Technical University of Cluj-Napoca

Faculty of Electronics, Telecommunications and Information Technology

Microprocessor Architecture

Musical instrument recognition implemented with Python

Coordinating teachers:

Prof. Dr. Ing. Mircea Giurgiu

Ing. Alexandra Drobut

Student:

Jimon Lucian-Daniel

Group 2031

2022

Table of Contents

Introduction

About the project

The present project entitled ‘Musical instrument recognition implemented with Python’ was structured and developed as a part of the ‘Microprocessors Architecture’ lecture, being a main aspect of its laboratory aims. The project is using several data and computer science theories and algorithms, which are all based on the microprocessors working principles, which were studied during the mentioned lecture.

The aim for the project was to develop a software, which based on genuine machine learning techniques can recognize and classify different musical instruments based on their specific sound.

The problem

The main problem for the project’s goal was to use a specific algorithm which can extract from sound files (specifically .wav files) the useful information, it can convert it to a data structure well understood by the PC’s microprocessor and it can learn, based on the data, the specific information about all the test musical instruments, which then can be used to show a human user the wanted result based on a sound file.

The solution

As the title of the paper suggests, the used programming language was Python, which among other extraordinary benefits comes with an extended usability of different libraries and pre-defined or open-source functions, specifically in the data science domain. This way Python was chosen as the main tool of the project.

The ‘translation’ of an audio file can be made in different ways, but the most simple and utile form of it is using the Fast Fourier Transform, which combined with the Mel Spectrogram made the essence of the project. While the FFT algorithm decomposes a signal into the spectral components, the Mel Spectrogram is a mapped form of the extracted spectra, which then can be transformed into data chunks, saved in a data structure. The open-source Python library ‘python_speech_features’ was used to get the Mel-frequency cepstral coefficients (MFCCs) of the audio files used.

The deep learning aspect of the project was solved by the K-Nearest Neighbors algorithm, which is largely used in classification projects and statistics.

Finally, as the scripts were intended to be used by human users, the GUI for the project was made with Tkinter, which is the standard GUI library for Python.

Python programming and machine learning techniques

Python is one of the most popular programming languages, being a fast, easy to understand and approachable language, with thousands of available libraries, making it the most used programming

language in several years (at the present hour Python is still the most used programming language, and has won the '2021 best programming language' award[1]).

One of the main uses of Python is in data science, and this can be proved by the multitude of machine learning libraries, the simplicity of the use of them in scripts and the ease of the language's syntax, making it perfect for purposes like this project.

Theoretical background

Fast Fourier Transform

The Fast Fourier Transform (FFT) is the algorithm which transposes a signal from time domain to frequency domain, by computing the discrete Fourier transform of the signal, which will result a set of values for amplitudes having different frequencies. As the original DTF algorithm can be slow when working with bigger sized data, the FFT factorizes the DFT matrix into a product, which will decrease complexity. It is mainly used in engineering, music industry, mathematics.

Mel Spectrogram

The Mel Spectrograms are graphical representations of audio signals, which were transformed with FFT into spectral values, which are then mapped based on their powers to the Mel scale. The Mel scale is a scale of frequencies, which by the human ear seem at equal distance from each other, but, their values are differing by a logarithmic function.

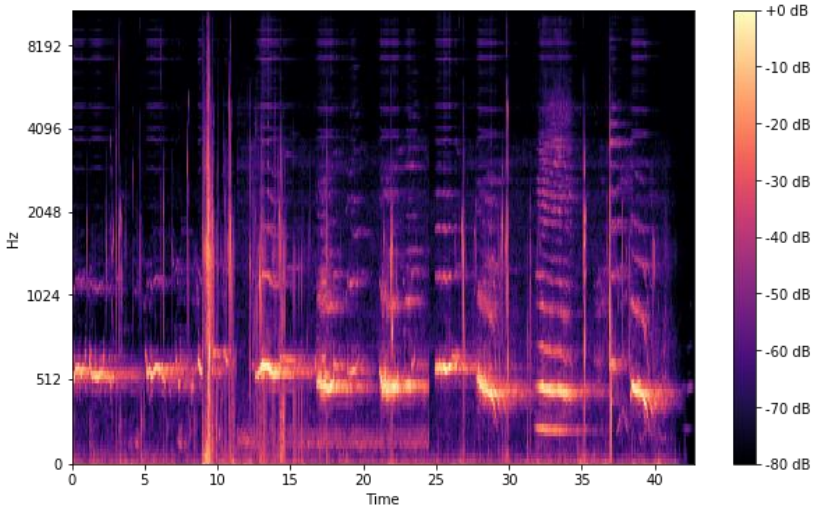

Figure 1: MEL Spectrogram

To plot the Mel Frequency Cepstral Coefficients (MFCC), which is the data to be plotted, there are four steps which need to be followed[2]:

1. Divide the input sample into several subsamples (usually 1024 or 2048).
2. Compute FFT for each subsample
3. Map the data on the Mel Scale
4. Generate Spectrogram with the obtain MFCC

[1] https://www.tiobe.com/tiobe-index/
[2] According to multiple sources mentioned in the References section

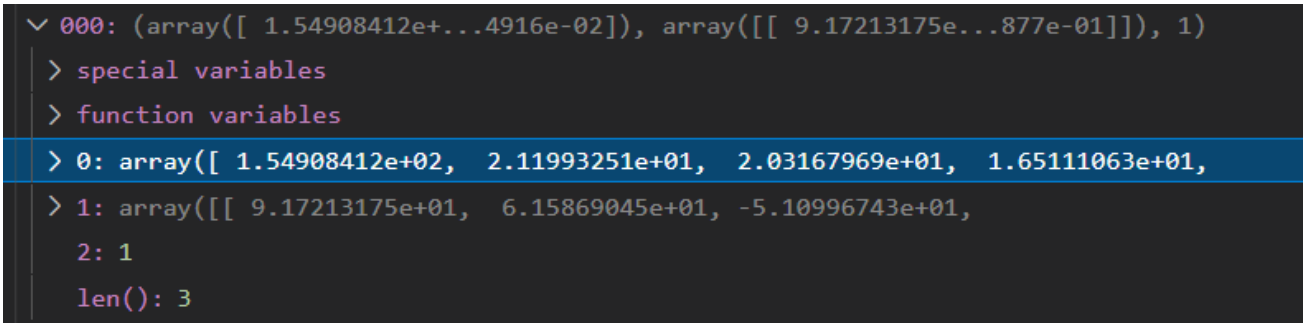

Figure 2: The structure of data for a single file, being an array of arrays

This will conclude a different graphic representation (spectrogram) for every unique sound signal, meaning a set of different coefficients for every sample, making it possible to compare them based on their frequency representation.

K-Nearest-Neighbours Algorithm

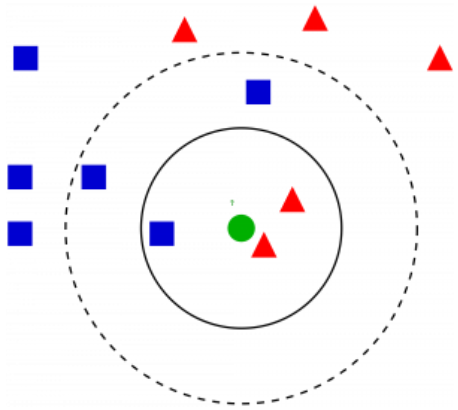

Figure 3: K-Nearest-Neighbours

The K-Nearest-Neighbours is one of the most common classification algorithms. It is considered a supervised machine learning technique, and is used for pattern recognition, data mining and intrusion detection[3].

The algorithm classifies a set of data based on their place in the dataset, sorted by their characteristics. This follows the principle that if a single entity of data is encircled with other entities having the same class, the data entity is question is most probably part of that class. The name of the algorithm provides from the fact that the number of nearby elements which are analysed for a piece of data is k. This k number is often an odd number, because if the surrounding items have different classes, to make possible a majority. As ‘k’ is increasing, the more precise the classification can be, but a too big value can cause important errors.

Implementation

The implementation of the project was realized within three ‘.py’ script files.

The ‘music_instruments.py’ file contains the machine learning section of the work, as it reads the music samples from the dataset, performs the analysation of the data (subsampling, calculating the MFCC, ordering the data, saving the data etc.) and creates a ‘.dat’ file which will store the trained set of the program.

The second file is name ‘test_instruments.py’ and it contains two parts: the implementation of the trained model by the first script and the implementation of the GUI which includes the classifier functionality developed.

[3] https://www.geeksforgeeks.org/k-nearest-neighbours/

The third file name ‘main.py’ is a main application script which calls the ‘test_instruments.py’ file. Its use in the case of an upgrade for the project, as it is optimal for a ‘py2exe’ conversion into an executable file.

Data set

The test subjects which are called ‘classes’ by the KNN algorithm were ten different musical instruments: bass, clarinet, double bass, flute, guitar, piano, saxophone, synth, trumpet and violin. Each folder for an instrument contains between 95 and 110 ‘.wav’ files, with lengths between 1 seconds and 4-5 seconds, a few of them having even a longer length, which is reasonable, as it is difficult to find this number of samples with no copyright.

The samples were taken from the Philharmonia UK website, from MARSYAS website and 10 files for the bass section were created manually with a Cort electric bass and a Harley Benton amplifier.

GUI

In the followings, the GUI of the application will be presented, which is developed within the ‘test_instruments.py’.

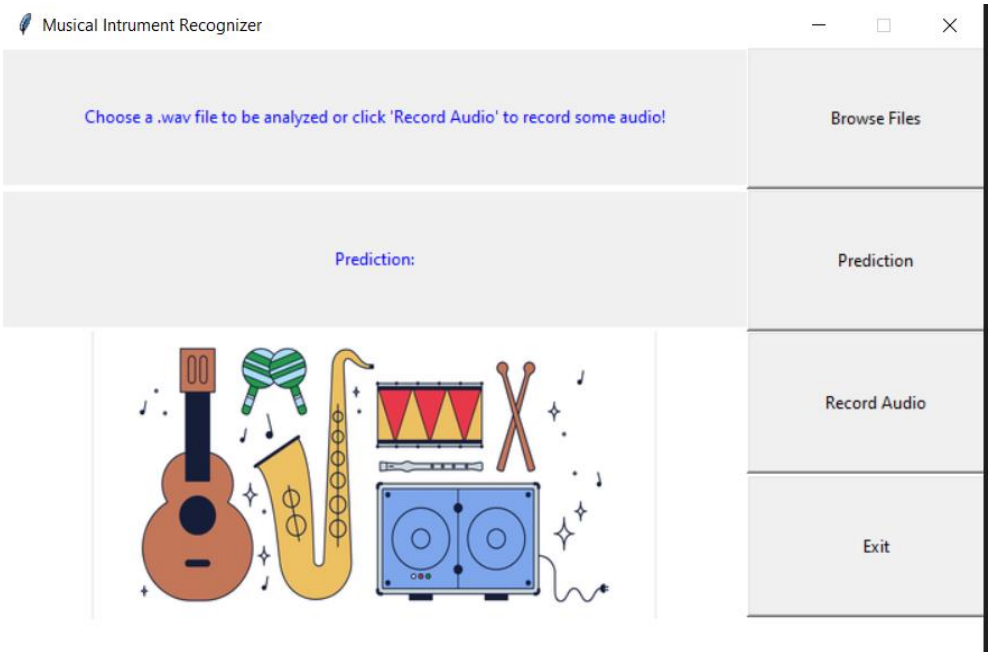

Figure 4: GUI of the application

As Figure 4 shows, the designed GUI has three main functionalities: to browse a .wav file using the ‘Browse Files’ button, to record audio with ‘Record Audio’ button and to make a prediction by pressing the ‘Prediction’ button, which will compare the data (the .wav file selected or recorded) with the trained model and will predict the musical instrument present in the sound file.

If a .wav file is chosen, the first label will change its content into the path of the selected file.

After pressing the ‘Record Audio’ button, the second label will change its text into “Recording audio...” and after five second the recording function will stop, the label will “Recording finished” and the recorded audio file can be used for a prediction, by pressing the “Prediction” button.

The result of the prediction is shown in the second label, initially filled with the ‘Prediction:’ plain text. When making a prediction, one of the ten subject instrument’s names will always appear in the specified place, like it can be seen in Figure 4.

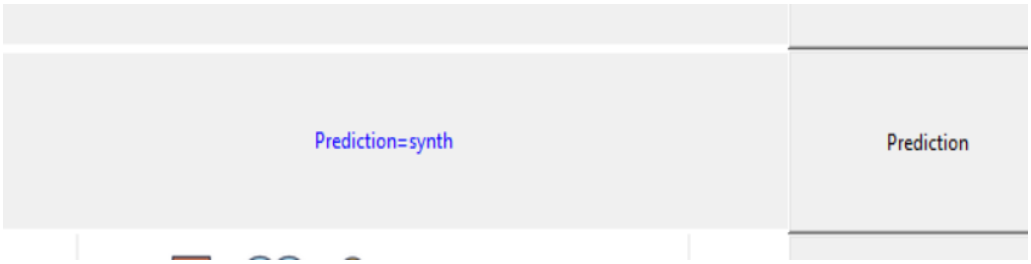

Figure 5: Prediction result shown

Additionally, an exit button can send the user out of the application, closing the Python-app window.

Results

The incorporated algorithm in the first Python script file was used with a split of the dataset of 80-20. This means that 80% of the files were used in the training process, and 20% were after that used to automatically test the classifier and to get a precision.

Also, the used k value for the KNN algorithm was 5, a commonly used value in literature.

The obtained result is shown below as it appears in the terminal.

```
(base) D:\Materii\EA-3\SEM1\up\proj\music instrument recognition>python -u "d
n-master\music_instruments.py"
The accuracy of the recognition model is: 0.926829268292683
```

Figure 6: Accuracy results

Another test phase was made manually with 10 randomly chosen sound data containing audio content of a single musical instrument, taken from the dataset not used from Philharmonia UK, from personal creation with the mentioned electric bass or from no copyright sources mentioned in References. The results of the inferences can be observed in the table below.

Table 1: Accuracy results 2

Index of sample	Musical instrument	Predicted instrument	Accuracy point
1	Bass	Bass	1
2	Violin	Violin	1
3	Cello	Violin	0
4	Bass	Bass	1
5	Guitar	Guitar	1
6	Synth	Synth	1
7	Synth	Synth	1
8	Bass	Bass	1
9	Piano	Synth	1
10	Flute	Flute	1
Total accuracy:			**90%**

It can be observed that the obtained 90% accuracy may be influenced from the small size of the testing set, but it is a reasonable start.

The accuracy and the overall functioning of the classifier program is although ponderously influenced by the type of the test subject: the clarity, the length and the texture of the sound file can influence the inference result, as the initial nearly 1000 pieces of data is considerably a small amount in comparison to the multitude of musical instrument sounds even for just these ten example instruments, as both the artistic variety and the overall recording quality are vast enough to minimalize the use of this project result for daily use.

Conclusion and possible improvements

As the goal for the project was to create a software which can recognize and classify different musical instruments based on their specific sound, the initial aim was successfully achieved with the help of the Mel Spectrogram, the KNN machine learning algorithm and in overall with Python.

As both the automatically and manually calculated accuracies are over 90% (92% and 90%), the functionality of the trained classifier model can be considered a reasonably good one.

Possible improvements include the use of a bigger data set for the training process, the optimization of the KNN algorithm to increase the inference process speed and to develop an executable file, based on the three '.py' script files developed. The first two improvements can enlarge the precision and usability of the software, while the third can make it more user-friendly.

In conclusion, the presented work was a successful development of a machine learning application, which can identify ten different musical instruments based on the sound, and it used Python as programming language, FFT and Mel Spectrogram as sound signal analysis techniques and KNN as machine learning algorithm. The project can be further developed by the mentioned improvements, and it can be used as a base of an embedded of mobile application with the same thematic.

Bibliography

1. Seipel, Fabial, Technische Universität Berlin, '*Music Instrument Identification using Convolutional Neural Networks',* Master Thesis, 2018, available at: https://www2.ak.tu-berlin.de/~akgroup/ak_pub/abschlussarbeiten/2018/Seipel_MasA.pdf, accessed on 11.01.2022
2. Geeks for Geeks website, '*Create a Voice Recorder using Python*', https://www.geeksforgeeks.org/create-a-voice-recorder-using-python/, accessed on 11.01.2022
3. Geeks for Geeks website, '*Build a Voice Recorder GUI using Python*', https://www.geeksforgeeks.org/build-a-voice-recorder-gui-using-python/?ref=lbp, accessed on 11.01.2022
4. Ramos, Leonidas Pozo, Peal Python website, '*Python and PyQt: Building a GUI Desktop Calculator*', https://realpython.com/python-pyqt-gui-calculator/, accessed on 11.01.2022
5. Guignard, Lewis and Kehoe, Greg, Stanford University, '*Learning Instrument Identification*', available at: http://cs229.stanford.edu/proj2015/010_report.pdf, accessed on 11.01.2022
6. Gupta, Shikha; Jaafar, Jafreezal; Fatimah, Wan wan Ahmad and Bansal, Arpit, Universiti Tecknologi PETRONAS, '*FEATURE EXTRACTION USING MFCC'*, available at https://aircconline.com/sipij/V4N4/4413sipij08.pdf, accessed on 11.01.2022
7. Downey, B. Allen, '*Think DSP - Digital Signal Processing in Python',* Green Tea Press Needham, Massachusetts, 2014, available at https://greenteapress.com/thinkdsp/thinkdsp.pdf, accessed on 11.01.2022
8. Doshi, Ketan, Towards Data Science website, '*Audio Deep Learning Made Simple: Sound Classification, Step-by-Step*', https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5, accessed on 11.01.2022
9. Saska, Colton, DiValerio, Mark and Molter, Matthew, Medium website, '*Building an Audio Classifier*', https://medium.com/@anonyomous.ut.grad.student/building-an-audio-classifier-f7c4603aa989, accessed on 11.01.2022
10. Data Flair website, '*Python Project – Music Genre Classification*', https://data-flair.training/blogs/python-project-music-genre-classification/?ref=morioh.com&utm_source=morioh.com, accessed on 11.01.2022
11. Marsyas website, GTZAN genre collection, http://marsyas.info/downloads/datasets.html, accessed on 11.01.2022
12. Carleton website, Computer Science Comps Project, Netflix Prize, '*k Nearest Neighbours*', https://cs.carleton.edu/cs_comps/0910/netflixprize/final_results/knn/index.html, accessed on 11.01.2022
13. Geeks for Geeks website, '*File Explorer in Python using Tkinter*', https://www.geeksforgeeks.org/file-explorer-in-python-using-tkinter/, accessed on 11.01.2022
14. Holy Python, '*GUI with Python: File Browser and Input Form (PySimpleGUI Part III)*', https://holypython.com/gui-with-python-file-browser-and-input-form-pysimplegui-part-iii/, accessed on 11.01.2022
15. Wikipedia website, '*k-nearest neighbors algorithm'*, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, accessed on 11.01.2022

16. Harrison, Onel, Towards Data Science website, '*Machine Learning Basics with the K-Nearest Neighbors Algorithm*', https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, accessed on 11.01.2022
17. TopPNG website, '*music instruments vector pack free png graphic cave - vector music instrument PNG image with transparent background'*, https://toppng.com/music-instruments-vector-pack-free-png-graphic-cave-vector-music-instrument-PNG-free-PNG-Images_219491, accessed on 11.01.2022
18. Doshi, Ketan, Towards Data Science, '*Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better*', https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505, accessed on 11.01.2022
19. Wikipedia website, '*Python (programming language)*', https://en.wikipedia.org/wiki/Python_(programming_language), accessed on 11.01.2022
20. Statista website, '*Most used programming languages among developers worldwide, as of 2021'*, https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/, accessed on 11.01.2022
21. Gartzman, Dalia, Towards Data Science website,' Getting to Know the Mel Spectrogram', https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0, accessed on 11.01.2022
22. Wikipedia website, '*Mel scale*', https://en.wikipedia.org/wiki/Mel_scale, accessed on 11.01.2022
23. Wikipedia website, '*Mel-frequency cepstrum'*, https://en.wikipedia.org/wiki/Mel-frequency_cepstrum, accessed on 11.01.2022
24. Wikipedia website, '*Fast Fourier transform'*, https://en.wikipedia.org/wiki/Fast_Fourier_transform#Cooley%E2%80%93Tukey_algorithm, accessed on 11.01.2022
25. Mlearnere, Towards Data Science website, '*Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients*', https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8, accessed on 11.01.2022
26. PNSN website, '*What is a Spectrogram?*', https://pnsn.org/spectrograms/what-is-a-spectrogram, accessed on 11.01.2022
27. Geeks for Geeks website, '*K-Nearest Neighbours*', https://www.geeksforgeeks.org/k-nearest-neighbours/, accessed on 11.01.2022
28. Srivastva, Tavish, Analytics Vidhya, '*Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R)*', https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/, accessed on 11.01.2022
29. Philharmonia, Sound samples, https://philharmonia.co.uk/resources/sound-samples/, accessed on 11.01.2022
30. Roberts, Leland, 'Analytics Vidhya website, '*Understanding the Mel Spectrogram'*, https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53, accessed on 11.01.2022