CS229 Final Report - Music Generator

Abstract

Music is known to be a beautiful art and a passion to many. Instrumental music is a very important part of music and one that not everybody can claim to master. This project explores the possibilities of bringing the power of instrumental music generation to anybody whether they're musically trained or not. The work in this project considers the use of a deep learning model in order to extract the featured musical chords from an audio, the use of those extracted chords to generate an internal music representation and the generation of an audible instrumental music. In the chord extraction, we observe how it is a little easier to identify more general chords from an audio and how much of a challenge it becomes when we want to dive deeper into the more specific chord notations. For the representation and playback of the music, we use a MIDI approach and observe that acquiring a perfect sense of the Tempo presents a challenge.

Introduction

Music is an art that is enjoyed by many people. Whether they are musically trained or not, whether they can perform their own music or not, people still know to recognize what music fits their desire. The tragedy resides in the fact that not everyone is able to create their own music (or at least instrumental music) if they don't have the proper training, even when they feel really drawn to, if . Thankfully, such people can always seek the help and service of the trained musicians. However, this usually doesn't come free. Whether it's in money, time, convenience or availability, there is a price to pay for not being able to be self-dependent, every so small.

This project arose from a very personal experience. Indeed, I am myself musically [self] trained since a very young age. It's always been a pleasure to work with friends and family on their various projects and provide help whenever possible. However, although this type of collaboration is done for free, time and availability are not always guaranteed. Thus, the idea came of a system that would replace this music expert and be able to help people with their music requests, on their own terms, as efficiently as a music expert would.

The milestone set for this project was to produce a system that would work in three steps: (1) take in a piece of audio/song and analyze it to correctly identify the different musical chords in the song; (2) produce an internal representation of the chords in a way that they can be used to perform actual music; (3) play back the music to a user. That is just what is being done in this work. We start off with an audio input (a wave file). This file is turned into its sound wave representation and fed into a trained Neural Net model that will identify the chords sequence from the audio. Those chords are then arranged in time in such a way that they are as coherent as possible with the audio and this arrangement is recorded in a [.chtr] file (stands for chord transcription). From those chords, we then use the MIDI library from PyGame to play the instrumental through the speakers.

Related Work

There are lots of work currently produced in the general area of chord recognition. More specifically, some work has been done in the area of music transcription using Neural Networks but the issue still remains unsolved to this day[1].

Dataset and Features

Lots of data was needed in order to correctly train our model and get reliable results. One benefit that we had in this work is that we were able to generate our own data, in whatever combination or nature we desired. This is that we were able to use the famous Music Software Fruity Loops Studio (aka FL Studio) to generate the audio samples that we needed. Indeed, to avoid facing the issue of overfitting the learning of our model to a very local and corner case, we took the precaution to avoid training using only audio produced by a single instrument or by similar sounding instruments. This is that we needed to generate audio samples with instruments such as pianos, string instruments, wind instruments, in any combination. Thankfully this process was made very easy through the software mentioned above.

Since we are using a supervised learning approach, the data needed to be labeled in order to be useful. We made that possible by ensuring that each audio file generated as data only represented one single ‘chord’ and was placed in the corresponding folder (bearing the name of the chord/label itself). Later, all those files are used in generating the actual data used by the model. This is that each file is read and their sound wave data are extracted as features, with the corresponding label. However, we have to consider that audio files such as these ones are typically sampled at a rate of 44100. Which means that for a 1 second excerpt, there will be 44100*2 sample points. This observation coupled with the fact that chords are susceptible to change in less time than one second, we decided to split these samples into smaller chunks. Therefore, each sample ends up with about 4410 sample points.

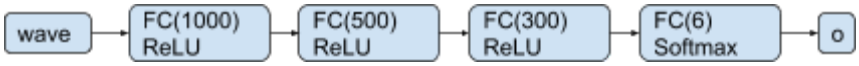
One issue is that these sound wave points are really the only features we get. Later on it became apparent that using more than one feature might help in improving the performance of our chord predictor. We then introduced additional features such as the timestamp associated with each point, as well as a modified sequence of those points where the wave was slightly smoothed out.

Work (Methods, Experiments & Results)

One Fully Connected Neural Network Model

The very first approach we considered for the chord recognition problem was using a Fully Connected NN. In this approach, we are using one model to predict the presence of any chord in a given audio. For the purpose of this research, we focused on only a subset of all the chords that can exist. More specifically, we focused on simple major and minor chords in the key of C. This is that we wanted out model to be able to analyze any audio with music presumably played in the key of C, and be able to correctly identify what chords were being played.

We have experimented with different architectures of the network to see where we were getting the most accurate results. The general graph followed the model



where the ReLU and softmax functions are of the form

$$ReLU(z) = max(0, z)$$

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

As far as evaluation metrics over all the methods, we put a bigger emphasis on the accuracy of the prediction as a bad accuracy pretty much invalidates everything else here.

What we observed from this method was that the model is capable of identifying chords quite well, as long as we don't get too technical with the variations of these chords, i.e it is able to correctly differentiate between chords such as C Major and D minor. But when go deeper into different variations such as differentiating between C Major and C minor, or "C minor 7" and "C minor 9", the model struggled a bit more.

An example of a confusion matrix is with only the 6 given chords

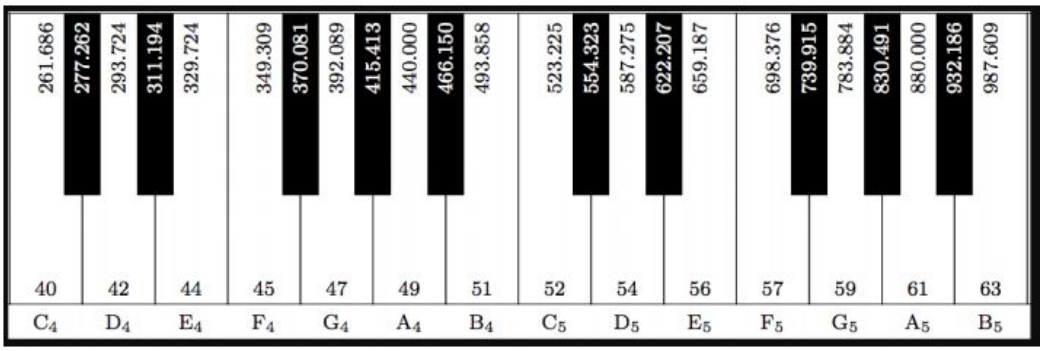
	Predicted Chord is present	Predicted Chord isn't present
Chord present	748	51
Chord not present	35	650

With more nuanced chords and a different sample set (namely C major and C minor) we have

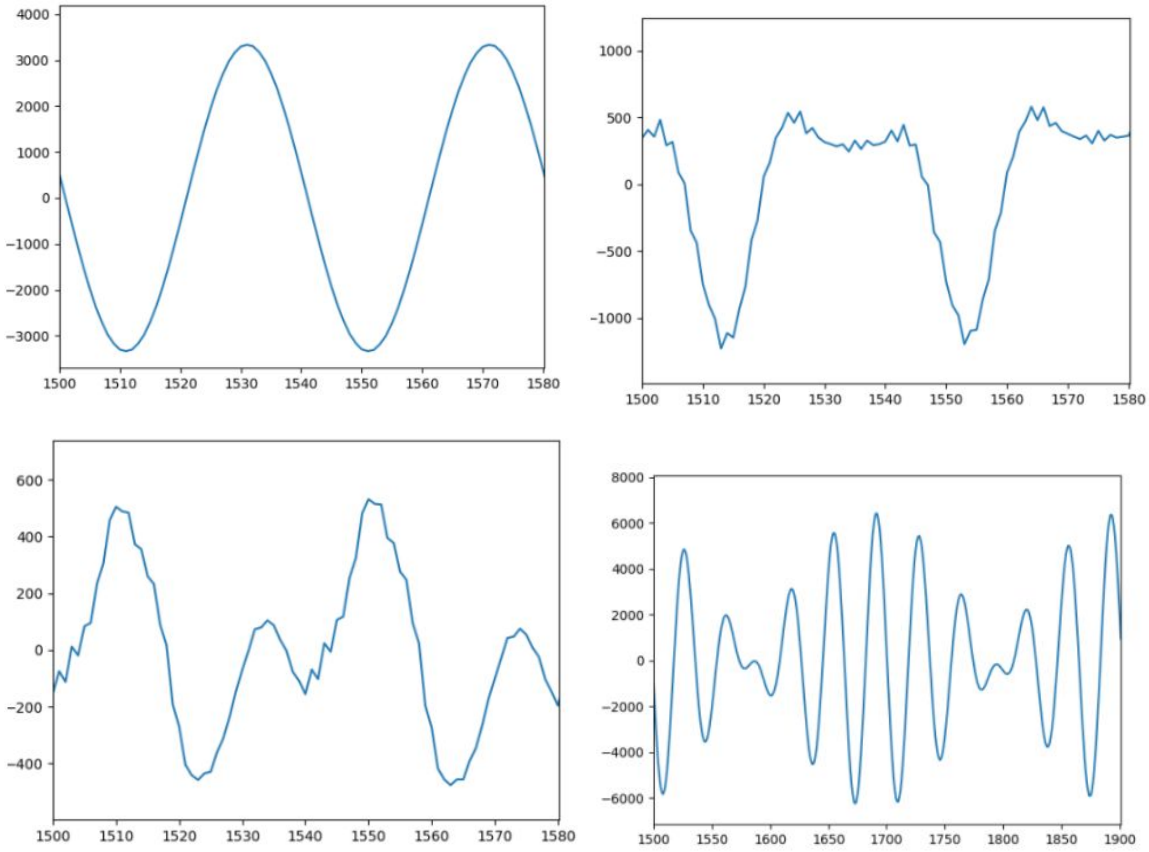
	Predicted Chord is present	Predicted Chord isn't present
Chord present	235	12
Chord not present	240	253

Multiple Fully Connected NN Models

This method is very similar to the first one in most of its architecture. The only difference here is that, instead of training the model on identifying all the given chords that we want to classify, we want the model to identify the presence of each musical note. We base this approach on the knowledge that there are only 12 musical notes that are repeated on different octaves. This allows us to focus on only one octave and generalize the results for any notes on any octaves. More specifically, we use the knowledge that the frequency of notes are related to each other. Indeed, as we can see in the figure below, the frequency of a note is twice the frequency of the same note on the octave below.



Furthermore, we use the knowledge that we have of sound and the frequencies of musical notes to utilize in our favor the concept of transferability. Indeed, for any instrument that we have, we can observe similarities in the sound that they produce if those instruments happen to be playing similar patterns/notes. For example, if we play a note of A on the 4th octave (aka A4) we obtain the following waves for a sine wave instrument on the top left, a piano on the top right and a violin on the bottom left, all sampled for $2 * \frac{1}{440}$ seconds (aka, two periods):



The similarity between these is that they all present a pattern that seems to be repeated 2 times. This is as expected considering that these instruments are all playing the same note. However, another difficulty arises when we combine multiple of those notes together as they produce only one audio signal which is the combination of all the constituting notes' singlas. The last graph above (bottom right) shows the result of the combination of a A4 and C5 from a sine wave instrument. We know that the resulting wave will still carry characteristics of the different waves that have been combined together. In other words, we know that the periods of the different notes will somewhat be reflected in the output signal. Therefore, this

method attempts to capture any semblance of a period in the given sound wave. More precisely, for each of the 12 models, we try to identify whether or not a given note is present in the audio. We then get 12 estimates that we are able to combine together to predict what is the most likely chord being played. While this sounds reasonable in theory, in turned out less efficient in practice. Indeed, we have the following matrix for one run.

	Predicted Chord is present	Predicted Chord isn't present
Chord present	241	330
Chord not present	326	156

Indeed, the accuracy seems to always be very low around 20 to 30%. We observe that this is very likely due to the fact that the model is trying to overfit to the training data which is being fed into it. More specifically, the model was having difficulties following our desired method which was to compare 2 periods-length of the wave (for a given note) and evaluate whether or note there seemed to be a repeated pattern.

Representation & Music Playing

Once we have the chords generated, we then save them in a .chtr file which keeps track of the chord predicted, the start time of the chord and the end time of the chord. Our music player will then use this file to perform the music. We use the MIDI package of the PyGame library to play the music.

Conclusion

We observe that it was much easier for us to use a fully connected model to predict the more general chords than it was to produce the more specific chords. This is understandable and consistent with the results that the state of the art methods are currently discussing as far as detecting each individual notes in a piece of music. The code source can be found at www.github.com/kndombe/muge

References

[1] Benetos E., Dixon S., Automatic Music Transcription. IEEE Signal Processing Magazine, 2019