eSTUDENT

Instrument Detection in Audio Signals Using Machine Learning Methods

Technical Documentation

LUMEN Data Science Competition 2023
at the University of Zagreb

Contents

1 Introduction

Music is present in our daily lives, appearing in all cultures throughout history and constantly evolving. Accordingly, services related to music are also developing. Major streaming services strive to improve their user experience by editing content and personalizing recommendations for their users. To do all of this, they possess powerful tools for musical analysis and retrieving various information about songs.

One of the most important pieces of information is the presence of various instruments. The ability to automatically classify instruments (including vocals) within a song enables various applications, such as easier searching, more accurate recommendations, and even further detailed and complex analysis. Although detecting instruments may seem like a relatively easy task to the human ear, automating such a process is not trivial. The high entropy of information contained in audio signals, the wide range of sources, the mixing process, and the difficulty of analytical description require somewhat more complex solutions.

This technical documentation will serve as a guide to our solution of the given problem at hand: classifying audio signals with regard to the instruments contained in them.

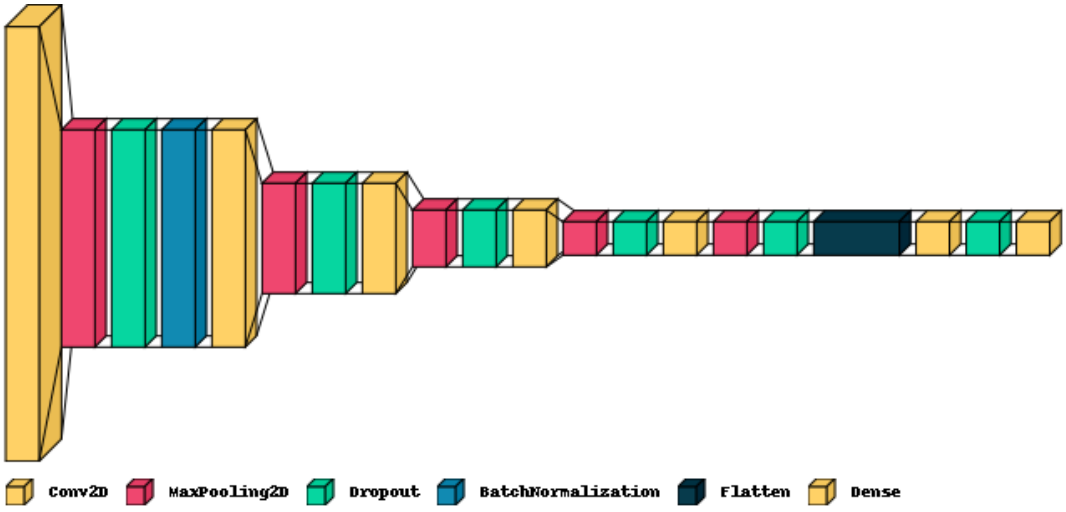

Figure 1: Architecture of our Solution Model

2 Setup

2.1 Technical Specifications

The technical specifications of three personal computers used for this project are listed below.

2.1.1 Computer specifications

- Computer 1:
  - Processor: AMD Ryzen 3 5300U 2.60GHz (4 cores)
  - RAM: 7.31 GB (usable)
  - Operating System: 64-bit OS
  - Processor Architecture: x64-based
- Computer 2:
  - Processor: Intel® Core™ i5-10310U CPU @ 1.70GHz (4 cores)
  - RAM: 15.8 GB (usable)
  - Operating System: 64-bit OS
  - Processor Architecture: x64-based
- Computer 3:
  - Processor: 12th Gen Intel® Core™ i5-1235U 1.30GHz (4 cores)
  - RAM: 15.7 GB (usable)
  - Operating System: 64-bit OS
  - Processor Architecture: x64-based

2.1.2 Package Versions

- Computer 1:
  - Python: 3.9.5
  - librosa: 0.10.0
  - numpy: 1.23.5
  - scikit-learn: 1.2.1
  - tensorflow: 2.12.0
- Computer 2:
  - Python: 3.6.8
  - librosa: 0.9.2
  - numpy: 1.19.5
  - scikit-learn: 0.24.2
  - tensorflow: 2.6.2
- Computer 3:
  - Python: 3.9.16
  - librosa: 0.10.0
  - numpy: 1.22.1

- scikit-learn: 1.2.1
- tensorflow: 2.12.0

2.2 Dataset Details

The dataset provided for this project consists of 6,705 monophonic *.wav* files, with each file being 3 seconds in length.

After some initial research and tests, a new dataset named MIX was created. The MIX dataset contains approximately 6,700 *.wav* files, where each file is a combination of exactly two monophonic *.wav* files from the original dataset. The aim was to maintain the distribution of audio files for each instrument relative to each other similar to the original dataset. The labels for each *.wav* file are saved in separate *.txt* files with the same name and *.txt* extension, where each *.txt* file contains shortcuts for the instruments listed one below the other.

Additionally, new training examples were created by combining two, three, and four monophonic audio files from the original training dataset. This resulted in 4,000 new combinations for each case, totaling 12,000 new training samples. For each set of 4,000 combinations, 2,000-3,000 examples were labeled with 1 for the instrument of interest, and the remaining examples were labeled with 0. The new training samples were organized into separate folders named `MIX2_InstrumentShortcut` for each instrument, resulting in 12,000 samples per instrument.

2.3 Data Preprocessing

Through some experimentation (see more in the project documentation of our project), we have deduced that the best results are obtained when training our models on datasets where the classes are equally represented. Therefore, we decided to create additional training examples by combining two, three, and four monophonic audio files from the original training dataset. We created 4,000 combinations of each, resulting in a total of 12,000 new training samples. For each set of 4,000 combinations, we generated 2,000-3,000 examples with label 1 for our instrument of interest, and the rest with label 0. This process resulted in the creation of a new folder named `MIX2_InstrumentShortcut` for each instrument we investigated, containing 12,000 new samples.

For our final models, we used all the samples from the original dataset with label 1, along with approximately half of the remaining samples. Additionally, we included all samples from the `MIX` folder with label 1, and approximately half of the remaining samples. Furthermore, we incorporated all the samples from the `MIX2_InstrumentShortcut` folder. Depending on the instrument, this resulted in approximately 22,000 samples, with approximately 45% having label 1 and the remaining 55% having label 0. These datasets were used to train our models. For a better understanding of the process, please refer to `Create_training_datasets.ipynb` and `gel_final_model.ipynb`.

For training purposes of our network, 20% of these samples were set aside as a temporary validation set, while the rest were used for training. We reported our results on the three validation datasets provided by the organizers.

2.4 Data preprocessing execution

We will now explain the setup required to reproduce our results. It is important to note that the Python libraries used may differ among the members of our team, and the control of random states was not uniformly applied. As a result, it is not possible to reproduce the results exactly, but similar results can be achieved.

The first step is to download and unzip the original training dataset and the three validation datasets provided by the organizers. If the paths to the unzipped datasets are:
`C:\Users\YOUR_USERNAME\Downloads\IRMAS_Training_Data\IRMAS_Training_Data` and
`C:\Users\YOUR_USERNAME\Downloads\IRMAS_Validation_Data_part2\IRMAS_Validation_Data_part2`,
the only adjustment required to run our notebooks is to replace `YOUR_USERNAME` with your personal computer username at the start of each notebook. Otherwise, you should adjust all the paths in the notebooks to match your system configuration.

The next step is to run the notebook `Create_training_datasets.ipynb` and follow the instructions in there. After we do so for one instrument, we are left three datasets that we will later use for training, which were already mentioned in this report.

2.5 Training Process

We made use of ML methods for our models. Specifically, we trained CNNs for each intrument to act as a binary classifier. In ML training, one always needs to balance between eliminating bias (fitting the model) and limiting variance (preventing overfitting). Generally, this often looks like this:

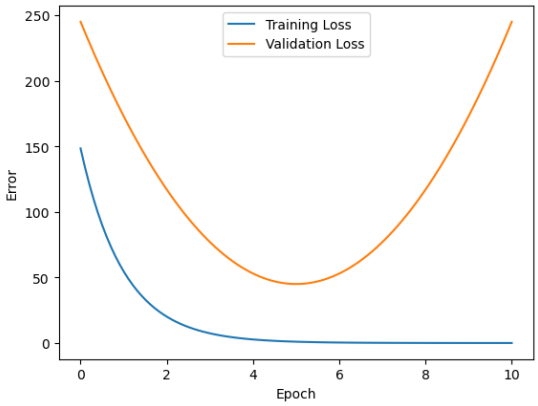

Figure 2: Training and Validation Loss

The training loss further decreases with each epoch, while the validation loss at some point starts stagnating due to overfitting. We would ideally like to exit just before that happens, at the vertex of the parabola. Concrete results of the training can be found in the project documentation.

3 Web application

In this section, we will describe each content of the final solution we have submitted for the competition **LUMEN Data Science 2023**. To run the web application, please follow the steps below:

1. Unpack the downloaded .zip file into a folder of your choice.
2. Open the terminal and navigate to the destination folder.
3. Run the command `cd loomen` to enter the application directory.
4. Install the required dependencies by running the command `npm install`.
5. Start the application by running the command `ng serve`.
6. Open a new terminal window and navigate to the unpacked folder's destination.
7. Install the necessary Python modules by running the following commands one by one:
   - `pip install IPython`
   - `pip install tensorflow`
   - `pip install os`
   - `pip install numpy`
   - `pip install librosa`
   - `pip install wave`
   - `pip install soundfile`
   - `pip install math`
   - `pip install flask`
   - `pip install flask_cors`

   Run each command separately to install the specified module.
8. In the terminal, run the command `python kontroler.py`.
9. Open your web browser and visit `http://localhost:4200`.

Four images below show the application being used.

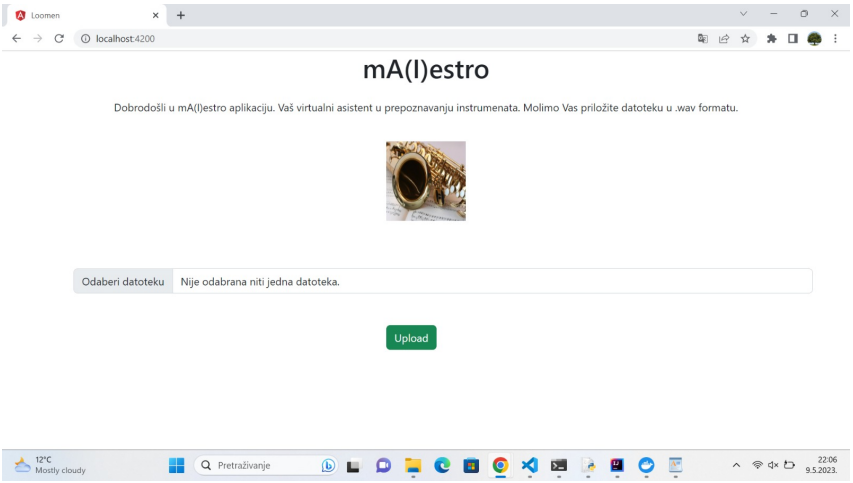

Figure 3: Start of the Interface of our Application mA(I)estro

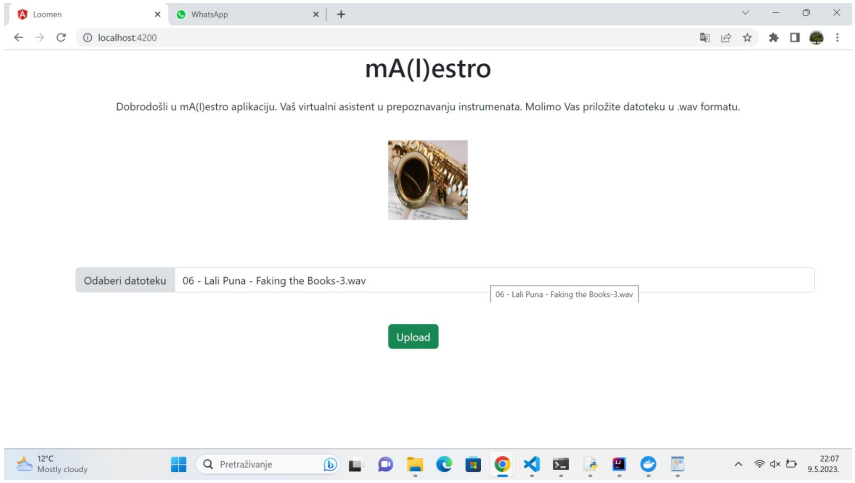

Figure 4: How to Upload a File

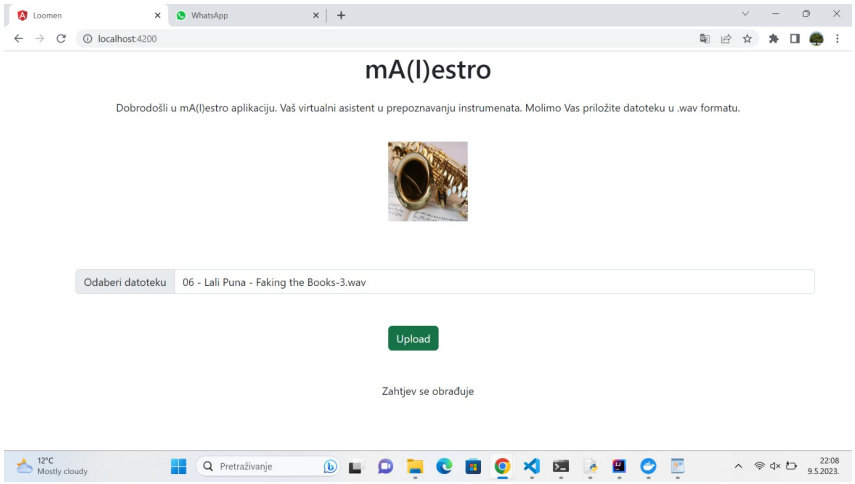

Figure 5: The Request gets Processed

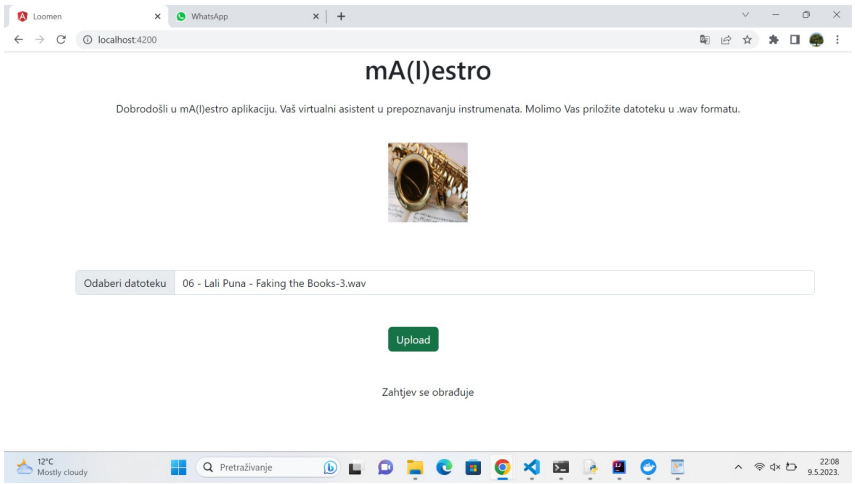

Figure 6: The App Predicts

4 Contents

In this section, we will provide a detailed description of the contents included in the final solution submitted for the **LUMEN Data Science 2023** competition.

4.1 Notebooks

The following notebooks are included in the submission, each serving a specific purpose:

- `Create_training_datasets.ipynb`: This notebook is utilized to configure the training dataset for our final approach using MFCC images. It contains comprehensive information and instructions on the dataset configuration.
- `gel_final_model.ipynb`: This notebook encompasses the creation of the electric guitar recognition model trained on the dataset created from the original validation data. It represents the model we have chosen as our final submission for the competition. By modifying the *instrument_of_interest* variable at the beginning of the notebook, this model can be adapted for other instruments, with having the datasets prepared as explained in the notebook. For instance: *instrument_of_interest='gel'*.
- `gac_final_model.ipynb`: This notebook is similar to the predecessor one for the acoustic guitar recognition.
- `org_final_model_mix.ipynb`: This notebook contains the instructions and the code needed to create our model for the organ recognition where the training dataset *MIX2_InstrumentShortcut* is used. One can change the *instrument_of_interest* variable at the start to recreate the process for other instruments as well, provided that the *MIX2_InstrumentShortcut* has previously been created.
- `Spectrogram_model.ipynb`: This notebook can be used to train one of our models in a similar fashion as before, using the *MIX* training datasets, but using mel spectrograms instead of MFCC images.
- `Tensor_validation_datasets.ipynb`: This notebook is responsible for generating tensors that incorporate MFCC images for the provided validation samples. The validation data is divided into 3-second intervals, as explained in detail in our project documentation. Each interval's corresponding MFCC image and label are saved in this process.
- `Audio_predict.ipynb`: This notebook tests our final models for an *.wav* audio file input. The same code is being used in our web application.
- `Test_to_json.ipynb`: This notebook contains the final evaluation of our models on the test dataset provided by the organizers of the competition. It creates a *.json* file with our final results.

4.2 Models

This folder contains all of our final models, on which we have based our performance evaluation. It includes the models employed for predicting the provided test dataset. About half of the models are based on the *MIX2_InstrumentShortcut* datasets, while the other half is based on the training dataset created from the given validation sets.

4.3 mA(I)estro

This folder contains the implementation of our web application `mA(I)estro`. The application allows its users to upload a *.wav* audio file and it returns our prediction for the 11 music instruments of interest we modeled. See the Web Application section for the instructions manual.

4.4 Documentation

The following files, located in the folder `Documentation`, are included in the submission:

- `LUMENDS2023_Project_documentation.pdf`: This *.pdf* file encompasses a comprehensive project documentation, outlining our research progress and findings throughout the project. It covers initial approaches, methodologies, and final results.
- `LUMENDS2023_Technical_documentation.pdf`: The current file being read is a *.pdf* that serves as detailed documentation. It includes instructions for executing our notebooks and models, an explanation of the web application and its usage, as well as a comprehensive list of the submitted solution's contents.
- `LUMENDS2023_Final_predictions.json`: This *.json* file contains our final predictions on the test dataset provided by the competition organizers. The file adheres to the same format as the example provided by the organizers, ensuring consistency in the submission.

Please note that the contents provided in this section form the entirety of our submission for the **LUMEN Data Science 2023** competition.

5 References

[Ash23] Moshin Ashraf. *A Hybrid CNN and RNN Variant Model for Music Classification*. https://www.mdpi.com/2076-3417/13/3/1476, 2023.

[Bla22] Maciej Blaszke. *Musical Instrument Identification Using Deep Learning Approach*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9025072/, 2022.

[Dos21] Ketan Doshi. *Audio Deep Learning Made Simple (Part 1, 2, 3): State-of-the-Art Techniques*. https://towardsdatascience.com/audio-deep-learning-made-simple-part-1-state-of-the-art-techniques-da1d3dff2504, 2021.

[Gur19] Siddharth Gururani. *An Attention Mechanism for Musical Instrument Recognition*. https://arxiv.org/abs/1907.04294, 2019.

[Han16] Yoonchang Han. *Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music*. https://arxiv.org/pdf/1605.09507.pdf, 2016.

[Kaw19] Nadim Kawwa. *Can We Guess Musical Instruments With Machine Learning?* https://medium.com/@nadimkawwa/can-we-guess-musical-instruments-with-machine-learning-afc8790590b8, 2019.

[Li15] Peter Li. *Automatic Instrument Recognition in Polyphonic Music Using Convolutional Neural Networks*. https://arxiv.org/abs/1511.05520, 2015.

[Sze22] Dominika Szeliga. *Musical Instrument Recognition with a Convolutional Neural Network and Staged Training*. https://www.sciencedirect.com/science/article/pii/S1877050922011966, 2022.

[Tae21] Michael Taenzer. *Deep Learning-Based Music Instrument Recognition: Exploring Learned Feature Representations*. https://cmmr2021.github.io/proceedings/pdffiles/cmmr2021_24.pdf, 2021.

[Yun17] Mingqing Yun. *Deep Learning for Musical Instrument Recognition*. https://www.semanticscholar.org/paper/DEEP-LEARNING-FOR-MUSICAL-INSTRUMENT-RECOGNITION-Yun/ad4201d862fd0952d8028697d505ad7697337292, 2017.