Guitar Tuning Identification

Lukas Dillingham and Varun Khatri

Department of Electrical and Computer Engineering, University of Rochester

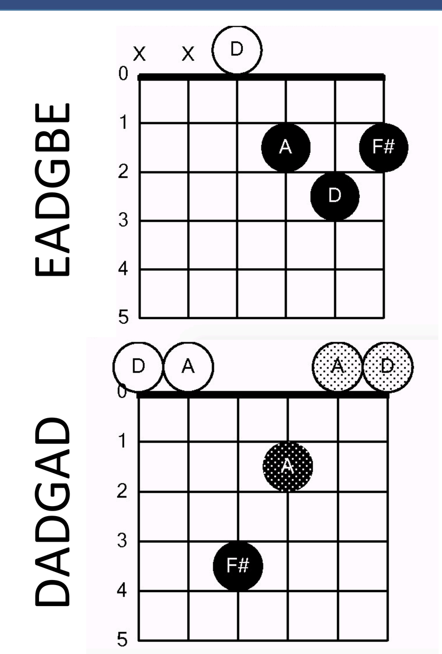
HAJIM SCHOOL OF ENGINEERING & APPLIED SCIENCES UNIVERSITY of ROCHESTER

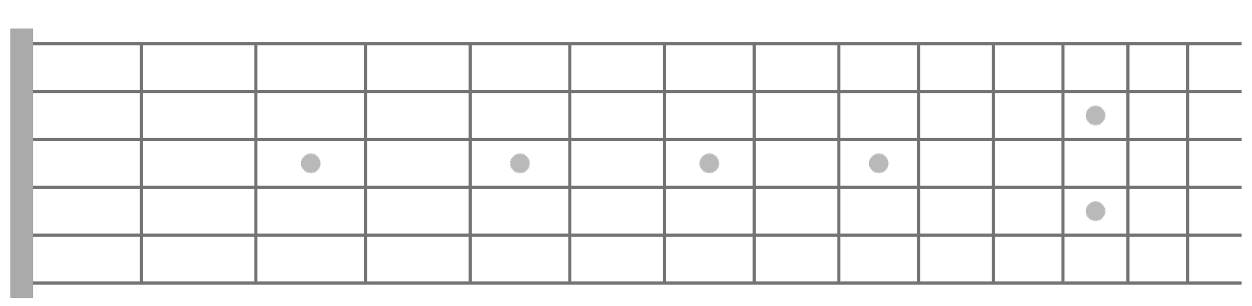
Abstract

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 is on. To address this problem, we propose two methods: one, a supervised learning algorithm to identify the tuning from a fixed set of tunings using an LSTM-based neural network; two, 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.

The Problem

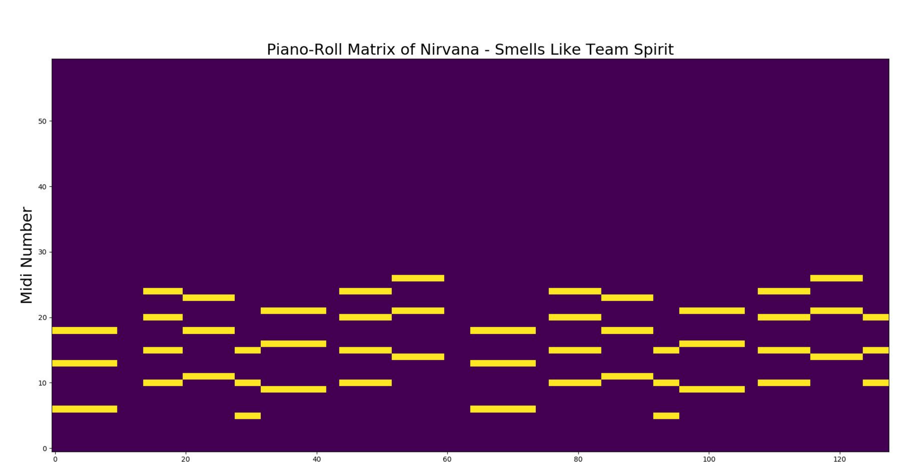
- Timbre Modeling
- Key Identification
- Interval Analysis
- Chord Recognition (Inversion, Drop chords)
- Identifying Lowest Note





Dataset

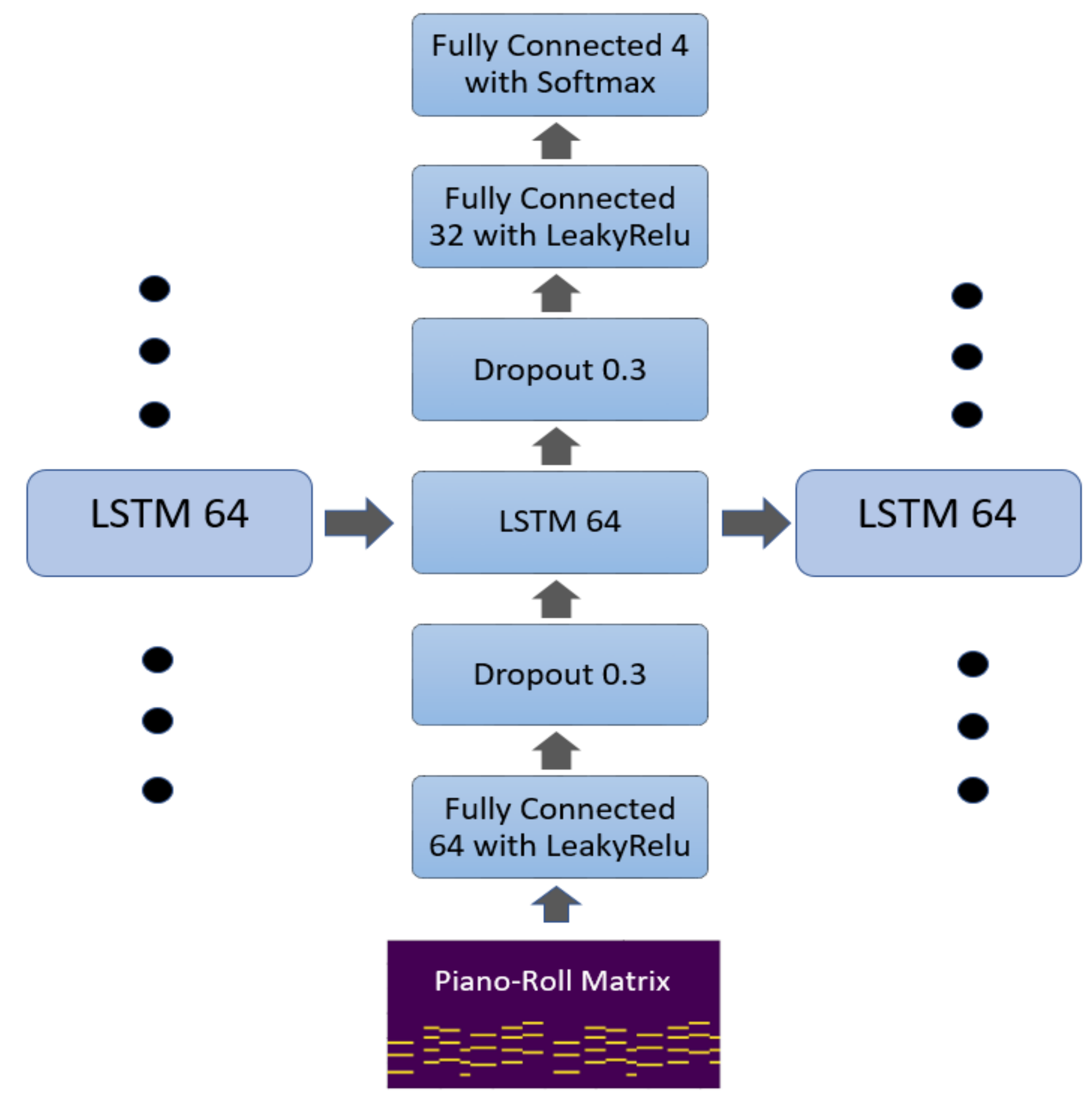
- Guitar Pro files containing tablature was downloaded from *Ultimate-Guitar.com*
- Python library *music21* was used to parse files
- Notes were stored as a piano-roll matrix with midi note number and four measures with 32nd note intervals as axis



Collected Songs in Dataset

	Standard (EADGBE	Open D (DADF#AD)	Open G DGDGBD	Open C CGCGCE
Number of Songs	55	126	90	92
Number of 4 Measure Bars	6523	5114	4289	6011

Approach 1 – Deep Learning



- Input is passed through a fully connected layer to reduce dimensionality.
- An LSTM captures time dependency between notes and models the transitions from one note to the next.
- Output is a vector of soft probabilities for 4 classes: Standard, Open D, Open G and Open C

Approach 2 - Dynamic Programming

- A difficulty based model.
- Can use dynamic programming to calculate the optimal combination of note positions for a given note sequence and tuning

Note Sequence and Possible Locations for Standard Tuning

	Note i -1 C4 (60)	Note i E4 (64)	Note i + 1 G4 (67)
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)

$$Cost[i,j] = \min(Cost[i-1,k] + \mathrm{dist}(x_{i,j}, x_{i-1,k}), for\ k \in \{1, \dots, N\}$$

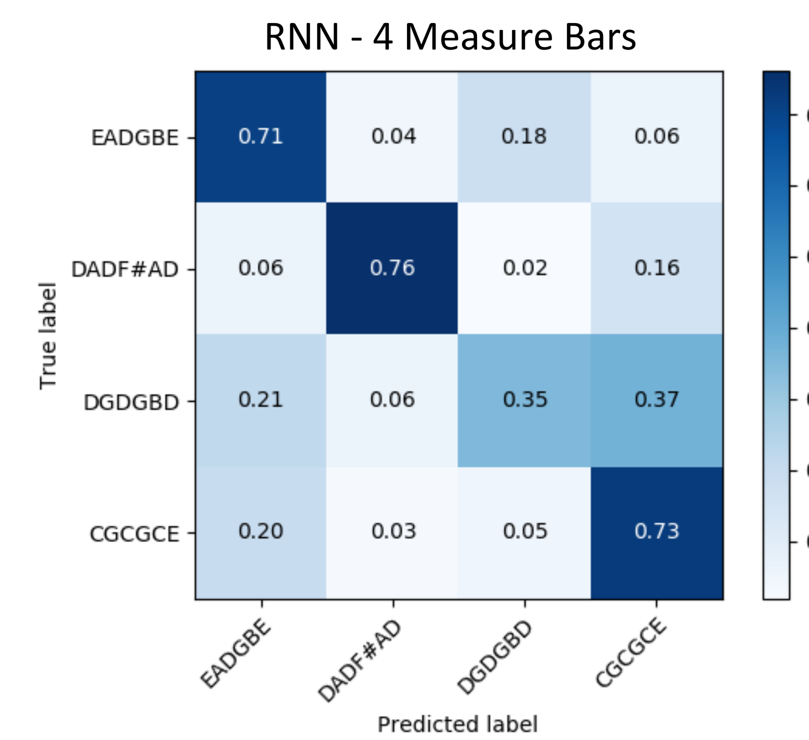
where $Cost$ is the accumulated cost up to note i at position j, and N is the number of possible note locations for note $i-1$.

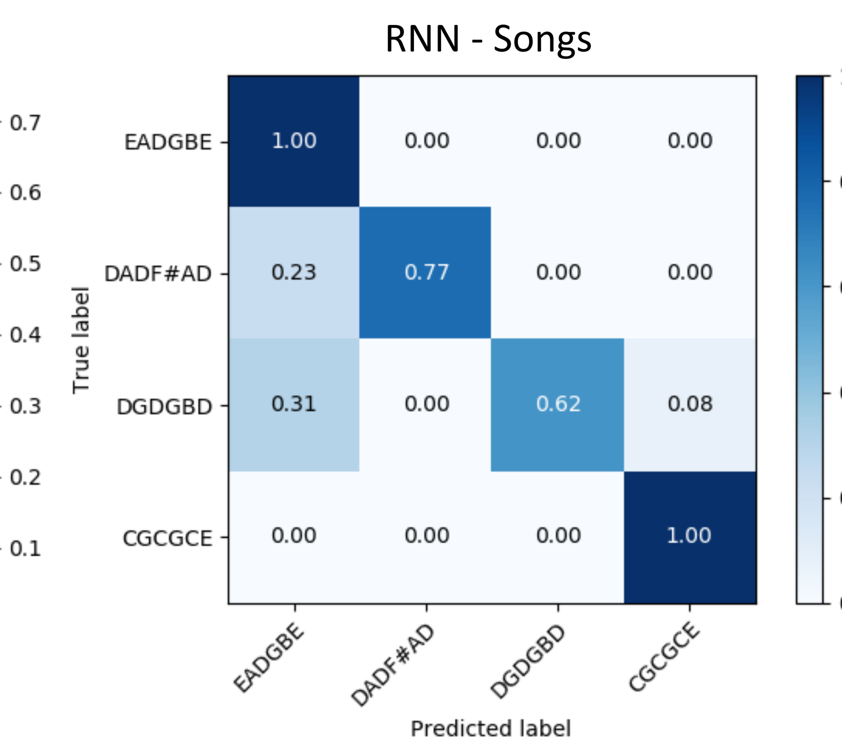
$\mathrm{dist}(x_1, x_2)$ is the distance function calculated as follows:

$$\mathrm{dist}(x_1,x_2) = \begin{cases} 0, & x_1\ or\ x_2 = 0 \\ \dfrac{|x_1 - x_2|}{(x_1 - x_2)^{\frac{1}{12}}(\min(x_1,x_2))}, & otherwise \end{cases}$$

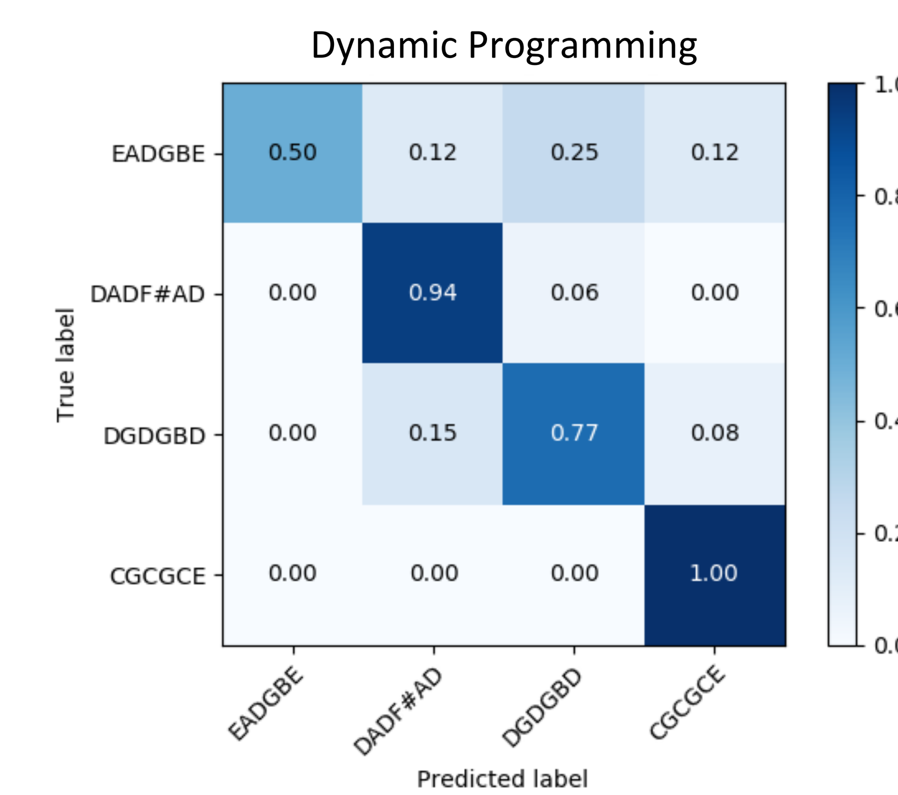
- Open strings have no distance (when the fret number is zero)
- Vertical change (change in y) between strings have no distance
- If a chord is played, the minimum distance of all valid position combinations of the notes in the chord is taken

Results - Evaluation - Future Work





- Confusion matrix by song is made by summing all predictions 4 measure bars and taking the maximum.
- Confusion in Open G (DGDBGD) could be a result of having less information in the dataset.



- The cost matrix purely calculates cost. We assume that the original tuning of a song will have the minimum cost in most cases, but this is not always the case.
- Low score on standard is likely due to:
  - -> Power chords are easier to play in Open D.
  - -> Many songs are in Am and Em which are easier to play in Open C and G.

Conclusion and Future Work

The problem of identifying the optimal guitar tuning to play 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. We also showed that dynamic programming can be used to calculate the cost a song has for a given tuning based on difficulty. Both methods had decent levels of success.

To improve our results some future work could include:

- Collecting more data for the dataset
- Adjusting the hyperparameters of our model
- Experimenting with different distance functions in the DP algorithm