TABLATURE ESTIMATION OF ACOUSTIC GUITAR RECORDINGS WITH A CONVOLUTIONAL NEURAL NETWORK

Seth Roberts
University of Rochester
Audio and Music Engineering
srober35@u.rochester.edu

ABSTRACT

Automatic music transcription has been an important field of study in recent years. It involves converting an audio signal into musical notation. Guitar transcription is usually in the form of tablature, a style of notation that indicates the position of musical notes on a guitar's fretboard rather than musical pitches. This requires a level of expertise not many guitar players have and is very time consuming. We attempt to solve this problem by automating this task. There are approaches to automatic tablature transcription in which it breaks down the problem into two steps. The pitch is first estimated and then it is converted to tablature fingering. In this work, we will be using a convolutional neural network (CNN) in order to join these two steps together [1]. This approach allows us to map the audio data directly to the tablature. Data is used from the GuitarSet dataset and used to train, validate, and test our network [2]. Additionally, we use various metrics in order to evaluate the performance of our estimation and discuss relevant work within the field of music information retrieval.

1. INTRODUCTION

Until recently, the problem of transcribing music audio into music notation has relied solely on humans. Automatic Music Transcription (AMT) uses computational algorithms in order to convert an acoustic audio signal into musical notation. This has been achieved in a variety of different approaches. Some methods use pitch estimation and onset detection, while others use artificial intelligence.

Piano transcription is very popular because there is a plethora of data. The recordings for piano are also less complex because of the nature of the instrument. On a piano, you can only vary the duration and intensity of the note and a note can only be played in one location. With guitar, the same note can be played in multiple places on the fretboard, which makes pitch detection not as optimal for transcription. Additionally, strings can be played in a variety of different ways. Notes on a guitar can be played by bending, sliding, hammering on and off, and can be plucked or strummed with your fingers or a guitar pick. Because of this, there is a lack of data for guitar transcription.

We develop an approach that employs a CNN in order to determine the fingerings of the notes being played instead of a pitch estimation. This allows us to map guitar tablature from a raw audio signal. The GuitarSet dataset provides us with sufficient information in order to train, validate, and test our CNN. We use Pytorch to implement the proposed model.

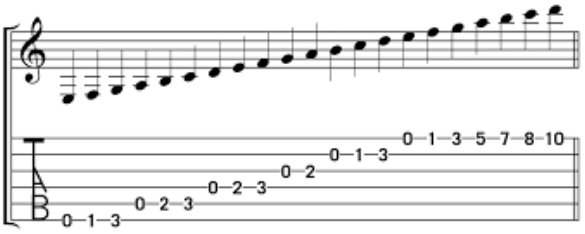

Figure 1. Translation of music notes to tablature. This shows notes from low E to high D. A zero in the tablature represents an open string being played.

2. RELATED WORK

There have been a number of works that address the problem of automatic tablature transcription. A chord estimation algorithm using pitch detection was used in [3]. This approach models the mechanics of the guitar with a deep convolutional network. A set of finite chord shape templates are used and the neural network is trained by minimizing the distance between its output and the best template. This system is, however, constrained by the chord shape templates that it uses to differentiate between input chords.

Another method that has been explored involves using a Hidden Markov Model (HMM) [4]. This model measures the strength of different fundamental frequencies within each time frame and, based on the probabilities of different chord sequences, predicts the output using a number of hidden states. This model is constrained to only four different chord types: major, minor, major 7^{th}, and minor 7^{th}.

The most relevant approach to our model is from TabCNN. This approach uses the GuitarSet dataset in order to train and test their CNN and uses a variety of different metrics in order to evaluate the performance [1]. However, there is no validation within their model. In our implementation we will use different network architecture and use a portion of the data for validation to train hyper-parameters within the model. Additionally, we use a different training optimization algorithm.

3. DATASET

The GuitarSet Data set consists of 360 audio excerpts of solo acoustic guitar that are around 30 seconds long [2]. The dataset was created by having six different guitarists

play the same thirty lead sheets. Each guitarist recorded themselves comping chords and then soloing over their own comping. There are 5 different styles of lead sheets: Rock, Singer-Songwriter, Bossa Nova, Jazz, and Funk. These styles vary in chord progression and tempo. These excerpts were recorded with both a hexaphonic pickup and a Neumann U-87 condenser microphone as reference [2]. The audio files can be classified into three groups: hexagonal pickup, hexagonal pickup with interference removal applied, and monophonic recording from a reference microphone. The labels for the excerpts are in the form of *.jams* files. These *.jams* files contain information of pitch contour, MIDI note, beat position, and tempo. Additionally, the ground-truth note annotations are provided in order to evaluate the quality of our transcription.

4. METHODOLOGY

4.1 Audio Preprocessing

In the preprocessing stage, we take in each of the monophonic *.wav* files and downsample from 44,100 Hz, what the audio was recorded at, to 22,050 Hz. This allows us to use frequencies up to 11,025 Hz, the Nyquist frequency [5]. This covers most of the harmonics generated by the guitar and reduces the size of the input signal for processing. We also normalize the data in order to have a uniform amplitude across audio clips. We will be training our CNN with spatial data from the files, and therefore will convert our raw audio data from the time domain into the frequency domain. The most common way to do this is with the Short-Time Fourier Transform (STFT). The STFT determines the sinusoidal frequency and phase content of sections in a signal. It additionally spaces frequency bins linearly and has the same window length for each frequency. This is undesirable because musical notes are spaced apart logarithmically, not linearly. In this work, we will use a Constant-Q Transform (CQT), in which the frequency scale is logarithmic and the ratios of the center frequencies to band-width of all bins are equal [6]. This means that there are longer windows for lower frequencies and shorter windows for higher frequencies,reducing the dimensionality of the input signal in our CNN.

For our CQT, motivated by previous work, we use 24 bins per octave, over 8 octaves [6]. This gives us 192 bins, with 2 bins per semitone. We use a hop size of 512 samples, at a sampling rate of 22,050 Hz, equating to around 43 frames per second. A context window of size 9 frames is used and is padded on either side to ensure that the input data is uniform. This also ensures features are preserved that exist at the edges of the matrix. Input samples of our CNN are 192 x 9.

4.2 Labeling Preprocessed

In order to properly train our CNN we need to have labels for each input sample. We will be labeling our data from the information in the *.jams* files that are provided within the dataset. We will use the same number of frames that are used when calculating the CQT, 43 frames. From here, we convert the frames to seconds because the *.jams* labels are in terms of seconds. We then loop over all 6 strings and replace the given MIDI pitch with the correct fret number for each input frame. To do this, we round each MIDI number to a whole number, which corresponds to the nearest musical pitch, after subtracting the corresponding string's open pitch value. This outputs the fret that was activated in order to produce the note. There are 21 different fret classes: the 19 frets on the guitar plus open string (played with no fret pressed) and closed string (not played). This is then converted into a label array of size 6 x 21 for each frame.

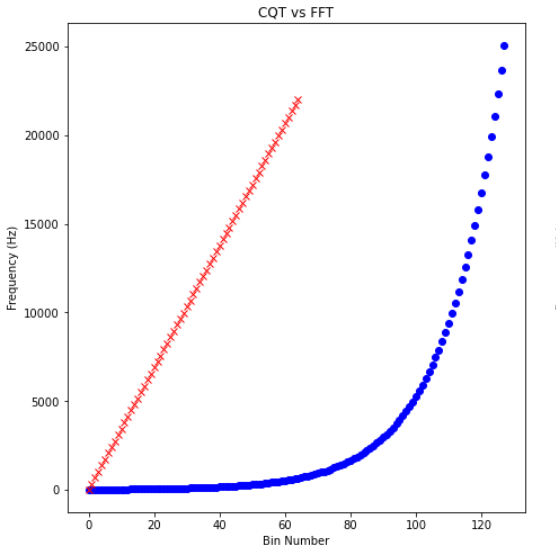

Figure 2. Bin number to frequency comparison between CQT and STFT. This shows the logarithmic spacing between frequency bins for the CQT.

4.3 Network Architecture

The neural network structure that we will use is inspired by the TabCNN model [1]. Using a CNN, we are able to filter the input with convolutional layers and extract spectral features. The weights of these filters are learned throughout the training process. The CNN concludes with two dense layers which create a shallow neural network to perform our classification. It receives the output from each neuron of its preceding layer in order to predict the output.

The first stage of the proposed network is a series of 2 convolutional layers, with sizes 3 x 3 and 5 x 5 respectfully, with a stride of 1. Our first layer contains 32 filters and the second layer has 64 filters. Each convolution layer is followed by a Rectified Linear Unit (ReLU) activation, which performs nonlinear mapping to

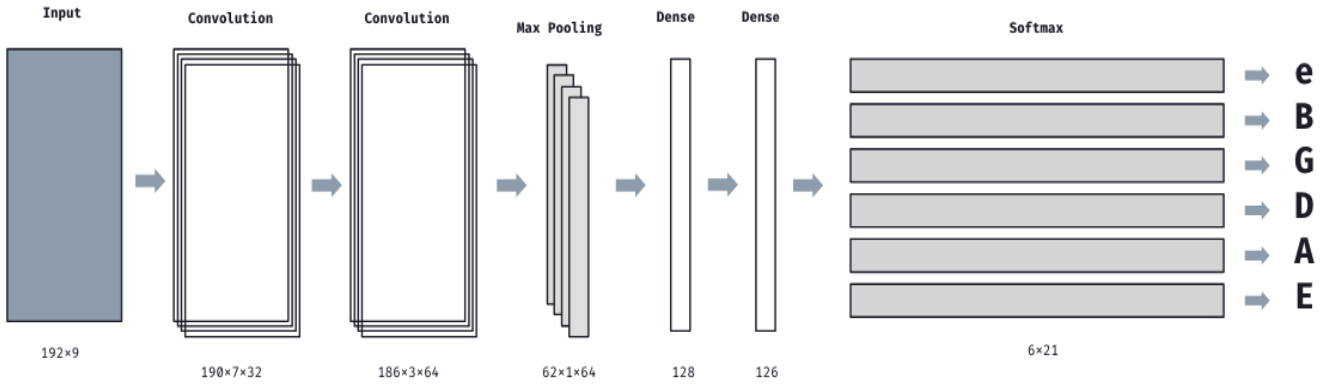

Figure 3. The proposed network architecture. The input is a constant-Q spectrogram of acoustic guitar audio. This is input to 2 convolutional layers then a max pooling layer. These layers extract spatial information in order to output a tablature estimation. The two dense layers and softmax predict the fret-number labels for each of the 6 strings.

be learned. It defines the output of a neuron given a set of inputs, with an input above zero returning the output of the neuron, and an input below zero returnings an output of zero.

The next stage is our max pooling layer. This reduces dimensionality of the images, the number of parameters, and computational load. We use a max pooling layer of size 3 x 3 and stride of 1. This extracts the highest, or most activated, value from the 3 x 3 portion of the image we are looking at. This is then flattened to one dimension to be input into our first dense layer. After our first dense layer, of size 128, ReLU activation is applied. This is then input to our second dense layer of size 126, and reshaped to 6 x 21, which corresponds to the 6 different strings and 21 different fret classes. Finally, a softmax activation is performed. This activation converts values into probabilities such that all values in the list sum to 1. It learns to output the probability of each fret class for each string.

4.4 Training

For our training model, we use the ADAM optimization algorithm [7]. This model works by adapting the parameter learning rates based on the average first and second moment's gradients using an exponential moving average and correcting its bias. Using a learning rate of 0.1 and mini-batch size of 128, we train for 4 epochs. Based on previous work [1], overfitting occurred when training for longer. To combat overfitting, we use a dropout rate of 0.25 after the max pooling layer and the first dense layer. We use a portion of the dataset for validation in order to do some hyper-parameter optimization that previous work did not implement.

With our output being a representation of a fretboard, we now design a loss function so our machine can learn to reproduce it. Because we are dealing with a multi-label classification for each of the 6 strings, it is best to use cross-entropy as a loss function. The proposed loss function is described as follows:

$$L = \frac{1}{N}\sum_{n=0}^{N} y_n \log \hat{y_n} + (1 - y_n) \log(1 - \hat{y_n})$$

In this loss function, N represents the total number of labels to be predicted. This function is used because it enables us to measure the loss between the predicted and ground truth vectors with multiple labels [8]. Once we calculate the cross-entropy for each string, they are summed in order to produce our final result.

5. RESULTS

The results did not produce data that was sufficient enough to conclude that this proposed architecture works better than that of TabCNN. During training of the model, the loss was decreasing through each epoch. However, during the validation process, the loss was increasing. This means that the model was overfitting to the training data. Due to the time constraints of the project, we did not have time to fine tune the hyperparameters of the architecture in order to fix this overfitting issue. In the future, the learning rate, optimizer, and context window size will be adjusted in order to resolve this problem.

6. CONCLUSION

In this paper, we propose a convolutional neural network for estimating guitar tablature. This network converts audio of solo acoustic guitar into musical notation. Although this model did experience over-fitting issues, we believe that it will be able to produce results similar to that of TabCNN once this issue is resolved. We hope that in the future we can use this as a framework for transcription of other musical notations as well, not just tablature.

7. REFERENCES

[1] Andrew Wiggins and Youngmoo Kim. Guitar tablature estimation with a convolutional neural network. In ISMIR, 2019.

[2] Q. Xi, R. Bittner, J. Pauwels, X. Ye, and J. P. Bello. Guitarset: A Dataset for Guitar Transcription. in 19th International Society for Music Information Retrieval Conference, Paris, France, Sept. 2018.

[3] Ana M Barbancho, Anssi Klapuri, Lorenzo J Tardon, and Isabel Barbancho. Automatic transcription of guitar chords and fingering from audio. IEEE Transactions on Audio, Speech, and Language Processing, 20(3):915–921, 2011.

[4] Chang, W., A. W. Su, C. Yeh, A. Roebel, and X. Rodet. 2008. Multiple-F0 tracking based on a high order HMM model. In Proceedings of the International Conference on Digital Audio Effects, Espoo, Finland.

[5] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 25(1):208–221, 2017.

[6] A. Klapuri, C. Schorkhuber, Constant-Q Transform Toolbox For Music Processing. University of London, 2010.

[7] Eric J Humphrey and Juan P Bello. From music audio to chord tablature: Teaching deep convolutional networks to play guitar. In 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), pages 6974–6978. IEEE, 2014.

[8] J. Sleep, Automatic Music Transcription With Convolutional Neural Networks Using Intuitive Filter Shapes, California Polytechnic State University, October 2017.