Su, T.-W., et al. (2019). TENT: Technique-Embedded Note Tracking for Real-World Guitar Solo Recordings. *Transactions of the International Society for Music Information Retrieval,* 2(1), pp. 15–28. DOI: https://doi.org/10.5334/tismir.23

RESEARCH

TENT: Technique-Embedded Note Tracking for Real-World Guitar Solo Recordings

Ting-Wei Su[*,‡], Yuan-Ping Chen[†,‡], Li Su[‡] and Yi-Hsuan Yang[‡]

The employment of playing techniques such as string bend and vibrato in electric guitar performance makes it difficult to transcribe the note events using general note tracking methods. These methods analyze the contour of fundamental frequency computed from a given audio signal, but they do not consider the variation in the contour caused by the playing techniques. To address this issue, we present a model called technique-embedded note tracking (TENT) that uses the result of playing technique detection to inform note event estimation. We evaluate the proposed model on a dataset of 42 unaccompanied lead guitar phrases. Our experiments showed that TENT can nicely recognize complicated skills in monophonic guitar solos and improve the F-score of note event estimation by 14.7% compared to an existing method. For reproducibility, we share the Python source code of our implementation of TENT at the following GitHub repo: https://github.com/srviest/SoloLa.

Keywords: Automatic music transcription; guitar performance analysis; music signal processing; playing technique detection; expression style recognition; expressive music performance

1. Introduction

Recent years have seen an increasing number of on-line services such as Chordify and Riffstation for transcribing the chord progression of real-world guitar performance (de Haas et al., 2012). Although the accuracy of such chord transcription services is not perfect, they make it easier for music lovers and novice learners to comprehend and learn guitar music. Beside chords, transcribing the melody line of the lead guitar is also important for educational and archival purposes (e.g., to transcribe improvised music) (Xi et al., 2018). Similar to manual chord transcription, manual transcription of guitar solos demands musical training and is time consuming. However, compared to chord transcription, the transcription of solo guitar has received relatively less attention thus far.

A main difficulty of solo guitar transcription, compared to general automatic music transcription (AMT), is that guitar playing often involves heavy use of specific playing techniques or expression styles. The term playing technique here refers to a kind of skill played by the string-pressing hand, usually the left hand, performed within a note event or during the transition between two adjacent note events. For example, `Vibrato` is a technique used while playing a note, whereas `Slide` happens between two note events. These playing techniques have the effect of modulating the pitch of the involved notes, so they may confuse an AMT system and create errors in tracking the fundamental frequency (F0), onset and offset of the note events. For example, a note event played with `Vibrato` may be misinterpreted as multiple consecutive note events. Similar errors arise for other techniques such as `Slide` and `Bend`.

1.1 Lead Guitar Playing Techniques

Electric guitar is characterized by the flexibility of its strings, facilitating the employment of various playing techniques. In this work, we regard playing techniques as skills played by the string-pressing hand to modify the pitched sound ringing on a string. Specifically, we consider the following basic techniques (see **Figure 1** for examples):

- **`Bend:`** stretch a string with the string-pressing hand to increase the pitch of a ringing note, gradually or instantly.
- **`Release:`** loosen a 'bended' string to decrease the pitch of a ringing note; it is the opposite of `Bend`.
- **`Vibrato:`** a periodic oscillation of pitch.
- **`Hammer-on:`** when a note is sounded, use a finger of the string-pressing hand to quickly press down a higher fret on the same string while the first note is still ringing.
- **`Pull-off`** after playing a note, pull the fretting finger off the string to generate a lower note.

* Stanford University, US

† University of California, Santa Cruz, US

‡ Academia Sinica, TW

Corresponding author: Ting-Wei Su (twsu@ccrma.stanford.edu)

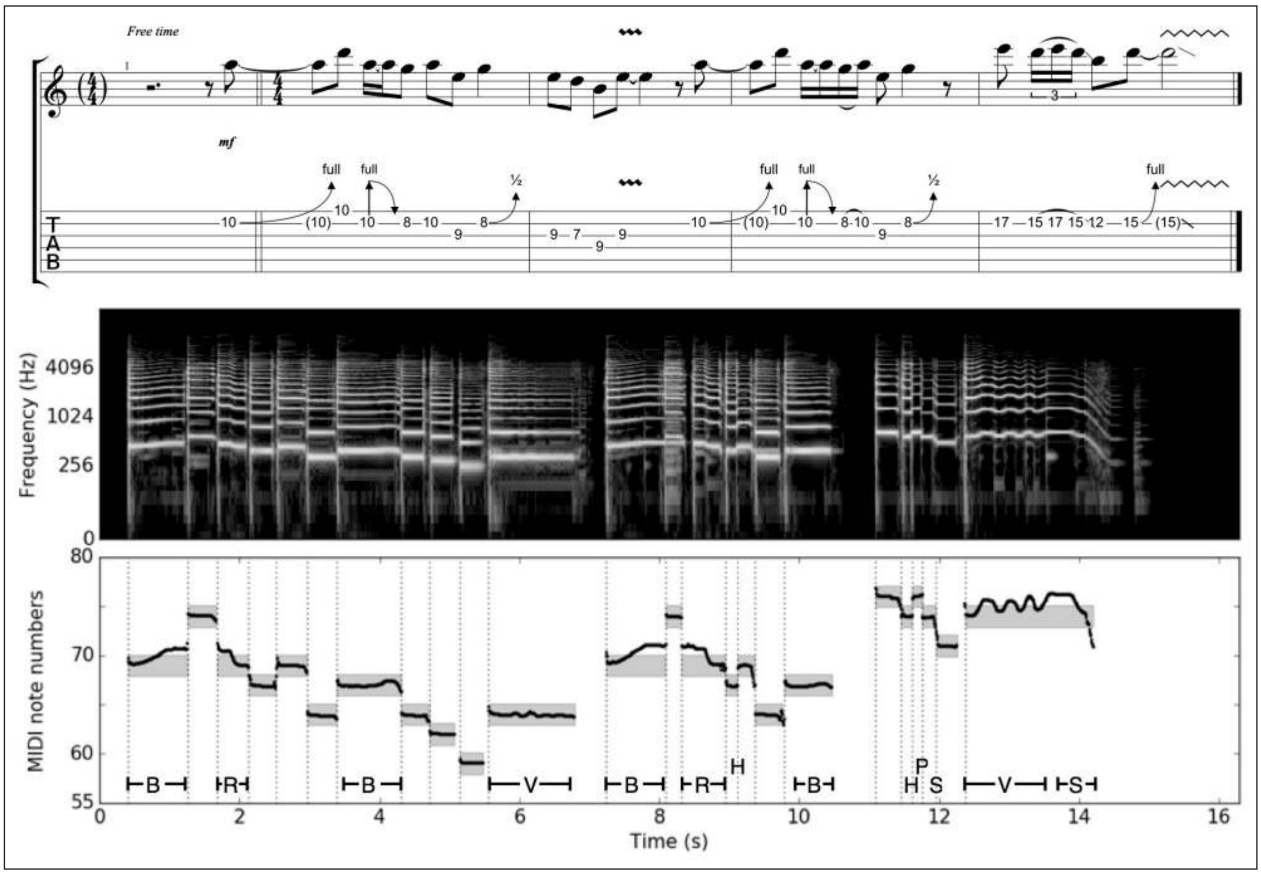


Figure 1: From top to bottom: **(a)** guitar sheet music and tablature generated by Guitar Pro (https://www.guitar-pro.com) **(b)** spectrogram of the example guitar phrase **(c)** discrete note events, pitch contour, and the intervals of playing techniques. In (c), 'B' denotes `Bend`, 'R' stands for `Release`, 'V' for `Vibrato`, 'H' for `Hammer-on`, 'P' for `Pull-off`, and 'S' for `Slide` (see Section 1.1 for definitions). From left to right, the first 'B' shows a note *full-bended* from `A4` to `B4` gradually. The second 'R' is a note *pre-bended* to `B4`, i.e., bend the note without sounding it, and then *released* to `A4` after playing the note. The third 'B' shows a *half-step* bend. The two 'V's are respectively a very subtle vibrato with smaller extent and a wide vibrato with larger extent. The last 'S' is a slide-out.

- **`Slide:`** slide a finger of the string-pressing hand across one or more frets to reach another note. A guitar solo often begins/ends with a variant known as "slide from/into nowhere," or `Slide-in/out` for short.

We do not consider plucking styles such as those discussed by Abeßer et al. (2010), for they are not from the string-pressing hand and do not affect the pitch. Moreover, when we have to assign a class label to each time instance, we refer to the case with no specific playing technique (i.e., simply plucking the strings) as **`Normal`** hereafter. In this work, we treat any pitch contour created using only `Bend`, `Release` and `Vibrato` techniques as a single note event.

1.2 Playing Technique Detection and Note Tracking

For transcribing solo guitar recordings, the detection of playing techniques is needed. While a sequence of note events comprises a melody, playing techniques determine how the note events are played and accordingly influence the expression of the guitar performance. As shown by the guitar tablature in **Figure 1**, a complete transcription of a guitar performance should contain annotations of the playing techniques. In addition, the two tasks, *playing technique detection* (i.e., predicting the techniques employed while playing the notes) and *note tracking* (i.e., estimating the F0, onset and offset of note events), also benefit each other when they are approached together. For example, from the pitch contour computed for note tracking, we can look for the parts with large variation (i.e., places with obvious F0 changes) to temporally localize possible use of playing techniques.

On the other hand, note tracking can also be improved with the aid of playing technique detection. The playing techniques mentioned in Section 1.1 can be categorized into two groups by whether or not they are employed during a transition between note events. Those that are employed between two adjacent notes include `Hammer-on`, `Pull-off`, and `Slide`, and those that are not include `Slide-in`, `Slide-out`, `Vibrato`, `Bend`, and `Release`. The second group are those prone to being incorrectly split into multiple notes with different pitches during note tracking. Therefore, when the techniques are detected, we know better how each note is played and may therefore avoid falsely splitting a single note event into several notes.

To our best knowledge, playing technique detection and note tracking have been mostly studied separately and little has been done in the literature to jointly consider them. The major technical contribution of the paper therefore lies in the design of a note tracking model that takes advantage of the result of playing technique detection for transcribing electric guitar solo recordings. We call the model *technique-embedded note tracking*, or TENT for short. The output of TENT is a sequence of note events with estimated notation of their pitches, onsets, offsets, and the involved playing techniques. Such a notation can be later used to produce more comprehensive sheet music.

1.3 Challenges and Proposed Solutions

Localizing and recognizing techniques of lead guitar in music audio is the first challenge of this work. As **Figure 1** illustrates, guitar techniques appear at arbitrary times, pitches, and phases, with various durations. We first need to *localize* candidate regions (i.e., time intervals) of playing techniques, and then *recognize* (or classify) the technique employed in each candidate region.

Since the main effect of playing techniques is pitch modulation, we propose to localize them not from the audio waveform itself, but from the estimated contour of F0, or *pitch contour* (see **Figure 1(c)** for an example). Research on automatic melody extraction has received great attention over the years, with easily accessible implementations of state-of-the-art algorithms (Salamon and Gómez, 2012). While notes that are played without any playing technique (i.e., simply plucking the strings) may correspond to horizontal regions in the pitch contour, the use of different techniques would lead to different patterns in the contour. We devised different methods to localize and recognize different playing techniques. For regions in a pitch contour that have mild pitch modulation within a short duration, we compute spectral features from the localized regions and then employ a pre-trained convolutional neural network (CNN) (LeCun et al., 1998) to determine whether or not those are playing techniques, and what kind of techniques they are.

We remark that our model associates playing techniques with temporal regions in the pitch contour, not with individual note events. This is important because a playing technique may not be used throughout the note event (e.g., employed only in the beginning of the note), and because some playing techniques are used in between two note events. Moreover, if multiple playing techniques are employed one after another while playing a note (e.g., for the last note event in **Figure 1**, there is a `Vibrato` in the beginning and a `Slide-out` at the end), our model can likely detect them all. As a result, the proposed model can generate symbolic notation which is designed for electric guitar solo and can be labeled on sheet music.

Another strength of such a pitch contour-based approach is that our model would be less sensitive to variation in tone colors (i.e., timbre) created by the use of sound effects such as distortion (Dattorro, 1997; Stein, 2010; Fohl and Meisel, 2012).

The second challenge is to properly leverage the result of playing technique detection (which involves both localization and recognition) to improve the result of note tracking. Instead of inventing a whole new note tracking algorithm from scratch, we propose to use an existing note tracking algorithm that is designed for general music (Mauch et al., 2015) to obtain an initial estimate first, and then use the result of playing technique detection in a post-processing stage to refine the result of note tracking. For example, for a note event played with `Bend`, the initial result of note tracking may falsely split it into two note events. We can correct this by merging the two note events and setting the F0 of the merged note event according to the F0 of the first note. In this way, we benefit from the cumulative efforts in the research community for general note tracking, and at the same time take into account the specialties of guitar music for guitar solo transcription.

1.4 Organization of the Paper

The remainder of this paper is organized as follows. Section 2 reviews related work on melody extraction, note tracking and guitar playing technique detection. Section 3 presents the details of the proposed TENT model. Section 4 describes the experimental setting we use to evaluate the proposed model and Section 5 discusses the results. Finally, Section 6 concludes the paper.

2. Related Work

2.1 Melody Extraction

Melody extraction has many different applications (Salamon et al., 2014), and for this paper, the extracted pitch contour provides essential information about the playing techniques being used.

Previous work tried to extract melody from different aspect. **Peeters**' temporal and spectral representation method (2006) performs F0 estimation by finding peaks in the dot product of a *spectral representation*, such as the spectrum of Discrete Fourier Transform, and a *temporal representation*, such as the real cepstrum, along the frequency axis. The idea is to exploit the inverse octave errors seen in the spectral and temporal representation of periodic signals. **pYIN** (Mauch and Dixon, 2014) performs F0 estimation directly in the time domain. It calculates the difference between the audio waveform and a time-shifted version. If the signal is periodic, we find a local minimum when the amount of shift is equal to the period. After finding several frequency (period) candidates at each temporal moment, and given a probability distribution of these frequencies, pYIN then uses a pre-defined Hidden Markov Model (HMM) to calculate the most likely pitch contour. It is based on its predecessor, YIN (de Cheveigné and Kawahara, 2002), which only finds one frequency in each time frame and uses it as the final estimation.

Melodia (Salamon and Gómez, 2012) extracts several frequency peaks at each temporal moment and computes their *saliency* by summing up the energy of the partials. It tracks all possible pitch contours according to the salience function. Then, it distinguishes between melodic contours and non-melodic contours based on heuristics such as the contour's average pitch height and its salience, the amount of deviation in the contour's pitch trajectory, and whether the contour contains vibrato or not. The final F0 trajectory is obtained by filtering out the non-melodic contours. Some other extensions of Melodia include changing the heuristic algorithm of melody selection into a generative classification model (Salamon et al., 2012) or a data-driven discriminative model (Bittner et al., 2015).

While most of the methods mentioned above make use of digital signal processing, recent work also tried to estimate the fundamental frequency directly from the raw waveform by means of deep learning techniques such as deep convolutional neural networks (Kim et al., 2018).

2.2 Note Tracking

Note tracking has been viewed as one of the most fundamental yet challenging tasks in the music information retrieval (MIR) community (Benetos et al., 2013). This problem has been mostly approached by incorporating the features for onset and offset detection, or by state-space modeling on the attack-decay-sustain-release (ADSR) curve and silence/non-silence behaviors. For the feature-based approach, Chang and Lee (2014) considered using the spectral correntropy, which is relevant to onset and offset events of a music signal. For the state-space model approach, Mauch et al. (2015); Cheng et al. (2015); Yang et al. (2017) utilized the hidden Markov model (HMM) to model the transition of note-level dynamics. Besides these approaches, side information other than onset, offset and F0 has also been utilized to enhance the accuracy of note tracking. For example, based on the observation that note onsets are correlated with beats, Nishikimi et al. (2016); Dzhambazov et al. (2017) utilized beat information to solve the note tracking problem. Playing techniques represent another source of important side information, but to our best knowledge, nothing has been done to leverage them for note tracking.

2.3 Playing Technique Detection

Unlike F0 estimation or chord recognition, research on playing technique detection is still in its early stages. The focus of much existing work is on playing technique recognition (classification) only, using audio recordings of pre-segmented individual notes. Abeßer et al. (2010) compiled a dataset of around 4,300 single bass guitar notes to investigate the classification of 5 plucking styles, techniques made by the plucking hand, and 5 expression styles, techniques made by the string-pressing hand. They extracted features motivated by the playing techniques. For example, their features include parameters that characterize the shape of a note's harmonic frequencies through time to better identify muted plucking style, spectral crest factor to detect dead-notes, and spectral centroid to discriminate different kinds of finger slapping. After reducing the dimensionality by a feature selection technique and a feature space transformation method, they evaluated the extracted and reduced features on several classifiers such as support vector machines (SVM).

Different from Abeßer et al. (2010), Reboursière et al. (2012) further discriminated whether a technique is played by the plucking hand or the string-pressing hand. First, their system observed the energy slope several milliseconds before the onset to distinguish the hand by which a note is played. Then, 4 techniques from the string-pressing hand were classified by measuring the pitch time derivative within a note, and 2 techniques from the plucking hand were identified by comparing the characteristics of the attack to pre-defined thresholds. Overall, it classified 6 playing techniques from 1,416 samples of single guitar notes.

Su et al. (2014) recorded 11,928 single electric guitar notes and investigated features extracted from the cepstrum and phase derivatives to classify 7 playing techniques using an SVM. Follow-up research has managed to improve the accuracy of playing technique recognition for the same dataset using Gaussian hierarchical latent Dirichlet allocation (Chen et al., 2017) and a variational auto-encoder with a Gaussian process (Chen et al., 2018). It is, however, not clear how these methods can detect playing techniques in a real-world guitar solo track, due to the lack of a playing technique localization module.

One exception is the work presented by Kehling et al. (2014), which considered playing technique detection in 12 phrases of guitar solos. They proposed to use onset and offset detection first to identify each note event in a guitar solo track. The statistical values (*e.g.* minimum, maximum, mean, or median) of frame-level spectral features over the duration of each note event were then extracted and fed to a pre-trained classifier for playing technique recognition. Using a multi-class SVM, they obtained 83% average accuracy in recognizing the following 6 classes: `Normal`, `Bend`, `Slide`, `Vibrato`, `Harmonics`, and `Dead notes`. Lower recall rates were found for `Slide`, `Vibrato`, and `Bend`: they were 50.9%, 66.7%, and 71.3%, respectively.

The work by Kehling et al. (2014) represents an important step forward in playing technique detection, but we note that their approach has a few limitations. First, using the whole note event as a fundamental unit in classification cannot deal with techniques that are concerned with the transition between successive notes, such as `Hammer-on` and `Pull-off`, which are widely used in guitar playing. Second, extracting features from the whole note may include information irrelevant to techniques that appear only in a small portion of the note event. Third, existing techniques for onset and offset detection may not be robust to timbre variations commonly seen in guitar performance. We attempt to address these limitations by using the pitch contour for localization, and by associating playing techniques with temporal regions in the pitch contour, not with individual note events.

This work is extended from Chen et al. (2015), which focuses only on playing technique detection. We propose new methods for playing technique localization, use CNN instead of SVM for playing technique recognition, and design methods to incorporate the result of playing technique detection for improving note tracking.

3. Proposed Model

Figure 2 depicts the flowchart of the proposed model, which consists of three stages. Firstly, we extract the pitch contour from the given audio track. Second, the pitch

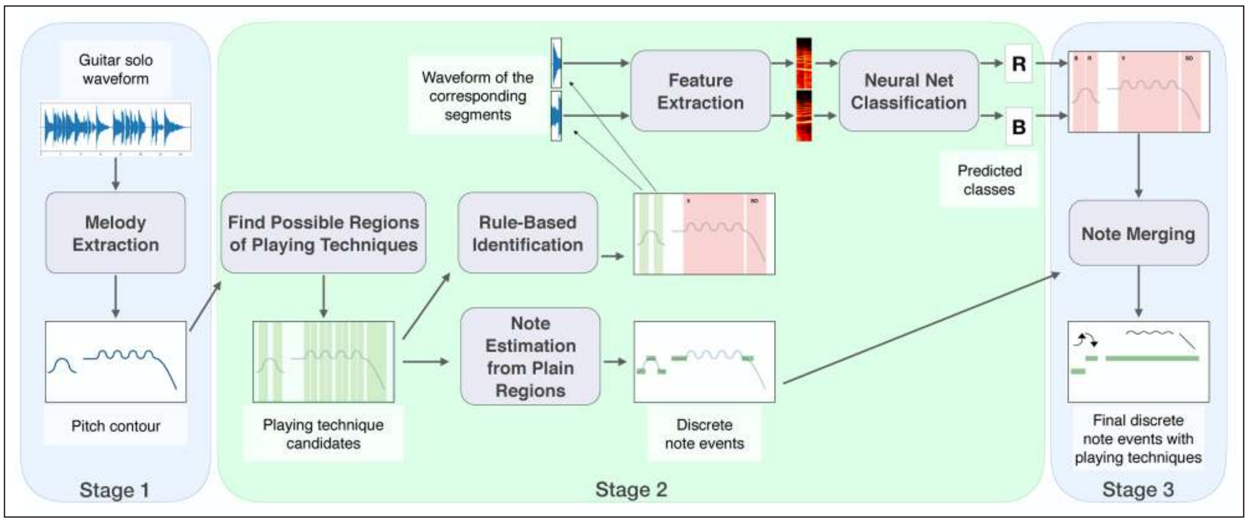

Guitar solo waveform
Melody Extraction
Pitch contour
Stage 1
Find Possible Regions of Playing Techniques
Playing technique candidates
Waveform of the corresponding segments
Rule-Based Identification
Note Estimation from Plain Regions
Discrete note events
Feature Extraction
Neural Net Classification
R
B
Predicted classes
Stage 2
Note Merging
Final discrete note events with playing techniques
Stage 3

Figure 2: Flowchart of the proposed model for solo guitar transcription. The three stages are: 1. melody extraction, 2. playing technique detection and note tracking, and 3. note merging.

contour and the audio are used to localize possible playing techniques and to get the initial result of note tracking. The candidate regions of playing techniques then go through a combination of rules and a CNN classifier for playing technique recognition. Finally, we use the result of playing technique detection to refine the result of note tracking. We call this process 'note merging' in **Figure 2**.

We can use existing methods for the first stage, but new methods are required for the last two. Below, we give details of these three stages.

3.1 Melody Extraction

We adopt and empirically compare the performance of the following three melody extraction methods in this work: Peeters' method (Peeters, 2006), pYIN (Mauch and Dixon, 2014), and Melodia (Salamon and Gómez, 2012). For **Melodia**, we use the implementation provided by Essentia (Bogdanov et al., 2013). All of them have been shown effective for monophonic F0 estimation. Also, implementations of these methods are publicly available.

We note that melody extraction can be considered as a monophonic F0 detection task. However, while the output of standard F0 detection or note tracking usually involves discrete F0 expressed in semitones, the desired output for our task is a quasi-continuous curve in Hertz.

3.2 Playing Technique Detection

After getting the pitch contour, the next step is to detect the note events and their corresponding playing techniques from the pitch contour. This is the most crucial stage and the main contribution of TENT. To better illustrate the following processes, we show an example of TENT in action step-by-step in **Figure 3**.

3.2.1 Sub-melodies

First, the pitch contour is segmented into several *sub-melodies* at points where the differences in frequency (in Hertz or in semitones) between adjacent frames are higher than a pre-defined value. This is to ensure that each sub-melody is a continuous curve. For example, we intend to segment the pitch contour shown in **Figure 3(a)** into two sub-melodies. In our implementation, we found that setting the threshold value to 0.5 semitones works well. Ideally, we expect this threshold to be 1 since a semitone is the pitch unit in Western music. However, it is possible that the pitch is shifted at the start or at the end of a note. Therefore, we gave a 0.5 semitone tolerance and set the threshold to 0.5. Moreover, we found that noises and errors in melody extraction would lead to overly short sub-melodies. To get rid of them, we discard sub-melodies that are shorter than 0.1 seconds. This is because a note is generally longer than 0.1 seconds.

Since we cut the pitch contour into sub-melodies using 0.5 semitones as the threshold, it is likely that each sub-melody corresponds to a note event, and accordingly, the cutting point between two sub-melodies corresponds to the transition between two note events. Playing techniques such as `Hammer-on` and `Pull-off` may be used during such note transitions. We consider cutting points with frequency difference less than 3.5 semitones as *candidates* of playing technique recognition (i.e., to decide whether it is a `Hammer-on`, `Pull-off`, or `Normal`). We set the threshold here to 3.5 semitones because larger intervals during note transitions are not very common for these playing techniques, due to the limit imposed by human hand size and the tension of general guitar strings.

The other playing techniques may occur within the sub-melodies, including `Slide`. The use of `Slide` may not introduce sudden frequency changes larger than 0.5 semitones, so it is possible for a sub-melody to contain two note events, when there is `Slide` in between. We describe next how we use the slope of the sub-melodies to recognize these playing techniques.

3.2.2 Patterns

A sub-melody may contain segments with positive or negative slope. The slopes of the segments within sub-melodies is important, since different techniques modulate the pitch in different ways. For example, `Slide` is

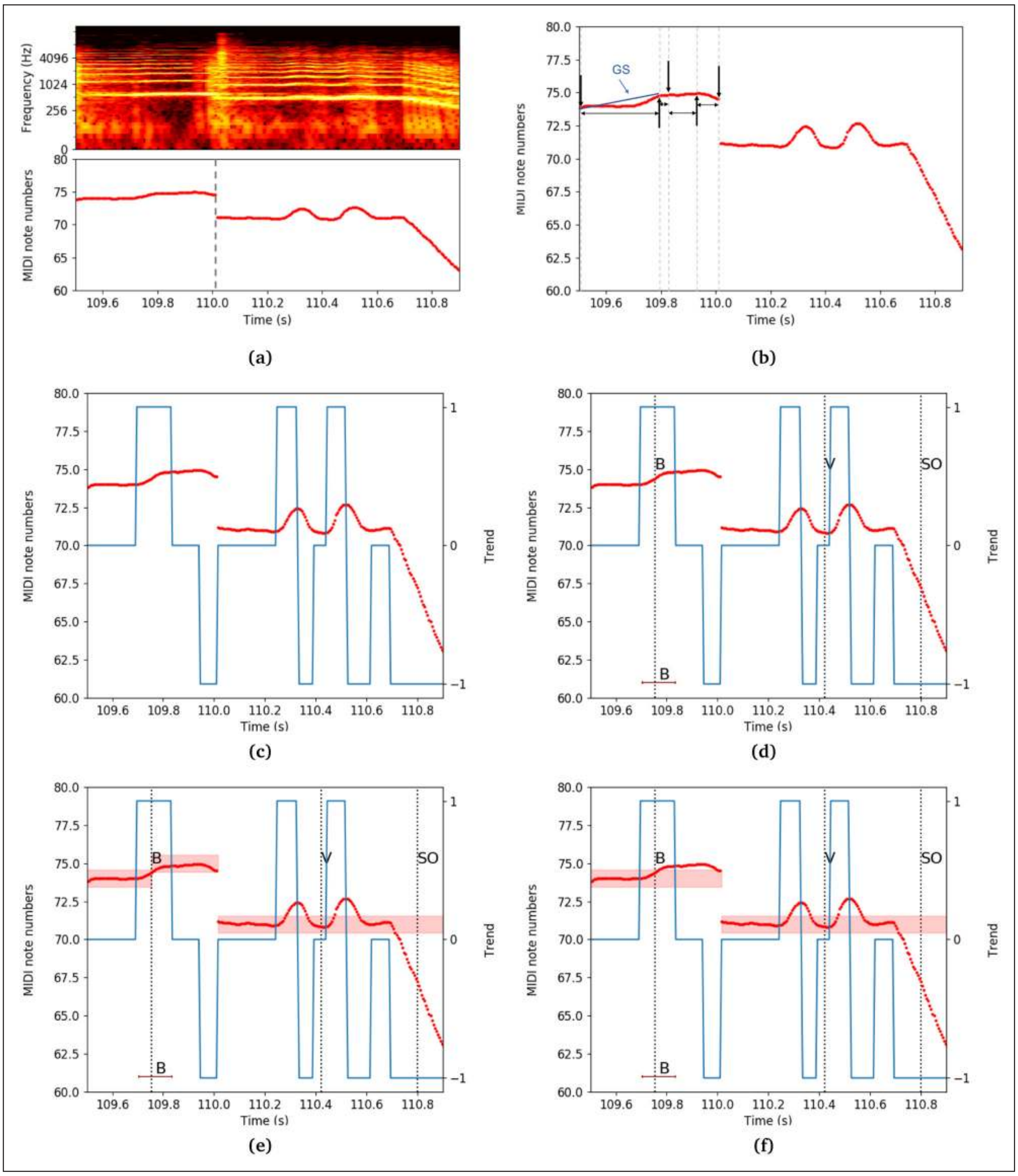


Figure 3: A running example of TENT in action. 3a): Cut melody into sub-melodies. The upper half shows the mel-spectrogram, and the lower half is the pitch contour. The vertical dash line denotes the border of the two sub-melodies. **3b): Find local extrema and calculate the slope of each pattern.** The vertical arrows indicate the local maximum or minimum, and the intervals between two adjacent extrema are referred to as patterns. GS means the slope of a pattern. **3c): Get trend.** The blue line represents the trend labels, with values 1, 0, or −1. **3d): Get techniques by rules and a classifier.** In this example, a `Bend` (B), a `Vibrato` (V), and a `Slide-out` (SO) are detected. `Bend` is decided by a CNN classifier, whereas `Vibrato` and `Slide-out` are recognized by rules. **3e): Get the initial estimate of note tracking.** The pink highlights are the estimated note events before merging. **3f): Merge note events.** The adjusted red highlights are the final note events.

characteristic of continuous pitch ascending or descending, whereas `Vibrato` would lead to segments with alternating positive and negative slopes.

We analyze the slopes of the segments within a sub-melody in the following way for playing technique detection. First, we partition a sub-melody into several *patterns* by finding points with local maxima or minima in frequency, as **Figure 3(b)** illustrates. We discard a pattern if it is shorter than 0.045 seconds since those are usually due to noise in the pitch contour.

3.2.3 Trend Labels
After obtaining the patterns, we calculate the slope for each pattern by dividing the absolute difference in frequency of the two end points by the length of the pattern. Then, each time point of a pattern is assigned with one of the following three *trend labels*: *ascending* (+1), *descending* (–1), or *none* (0). If the difference in frequency between a point and the next point is positive (negative) and if the absolute value of the difference is larger than α times the slope of that pattern, we label the time point as ascending (descending). The *slope parameter* α is a pre-defined value within [0,1]. We will evaluate its effect in Section 5.2.

3.2.4 Segments
The result of the trend analysis mentioned above is a sequence of trend labels, as **Figure 3(c)** illustrates. We can then divide a sub-melody into a number of *segments* at time points where the trend label changes. The regions with non-zero trends are called *segments*. In this way, we ensure that all the time points within a segment share the same trend label. For example, the first sub-melody in **Figure 3** would have four segments with labels 0, +1, 0, –1, respectively.

The core idea is then use segments with no trend (i.e., trend label 0) for predicting the F0 of the corresponding note events. Since there is little pitch variation for those segments, we may use a simple method for F0 estimation here to improve efficiency. On the other hand, there might be playing techniques in ascending and descending segments. We consider a segment as a *candidate* of playing technique recognition, when the difference between the maximal and minimal frequency values of that segment is larger than a parameter β_1, which is set to 0.3 semitone. We refer to β_1 as a *playing technique candidate threshold* and will also empirically evaluate its effect in the experiment reported in Section 5.2.

3.2.5 Rule-based Playing Technique Recognition
We design the following rules to recognize playing techniques with obvious features:

- **Long Slides:** A segment is a long slide when the difference in frequency between the maximum and the minimum of the segment is larger than a pre-defined value, which is empirically set to 3.5 semitones. Again, it is set to 3.5 because, generally, no other techniques can be performed in such a large range due to the physical limitation of human hand size and the tension of guitar strings. If the slide appears in the beginning (or end) of a sub-melody, it is a `Slide-in` (or `Slide-out`). Otherwise, it is a `Slide`.
- **Long Bend/Release:** If the difference in frequency between the maximum and the minimum of the segment is smaller than 3.5 semitones but the temporal duration of the segment is long enough, we call it a long `Bend` or a long `Release`, depending on whether it is an ascending or descending segment. According to domain knowledge in guitar solo playing, we empirically set the minimal duration to be 0.2 seconds.
- **Vibrato:** A region with more than three continuously and alternately ascending or descending segments within a sub-melody is claimed to be a `Vibrato`, as **Figure 3(d)** shows. It can be seen that β_1 can also be interpreted as the minimal vibrato extent in our model, since all the `Vibrato` detected in this way will have vibrato extent larger than β_1.

3.2.6 Classification
As **Figure 2** shows, candidates of playing technique recognition will firstly go through a rule-based identification module (see Section 3.2.5), followed by a CNN classifier. The purpose of the CNN classifier is to use additional timbre features to recognize playing techniques that cannot be identified via rules from the pitch contour, including (short) `Bend`, (short) `Release`, `Hammer-on`, `Pull-off`, and (short) `Slide`. We actually have to distinguish between six classes, since we also need to consider `Normal`.

As we suppose all the `Vibrato` cases would be recognized by the rules, we consider a segment a candidate input to the CNN classifier only if the difference between the maximal and minimal frequency values of that segment is larger than a slightly larger *playing technique candidate threshold* β_2 (i.e., $\beta_2 > \beta_1$). We empirically set β_2 to 0.5 semitone but will also justify this choice through an experiment.

The candidates for technique recognition are either transient regions at note transitions (see Section 3.2.1) or segments with variable length (see Section 3.2.4). For simplicity, we expand or truncate the length of each candidate so that each of them becomes an audio slice of 0.14 second. The length 0.14 second is chosen because we found empirically that it is long enough to cover the length of most playing techniques as well as part of their previous and next note events. Moreover, it is not too long to involve two adjacent techniques (or transitions) in a candidate.

We then extract the following three types of features for these audio slices and use them as input to the CNN classifier. The sampling rate is fixed to 44,100 samples per second.

- Log Mel-spectrogram (LMSpec): Compute the the power spectrogram of the audio slices by short-time Fourier Transform (STFT), with half-overlapping windows each 512 samples in length. We then convert the frequency axis into the Mel scale, reducing the feature dimension from 256 to 128, and finally take the log values.
- Mel-frequency cepstral coefficients (MFCC): Take the Discrete Cosine Transform (DCT) of the log Mel-spectrogram and use the amplitudes of the resulting spectrum as the features. Such MFCC features are commonly used in timbre-related audio processing tasks such as speech recognition and instrument recognition. We take the first 13 MFCCs and their deltas and delta-deltas, summing up to 39 features per frame.
- Pitch contour features (PC): We also use values of the normalized pitch contour and its first derivative as features. We normalize the pitch contour of the candidates by z-score normalization, to discard pitch information but retain information regarding the shape of pitch contour.

LMSpec and MFCC are computed with the LibROSA library (McFee et al., 2015). We will compare the performance of different combinations of these features for playing technique recognition in Section 5.1.

We use a CNN to build the classifier. Instead of using only one classifier, we train two classifiers, one for segments/note transitions with ascending pitch, and the other for those with descending pitch. The former classifies `Normal`, `Slide`, `Bend` and `Hammer-on`, whereas the latter classifies `Normal`, `Slide`, `Release` and `Pull-off`. We call them the *ascending type classifier* and *descending type classifier*, respectively.

3.3 Note Tracking and Note Merging

Given a sub-melody and the trend labels, we can use the segments of the sub-melody that have no trend (we call such a segment a *flat segment* hereafter for convenience) to estimate the F0 of the corresponding note events. The onset and offset of a note event are decided according to the type of playing techniques employed at the two sides of that flat segment. For example, if there are two flat segments in a sub-melody and a candidate segment for playing technique recognition in between, we claim that there are two note events in the sub-melody. The midpoint of the offset of the first note event and the onset of the second note event will be assigned to the middle of the candidate segment between them. If there is a `Slide-in` at the beginning of this sub-melody, the onset of the first note event will be assigned to the onset of that `Slide-in`; otherwise, the onset will be the starting point of the flat segment. We do the same thing for offsets according to whether there is a `Slide-out` at the end of a sub-melody. An example result of this note tracking method is shown in **Figure 3(e)**.

We further use the results of playing technique recognition to refine the result of the aforementioned note tracking method, as illustrated in **Figure 3(f)**. Specifically, we embed the detected playing techniques in the previously estimated notes, and merge the adjacent note events with either Bends or `Releases` in between. If it is a `Bend`, the F0 of the new note event will be set to the F0 of the original first note event. On the other hand, if it is a `Release`, the F0 will be the same as that of the second note event. The merged and technique-embedded note events are the final output of TENT.

4. Experimental Setup

4.1 Datasets

We use a dataset collected from the CD accompanying a textbook written by Gill and Nolan (1997), or **G&N** dataset for short. It contains 42 unaccompanied monophonic electric guitar solo tracks. The timestamps of all the note events and the involved playing techniques were carefully annotated by an experienced electric guitar player (i.e., the second author of the paper), with the help of the guitar tablatures provided by the book.[1] The labels were then checked by another electric guitar player (i.e., the first author of this paper) to make sure every label is correct. This newly annotated dataset contains 1,113 note events, where 137 have `Bend/Release`, 63 have `Slide`, 70 have `Hammer-on`, 143 have `Pull-off`, 61 have `Vibrato`, and the others are `Normal`. The length of the tracks ranges from 20 to 40 seconds, amounting to 19 minutes and 31 seconds in total. The tracks were recorded using a standard tuned electric guitar with either clean tone or distortion sound effect.

To train and evaluate the playing technique classifier, we extracted 1,046 0.14-second audio clips from the 42 songs, where all audio clips contain a short period of either ascending or descending pitch variation. The 1,046 audio clips include all the techniques as well as 633 `Normal` transitions in G&N. For short, we will call this segmented dataset G&N-Seg. All these selected clips have obvious pitch variations between two adjacent note events. Since `Normal` largely outnumbers the other classes, we performed data augmentation for the other classes, by shifting their F0 by ±1 and ±2 semitones. Eventually, the augmented dataset has 685 `Bends/Releases`, 315 `Slides`, 350 `Hammer-ons`, and 715 `Pull-offs`. Since the classifiers do not have to classify `Vibrato`, we did not put `Vibrato` into G&N-Seg.

Please note that we do not distinguish between `Bend` and `Release` in G&N-Seg and consider them the same class. Yet, we can easily distinguish between the two classes for technique detection for G&N, using the trend type of the segments.

4.2 Evaluation Metrics

Since the number of samples is very limited, we performed stratified five-fold cross-validation in our experiments. To do this, the 42 songs of G&N are separated into 5 folds in a way that the number of each technique inside every fold is close to one another fold. Therefore, this five-fold structure can be applied to experiments of both playing technique classification and note tracking.

For evaluating the accuracy of the playing technique classifiers, we trained ten different models with the same set of hyperparameters on the G&N-Seg dataset and reported the average evaluation result. We recorded the precision, recall, and F-score, the harmonic mean of the first two. In this part, the dataset is composed of only note transitions, so there is no need for playing technique localization.

In the second part, the G&N dataset is used to evaluate the result of note tracking and playing technique detection both also in terms of their precision, recall, and F-score. First, we tuned the parameters of TENT, α, β_1, and β_2. For note tracking, we used the evaluation function from `mir_eval` (Raffel et al., 2014), considering the correctness of F0 and onset for each note event. For playing technique detection, we require that both the type of technique and the onset are correct. An onset estimate is considered correct if it is within 0.1 seconds of the ground-truth time marks.

We note that the default threshold used by `mir_eval` in onset evaluation is 0.05 seconds, but we found this value is too small to cope with the ambiguity created by the use of longer-duration playing techniques, such as Bend or Slide, between two note events. In this situation, the offset of the first note and the onset of the second note depend on how we define the way to split the notes. If the playing technique is 0.12 seconds long, then splitting them at the beginning of the playing technique will have 0.12-second difference than splitting at the end, causing the onset and offset of the notes to be very different. Therefore, we chose a larger but still endurable tolerance range. Also to overcome this problem, two notes are split at the center of the playing techniques in our implementation.

After parameter tuning, the best set of parameters is compared with the note tracking function of Tony (Mauch et al., 2015), given the same F0 contours, in three different conditions: considering the correctness of onset only, the correctness of both onset and F0 (i.e., 'onset+F0'), and the correctness of onset, F0 and offset (i.e., 'onset+F0+offset'). An offset estimate is considered correct if its distance from the corresponding ground-truth time mark is less than 0.05 seconds, or 20% of the duration of the note event (whichever is larger).

4.3 Implementation Details of the Classifier

For the classifier for playing technique recognition mentioned in Section 3.2.6, we tested both CNNs and the simpler multi-layer perceptrons (MLP; a.k.a. DNN) with different numbers of layers. **Table 1** shows the architecture of the four models we implemented: CNN2, CNN3, MLP4 and MLP7, where the last number represents the number of convolutional layers (for CNN) or the number of fully-connected layers (for MLP). We tried adding more layers but found doing so does not improve the recognition accuracy, possibly because the dataset is not that large.

As mentioned at the end of Section 3.2.6, we have two classifiers, the ascending and descending type classifiers. We use the same network architecture for both.

Table 1: Architecture of the four neural network based classifiers we implemented and evaluated for playing technique recognition. Notation: 'Conv(number of filters, filter size, stride size)' denotes a convolutional layer and its hyperparameters. 'Max-Pool(stride size)' and 'MeanPool(stride size)' are max-pooling and mean-pooling layers. 'FC(number of neurons)' is a fully-connected layer. All convolutional and fully-connected layers have 0.5 dropout rate. The total number of parameters for these four models are around 2.9M, 2.3M, 7.5M, and 11M, respectively.

CNN2	CNN3	MLP4	MLP7
Conv (256,3,1)	Conv (256,3,1)	FC (1800)	FC (1800)
MaxPool (2)	MaxPool (2)	FC (1800)	FC (1800)
Conv (128,3,1)	Conv (128,3,1)	FC (900)	FC (1800)
MeanPool (2)	MaxPool (2)	FC (900)	FC (1800)
FC (1800)	Conv (128,3,1)	FC (4)	FC (900)
FC (900)	MeanPool (2)		FC (900)
FC (4)	FC (1800)		FC (900)
	FC (900)		FC (4)
	FC (4)		

The input of the classifiers is a tensor of [batch size, number of features, number of frames]. For CNNs, we used 1D convolutions (along the time axis) instead of 2D convolutions. While 2D convolutions analyze the input data as a chunk and convolve on both spectral and temporal dimensions, the 1D convolutions might better capture frequency and timbral information in each time frame, as has been shown in previous work on music and sound classification (Liu and Yang, 2016; Chou et al., 2018). The activation function for all but the last layer is the rectified linear unit (ReLU), and that of the last layer is softmax. All convolutional and fully-connected layers have 0.5 dropout rate. We use cross-entropy as the cost function.

We note that all the four models listed in **Table 1** can only deal with fixed-length input. That is a major reason why we require the input audio slice to have a fixed length of 0.14 seconds. Future work may discard the fully-connected layers in the CNN to realize a so-called *fully-convolutional network* (Liu and Yang, 2016) to deal with variable-length input.

5. Results and Discussion

Our experiments have three parts. First, we evaluated the accuracy of playing technique detection using different features and different models. Second, we performed sensitivity tests for some key parameters of TENT for both playing technique detection and note tracking. Lastly, we compared the performance of TENT against existing methods for note tracking.

5.1 Playing Technique Recognition

We firstly compared the results for different features, using CNN2 as the classifier. **Table 2** shows that MFCC outperforms LMSpec. Moreover, combining the spectral

Table 2: Performance of CNN2 using different input features on the G&N-Seg dataset. The numbers in the parentheses of the latter three columns in this table, and Tables 3 and 4, are the standard deviation of the results of the five-fold cross validation.

Features (#Dim)	Precision	Recall	F-score
MFCC (39)	0.771	0.752	0.759
	(±0.059)	(±0.058)	(±0.057)
MFCC+PC (41)	**0.809**	**0.792**	**0.797**
	(±0.054)	(±0.050)	(±0.050)
LMSpec (128)	0.718	0.670	0.686
	(±0.044)	(±0.028)	(±0.032)
LMSpec+PC (130)	0.720	0.667	0.683
	(±0.045)	(±0.023)	(±0.029)
MFCC+LMSpec (167)	0.769	0.749	0.757
	(±0.035)	(±0.034)	(±0.034)
All (169)	0.778	0.760	0.767
	(±0.040)	(±0.029)	(±0.033)

representation MFCC with the pitch contour feature PC leads to the highest F-score 0.7972, showing that the two features are complementary and useful for the task. Therefore, we use MFCC+PC as the input features hereafter.

We then compared the four architectures listed in **Table 1**. **Table 3** shows that CNNs perform generally better than MLPs as expected, since CNNs can better learn local spectral-temporal patterns in audio signals. The results of CNN2 and CNN3 are comparable, so we choose CNN2 as the classifier hereafter.

Table 4 and **Figure 4** show the per-class result and the confusion matrix. Our classifier performs generally well for different classes, but it does not work well for `Slide`. From **Figure 4** we see `Slide` can be easily confused with `Bend/Release`. There are two possible reasons for this. First, `Slide` is indeed acoustically similar to `Bend/Release`; even humans have to listen carefully to distinguish them. Second, `Bend/Release` are more often used than `Slide` in guitar solo, so there might be a class imbalance problem.

Empirically we found that the performance of note tracking is more sensitive to errors in detecting `Bend/Release` than in detecting `Slide`. Therefore, the low accuracy of our classifier (CNN2 MFCC+PC) in detecting `Slide` does not hurt the performance of note tracking much. However, when the target is to create symbolic notation for guitar solos, future work is needed to improve the accuracy for `Slide`.

Table 3: Performance of different models on the G&N-Seg dataset using MFCC+PC as the input features.

Model	Precision	Recall	F-score
CNN2	0.809	**0.792**	**0.797**
	(±0.054)	(±0.050)	(±0.050)
CNN3	**0.812**	0.784	0.793
	(±0.053)	(±0.049)	(±0.050)
MLP4	0.778	0.7534	0.762
	(±0.057)	(±0.055)	(±0.053)
MLP7	0.780	0.759	0.765
	(±0.053)	(±0.052)	(±0.049)

Table 4: Per-class result of CNN2 MFCC+PC model.

Technique	Precision	Recall	F-score
Bend/Release	0.714	0.829	0.767
	(±0.088)	(±0.108)	(±0.086)
Hammer-on	0.644	0.723	0.681
	(±0.331)	(±0.222)	(±0.281)
Normal	0.933	0.832	0.880
	(±0.044)	(±0.070)	(±0.045)
Pull-off	0.665	0.810	0.730
	(±0.089)	(±0.118)	(±0.074)
Slide	0.383	0.394	0.388
	(±0.113)	(±0.159)	(±0.132)
All	0.809	0.792	0.797
	(±0.054)	(±0.050)	(±0.050)

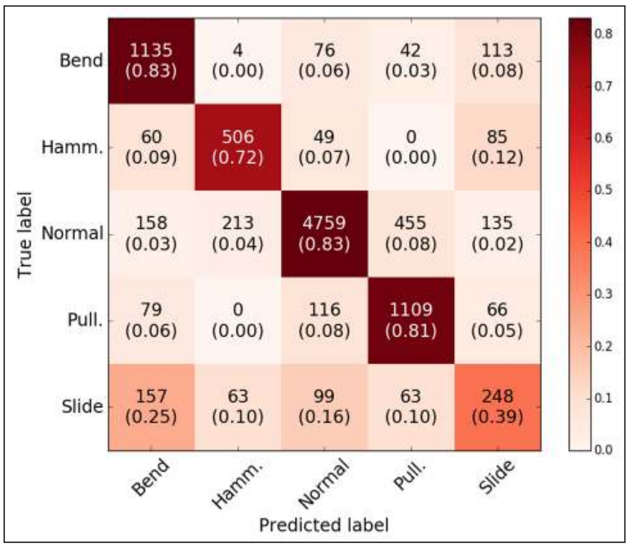


Figure 4: Confusion matrix of the 10 CNN2 MFCC+PC models, aggregated over all the evaluation runs. Each box shows the total classified number and the recall rate.

5.2 Parameter Sensitivity Test

As described in Section 3, TENT has a few parameters. While some of them are set according to the characteristics of playing techniques or musical domain knowledge, others can be set flexibly. Here, we varied the values of the three key parameters α, β_1 and β_2 and investigated how they affect the results of playing technique detection and note tracking. The default values we used for α, β_1 and β_2 in this work were 0.05, 0.3 and 0.5, respectively. We justify this choice by examining the results presented in **Table 5**.

The slope parameter α affects the assignment of trend labels. When the value is close to 0 (1), a frame would be more (less) likely labeled with a non-zero trend (i.e., +1 or –1). **Table 5** shows that setting α smaller leads to slightly better results for recognizing `Hammer-on` and `Slide`. However, in general the note tracking results are not sensitive to the value of α. This is because changing α only slightly changes the length of a segment, but usually the changes are not large enough to affect the result of the rule-based recognition algorithm. Furthermore, since we fix the length of technique classification candidates to 0.14 seconds, a mild change in the segment length would not affect the results much.

The parameters β_1 and β_2 control how many audio slices would be considered as candidates for playing technique recognition by rules and the classifier, respectively. The first one, β_1, can also be interpreted as the minimal vibrato extent expected by our model. **Table 5** shows that setting β_1 to 0.3 semitones leads to better results in detecting `Vibrato` as well as other techniques. The second one, β_2, affects the detection of all other techniques. In theory, we can set β_2 to 1 semitone, since all these playing techniques would involve at least two note events that are different in F0. However, due to errors in melody extraction and inaccuracy of human performance, setting β_2 smaller than 1 increases tolerance to such errors and leads to better results in practice. **Table 5** shows that setting β_2 smaller improves the accuracy of detecting `Bend` and `Release`, which often involve pitches that are one semitone apart, but setting β_2 closer to 1 performs better for other techniques, in particular `Slide`.

From **Table 5** we see that smaller α and moderate β_1 and β_2 lead to better results for note tracking. Although setting β_2 to 0.8 seems slightly better, we decided to set it to 0.5 for better technique recognition.

5.3 Note Estimation
We then evaluate the performance of TENT for note tracking against the note tracking function of Tony (Mauch et al., 2015), a widely used tool for pitch analysis in the MIR community. It is expected that TENT would perform better, since Tony is not specifically designed for guitar music and it does not take playing techniques into account. We evaluate the combination of the three melody extraction methods described in Section 3.1 with Tony.

Table 6 shows the result. We can see the importance of having a good pitch contour. Peeters' algorithm gets a lower score, because it is more sensitive to the distortion or the clicking sounds in the guitar solo recordings, creating many false alarms. When using Tony, pYIN and Melodia perform comparably in terms of F-score across all the three scenarios. However, Melodia outperforms pYIN when we use TENT as the note tracking method.

Table 6 also shows that TENT greatly outperforms Tony across all the three scenarios as expected. For the 'onset+F0+offset' scenario, the F-score is improved from 0.5726 to 0.6879 if we replace Tony by TENT.

A closer look into the results reveals that the performance gap is largely due to the note merging operation presented in Section 3.3. There are 145 `Bend/Release` and 72 `Vibrato` among the 1,113 note events of the G&N dataset. These three playing techniques affect almost 20% of the notes. For such note events, a general-purpose method such as Tony would naturally split them into several note events, due to the changes in pitch caused by the playing techniques. Because TENT is based on a different definition of note events, it is free of such false positives, and greatly outperforms Tony in the precision rate, as **Table 6** shows.

5.4 Example Results
We use three examples to demonstrate our results. **Figures 5a** and **5b** show that TENT performs well in recognizing `Bend`, `Release`, and `Vibrato`. In these two examples, most of the playing techniques are correctly

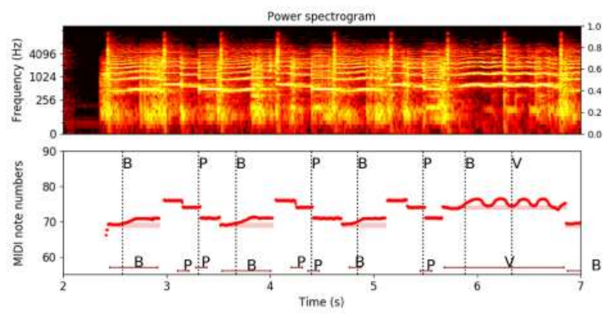
Table 5: Sensitivity test for the following three parameters of TENT: the slope parameter α, and the first and second playing technique candidate thresholds β_1 and β_2 (in semitones). We show the F-scores of playing technique recognition within the note tracking experiment and also the onset+F0 F-score for note tracking over the G&N dataset. Here, the default values of (α, β_1, β_2) are (0.5, 0.3, 0.5), although we use (0.05, 0.3, 0.5) in other experiments in the paper.

	α			β_1			β_2		
	0.05	**0.5**	**0.95**	**0.1**	**0.3**	**0.5**	**0.3**	**0.5**	**0.8**
Bend+Release	**0.606**	0.602	0.602	0.570	**0.602**	0.585	**0.633**	0.602	0.561
Hammer-on	**0.606**	0.544	0.556	**0.561**	0.544	**0.561**	0.333	**0.544**	0.500
Pull-off	0.577	**0.578**	0.526	0.527	**0.578**	0.570	0.527	**0.578**	0.574
Slide	**0.414**	0.355	0.362	0.327	**0.355**	0.333	0.021	0.355	0.491
Vibrato	0.675	0.688	**0.711**	0.463	**0.688**	0.648	0.652	**0.688**	0.588
Note Tracking (onset+F0)	**0.859**	0.858	0.856	0.835	**0.858**	0.851	0.834	0.858	**0.862**

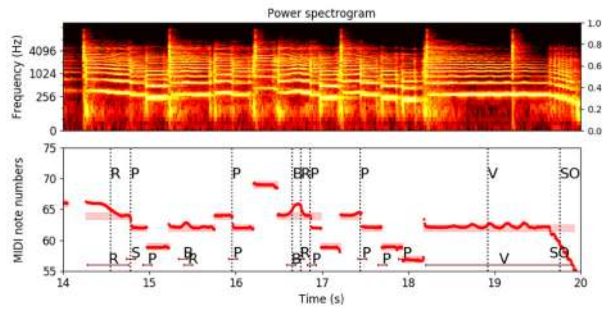
Table 6: Performance of different methods for note tracking on the G&N dataset, in terms of precision, recall, and F-score. The evaluated methods are Peeters (2006), pYIN (Mauch and Dixon, 2014), Melodia (Salamon and Gómez, 2012), the note tracking function of Tony (Mauch et al., 2015), and the proposed TENT model.

Melody extraction method	**Note tracking method**	**Onset + F0 + Offset**			**Onset + F0**			**Onset**		
		Prec.	**Recall**	F_1	**Prec.**	**Recall**	F_1	**Prec.**	**Recall**	F_1
Peeters	Tony	0.384	0.645	0.481	0.517	0.869	0.648	0.542	0.912	0.680
pYIN	Tony	0.538	0.659	0.592	0.674	0.826	0.742	0.729	0.893	0.803
Melodia	Tony	0.499	0.671	0.573	0.652	**0.877**	0.748	0.679	**0.913**	0.779
Peeters	TENT	0.477	0.562	0.516	0.661	0.778	0.715	0.696	0.819	0.752
pYIN	TENT	0.518	0.607	0.559	0.743	0.872	0.802	0.779	**0.913**	0.841
Melodia	TENT	**0.679**	**0.697**	**0.688**	**0.841**	**0.877**	**0.858**	**0.879**	0.903	**0.891**

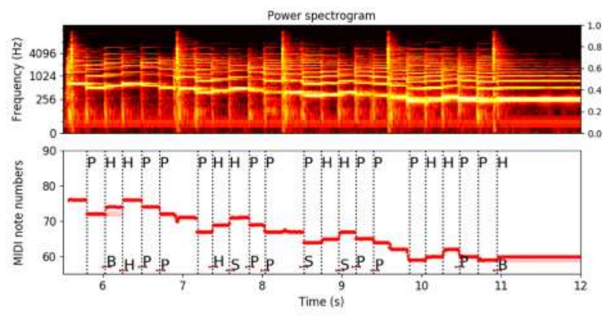
predicted. But, TENT may sometimes be too sensitive to pitch changes. For example, we see false alarms of `Bend` and `Release` for the arc-shape pitch contour between 15 and 16 seconds in **Figure 5b**. Moreover, TENT can not distinguish well between `Hammer-on`, `Slide`, `Pull-off`, and `Normal` transitions. **Figure 5c** shows that many of these techniques are wrongly predicted. We found that some cases are indeed challenging, even for human listeners. Moreover, in **Figure 5c**, TENT mistakes the `Hammer-on` between the second and third notes for `Bend` and wrongly merges the two notes. These results show that there is still much room for improvement. For more examples, please see the accompanying website.



(a) Example 1: lick_7.wav



(b) Example 2: lick_25.wav



(c) Example 3: lick_73.wav

Figure 5: In these examples, the upper image is the power spectrogram. The red lines on the lower graph are the pitch contours. Characters above the pitch contours as well as the vertical dotted lines are the ground-truth labels, and characters below are the predicted labels. Behind the pitch contours, there are some pink horizontal highlights (best viewed in color) that represent the note events.

6. Conclusion and Future Work

In this paper, we have shown that note tracking and playing technique detection can benefit each other when considered jointly. We can highlight the potential areas of playing techniques by tracking the variation of a pitch contour. On the other hand, playing technique detection reduces the false alarms in note tracking. Moreover, as TENT transcribes the F0, onset, offset and all the playing techniques involved in playing each note, it can be used to generate symbolic notation for electric guitar solo. Also, as we have kept the computation of TENT simple, it is possible that such a transcription can be made in real-time, after some more optimization.[2]

In TENT, we use hand-crafted rules to detect long playing techniques such as `Slide-in`, `Slide-out` and `Vibrato`. The candidate selection rules are also human-defined, making TENT sensitive to errors from melody extraction. Moreover, we do not consider the relations between adjacent playing techniques while recognizing them, which may lead to poor fingering arrangement. In the future, we intend to replace the rules by machine learning-based methods, and to build a language model of playing techniques that consider the playability of the resulting notation. We can also replace the melody extraction part with a neural network that detects F0 and techniques together. With more labeled data (e.g., by using the recently-released GuitarSet (Xi et al., 2018)) and more advanced network architectures (e.g., fully-convolutional or recurrent networks (Nam et al., 2019)), it is possible to further improve the accuracy of playing technique detection. We are also interested in implementing a guitar solo detector so that the model may be applied directly to a complete piece of guitar performance.

Notes

[1] The guitar tablatures in the book labelled all of the techniques. Therefore, we did not have to classify the techniques from the audio files by ourselves.

[2] Currently, with a 2.4GHz Intel Core i5 CPU, TENT, with Melodia for melody extraction, only needs about 1.2 seconds to process a 4-minute audio file. The time measured here excludes the loading time of Python packages and pre-trained classifier model files since these can be done beforehand in a real-time service.

Acknowledgement

This work was supported by a grant from the Ministry of Science and Technology of Taiwan under contract MOST 105-2221-E-001-019-MY2.

Competing Interests

The authors have no competing interests to declare.

References

Abeßer, J., Lukashevich, H., & **Schuller, G.** (2010). 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. DOI: https://doi.org/10.1109/ICASSP.2010.5495945

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & **Klapuri, A.** (2013). Automatic music transcription: challenges and future directions. *J. Intelligent Information Systems*, *41*(3), 407–434. DOI: https://doi.org/10.1007/s10844-013-0258-3

Bittner, R. M., Salamon, J., Essid, S., & **Bello, J. P.** (2015). Melody extraction by contour classification. In *Proc. International Society for Music Information Retrieval Conference*, pages 500–506.

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J., & **Serra, X.** (2013). Essentia: An audio analysis library for music information retrieval. In *Proc. International Society for Music Information Retrieval Conference*, pages 493–498. [Online] http://essentia.upf.edu. DOI: https://doi.org/10.1145/2502081.2502229

Chang, S., & **Lee, K.** (2014). A pairwise approach to simultaneous onset/offset detection for singing voice using correntropy. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 629–633. DOI: https://doi.org/10.1109/ICASSP.2014.6853672

Chen, S.-H., Lee, Y.-S., Hsieh, M.-C., & **Wang, J.-C.** (2018). Playing technique classification based on deep collaborative learning of variational autoencoder and Gaussian process. In *Proc. IEEE International Conference on Multimedia and Expo*. DOI: https://doi.org/10.1109/ICME.2018.8486467

Chen, S.-H., Wu, S.-H., Lee, Y.-S., Lo, R., & **Wang, J.-C.** (2017). Hierarchical representation based on Bayesian nonparametric tree-structured mixture model for playing technique classification. In *Proc. ACM Multimedia Thematic Workshops*, pages 537–543. DOI: https://doi.org/10.1145/3126686.3126757

Chen, Y.-P., Su, L., & **Yang, Y.-H.** (2015). Electric guitar playing technique detection in real-world recordings based on F0 sequence pattern recognition. In *Proc. International Society for Music Information Retrieval Conference*, pages 708–714.

Cheng, T., Dixon, S., & **Mauch, M.** (2015). Improving piano note tracking by HMM smoothing. In *Proc. European Signal Processing Conference*, pages 2054–2058. DOI: https://doi.org/10.1109/EUSIPCO.2015.7362736

Chou, S.-Y., Jang, J.-S., & **Yang, Y.-H.** (2018). Learning to recognize transient sound events using attentional supervision. In *Proc. International Joint Conference on Artificial Intelligence*, pages 3336–3342. DOI: https://doi.org/10.24963/ijcai.2018/463

Dattorro, J. (1997). Effect design, part 2: Delay line modulation and chorus. *J. Audio Engineering Society*, *45*(10), 764–788.

de Cheveigné, A., & **Kawahara, H.** (2002). YIN: A fundamental frequency estimator for speech and music. *J. Acoustical Society of America*, *111*(4), 1917–1930. DOI: https://doi.org/10.1121/1.1458024

de Haas, W. B., Magalhães, J. P., & **Wiering, F.** (2012). Improving audio chord transcription by exploiting harmonic and metric knowledge. In *Proc. International Society for Music Information Retrieval Conference*, pages 295–300.

Dzhambazov, G., Holzapfel, A., Srinivasamurthy, A., & **Serra, X.** (2017). Metrical-accent aware vocal onset detection in polyphonic audio. *arXiv preprint arXiv:1707.06163.*

Fohl, W., & **Meisel, A.** (2012). A feature relevance study for guitar tone classification. In *Proc. International Society for Music Information Retrieval Conference*, pages 211–216.

Gill, D., & **Nolan, N.** (1997). *Rock Lead Basics: Master Class Series*. Musicians Institute Press.

Kehling, C., Abeßer, J., Dittmar, C., & **Schuller, G.** (2014). Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters. In *Proc. International Conference on Digital Audio Effects*, pages 219–226.

Kim, J. W., Salamon, J., Li, P., & **Bello, J. P.** (2018). CREPE: A convolutional representation for pitch estimation. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 161–165. DOI: https://doi.org/10.1109/ICASSP.2018.8461329

LeCun, Y., Bottou, L., Bengio, Y., & **Haffner, P.** (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, *86*(11), 2278–2324. DOI: https://doi.org/10.1109/5.726791

Liu, J.-Y., & **Yang, Y.-H.** (2016). Event localization in music auto-tagging. In *Proc. ACM International Conference on Multimedia*, pages 1048–1057. DOI: https://doi.org/10.1145/2964284.2964292

Mauch, M., Cannam, C., Bittner, R., Fazekas, G., Salamon, J., Dai, J., Bello, J., & **Dixon, S.** (2015). Computer-aided melody note transcription using the Tony software: Accuracy and efficiency. In *Proc. International Conference on Technologies for Music Notation and Representation*, pages 23–30.

Mauch, M., & **Dixon, S.** (2014). pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 659–663. DOI: https://doi.org/10.1109/ICASSP.2014.6853678

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & **Nieto, O.** (2015). LibROSA: Audio and music signal analysis in Python. In *Proc. Python in Science Conference*, pages 18–25. DOI: https://doi.org/10.25080/Majora-7b98e3ed-003

Nam, J., Choi, K., Lee, J., Chou, S.-Y., & **Yang, Y.-H.** (2019). Deep learning for audio-based music classification and tagging: Teaching computers to distinguish Rock from Bach. *IEEE Signal Processing Magazine*, *36*(1), 41–51. DOI: https://doi.org/10.1109/MSP.2018.2874383

Nishikimi, R., Nakamura, E., Itoyama, K., & **Yoshii, K.** (2016). Musical note estimation for F0 trajectories of singing voices based on a Bayesian semi-beat-synchronous HMM. In *Proc. International Society for Music Information Retrieval Conference*, pages 461–467.

Peeters, G. (2006). Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 53–56. DOI: https://doi.org/10.1109/ICASSP.2006.1661210

Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., & **Ellis, D. P.** (2014). mir_eval: A transparent implementation of common MIR metrics. In *Proc. International Society for Music Information Retrieval Conference*, pages 367–372.

Reboursière, L., Lähdeoja, O., Drugman, T., Dupont, S., Picard-Limpens, C., & **Riche, N.** (2012). Left and right-hand guitar playing techniques detection. In *Proc. International Conference on New Interfaces for Musical Expression*, pages 7–10.

Salamon, J., & **Gómez, E.** (2012). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Trans. Audio, Speech & Language Processing*, *20*(6), 1759–1770. DOI: https://doi.org/10.1109/TASL.2012.2188515

Salamon, J., Gómez, E., Ellis, D. P., & **Richard, G.** (2014). Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, *31*(2), 118–134. DOI: https://doi.org/10.1109/MSP.2013.2271648

Salamon, J., Peeters, G., & **Röbel, A.** (2012). Statistical characterisation of melodic pitch contours and its application for melody extraction. In *Proc. International Society for Music Information Retrieval Conference*, pages 187–192.

Stein, M. (2010). Automatic detection of multiple, cascaded audio effects in guitar recordings. In *Proc. International Conference on Digital Audio Effects*, pages 4–7.

Su, L., Yu, L.-F., & **Yang, Y. H.** (2014). Sparse cepstral and phase codes for guitar playing technique classification. In *Proc. International Society for Music Information Retrieval Conference*, pages 9–14.

Xi, Q., Bittner, R. M., Pauwels, J., Ye, X., & **Bello, J. P.** (2018). GuitarSet: A dataset for guitar transcription. In *Proc. International Society for Music Information Retrieval Conference*, pages 453–460.

Yang, L., Maezawa, A., Smith, J. B. L., & **Chew, E.** (2017). Probabilistic transcription of sung melody using a pitch dynamic model. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 301–305. DOI: https://doi.org/10.1109/ICASSP.2017.7952166

How to cite this article: Su, T.-W., Chen, Y.-P., Su, L., & Yang, Y.-H. (2019). TENT: Technique-Embedded Note Tracking for Real-World Guitar Solo Recordings. *Transactions of the International Society for Music Information Retrieval,* 2(1), pp. 15–28. DOI: https://doi.org/10.5334/tismir.23

Submitted: 02 October 2018 **Accepted:** 27 February 2019 **Published:** 09 July 2019