GUITAR TUNING IDENTIFICATION

Varun Khatri[1] **Lukas Dillingham**[2]
Department of Electrical and Computer Engineering
University of Rochester
[1]vkhatri2@ur.rochester.edu, [2]ldilling@ur.rochester.edu

ABSTRACT

In this paper, we propose two methods to identify tuning on a guitar from its MIDI transcription. Recently, a lot of musicians have been using alternate tunings and defining new styles of guitar based music. However, it is not easy to transcribe, study and learn such pieces without the knowledge of what tuning the guitar has. To address this problem, we propose two methods: first, a supervised learning algorithm to identify the tuning from a fixed set of tunings using an LSTM-based neural network; second, a dynamic programming algorithm that chooses a tuning with minimum distance measure on the optimal set of note locations a song can have for the given tuning.

1. INTRODUCTION

Guitar, along with piano, has been one of the most popular musical instruments in the past 70 years. While it has existed in many forms in the past, the current version was designed circa 1850, and has remained the same since then. It started off being widely used in blues and gypsy jazz music and it gained popularity with rock and metal music, where its use has been very streamlined. Due to this, the evolution in guitar music has been very technology based and not instrument-dependent. However lately, a lot of musicians have started to alternate tunings for guitar. This requires thinking beyond the existing "knowledge bank" of guitar, which might include heavy usage of power chords, barre chords, pentatonic scale, etc. This is because the fingering and the chord shapes, that the guitarists are used to, are not applicable anymore to the new tuning.

In this paper, we attempt to perform the task of identification of guitar tuning from MIDI transcriptions of this song. This hasn't been done before, but a similar task was performed in [2] where the author attempted to identify active stops on a pipe organ using Non-negative Matrix Factorization. However, unlike organ, guitars are post-processed using pedals, amplifiers, etc., which creating a generic model to represent timbre is difficult. Hence, we stuck on using MIDI as input to out model.

To further understand the differences existing in different tunings, we break down the problem into three tasks: Chord recognition, key identification and interval analysis.

Chord Recognition is a useful task in guitar tuning identification as a different tuning enables the guitarist to use certain chord voicings, which may be unique or more probable in that tuning than other tunings. Chord voicing is the process of arranging the notes in a chord in a different order in order to obtain different sound. For example, the notes F, A and C make up the F major chord. If the lowest note is F, followed by A and then C, that is a regular F major. However, if A is the lowest note, followed by F and then C, we call it the first inversion and it sounds different. Composers often use these different voicings to make interesting compositions, but the probability of occurrence of different voicings is more in alternate tunings.

Next, some of the most popular alternate tunings are open chord tunings such as open C, open D, open G, etc., in which all strings are tuned to the notes of a particular chord (e.g., CGCGCE for open C). It can be assumed that a song using an open C would be in the key of C major. Hence, determining the key helps in asserting the guitar tuning used in the song.

Lastly, the intervals used in a note progression are observed to be different for different songs. Rock music involves heavy usage of pentatonic scale, which can be very ergonomically played in standard tuning. However, recently a lot of math roch/indie musicians are composing songs which use alternate tunings and these songs use different intervals, which make their compositions sound different. A probabilistic model exploiting the aforementioned ideas would perform well in this task. However, it's somewhat difficult to model. We approach this problem using an LSTM-based model, which uses MIDI information of the songs and identifies the tuning. We also propose a dynamic programming based approach which determines optimal note locations for different tunings and chooses the one which is least difficult. This is a difficulty-based model, which assumes that the most ergonomic fingering positions are always chosen over the more difficult ones.

The problem of guitar tuning identification is interesting as it not only helps learners pick up and study new songs, but also helps with automatic guitar tablature generation. Further, it will also help in analysing different styles of guitar music by learning mathematical representations of the styles, as tuning is something very integral to the instrument.

The rest of the paper is arranged as follows: In Section 2, we detail the process we followed to create a new dataset for this task using guitar tabs from Ultimate-Guitar (`https://www.ultimate-guitar.com`). In Section 3, we detail the proposed methods for this classification task. In Section 4, we discuss the results from initial testing of the software and in Section 5, we provide our

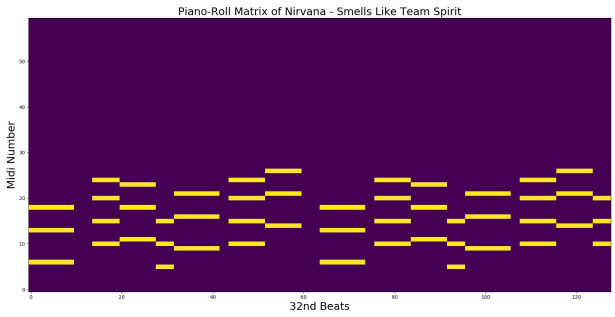

Figure 1. Figure captions should be placed below the figure.

conclusion and discuss future possibilities with this task.

2. DATASET

To prepare the dataset, around 90 *GuitarPro* files were collected for each tuning. The *GuitarPro* files were mostly downloaded from Ultimate-Guitar. *GuitarPro* files contain the tabulature and sheet music of the song. They contain information about guitar tracks as well as information for other instruments such as piano, drums, or even vocals. The open-source tabulature editor *TuxGuitar* was used to manage and inspect the *GuitarPro* files.

After being converted from guitar pro files to *.xml*, the python library Music21 was used to parse the files. Music21 essentially parses an *.xml* file and organizes a piece of music into classes of parts, measures, and notes, and also contains information such as the songs BPM and key, all of which can then be manipulated. From these *.xml* files matrices based on the notes in each measure were used as the input to the model. For every four measures, a 128x60 piano-roll matrix was created where 128 is four measures of 32^{nd} notes. 60 corresponds to the possible range of MIDI notes from the four tunings that were chosen with B1 being the lowest note and A#6 being the highest. For every note, a value of 1 is placed into the piano-roll matrix corresponding to its duration and MIDI number. Notes not being played and rests have a value of zero. These matrices were then labeled for training. An example of a piano-roll matrix can be seen in Figure 1.

Tuning	# Measures (x4)
Standard (EADGBE)	6523
Open D (DADF#AD)	5114
Open G (DGDBGD)	3292
Open C (CGCGCE)	6011

Table 1. Number of Four Measure Bars for Each Tuning in dataset

The piano-roll matrix is ideal for the input of the neural network as it preserves chord structures such as inversions as well as preserves the temporal information of the song. This multi-hot encoding is preferred here as providing a "note vector" would require an Embedding layer to encode the data. For example a C major chord could be represented as the midi notes (60, 64, 67), but would also require information about the starting beat and its duration, while the piano-roll matrix already contains this information.

To ensure that one tuning does not have vastly more information than others, the number of four bar segments were counted and it was made sure that the dataset is not too skewed. From the collected songs, we then divided the files to be trained, validated, and tested. for each tuning, 70% 0f the files were used for training, 15% was used for testing, and 15% was used for validation.

3. METHODOLOGY

3.1 Method 1: Deep Neural Networks

As mentioned earlier, we needed a method that could perform the tasks of key identification, interval analysis and chord recognition. This information needed to be interpreted from the multi-hot piano-roll that was generated using the *.xml* parser described earlier. An initial model, based on [1], used two-dimensional Convolutional Neural Network (CNN) which performs classification on single-measure inputs. The problem was perceived to be an image classification problem, in which the distances between the MIDI-note lines are intervals and could be interpreted as frequencies in the image. The offset of the MIDI-note would contribute to the key of the song. However, the time dependency between notes in a "piano-roll picture" was not captured by the CNN and hence, the CNN-based model didn't work well for this problem.

To solve that issue, we decided to use a Recurrent Neural Network (RNN), which would capture the time dependencies between the notes. The input to the network is a multi-hot encoded 32nd note frame. Since the input vector is already encoded, we do not use an embedding layer to generate an embedding. Instead, we use a single fully connected layer to represent the sparse, multi-hot data as float values. This is followed by an LSTM layer. The hidden state size for the LSTM layer is 128, which is 4 measures worth of data. In essence, the network looks back 4 measures and incorporates the information to do classification. The recurrent layer is followed by a fully connected layer and then the output. We used leaky ReLU as activations in both fully connected layers and softmax in the output to represent probabilities for each class.

3.2 Method 2: Dynamic Programming

Each note in a song can have up to six locations on the guitar fret board, and these locations vary for each tuning. By calculating the distance between note positions, it is possible to determine the optimal set of note locations for a song and in turn determine the optimal tuning for a song. An example of a three note sequence with possible note locations can be seen in Table 2.

The cost matrix for note i with possible positions j can is computed using the following formula:

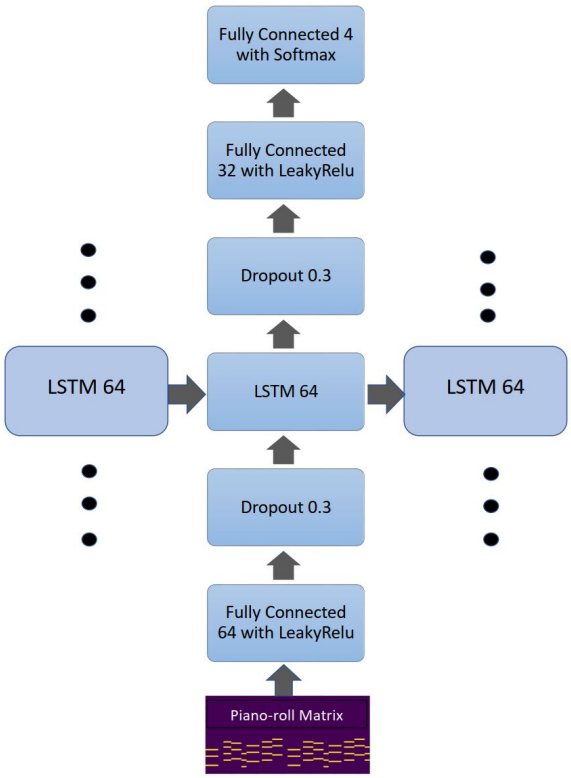

Figure 2. RNN Model

$$cost[i,j] = min(cost[i-1,k] + dist(x_{i,j}, x_{i-1,k})), \quad k \in \{1, ..., N\} \tag{1}$$

where N is the number of possible locations for note $i-1$ and $dist$ is a distance function defined by

$$dis(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 \text{ or } x_2 = 0 \\ \frac{|x_1 - x_2|}{(x_1 - x_2)\sqrt[12]{2}\, min(x_1, x_2)} & \text{otherwise} \end{cases} \tag{2}$$

This distance function accounts for open strings being easy to play by assigning it zero cost. Also, the same interval is easier to play higher frets, as they are smaller than lower frets. We used the ratio of the distances of two consecutive frets, $\sqrt[12]{2}$, or approx. 1.059463, to reduce the penalty for higher fret locations.

If in the sequence note i is a chord, all combinations of valid note locations in the chord are computed and the combination with the minimum distance between all notes is selected. The location of the chord is simply the mean of the x and y coordinates (approximately the center of mass) of all notes.

The cost of the optimal note location set is compared between all tunings to to then determine the optimal tuning.

	$i-1$	i	$i+1$
Possible Note Locations, j (string, fret)	(6, 20)	(6, 24)	(5, 22)
	(5, 15)	(5, 19)	(4, 17)
	(4, 10)	(3, 9)	(3, 12)
	(3, 5)	(2, 5)	(2, 8)
	(2, 1)	(1, 0)	(1, 3)

Table 2. Note Sequence and Possible Locations for Standard Tuning

4. RESULTS

We use normalized confusion matrices to visualize the performance of the system. The diagonals of the confusion matrices indicate the True Positive Rate (TPR).

Figure 3 shows the confusion matrix of the 48 songs used for testing. The results show that the architecture is successful in identifying the tuning of a song as three of the four turnings are successfully identified over 70% of the time. There are some problems however, as can be seen with the open G (CGCGCE) tuning. This problem is most likely due to the dataset being skewed, rather than a significant problem with the architecture. Currently open G has the least number of four measure bars out of the four tunings despite having a similar number of songs. One method to potentially improve the accuracy of the algorithm is to simply increase the data, and especially for the pen G tuning. A table showing the number of four bar measures can be seen in Table 1.

Summing all the predictions of four bar measures in a single song and taking the tuning with the most number of predictions is how the tuning of an entire song was predicted. It can be seen that the results for entire songs are quite better than those of single four measure bars. The confusion matrix for entire songs can be seen in Figure 4

The confusion matrix using dynamic programming can be seen in Figure 5. It must be noted that the dynamic programming method purely calculates the cost of a song and does not predict the tuning. The original tuning a song will often have the lowest cost, but this is not often the case. This could perhaps be a reason why there is confusion in standard tuning. For example, pop and rock songs often use power chords which would have a distance of zero in open D tuning.

5. FUTURE WORK AND CONCLUSION

The problem of identifying the tuning of a guitar from a set of notes is intriguing. We have shown that an RNN model can be trained using common patterns of notes and chords that occur in a given tuning. This in turn can predict the tuning for that set of notes to a decent level of success. The dynamic programming method also had good initial results, showing that an optimal tuning can be calculated for a set of notes in a song.

Our model does have some limitations, however. This can mainly be seen in the confusion matrix with the DADF#AD tuning. The dynamic programming method has some confusion in standard tuning as well. Future

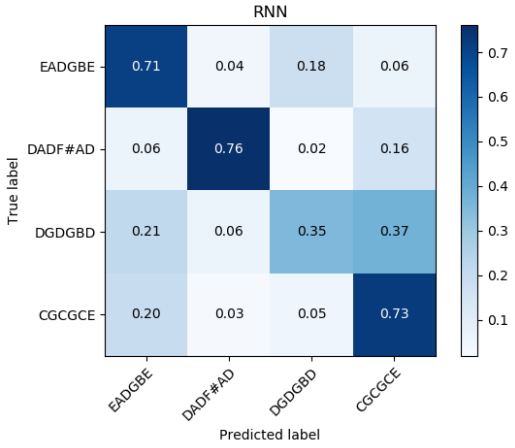

Figure 3. Confusion Matrix of Each Four Bar Measures.

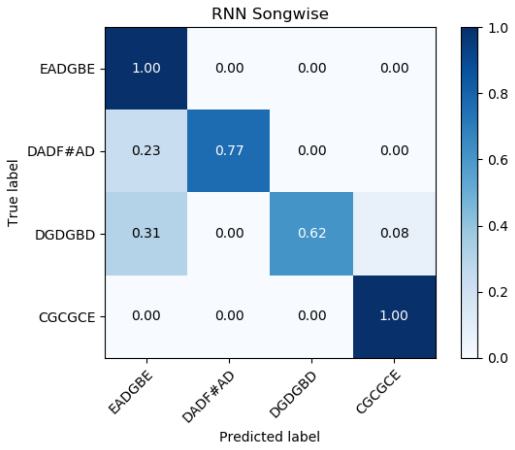

Figure 4. Confusion Matrix of per Song.

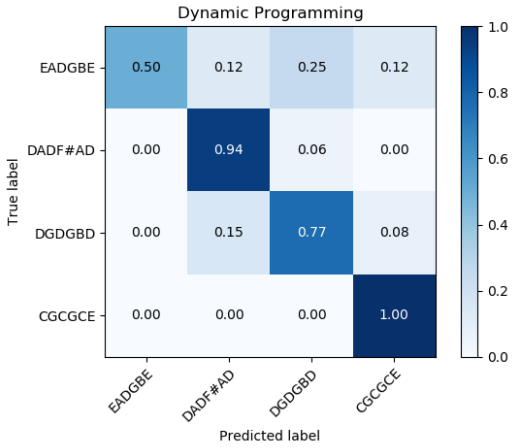

Figure 5. Confusion Matrix per Song using Dynamic Programming

work to improve our results could include collecting more data for the dataset. Adjusting the hyperparameters of the model could further improve performance as well. For dynamic programming, incorporating time between notes into the distance calculation as well as fine tuning the distance equation is something to consider. Incorporating an RNN model and dynamic programming into a single method could combine the strengths of both methods and lead to better results as well.

6. REFERENCES

[1] A. Maezawa B. Li and Z. Duan. Skeleton plays piano: online generation of pianist body movements from midi performance. In *Proc. International Society for Music Information Retrieval (ISMIR)*, 2018.

[2] A. Coe. Pipe organ stop identification via non-negative matrix factorization. In *Computer Audition, University of Rochester*, 2018.