Capstone Project Proposal - Dr. Davide Guidi

Domain background

As a person who loves music and works with computers, I have always been interested in the intersection between these two worlds. It's fascinating how tasks such as recognising a song or identify notes in a melody are relatively easy to perform by a human but so hard to achieve by a computer.

These kind of problems are very good candidates for machine learning algorithms, as trying to solve them in a programmatic way is a daunting task. As I am not an expert in the field of Audio Analysis, I don't expect to achieve results similar to the state-of-the-art. Nevertheless, I believe it makes a very good case study and will provide me with experience on how to attack similar problems.

One of the main tool used in Audio analysis is the Fast Fourier Transform, whose underlying ideas dated back as early as 1805, according to Wikipedia. Fourier analysis is used to convert a signal from its original domain (space) to a representation in its frequency domain. In other words, by applying FFT to a digital audio recording we can analyse its frequency components.

If we consider the guitar instrument as the source of the audio recording, then in this scenario detecting a single note being played (monophonic pitch detection), is a relatively easy task to solve. Applying an FFT function will in fact reveal one major frequency, which usually correspond to the note being played on the instrument.
However, the case is not so simple when multiple strings are played at the same time (polyphonic pitch detection). In this case, there are multiple concurrent pitched note activations at same time, which interact with each other and make the frequency detection harder. Polyphonic pitch detection is indeed a hard problem, and is the subject of ongoing research.

While I am not an expert in the field by any measure, I was intrigued by a blog post I found online entitled "Automated Guitar Transcription with Deep Learning" [1], based in turn on a poster presented at NEMISIG 2019 (Northeast Music Informatics Special Interest Group) [2].
The key idea of both approaches is to create a deep learning model trained using the GuitarSet [3] dataset, which contains high quality guitar recordings complete with annotations.
The model uses a convoluted neural network (CNN) trained on images extracted from the dataset using Constant-Q transform, a transformation very similar to FFT. According to the [1], the Constant-Q transform is better suited for fitting musical data than the Fourier transform, as its output is amplitude versus the log frequency. Also, the Constant-Q transform's accuracy is analogous to the

logarithmic scale and mimics the human ear, having a higher frequency resolution at the lower frequencies and a lower resolution at the higher frequencies [4].

By providing note annotations, the GuitarSet dataset enables the training and the validation of such model, making it easy to experiment with novel ideas.

Other material was also evaluated, such as the Git Hub project "polyphonic_track" [5].

Problem Statement

In this project I intend to replicate part of the research carried over in [1] and [2]. The goal is to extract a dataset from GuitarSet and use it to train a model to recognise note patterns in an audio file. The model should learn to detect the notes being played in a song with a certain accuracy.

While the final goal of both [1] and [2] is to map the actual finger position in the fret board corresponding to the notes, for this project I am only interested in detecting the notes being played in the audio file.

For this purpose, GuitarSet will be used for both training and validating the model.

Datasets and Inputs

This project would not be possible without using GuitarSet or a similar library. GuitarSet contains 360 audio files that are close to 30 seconds in length. The same guitar tune is recorded using both hexaphonic pickup and a mono microphone and it comes with high quality annotation, including:

- 6 pitch_contour annotations (1 per string)
- 6 midi_note annotations (1 per string)

GuitarSet has been published very recently (August 2019) under the Creative Commons Attribution 4.0 International license.
As the GuitarSet dataset contain 360 recordings of around 30 seconds each, we can expect to extract a training set containing over 50.000 unique entries, which should be big enough to train and validate the model.

Solution Statement

The project's goal is to develop a system that can identify notes being played in an audio file. The audio file consist of a mono audio recording containing a tune played by a guitar. To achieve this goal I will first create a dataset from GuitarSet audio files. Each audio file will be split into a chunk of appropriate length (for example 20ms) and then a Constant-Q transformation will be applied on it.

These transformations, which can be represented as images, denote the frequencies information related to the audio chunk. From GuitarSet I will also extract the annotations and convert them into the notes being played in each audio chunk. The resulting dataset will contain images (the audio features) and the associated notes being played (the ground truth).

Such training dataset will be used as input and validation for a CNN model. The model will output a 6-dimensional output of 21 possible results: 21 possible labels for each of the 6 strings. The 21 labels correspond to the closed position (the string is not played), the open position (the string is played open, no fingers on the fret) and the 19 different frets available on a standard guitar neck.

Benchmark Model

The idea here is to benchmark my result against the results obtained by [1] and [2].

In [1], the test accuracy has been calculated using 40828 training images and 4536 test images:

estring: 0.933
Bstring: 0.852
Gstring: 0.828
Dstring: 0.783
Astring: 0.805
Estring: 0.851

The average test accuracy is 0.842.

In [2], the results are very similar. Although the training and test size are not discussed, the accuracy is:

estring: 0.898
Bstring: 0.760
Gstring: 0.801
Dstring: 0.808
Astring: 0.884
Estring: 0.921

The average test accuracy is a very similar value: 0.845.

These values appear to be much higher than the average accuracy (which is more like 0.6-0.7 as reported in [6]).
I will use these values as target benchmark when developing my solution.

Evaluation Metrics

For this project I will use the same GuitarSet library for both training and evaluating the model. The evaluation will calculate the accuracy between the predicted notes and the annotations.

Project Design

The project workflow will proceed as follow. First of all I will research and familiarise with the state of the art Python libraries available for Audio Analysis.
I will then analyse the GuitarSet collection, validate and process its content. I will extract features from it using Constant-Q transformations to generate relevant images. I will also extract the associated annotations, which will be used as ground truth.
I will then create a CNN model based on the suggestions found in [1] and [2] and I will train it and validate it using the generated dataset.
Finally, I will discuss the obtained results and draw conclusions and possible further directions.

References

[1] Darren Tio, Automated Guitar Transcription with Deep Learning. Available at https://towardsdatascience.com/audio-to-guitar-tab-with-deep-learning-d76e12717f81
[2] Andy Wiggins and Youngmoo E. Kim, Automatic Guitar Tablature Transcription with Convolutional Neural Networks. Available at http://nemisig2019.nemisig.org/images/kimSlides.pdf
[3] 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.
[4] C. Schörkhuber and Anssi Klapuri, Constant-Q transform toolbox for music processing (2010), 7th Sound and Music Computing Conference.
[5] Jay Miller, polyphonic_track project. Available at https://github.com/jaym910/polyphonic_track
[6] Chris Liscio, A Note on “Auto Tabbing”. Available at https://supermegaultragroovy.com/2010/09/01/a-note-on-auto-tabbing/