Machine Learning for IoT – 2021/2022

Guitar Chords Recognition

Fabiana Vinci
s279918

Joana da Orada Gonçalves
s293583

1 Introduction

In the recent years the interest in music information retrieval (MIR) application has been continually growing. In particular, the automatic chord recognition is one of the most popular tasks. In the scope of the lecture Machine Learning for IoT a project based on sound recognition in the field of music was developed. It has the objective of recognizing the cords played by a guitar with the use of a Raspberry Pi. The missing data is one of the biggest barriers in this field so for the development of this project the creation of a dataset was mandatory. Consequently a Convolutional Neural Networks architecture was used for the classification of the chords. A MQTT communication protocol was implemented between the Raspberry and the local computer in use to preform a live recording and prediction.

2 Project Description

The aim of the project is chord recognition. The possible applications of the project are for either beginner guitar musician or orchestra application. On the one hand, for beginners, the possibility of learning the chords of a song based on the original played song. On the other hand the improvement of the already existing application[1] in the scope of the orchestra for automatic sheet music reading.

3 Dataset

The chords used correspond to the most basic ones played on the guitar, as follows: Am, Bb, Bdim, C, Dm, Em, F, G. In the music industry the different chords are played in different ways and for different guitar sounds, the strumming pattern may correspond to the rhythm of the song. In order to obtain the most general dataset the strumming patterns used were 4 and the types of guitar used were electric (many types), classical and acoustic. To have the most heterogeneous dataset possible, samples created virtually and real sounds were mixed. For the dataset completeness 20 different real guitars were used and played by different musicians, 5 virtual guitars from the application GarageBand and 25 computed audio with the usage of the[2] application based on plugin filtering.

Three different versions of the dataset were created and tested, each of them have a difference between the portion of virtual and real samples. The best performing one for our application corresponds to version 3 which contains in the training set 30 virtual guitars and 15 real guitars, and in the test set 5 virtual and 5 real guitars. More detailed specification can be found in the kaggle page[3].

4 Methods

4.1 Preprocessing

The first step was to verify if all the input audio had the frequency of 44100 kHz and Monochannel. Dealing with different length audios the zero padding option was considered and applied. The input data was all transformed, firstly with the Short-Time Fourier transform (STFT) in order to move from the time domain to the frequency domain. The second preprocessing corresponds to Mel-frequncy cepstrum coefficients

(MFCC) which extractions are considered a good approximation of the human perception.

The figure shows the preprocess pipeline for each audio.

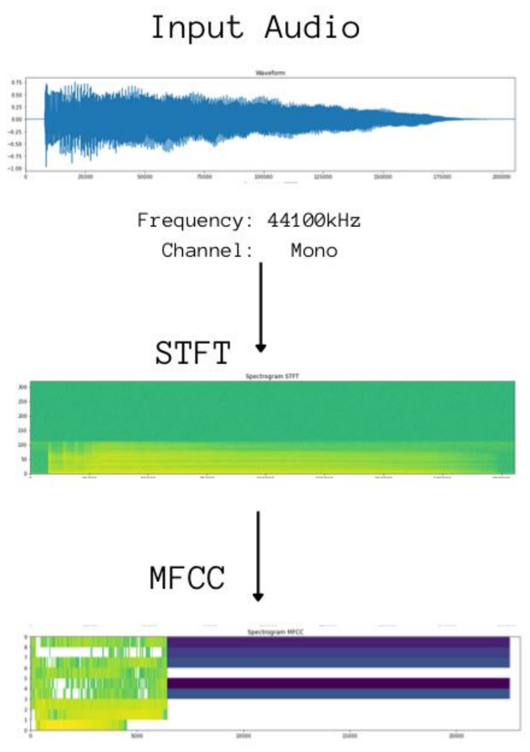


Figure 1: Preprocess pipeline

In the trail process, noise reduction was applied, but it did not bring improvements to the output result. This implementation was not particularize to our dataset, but in future, the noise reduction should be adapted to the dataset in question.

4.2 Models

The models used were CNN, DCNN and LSTM. Different testing and modification were made for example using the MaxAveragingPooling instead of GlobalAveragePooling, different numbers of filters, batch size and learning rate. LSTM model performances were not as good as the other two models. The best performance corresponded to the CNN model with 2D convolutional layers, Batch normalization, ReLU as activation function and Global average Pooling 2D as last step.

4.3 Communication protocol

In order to simulate the communication between the Raspberry Pi and the computer the MQTT protocol was implemented. The objective was to continue on having a live communication between the microphone/Raspberrypy, while the chord recognition is happening, and the computer terminal. The used broker was mosquitto and the topic "/279918/GuitarPred".

5 Results

The accuracy of this model corresponded to 80% for the test set. The two best results are the ones that follow.

Model	Epochs	Batch size	Accuracy Test
CNN	50	64	80%
DCNN	50	64	74%

The following graphs represent the loss and the accuracy of the train and validation set of the CNN model.

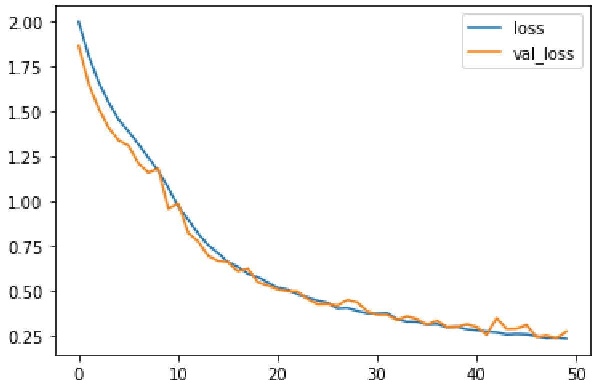


Figure 2: Loss graph

The Confusion Matrix shows the chords that are better predicted by the model.

During the testing with the raspberry, the chords that were more problematic correspond to the Em and to the F. It was also verified that the Environment in which the testing took place influences the audio entrance and also the prediction.

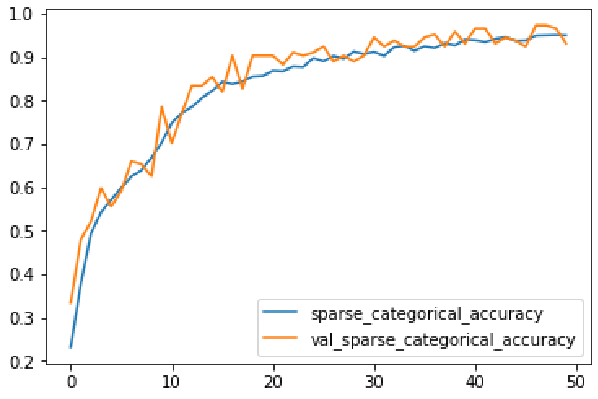
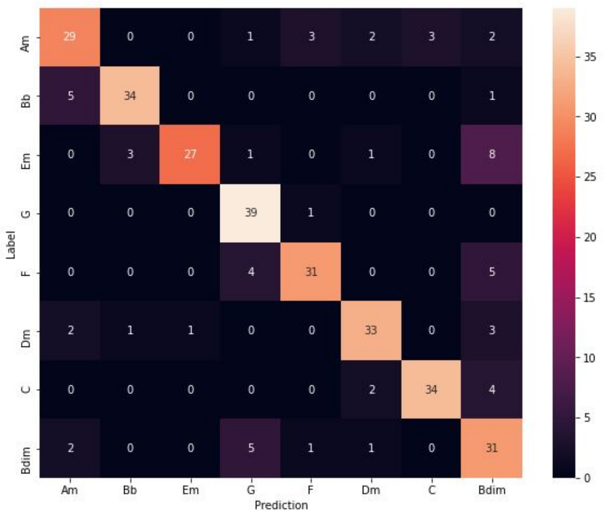


Figure 3: Accuracy graph



Label \ Prediction	Am	Bb	Em	G	F	Dm	C	Bdim
Am	29	0	0	1	3	2	3	2
Bb	5	34	0	0	0	0	0	1
Em	0	3	27	1	0	1	0	8
G	0	0	0	39	1	0	0	0
F	0	0	0	4	31	0	0	5
Dm	2	1	1	0	0	33	0	3
C	0	0	0	0	0	2	34	4
Bdim	2	0	0	5	1	1	0	31

Figure 4: Confusion Matrix

6 Conclusion

The application of the preprocess pipeline and the Convolutional Neural Network described in Section 4 constituted the turning point to obtain these results results. This project achieved the purpose of creating a new dataset and obtaining some promising results for future works.

References

[1] "Enote application."

[2] "Amplitube application."

[3] F. Vinci and J. Gonçalves, "Guitar chord recognition dataset," 2022.

[4] G. Byambatsogt, L. Choimaa, and G. Koutaki, "Guitar chord sensing and recognition using multi-task learning and physical data augmentation with robotics," *Sensors 2020, 20, 6077.*

[5] "Github code for guitar chord recognition at: https://github.com/joorada/guitar-sound-recognition."