GUITAR TABLATURE ESTIMATION

Audio Processing and Indexing

Final Project

Kamand Hajiaghapour s3058107

Nick van der Linden s2971976

May, 2021

1. Introduction

The common notation for composing, sharing and learning guitar for popular music, is guitar tablature, instead of score notation. Both of these notations can be seen in fig1: The top half representing score notation, and the bottom half representing tabs. In addition to displaying the arrangements of pitches in time, like music scores notation, guitar tablature notation illustrates activated guitar strings and their positions along the fretboard for producing the sounding pitches.

The task of creating guitar tablature is not easy because of two aspects. One is related to the challenges of polyphonic transcription. This task is challenging because it is hard to tell which pitches are sounding when multiple different pitches sound at once and their overtones may overlap in frequency domain. Another aspect is related to the characteristic of guitar that most notes can be played in numerous locations along the fretboard. Playing these identical pitches in different locations can make the overall performance different because of the difference in timbre in each location. Therefore, for making a guitar tablature from a guitar performance, we need to be able to estimate both the pitches and fingering changes over the course of the performance.

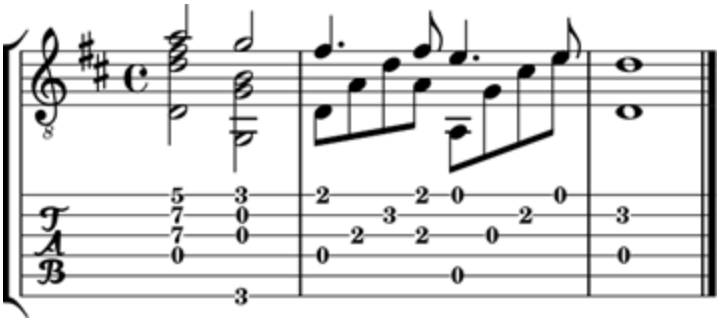


Fig1: guitar tablature notation and score notation

Guitar is a very popular instrument, both for professional musicians and amateur. To play different songs, especially as a beginner, one often needs to have the tablature of the songs. The task of creating tablature is not easy and it needs much time and energy even from a professional musician. Therefore; finding a way to create tablature automatically can be helpful. One way to automatically create guitar tablature is using Convolutional Neural Networks. Wiggins and Kim [1] worked on this approach and called the algorithm TabCNN. The proposed system uses a convolutional neural network (CNN) to learn a direct mapping from audio signal to guitar tablature. In this project, we try this system on our own electric guitar data set to see how much the results differ.

In the following sections we first explain our methodology including our data set, the preprocessing steps, the CNN model and the evaluation method. Then, we look at the result and compare it with the original paper. Lastly, there is a conclusion section.

2. Method

2.1. Data set:

For guitar transcription some previous studies used audio data that was automatically generated using MIDI to playback audio samples [2, 3, 4, 5]. However, we cannot use that approach in our project because it does not provide a reliable ground truth tablature, most notably since the information about the exact fingering used by the guitarist can't be determined using this process.

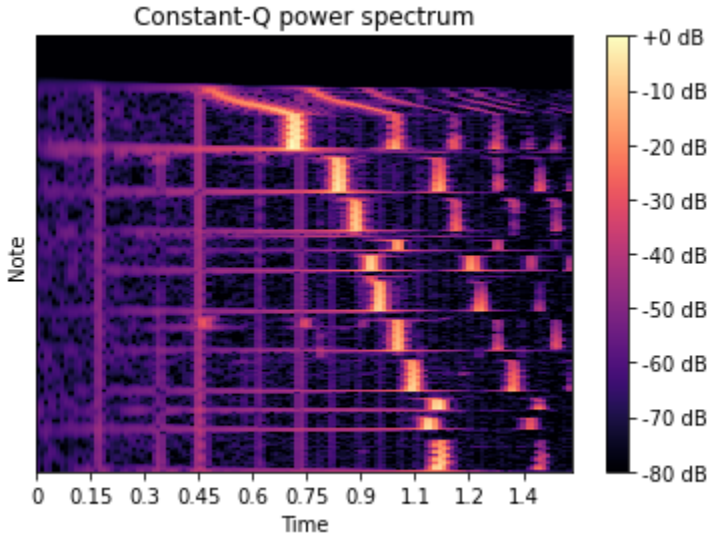
We created our own data set by recording electric guitar performances for 4 different songs. Contrary to the Wiggins and Kim [1], we did not have access to the hexaphonic pickups to generate the annotations. Instead, we recorded it using a standard electric guitar into Reaper, with the help of a DI box. Next, we created the annotation manually using Guitar Pro 7. We recorded 48 tracks that consist of 3 different songs playing in 4 different styles: singer-songwriter, jazz, rock and funk. Each song was

recorded at a fast and a slow tempo. For each of those ‘categories’, we played the chords, as well as the solo which contains mostly single notes, over those chords. The tracks don’t have the same duration. The shortest track has a length of 10 seconds and the longest track spans 26 seconds. This difference is evened out by adding silence at the end of the shorter tracks.

2.2. Data preprocessing and annotation

---"During the preprocessing stage, we first downsample the audio from 44100 Hz to 22050 Hz, making the assumption that there is not too much relevant information above 11025 Hz, in order to reduce the dimensionality of the input signals. We normalize each audio clip by its maximum value, to account for any major amplitude differences between clips.”---

For the input of our CNN model, we use CQT files (fig2), following the example of Wiggins and Kim [1]. The constant-Q transform (CQT) refers to a time-frequency representation in which the frequency bins are geometrically spaced and the Q-factors (ratios of the center frequencies to band-widths) of all bins are equal. The most common representation used for representing a signal is a STFT. For this task, CQT files were used for two reasons. First, STFTs are not as good of an option for CNNs due to its high dimensionality. Secondly, the CQT representation involves a frequency axis that is linearly spaced with respect to pitch and in this project we need to recognize musical pitches. Therefore, using CQT allows pitch-invariant features to be learned by the network.



In the work of Wiggins and Kim [1], CQTs of 192 bins were used, spanning 8 octaves. For this project, CQTs of 128 bins were used, to match the number of bins used by the midi files. Each octave spanned 24 bins, meaning our model spanned 5 octaves. The rest of the settings in the project were left unchanged; The hopsize was set to 512, and the input images were created using a sliding context window that spanned 9 frames, and a padding of 9 on either side.

For data annotation, we first use GuitarPro software to make the tablatures of the tracks (fig3). After that, we use GuitarPro to create the MIDI files of all the tablatures. These MIDI files are read as arrays using a class from the midi visualization library [6]. We round each sampled MIDI pitch value to the nearest integer for finding the nearest musical pitch. The corresponding string's open pitch value is subtracted based on the string that the pitch was played on. It results in some integers. These integers show the frets that were activated to produce the sounding pitches. Therefore, we have 21 different fret classes: open, close or any of the 19 frets used in electric guitar. A value of zero means that the string was plucked open and no frets were being pressed. As a result, we create arrays of one-hot vector encodings in a shape of the amount of the frames by 6 by 21.

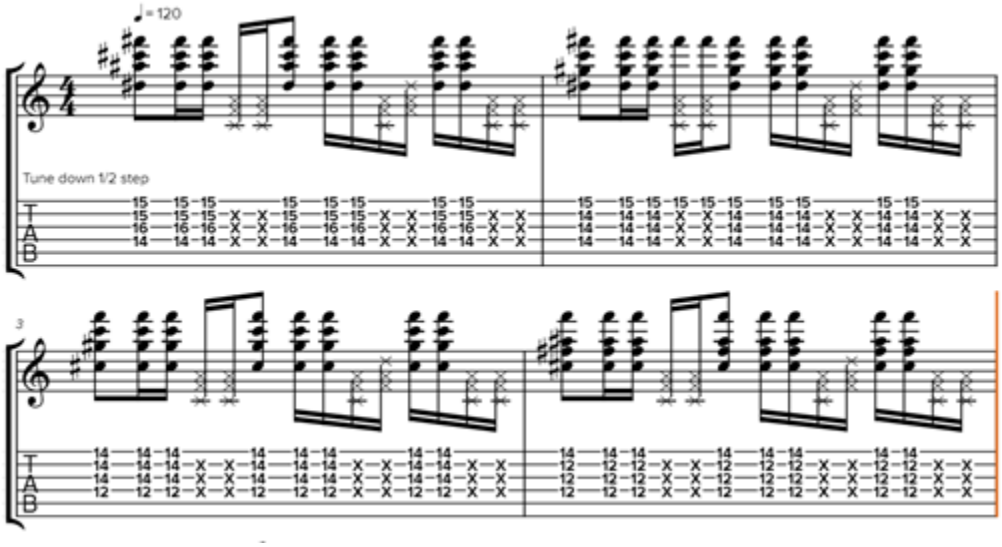


Fig3: part of one of the tracks tablature

2.3. Training and testing

2.3.1. Convolutional neural network

The network that is used here is based on the Wiggins and Kim,2019 paper. Regarding the structure of our neural network, it is inspired by popular models in computer vision, like AlexNet [7] and VGGNet [8]. These models include a series of convolutions with dense layer connection termination. In these architectures, the first layers act as feature extractors that learn filter coefficients which help the classification task. Therefore, the last dense layers are considered responsible for processing the features to result in a prediction. So, they act as classifiers.

Fig 4 illustrates the proposed network structure for TabCNN. At the beginning of the network there are three convolutional layers. After a Rectified Linear Unit (ReLU) activation, these layers are followed by a max pooling layer. After this layer there is a dense layer of dimension 128 which is the result of flattening the max pooling layer. By applying a Relu activation function this layer is connected to the second dense layer. The second dense layer of dimension 126 is then reshaped to 6*21. Finally, there is the output shape which is 6 by 21. 6 is related to the number of strings and 21 is related to the different fret classes a string can be assigned. Lastly, a softmax activation is applied to each of 6 rows. Therefore, the result of this model is an estimation of six probability mass functions, representing the probability of each fret class for each string. Also, it should be mentioned that, the 3 convolution layers here, have a filter size of 3*3. The number of filters for the first layers is equal to 32 and for the two others it is equal to 64.

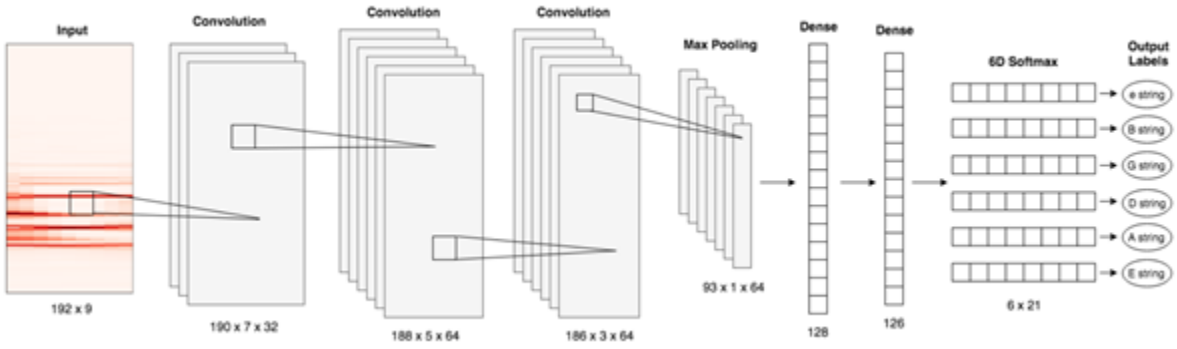


Fig4: network structure of TabCNN

For training our model, at first, we tried to use our own data set. However, because of the differences in the format of our dataset and the original paper's dataset, we decided to continue this procedure using the Guitar Pro dataset.

In the training process, we use the loss function designed in the Wiggins and Kim paper [1]. In that paper, the problem is viewed as 6 simultaneous multiclass classification problems. To this regard, the

categorical cross-entropy for each string and sum of these values is computed using the loss function indicated below.

$$Loss = -\frac{1}{N}\sum_{j=1}^{6}\sum_{i=1}^{N}\log p[z_{ij} \in C_{z_{ij}}]$$

The definition of each element in this equation is as follow:

Zij: an activation at frame i on string j that belongs to fret class Czij

$p[Zij \in Czij]$: the probability output by the network of *zij* belonging to class *Czij*

N : the total number of frames in the mini batch.

This model is trained using the ADADELTA optimization algorithm [9], which adapts learning rates for parameters based on a window of previous gradient values. The initial learning rate is set to 1.0 and a mini batch size of 128 training samples is considered in the model. Model is trained for 8 epochs. For combatting overfitting dropout regularization is employed, with a dropout rate of 0.25 applied immediately after the max pooling layer, and a second dropout rate of 0.5 applied after the first dense layer

Finally, we use our own data set to test the model (fig5). This way we can see if the model can be useful in other datasets, with different properties.

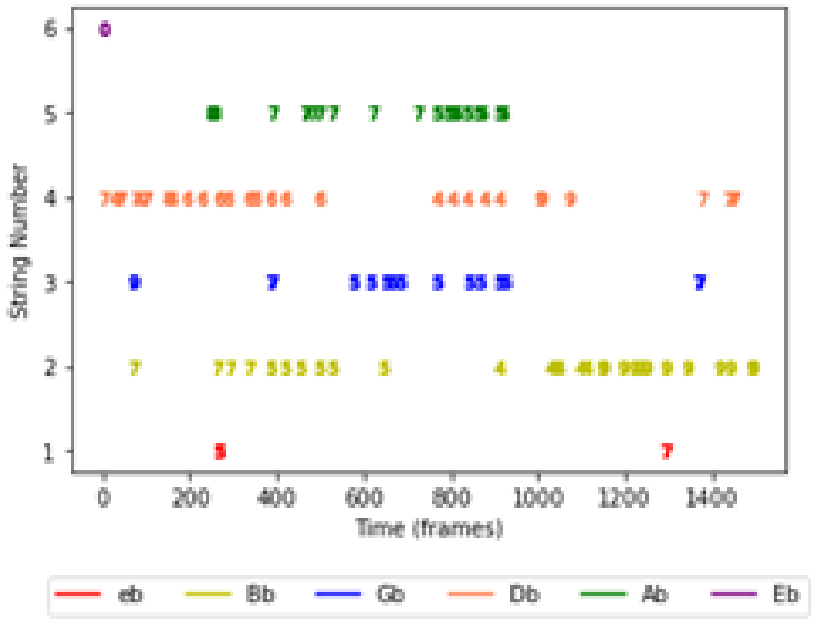


Fig5: sample of our output

2.3.2. Evaluation

For evaluating the model, two kinds of metrics are used: multipitch estimation metrics and tablature estimation metrics. The first metrics are based on [2] and for evaluating tablature estimation we use the approach proposed in [1]. Therefore, here we use 6-fold cross validation, holding out one of the 6 guitarists to test on, while training on the remaining 5. Then the numerical results are averaged over the 6-fold of data.

For multi-pitch estimation we use multi-pitch precision, recall and F measure. The formulation of these measures are as follow:

Multipitch precision: calculates the number of correctly identified pitches divided by the total number of predicted pitches.

$$P_{\text{pitch}} = \frac{e^T(Y_{\text{gt}} \odot Y_{\text{pred}})e}{e^T Y_{\text{pred}} e}$$

Multipitch recall: calculates the number of correctly identified pitches divided by the total number of ground truth pitches.

$$r_{\text{pitch}} = \frac{e^T(Y_{\text{gt}} \odot Y_{\text{pred}})e}{e^T Y_{\text{gt}} e}$$

Multipitch F measure: calculates the harmonic mean of multi-pitch precision and recall.

$$f_{\text{pitch}} = \frac{2p_{\text{pitch}} r_{\text{pitch}}}{p_{\text{pitch}} + r_{\text{pitch}}}$$

For tablature estimation we use tablature precision, recall, F-measure and Tablature Disambiguation Rate. Followings are the definition and formulation of each of these metrics.

Tablature precision: calculates the number of correctly identified string-fret combinations divided by the total number of predicted string-fret combinations.

$$p_{\text{tab}} = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T Z_{\text{pred}} e}$$

Tablature recall: calculates the number of correctly identified string-fret combinations divided by the total number of ground truth string-fret combinations.

$$r_{\text{tab}} = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T Z_{\text{gt}} e}$$

Tablature F-measure: calculates the harmonic mean of tablature precision and recall.

$$f_{\text{tab}} = \frac{2p_{\text{tab}} r_{\text{tab}}}{p_{\text{tab}} + r_{\text{tab}}}$$

Tablature Disambiguation Rate (TDR): computed by dividing the total number of correctly identified string-fret combinations by the total number of correctly identified pitches.

$$TDR = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T(Y_{\text{gt}} \odot Y_{\text{pred}})e}$$

3. Results

In the following table the results of the measure estimations of our project and the original paper can be found:

Measure	**Values**	**Original Paper**
Tablature precision	0.197	0.809
Tablature recall	0.197	0.696
Tablature F-measure	0.197	0.748
Multipitch precision	0.271	0.900
Multipitch recall	0.434	0.764
Multipitch F-measure	0.332	0.826
TDR	0.635	0.899

The results of tablature estimations are almost similar. Tablature precision, recall and F-measure are all 0.1997. The F-measure summarizes the system's overall performance for tablature estimation. Therefore, our model's overall performance is less than 0.2.

The estimations show better results regarding multi-pitch measures. Multipitch recall, which shows how frequently pitches existent in the signal are detected by the system, has the highest value in this group. It is equal to 0.434. However, the value of precision is less than recall, which means it is harder for the system to correctly detect the pitches than just detect them. It seems like our model predicts some pitches that are not actually there. Therefore, the number of predictor pitches which is the denominator of the precision is larger than the total number of ground truth pitches or the denominator of the recall measure. Because we have better values for multi-pitch recall and precision in comparison to tablature measures, it is unsurprising that we have a better F-measure here, too.

The highest value we got is 0.635 which is related to TDR. This metric measures how frequently pitches that are correctly identified are assigned the correct tablature. Therefore, the probability that our model assigns the correct tablature to the correctly identified pitches is a bit higher than 63 percent.

The original paper has better results in all manners and it is due to the fact that they had access to hexaphonic pickups, which allowed them to have detailed annotations of their playing.

4. Conclusion

This project was an attempt to make automatic tablature from a data set of electric guitar recordings based on the [1] paper. At first a dataset of 48 electric guitar recordings was built. To this regard we played 3 songs in 4 different styles, fast and slow. For each of those ‘categories’, we played the chords, as well as the solo over those chords. After this step, we made CQTs of these recordings. From these CQTs, frames are sampled in an overlapping window with a length of 9 frames. The paper we based our project on had access to hexaphonic pickups, which allowed them to have detailed annotations of their playing. Since we did not have access to that, we used a different method of creating the annotations. We made some handmade tabs of these recordings using Guitar Pro. From these tabs we generate midi files. From these midi files we create arrays of one-hot vector encodings in a shape of the amount of the frames by 6 by 21. We trained our network based on the dataset used in the original paper, which is the GuitarSet, and then used our own dataset to test it. At the last step, we calculated the related measures to evaluate the TabCNN model on our own dataset. Our model achieved a score of 0.635 for TDR measure that indicates the probability that our model assigns the correct tablature to the correctly identified pitches is a bit higher than 63 percent.

The result of our model was not as good as the result of the original paper. This can be related to two reasons. Firstly, in the original paper the detailed annotations of playings were available, however we did not have access to that. So we used another method for creating annotations. In addition, it is obvious that if we could train the model based on our own dataset, we got better results.

All in all, the objectives set for the scope of this assignment were met, and the results were overall satisfying. Some improvements can be made on both the problems that were formulated in this project, and in the original paper.

Future developments could adapt the CNN model in a way that can be useful for different datasets. In other words, make the annotation step, part of the model that can be applicable to any

recordings. This way the proposed TabCNN model can be closer to be integrated into an end-to-end tablature transcription system.

References

1. *Andrew Wiggins, Youngmoo Kim. "Guitar Tablature Estimation with a Convolutional Neural Network", 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands, 2019.*

2. *Gregory Burlet and Ichiro Fujinaga. Robotaba guitar tablature transcription framework. In Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR), Curitiba, Brazil, 2013.*

3. *Gregory Burlet and Abram Hindle. Isolated guitar transcription using a deep belief network. PeerJ Computer Science, 3:e109, 2017.*

4. *Kazuki Yazawa, Katsutoshi Itoyama, and Hiroshi G Okuno. Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3122–3126. IEEE, 2014.*

5. *Kazuki Yazawa, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi G Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 196–200. IEEE, 2013.*

6. *Exeex. (2017, August). exeex/midi-visualization. Github. https://github.com/exeex/midi-visualization.*

7. *Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.*

8. *Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.*

9. *Matthew D Zeiler. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.*