1. Introduction

Music is all around us and is largely influential in modern society. From amateurs to professional musicians, music can be played and enjoyed by nearly everyone. Tabs are a common way to transcribe music played on the guitar as they are easily understandable by all players, especially those without education on formal sheet music. While playing music from tabs is easy and accessible, writing tabs is time consuming and monotonous. Tabber sets out to address this problem by providing an online platform that uses machine learning and digital signal processing to automatically generate guitar tablature from short audio clips of individually-played notes, or licks.

2. Project Goals

The goal of this project is to create a webapp which is capable of automatically generating tablature for guitar licks, whether they are played directly into the computer's microphone, or uploaded from a pre-recorded file. These licks are then saved to a user's account, where they can be shared with other users.

Improvising licks is a great way for guitarists to explore new avenues in their songwriting; however, tabbing licks manually is time consuming and hinders the musician's creative flow. By utilizing machine learning and digital signal processing, Tabber will allow musicians to tab their licks without ever needing to put the guitar down - this will greatly speed up a musician's ability to save and iterate on their licks and enables them to remain in a state of continuous creative flow.

3. Design / Methodology

Tabber as an application can be broken down into two major components: the online platform and the tabbing algorithm. The online platform can be further divided into the frontend and the backend which are created using ReactJS and NodeJS respectively. The Tabbing algorithm is a standalone Python script which the Node server executes in a worker to prevent heavy processing on the server.

Online Platform

The frontend of Tabber was designed with the goal of accessibility in mind. For this reason Tabber was created as a web application that can be run on any of the major web browsers either on computer or mobile device. Some of the core features intended for the web application include uploading audio files, recording live instruments, sharing licks with other users, and archiving all of your previously created licks.

The backend of Tabber is the next major component which is used to authenticate and store user information. It is responsible for storing audio files in the cloud for users to securely share with other musicians. It is also responsible for storing all data related to the tabbing of audio files.

Machine Learning and Onset Detection

The final and most important component of Tabber is the tabbing algorithm. Based on our initial literature review we did not find any pre-existing solutions to audio-to-tab generation. Thus, a novel solution was created which can be broken down into the following steps:

1.) Find the pitch in each 10-millisecond frame with the CREPE machine learning model
2.) Detect onsets (i.e. when a note is played) with the librosa python library, which detects peaks in the spectral flux graph of the audio input using custom peak-picking parameters
3.) Take the detected peaks at the detected onsets, and assume that they're played on the highest string possible in order to generate the output tab string.

4. Results and Discussion

Results:

- The online platform and tabbing algorithm dramatically reduce the amount of time required to manually convert a lick into a tablature.
- Accuracy of the algorithm is above 90%
- Supports guitars on multiple different tunings

Limitations

- Accuracy of the algorithm is relative to the first note of the lick.
- The algorithm may generate a lick using a different position on the guitar but the notes will be the same when played.
- Limited to a maximum tempo of 140 beats-per-minute
- Does not support slides or bends

5. Conclusion

The development of the initial prototype of Tabbing algorithm proved to be successful and an early development build for the online application is fully functional. Moving forward in the development of this application there are many areas that need to be improved on. Architecturally the system needs to be redesigned to allow multiple tabbing algorithms to run in parallel due to the significant time constraints required to process a single lick. In addition, now that a proof of concept of the algorithm has been created, additional work can be put into optimizing the performance of the algorithm to minimize the overall execution time.