANSCRIPTION OF POLYPHONIC ELECTRIC GUITAR MUSIC: A NEW DATASET AND A MULTI-LOSS TRANSFORMER MODEL

Yu-Hua Chen[1,2,3], *Wen-Yi Hsiao*[1], *Tsu-Kuang Hsieh*[1], *Jyh-Shing Roger Jang*[3], *and Yi-Hsuan Yang*[1,2]

[1] Taiwan AI Labs, Taiwan, [2] Academia Sinica, Taiwan, [3] National Taiwan University, Taiwan

ABSTRACT

In this paper, we propose a new dataset named EGDB, that contains transcriptions of the electric guitar performance of 240 tablatures rendered with different tones. Moreover, we benchmark the performance of two well-known transcription models proposed originally for the piano on this dataset, along with a multi-loss Transformer model that we newly propose. Our evaluation on this dataset and a separate set of real-world recordings demonstrate the influence of timbre on the accuracy of guitar sheet transcription, the potential of using multiple losses for Transformers, as well as the room for further improvement for this task.

Index Terms— Dataset, guitar transcription, Transformer

1. INTRODUCTION

Automatic music transcription (AMT) is a fundamental task in audio signal processing and music information retrieval (MIR) [1–8]. Among musical instruments, the AMT of piano music has received the most attention in the literature [9–17], partly thanks to the availability of large-scale labeled datasets such as MAPS [1] and MAESTRO (∼200 hours) [5]. Creating labeled data for piano transcription has been found feasible, because the labels can be entered while playing the music using a MIDI keyboard, or a piano with note-capturing devices [2, 3]. Saving piano scores as MIDI files also comes handy, for a MIDI file can specify the onset, offset, velocity of notes, and the usage of piano pedals. Other instruments, however, do not have such an advantage, due to the differences in the physical mechanisms creating the sounds. Without the support of large labeled datasets, researchers cannot easily employ state-of-the-art machine learning algorithms to promote the progress of AMT for instruments other than the piano.

This paper aims at contributing to the AMT of music played by an electric guitar, which plays an important role in many musical genres such as rock and metal. Guitar transcription is technically challenging as it entails a similar polyphonic transcription problem as piano transcription. Our contributions are threefold.

First, extending the methodology proposed by Xi *et al.* [18], we propose a new dataset for electric guitar transcription. Xi *et al.* proposed to attach a special hexaphonic pickup [19, 20] to each string of an acoustic guitar to capture activity signals per string. The hexaphonic recordings were then analyzed using a semi-automatic approach combining the pYIN monophonic pitch estimator [21] and manual inspections to generate annotations for each string individually. They used a microphone to record the sounds of the acoustic guitar as the audio counterpart of the annotations. The resulting dataset, named GuitarSet [18], is rather small and comprises the annotations of 60 unique pieces (totaling 30 minutes) from a single guitar. In contrast, we collect the hexaphonic recordings from an electric guitar, and use a JUCE program [22] to control a digital audio workstation (DAW) to automatically re-render the audio recordings of the "Direct Input" (DI) using different amplifiers (Amps), including low-gain Amps and high-gain ones. Moreover, we employ a new collecting pipeline so as to reduce the effort of manual inspection. The final dataset contains six copies of 118 minutes of guitar playing, each copy being associated with a different timbre.

The new dataset, named "EGDB," is constructed in this way to account for the diverse timbre associated with electric guitar. Having multiple guitar tones makes it possible to test a trained model on held-out unseen tones for generalizability. We make the dataset publicly available. Link can be found at our demo website: `https://ss12f32v.github.io/Guitar-Transcription/`.

Our second contribution lies in the development of a new AMT model using Transformers [23]. Very recently, Hawthorne *et al.* [17] proposed a sequence-to-sequence (seq2seq) piano transcription model with Transformers, attaining the state-of-the-art accuracy on MAESTRO [5]. While previous work on AMT, such as the iconic "onset and frames" (OAF) model [14], tends to optimizes their network using frame-wise losses, this particular model [17] measures loss only in the symbolic space at the output of the Transformer decoder. In light of the possible complementary advantages of both loss functions, we propose an extension of this seq2seq model [17] by employing both losses from encoder and decoder. We will open source the code of this new model in the near future.

As the final contribution, we benchmark the performance of our implementation of the OAF model [14], the seq2seq Transformer model [17], and the proposed multi-loss seq2seq Transformer model for guitar sheet transcription over EGDB, considering both the scenarios where the timbre of the test data is seen or unseen during training. Moreover, we report an additional experiment transcribing real-world guitar recordings downloaded from YouTube. Our experiments validate the importance of considering multiple guitar timbres, and the effectiveness of adopting multiple losses.

2. RELATED WORK

Guitar transcription differs from piano transcription in many aspects, including the diversity of guitar tones, the intensive use of guitar playing techniques [24–29] such as bending and sliding, as well as the need of string (fingering) detection (i.e., to detect which string a note is played) [30, 31]. We differentiate below two types of guitar transcription tasks: *tab transcription*, which involves detecting not only the pitches but also the string playing the pitches, and *sheet transcription*, which does not deal with string detection. We focus on the latter in this paper, leaving the former as a future work.

Existing datasets for guitar transcription are rather small in size. For example, the IDMT-SMT-GUITAR database [26] contains only 17 unique guitar licks with groundtruth labels for building transcription models. The GuitarSet [18] provides only 30 unique comping

tracks and 30 solo tracks, in only one timbre (despite that the performance of multiple guitarists over these 60 tracks are recorded). The guitar playing techniques database [27] is solely for the detection of playing techniques, not the estimation of pitches and onset times. As a result, progress in guitar transcription has been slower than that of piano transcription, with few deep learning based attempts to date.

To our knowledge, the TabCNN model developed by Wiggins and Kim [31] may represent the state-of-the-art for guitar tab transcription. The model, which is fully-convolutional, is trained on the GuitarSet to estimate guitar tablature from audio of a solo acoustic guitar. We do not benchmark the performance of this model using our dataset, since our focus here is on sheet transcription. Moreover, in the context of piano transcription, it has been shown that models with recurrent layers tend to outperform fully-convolutional models [11]. While this needs to be further tested in the future, we consider in this work only recurrent and Transformer-based methods, which have been widely used in piano transcription.

Given an audio recording, the OAF model [14] uses separate but jointly-trained convolutional recurrent neural network heads for detecting note onsets, note velocities, and frame-wise note presence, respectively. The training objective is to minimize the frame-wise cross-entropy loss for onsets and velocity at specific frames (where there is an onset, according to groundtruth labels) and the frame-wise cross-entropy loss for note presence across all the frames. Kim and Bello [15] improved the performance of OAF by adding a fully convolutional discriminator discriminating real and generated piano rolls in an adversarial setting [32].

Recently, the use of Transformers [23] for AMT has been shown possible for piano transcription by Hawthorne *et al.* [17], who used a generic encoder-decoder Transformer architecture and considered the output of the decoder, i.e., a sequence of note-level "tokens", such as NOTE-ON and NOTE-OFF [33] (see Section 4.2 for more details), as the result of transcription. Specifically, the encoder of the model takes the spectrogram of an audio segment as input and converts each frame into a hidden representation via a number of bidirectional self-attention layers. The groundtruth transcription result of the audio segment is represented as a sequence of tokens beforehand while preparing the training data, and is treated as the target output of the decoder of the model. The decoder learns to autoregressively generate the target sequence by self-attending to tokens previously-generated by itself as well as cross-attending to the hidden states of all the input audio frames. The encoder and decoder are jointly optimized by minimizing the reconstruction error of the target sequence, measured by the cross-entropy (CE) loss at each timestep of the sequence using the teach-forcing training strategy. In this paper, we also refer to this model as an "CE-only Transformer," for its training process considers only the CE loss in the space of symbolic tokens at the decoder side.

Transformer decoders have been employed in the context of automatic guitar tab generation [34, 35], which learns to generate original music from a large collection of guitar tabs (without the corresponding audio files). It might be possible to use such tabs to pretrain the decoder of our model, but we leave this as a future work.

3. THE PROPOSED DATASET

3.1. Audio Collection

Following Xi *et al.* [18], we employ a hexaphonic pickup to capture audio from each string individually. The pickup is attached to a Stratocaster-type electric guitar by an experienced technician. Unlike the case of using a microphone to collect the sound of an *acous-* *tic guitar* as done by Xi *et al.* [18], we opt for collecting a clean "Direct Input" (DI) sound from our *electric guitar* using a DI box [36]. Specifically, we use high-quality Mogami cables (to reduce possible noises cause by equipment) to connect the hexaphonic pickup to a DI box with six output jacks, and then to the audio interface on a computer. The summation of the signals from the six output jacks (one corresponding to each string) can then be treated as an audio waveform, i.e., the *DI recording*. Using the DI reduces the effect of the recording environment and makes it easier to further apply Amps to re-render the sounds.

We collect over 500 guitar tabs encompassing compositions of solo, arpeggios and comping in various genres. The third author of the paper, who is a professional guitarist playing in a post-rock band, spends almost two months using that electric guitar to faithfully perform the tabs in a studio, following along through a headset a click track whose beat-per-minute (BPM) is set according to the corresponding tab. The throughput is roughly 5 minutes of recorded performance per working day. This yields the DI recordings of 240 unique tabs, amounting to 118 minutes worth of audio data.

3.2. Annotation

Since our musician is asked to faithfully follow the reference tabs and click tracks while playing, we use the following simple automatic process to get the annotations of onsets. First, we estimate the *expected* onset times of the notes from the tabs, using the BPM specified on the tabs to go from symbolic timing (in subbeats) to absolute timing (in milliseconds). Second, we run an onset detection algorithm [37] over the recorded DI signal per string to estimate the *actual* onset times of the notes. The expected onset times (from the tabs) and actual onset times (in the audio performances) will not be exactly the same, but for most cases there is a nice one-to-one mapping between them. Therefore, we can simply label the pitch of each actual onset according to that of the closest expected onset, without using sophisticated alignment algorithms.

However, the above process cannot be used for the annotation of offset times, due to the absence of a reliable offset detection algorithm. For now, we label the offset time for each note according to the actual onset time of the note "plus" the *expected* duration (in milliseconds) of the note.

After the aforementioned annotation process, the first author of the paper, who is an amateur guitarist, manual checks the annotations for all the 240 recordings. Only very few onset labels have to be (and are) manually corrected. In contrast, the offset labels are less reliable. We leave the improvement in offset labels as a future work, using the present offset labels in model training but not evaluating the performance of the benchmarked models in offset detection.

While performing the guitar tabs, our musician employs *playing techniques* such as bending when they are specified on the tabs. It is possible to get annotations of the usage of these playing techniques by referring to the tabs, but we leave this as a future work. Similarly, although we have access to *fingering* information from individual DI signals, for simplicity, we view string detection a future work. Moreover, we do not transcribe the note velocity.

3.3. Timbre Re-rendering

For timbre augmentation, we use the following five commonly-used and diverse Amplifiers selected by our musician to distort and re-render the DI recordings via Guitar Rig 5 [38] digitally: i) `Mesa Boogie Mark v`, ii) `Fender Twin amplifier`, iii) `Marshall JCM2000`, iv) `Roland JC120`, and v) `Marshall`
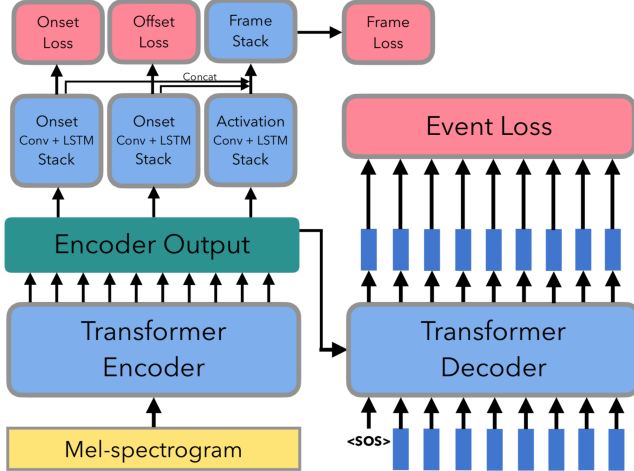
Fig. 1. Schematic diagram of the proposed multi-loss seq2seq Transformer model for AMT, with additional losses at the output of the encoder compared to the model proposed by Hawthorne *et al.* [17]. The input to the "frame stack" is a concatenation of the output from the "onset," "offset," and "activation" stacks.

`Plexi`. We automate the rendering process using an in-house program written in JUCE, an open-source cross-platform C++ application framework [22].

4. THE PROPOSED TRANSCRIPTION MODEL

Figure 1 shows a diagram of the proposed multi-loss seq2seq Transformer model, which falls back conceptually to the CE-only Transformer model [17] if dropping the three loss terms associated with the encoder output. We provide the details below.

We note that, while we develop the model in the context of guitar transcription, the architecture is general and can be adapted to other AMT tasks as well (the same holds for OAF [14] and the CE-only Transformer [17]). Moreover, we can further add string- and playing technique-related prediction heads and losses in the future.

4.1. Transformer Encoder

The encoder contains primarily a stack of self-attention layers. Unlike [17], we add an "onset stack," an "offset stack," and an "activation stack," each comprising a feed-forward dense layer, to estimate the presence of note onset, presence of note offset, and the holding of a pitch for each frame using the hidden representation produced by the last self-attention layer of the encoder, as shown in Figure 1. Moreover, we consider the following losses to provide supervisory signals to train the encoder. L_{frame} is from the "frame stack," which takes the concatenation of the output of the aforementioned three stacks as input, comparing the activation probability of the 45 possible pitches for guitar (`E2` to `C6`) of the output of this stack with that of the groundtruth (which is binary and possibly multi-hot) using cross entropy. L_{onset} and L_{offset} compare the probability of the presence of onset and offset of the 45 pitches with the groundtruth (also using cross entropy) at the output of the onset and offset stacks, respectively, concerning only the frames at the groundtruth onset and offset times (as done in OAF [14]). We call the summation of the three losses as L_{AM}, namely $L_{\text{AM}} = L_{\text{frame}} + L_{\text{onset}} + L_{\text{offset}}$.

4.2. Transformer Decoder

Our decoder follows pretty much the setting in [17], representing music as "event tokens" using a vocabulary with 3 types of tokens:

Note [90 possible values] Each of the 45 possible pitches from `E2` to `C6` has its `NOTE-ON` and `NOTE-OFF` events.

Time [512 possible values] Every `NOTE-ON` or `NOTE-OFF` will be followed by a `WHICH-FRAME` event that indicates the exact frame among the input frames the note-on or note-off event takes place, where 512 is the maximal number of input frames of the encoder considered in our implementation. It has been found in [17] that such `WHICH-FRAME` tokens work better than the alternative `TIME-SHIFT` tokens [33].

SOS, EOS [2 values] The start and end of a token sequence.

The decoder and encoder are jointly optimized by the "event loss," denoted as L_{LM}, calculated at the output of the decoder, formulated as the CE in the probability over all the possible tokens between the decoder output and the groundtruth at each time step of the decoder sequence. The prior model [17] uses only L_{LM} to train their whole model, while we use both: $L_{\text{total}} = L_{\text{AM}} + L_{\text{LM}}$.

4.3. Implementation Details

We split our dataset by 8:1:1 to create disjoint training, validation and test splits. We implement the proposed model and the two baselines [14, 17] by our own in PyTorch. The input to all the models are log-scaled mel-spectrograms computed by STFT with 2,048-pt Hann window, 512-pt hop size (each frame lasts for 11ms under 44.1 kHz audio), and 229 mel-filters. We train all the models with the proposed dataset on an NVIDIA Tesla V100 GPU (with 32 GB memory) using the Adam optimizer [39] with a learning rate (LR) of 1e–4, trained until the onset validation score converges. All the models converge in less than half a day. To the best of our knowledge, this represents the first attempt documenting the performance of the two baseline models for guitar transcription.

Unlike [17], which used eight Transformer stacks for both the encoder and decoder, we adopt only four stacks each for both our model and the CE-only baseline, because of the smaller size of our dataset and memory constraint of the GPU. We set the embedding size d_{model} to 256, hidden layer size d_{hidden} to 256, and use 8-head attention. Following [17], We use fixed absolute positional embedding for both the encoder and decoder.

For convenience, we fix the length of the source sequence (i.e., encoder input; the number of input frames) and target sequence (i.e., decoder output; the number of tokens corresponding to that input audio segment) to be the same, zero-padding at the back when needed. According to our dataset, the token sequence corresponding to a 256-frame audio segment (roughly 3 seconds) has on average 61.61 tokens, or 17.00±10.15 notes. We explore the effect of sequence length and batch size empirically in Section 5.1.

5. EXPERIMENT

We use the `mir_eval` library [40] to compute the "onset F1" and "frame F1" scores of each model. For onset F1, we consider an onset estimate as correct when it is within a ±50ms tolerance window of a ground truth onset (following [14]). We do not evaluate the offsets.

5.1. Evaluation on DI Recordings

We firstly consider the case where both the training and testing splits are DI recordings (or, "EGDB-DI"), therefore of the same timbre.

Model	Batch size	Sequence length	Parameters	DI (Section 5.1)			
				(Encoder output)		(Decoder output)	
				Onset F1	Frame F1	Onset F1	Frame F1
Onset and Frame (OAF) [14]	192	256	0.74M	0.591	0.583	—	—
CE-only Transformer [17]	32	256	4.58M	—	—	0.543	0.523
	128	128	4.58M	—	—	0.554	0.524
	256	64	4.58M	—	—	0.568	0.537
Proposed multi-loss Transformer	32	256	6.01M	0.598	0.579	0.507	0.495
	128	128	6.01M	0.604	0.573	0.515	0.493
	256	64	6.01M	**0.613**	**0.582**	0.514	0.496

Table 1. Transcription accuracy of various models trained on the training split of the DI recordings of EGDB (i.e., "EGDB-DI") and tested on the test split of EGDB-DI (i.e., same timbre). For the proposed model, we evaluate the result of both the encoder output and decoder output (since both are legitimate transcription results). We highlight the best result in bold.

Model	4, 5-th Amp (Section 5.2)		Real-world Data (Section 5.3)	
	Onset F1	Frame F1	Onset F1	Frame F1
[14] : DI	0.568	0.536	0.524	0.514
[14] : 3Amps	0.588	**0.546**	0.541	0.562
[17] : DI	0.503	0.478	0.513	0.492
[17] : 3Amps	0.532	0.523	0.521	0.484
Ours : DI	0.534	0.512	0.527	0.561
Ours : 3Amps	**0.592**	0.526	**0.550**	**0.586**

Table 2. Transcription accuracy on testing data with unseen timbres (left: the test split rendered with the two held-out Amps; right: real-world recordings) by the three models trained on either the training split of EGDB-DI or EGDB-3Amps. We report the result of [14] and our model at the encoder side, and that of [17] at the decoder side.

Fig. 2. (Top) the spectrogram of a real-world testing recording, (middle) human transcription, (bottom) transcription result of the proposed multi-loss Transformer model, trained on EGDB-3Amps.

Table 1 shows that both OAF and the proposed multi-loss Transformer model outperform the CE-only Transformer model, possibly because our training set is not large enough to train a reliable language model for the decoder, necessitating the use of loss terms at the encoder output side. We also see that, the Transformer models perform better using a larger batch size rather than a longer sequence length. The best result is obtained by the multi-loss Transformer model with batch size 256, using the output of its encoder. Therefore, in what follows, we evaluate the accuracy of the encoder output rather than decoder output of our model.

5.2. Evaluation on Unseen Timbres Rendered with Amps

Secondly, we evaluate the effect of timbre on the performance of guitar transcription. This time, we use the recordings of the test split rendered with the last two Amps (i.e., 'iv' and 'v' in Section 3.3) for testing; and recordings of the training split of either the DI or those rendered with the first three Amps (or, "EGDB-3Amps," which is three times larger than EGDB-DI) for training. The following observations can be made from the left-hand side of Table 2. First, for all the three models, training on "3Amps" works better than training on "DI," demonstrating the advantage of having a multi-timbre dataset. Second, if we use only EGDB-DI for training, we see that the onset F1 drops from 0.613 to 0.534 when we change the test set from DI to 4, 5-th Amp. Third, using EGDB-3Amps for training, our model achieves the best onset F1 of 0.592, while [14] attains the best frame F1 score. We plan to do further error analysis in the future to gain insights into the behavior of the models.
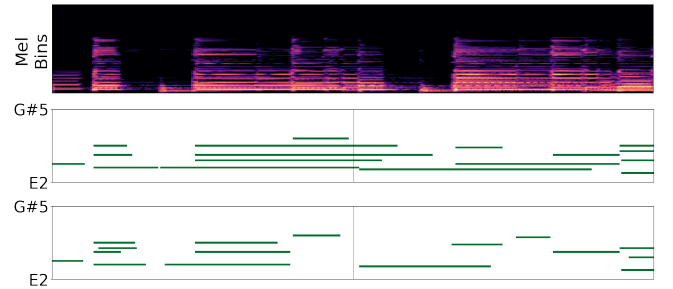
5.3. Evaluation on Real-world Recordings

Similar observations hold in our final evaluation, where we test the models on five real-world recordings downloaded from YouTube and manually annotated by our musician. The real-world recordings involve the use of pedals for effects such as delay and reverb, that are not seen in our data. Moreover, as often the case, the real-world recordings are not DI signals and can involve the use of guitar types and Amps not seen in our data. While the performance of all the models drops as expected, as shown in the right-hand side of Table 2 e.g., the best onset F1, as achieved by the proposed model, drops to 0.550), training on 3Amps is still advantageous. This adds empirical support of the importance of using a timbre-rendered dataset.

The real-world recordings and the transcription result of our model can be found at the demo website. The result of one such recording is illustrated in Figure 2.

6. CONCLUSION

In this paper, we have presented a new data collection methodology and a resulting new public dataset of multi-timbre electric guitar. We have also presented a multi-loss seq2seq Transformer model for AMT, and benchmarked its performance along with another two existing models [14, 17] using our dataset and a small collection of real-world recordings. The result of the best model in onset F1 falls within 0.550–0.613 across different test sets, while the onset F1 of state-of-the-art models for piano transcription (e.g., [17]) tends to be higher than 0.950 on MAESTRO [5]. This prompts future work to close the performance gap between piano and guitar transcription.

7. REFERENCES

[1] Valentin Emiya, Roland Badeau, and Bertrand David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, pp. 407–434, 2013.

[3] Li Su and Yi-Hsuan Yang, "Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription," in *Proc. CMMR*, 2015.

[4] Li Su and Yi-Hsuan Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE/ACM Trans. Audio, Speech and Language Processing*, vol. 23, no. 10, pp. 1600–1612, 2015.

[5] Curtis Hawthorne et al., "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. ICLR*, 2019.

[6] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, pp. 20–30, 2019.

[7] John Thickstun, Zaïd Harchaoui, Dean P. Foster, and Sham M. Kakade, "Invariances and data augmentation for supervised music transcription," in *Proc. ICASSP*, 2018, pp. 2241–2245.

[8] Yu-Te Wu et al., "Omnizart: A general toolbox for automatic music transcription," *arXiv preprint arXiv:2106.00497*, 2021.

[9] Emmanuel Vincent, Nancy Bertin, and Roland Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 528–537, 2010.

[10] Cheng-Te Lee, Yi-Hsuan Yang, and Homer H. Chen, "Multipitch estimation of piano music by exemplar-based sparse representation," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 608–618, 2012.

[11] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[12] Andrea Cogliati, David Temperley, and Zhiyao Duan, "Transcribing human piano performances into music notation," in *Proc. ISMIR*, 2016, pp. 758–764.

[13] Rainer Kelz et al., "On the potential of simple framewise approaches to piano transcription," in *Proc. ISMIR*, 2016.

[14] Curtis Hawthorne et al., "Onsets and frames: Dual-objective piano transcription," in *Proc. ISMIR*, 2018.

[15] Jong Wook Kim and Juan Pablo Bello, "Adversarial learning for improved onsets and frames music transcription," in *Proc. ISMIR*, 2019.

[16] Qiuqiang Kong et al., "High-resolution piano transcription with pedals by regressing onsets and offsets times," *arXiv preprint arXiv:2010.01815*, 2020.

[17] Curtis Hawthorne et al., "Sequence-to-sequence piano transcription with Transformers," in *Proc. ISMIR*, 2021, pre-print available at https://arxiv.org/abs/2107.09142.

[18] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello, "GuitarSet: A dataset for guitar transcription.," in *Proc. ISMIR*, 2018, pp. 453–460.

[19] Paul D. O'Grady and Scott T. Rickard, "Automatic hexaphonic and guitar transcription and using and nonnegative constraints," in *Proc. IET Irish Signals and Systems Conf.*, 2009.

[20] Raymond Vincent Migneco, *Analysis and synthesis of expressive guitar performance*, Ph.D. thesis, Drexel University, 2012.

[21] Matthias Mauch and Simon Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proc. ICASSP*, 2014, pp. 659–663.

[22] "JUCE: An open-source cross-platform C++ application framework," [Online] https://github.com/juce-framework/JUCE.

[23] Ashish Vaswani et al., "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[24] Jakob Abeßer, Hanna Lukashevich, and Gerald Schuller, "Feature-based extraction of plucking and expression styles of the electric bass guitar," in *Proc. ICASSP*, 2010.

[25] Loïc Reboursière et al., "Left and right-hand guitar playing techniques detection," in *Proc. NIME*, 2012, pp. 7–10.

[26] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters," in *Proc. DAFx*, 2014.

[27] Li Su, Li-Fan Yu, and Yi-Hsuan Yang, "Sparse cepstral, phase codes for guitar playing technique classification," in *Proc. ISMIR*, 2014, pp. 9–14.

[28] Yuan-Ping Chen, Li Su, and Yi-Hsuan Yang, "Electric guitar playing technique detection in real-world recordings based on f0 sequence pattern recognition," in *Proc. ISMIR*, 2015.

[29] Ting-Wei Su, Yuan-Ping Chen, Li Su, and Yi-Hsuan Yang, "TENT: Technique-embedded note tracking for real-world guitar solo recordings," *Trans. Int. Society for Music Information Retrieval*, vol. 2, no. 1, 2019.

[30] 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*, vol. 20, no. 3, pp. 915–921, 2012.

[31] Andrew Wiggins and Youngmoo Kim, "Guitar tablature estimation with a convolutional neural network," in *Proc. ISMIR*, 2019, pp. 284–291.

[32] Ian Goodfellow et al., "Generative adversarial nets," *Proc. Advances in neural information processing systems*, vol. 63, no. 11, pp. 139–144, 2014.

[33] Sageev Oore et al., "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, 2018.

[34] Yu-Hua Chen, Yu-Hsiang Huang, Wen-Yi Hsiao, and Yi-Hsuan Yang, "Automatic composition of guitar tabs by Transformers and groove modeling," in *Proc. ISMIR*, 2020.

[35] Pedro Sarmento et al., "DadaGP: A dataset of tokenized guitarpro songs for sequence models," in *Proc. ISMIR*, 2021.

[36] "DI unit," [Online] https://en.wikipedia.org/wiki/DI_unit.

[37] Sebastian Böck, Florian Krebs, and Markus Schedl, "Evaluating the online capabilities of onset detection methods.," in *Proc. ISMIR*, 2012, pp. 49–54.

[38] "Guitar Rig," Native Instruments. [Online] https://www.native-instruments.com/en/products/komplete/guitar/guitar-rig-6-pro/.

[39] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] Colin Raffel et al., "mir_eval: A transparent implementation of common MIR," in *Proc. ISMIR*, 2014, pp. 367–372.