Classification of tools sounds using machine learning methods

Student: Rakhlevski Ilia

Course: Machine Learning in Speech Processing Technologies.

Abstract

In this project is presented a method of sounds classification based on envelope shape descriptor recognition. This approach could be useful for such classification tasks, as music genre, speech/music, musical instruments classification and others [4]. I applied it for tools sounds classification.

Introduction

As a background for this project was my work at Magal company. I took part in development of security fence [5]. One of the tasks was recognition/ classification of sensor signals received from the tools which can be used by terrorists for digging a tunnel: hammer, jackhammer and etc. The application had to be real-time and had to run on a machine with very restricted resources, like CPU, memory, GPU, ideally on a microcontroller.

So, the chosen method must be simple, but effective.

I did not finish this project, but it really interested me and I want to perform it with some improvements. As a data I am going to use audio files containing relevant sounds.

Proposed method

Envelope models are common in speech and audio processing [4, 6, 7, 8]. An envelope is defined as a continuous, usually smooth shape describing a characteristic of the signal. See Figure 1 for an example.

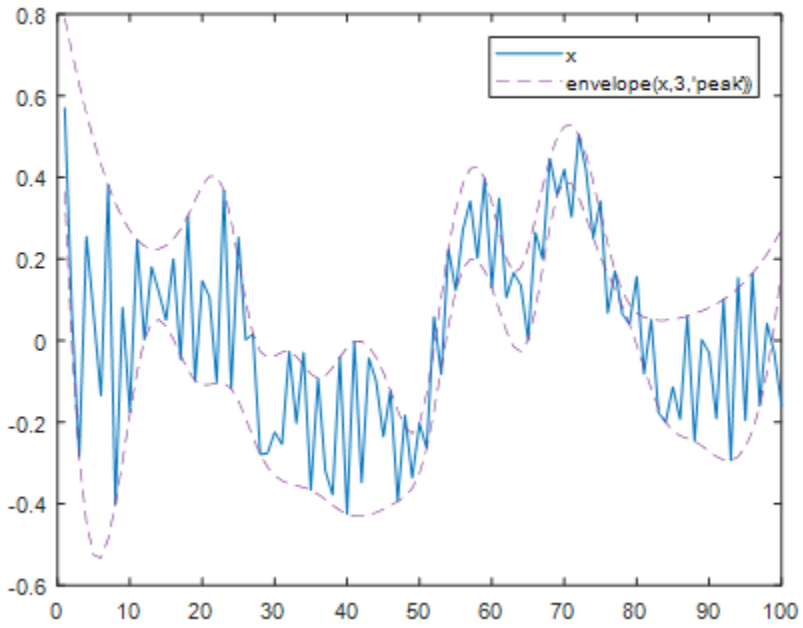

Figure 1. A signal with two-sided envelope

Example of using: phonocardiogram. Recording sounds of heart and processing them [9, 10].
The envelope can be either one-sided or two-sided. Also, it can be built on time domain, frequency domain and combined time-frequency domain. The representation is used in this project is two-sided and at this stage is built in time domain because the spectral domain requires more calculations and can worse the performance.
The received envelopes can be classified by machine learning methods. Due to the fact that we are talking about pattern recognition I use convolutional neural network well-proven in image classification. We can consider the two-sided envelope as an image.

Project

Development Tools: Python 3.7, Anaconda / Spyder / IPython.

Libraries: NumPy, SciPy, Pandas, Keras / TensorFlow.

Data: WAV files containing sounds of tools [1], [2].

Description: The goal of this project to develop a model that classifies sounds of tools, for example: hammer, jackhammer, drill and etc.
The model will use machine/deep learning methods.

Tools

The tools that are used in the project:

Hammer, jackhammer, drill. See Figures 1, 2, 3.

Hammer:

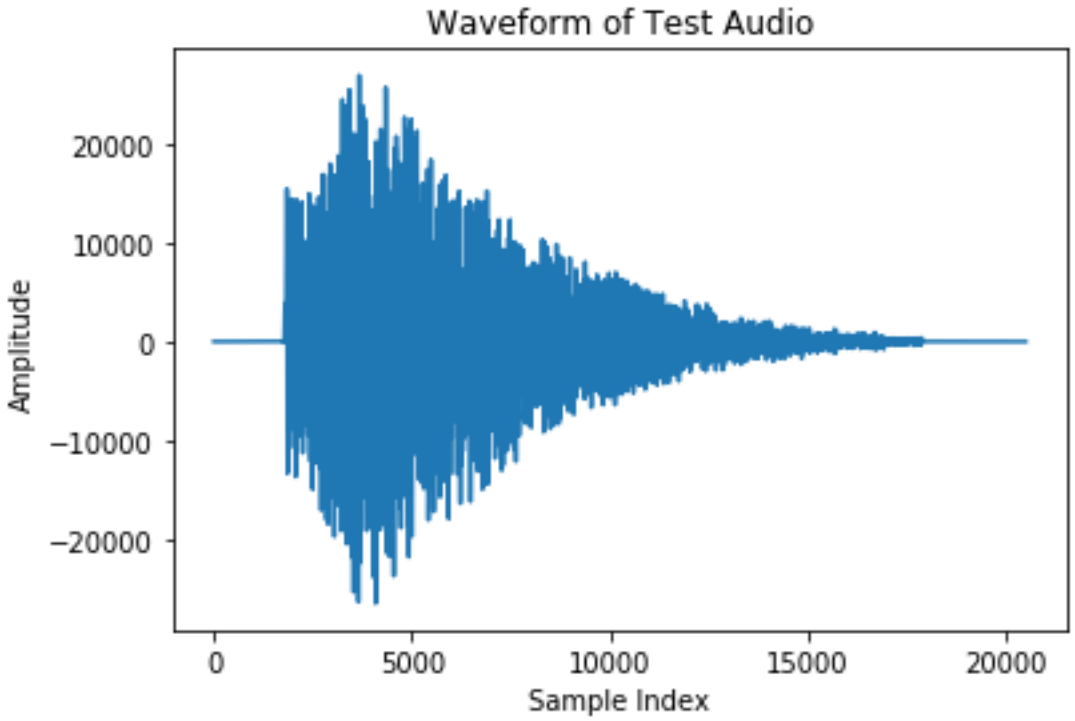

Figure 2. Waveform of hammer

Jackhammer:

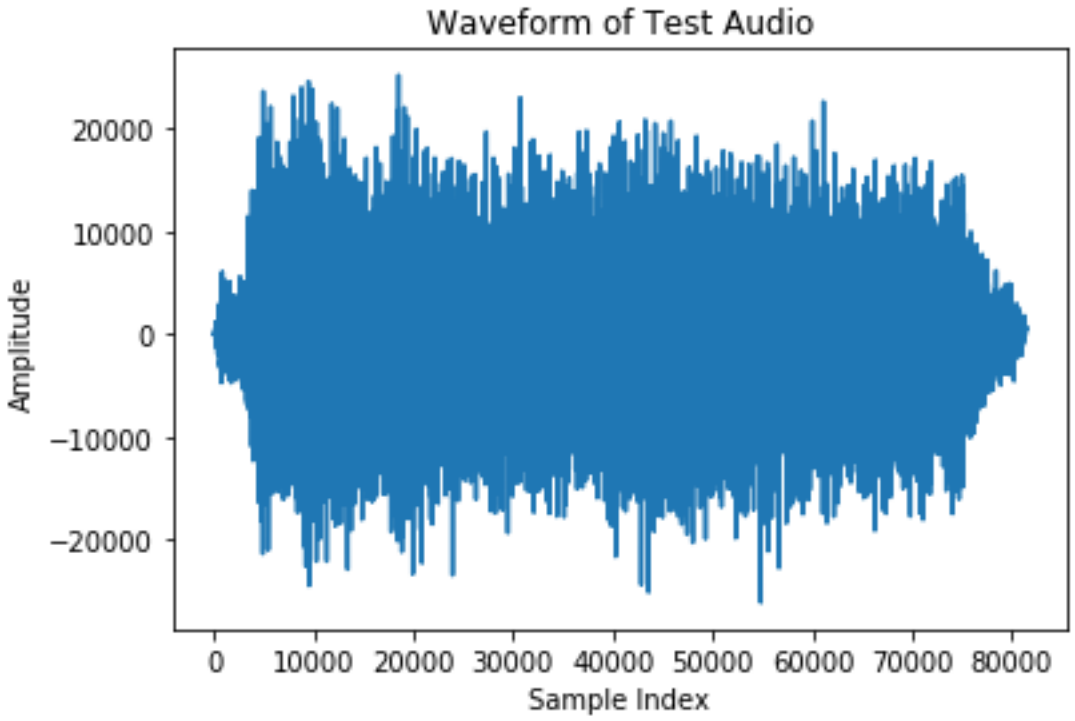

Figure 2. Waveform of jackhammer

Drill:

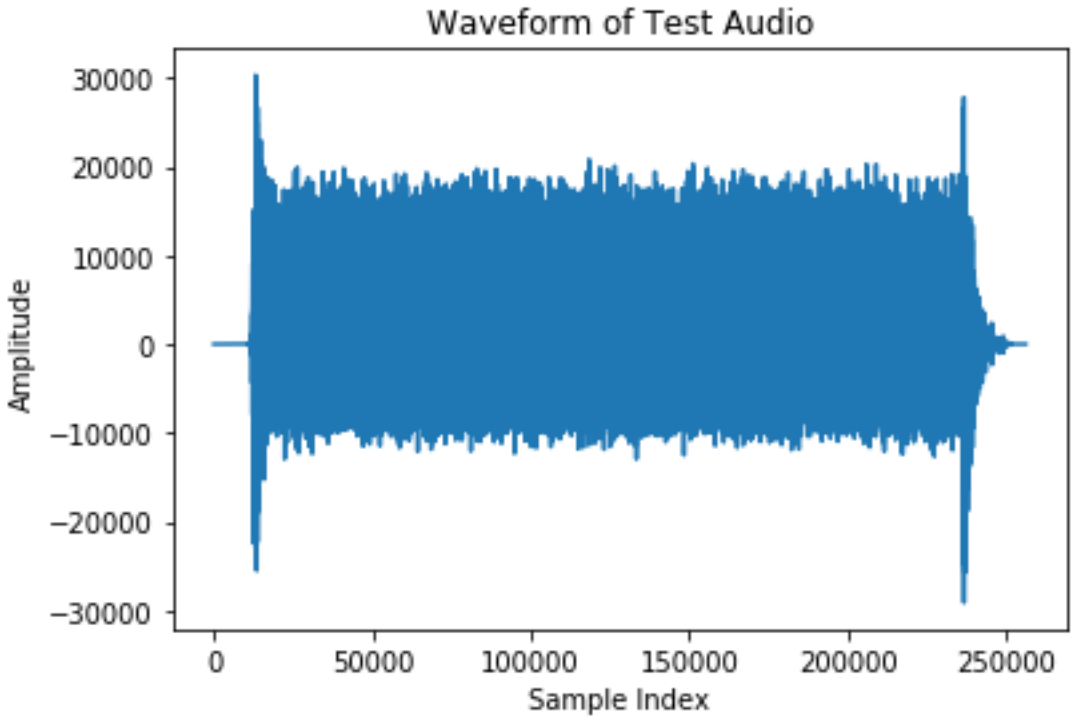

Figure 3. Waveform of drill

Method

Data preparation

1) Changing sampling rate: resample the original audio to 8000 Hz.
   The audio data can be sampled in many rates, for example:
   8 kHz, 44.1 kHz, 48 kHz, 96 kHz and etc.
   In this project I resample all the audio data to the same rate.
   I use the rate 8 kHz. This rate reduces the audio data size, but it still can be classified. See Figures 4, 5.

Example of an audio file resampling:

Original data: 48 kHz

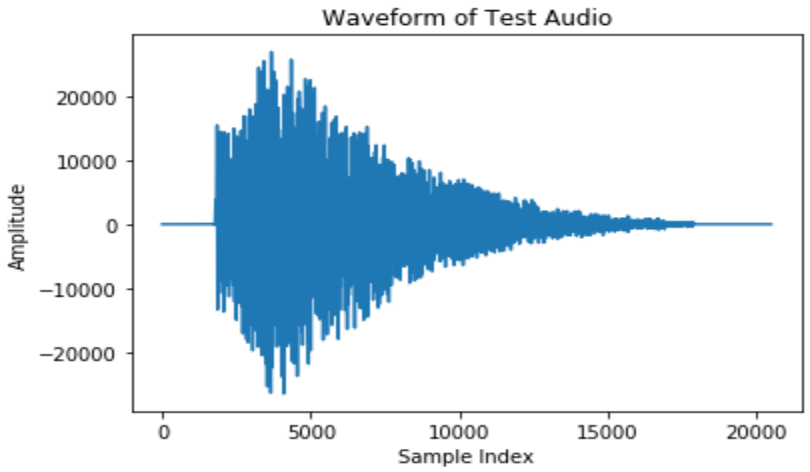

Figure 4. Original waveform

Resampled data: 8 kHz

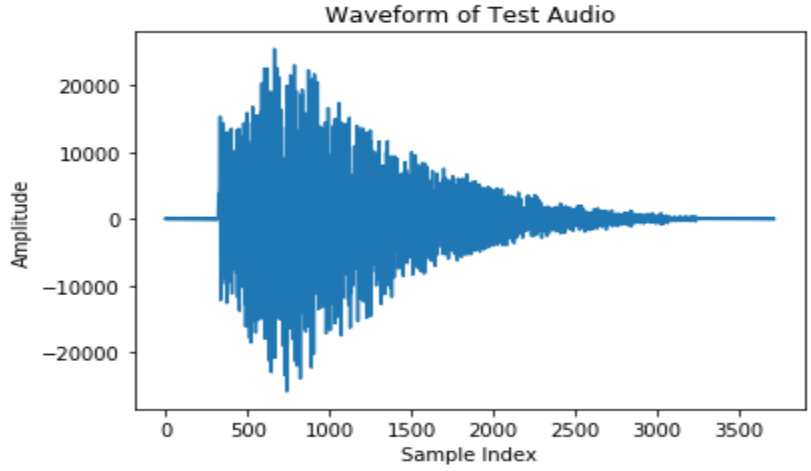

Figure 5. Resampled waveform

2) Normalization: normalize the data to range [-1, 1].
   Audio data can have different amplitudes. In order to train or to classify it we need to unify it. So, we bring it to the range [-1,1].

   See Figure 6.

   Example:

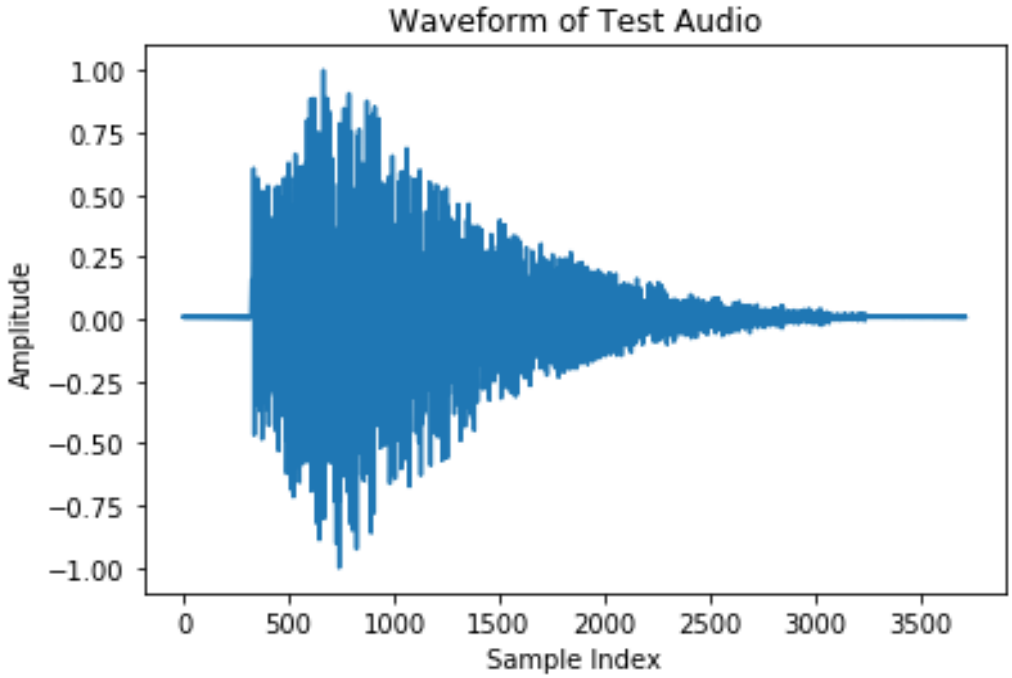

Figure 6. Normalized waveform

3) Fixing number of samples: adjust number of samples to the window size (4000 samples).
   If the current samples number is grater, then cut them. If it is less than the window size then zeros is added (padding).
   See Figure 7.

Example:

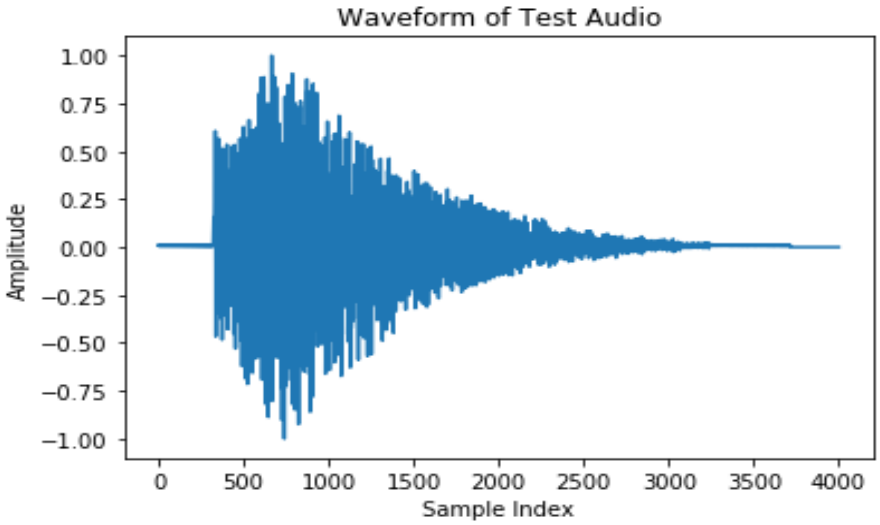

Figure 7. Waveform with adjusted size

4) Data peaks envelopes:

At this stage we calculate peaks envelope for the data from the previous stage.

This process can be repeated several times. It means to receive envelopes on previously created envelopes. In this project this process is performed twice. See Figure 8.

Audio data peaks envelope:

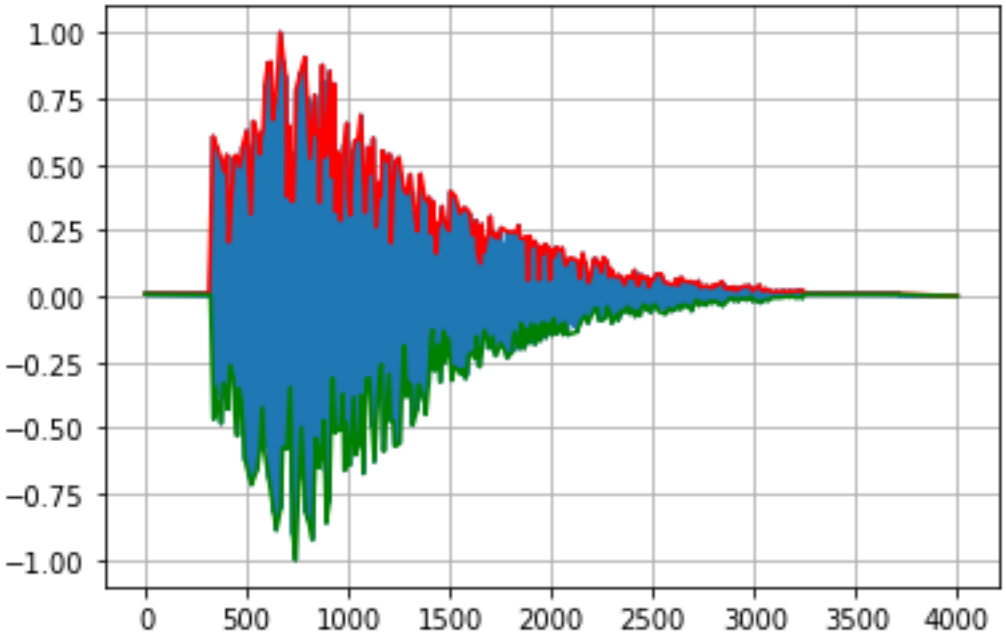

Figure 8. Waveform with peaks envelope

Neural network architecture

How it was mentioned above for audio data classification is used a convolutional neural network. In the project I use a network is based on Conv2D layer [3].

Input: envelopes that received from the stage of data preparation.

Output: tool class (hammer, drill or jackhammer).

Audio files classification

As a rule, an audio file is longer than the window (4000 samples) size. So we need to divide the file into chunks and to process each chunk.

This process is called windowing. See Figures 8, 9.

We take a window of constant size (in our case is 4000 samples) and move

it across the audio data with constant step size (in our case is half of window size – 2000 samples).

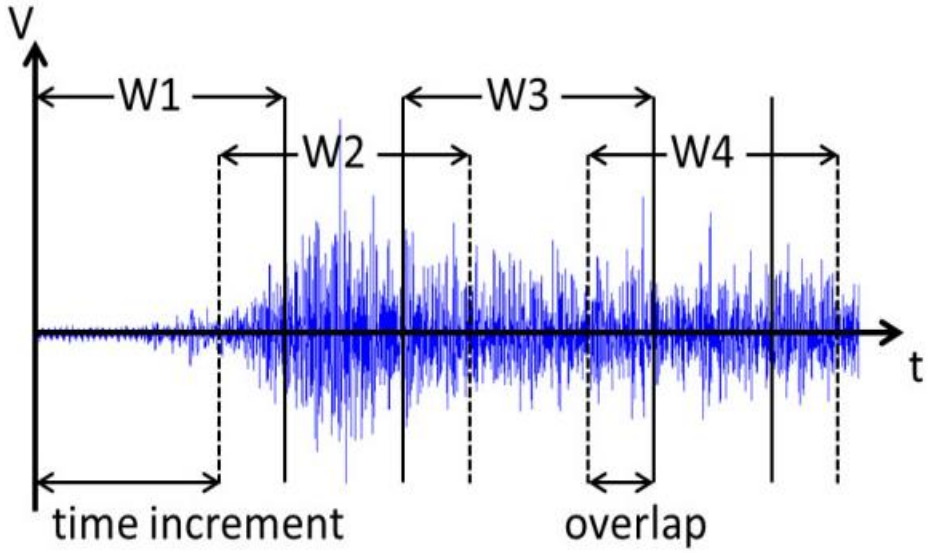

Figure 9. Overlapped windowing

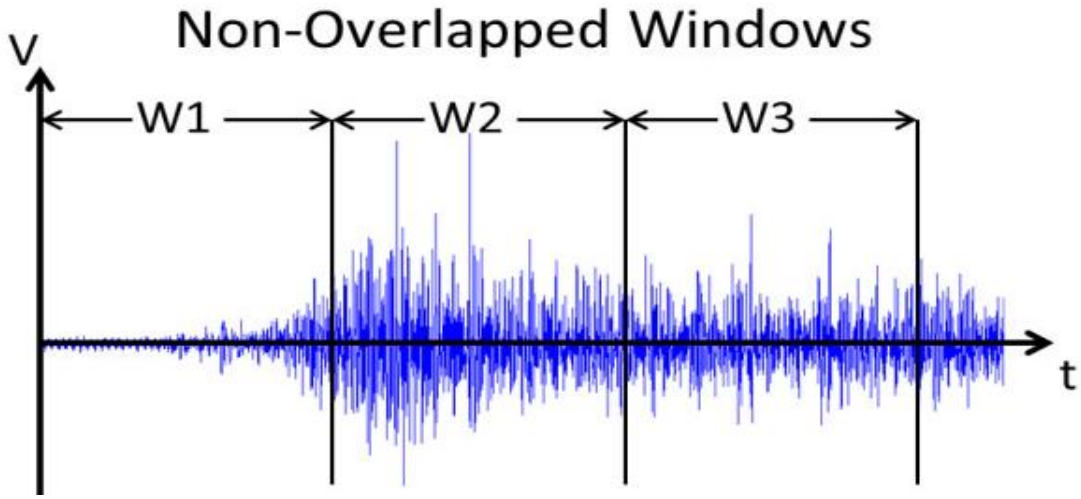

Figure 9. Non-overlapped windowing

Algorithm of audio data classification:

1) Find the maximal absolute value of the raw data in the audio file. From this value we calculate another value (25% of the maximal) that will be the minimal value for a data chunk to be classified. It means that if the maximal value of the data chunk is less than this value then the chunk is considered as weak signal and is ignored.

2) Allocate an array of the results. Each cell of the array concerns a specific class. Initialize this array with zeros. Each time that we classify a chunk ang get a result we increment the relevant cell value. Thus, we will know how many times each class was recognized.

3) Set the window at the beginning of the audio data.

4) Check if the maximal chunk value is larger than a minimal value that was calculated at the stage 1. If not then go to step 7.

5) Preprocess the data under the window. See *Data preparation* section.

6) Classify the preprocessed data. Update the results array according to the recognized class.

7) Move the window to next position. If the end of the data is reached then go to the step 8. Otherwise go to the step 4.

8) Scan the results array and find the maximal value. The class matches the index is selected as recognized.

Implementation

Project files:

settings.py – global settings of the project: variables and constants.

wav_files_processing.py – WAV files preparation: dividing audio file into several files, extracting sequences from the original file and writing them to new files, resampling.

samples_processing.py – processing samples loaded from audio files: resampling, padding, normalization, peaks envelopes creation, reading/writing samples from/to audio files.

peaks_envelope.py – peaks envelopes creation.

train.py – creation/training/testing of the model.

classification.py – classification of audio files.

main.py – main module of the project.

Training Results.html – results of the model training/testing. The model training was divided into several trainings. The results are presented in this file these are the results of the last training.

Test Classification Results.html – results of the test audio files classification.

Audio data:

In the directory “Data” are found audio files that are used for training/testing. Each directory contains audio files belong to the specific class: Drill – 32 files, Hammer – 23 files, Jackhammer – 35 files.

In the directory “Test” are found 5 audio files are used for testing.

Training:

For the training/validation are used the data are located in the directory “Data”. The training is performed several times. The final results are stored in the file *Training Results.html.* See Figure 10 for the results.

```
Test loss: 0.002416343195363879 / Test accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00         8

    accuracy                           1.00        18
   macro avg       1.00      1.00      1.00        18
weighted avg       1.00      1.00      1.00        18
```

Figure 10. Final training/validation results

Testing:

At this moment only 5 audio files were tested. All the files were recognized correct. See Table 1 for the summary.

The best results are shown for drill tool, good for jackhammer, satisfied for hammer.

File	Real Type	Prediction Accuracy		
		Hammer	Jackhammer	Drill
drill_1.wav	Drill	0.0 %	1.08 %	**98.91 %**
drill_2.wav	Drill	0.0 %	3.125 %	**96.875 %**
hammer.wav	Hammer	**52.17 %**	34.78 %	13.04 %
jackhammer_1.wav	Jackhammer	0.0 %	**74.50 %**	25.49 %
jackhammer_2.wav	Jackhammer	0.0 %	**81.81 %**	18.18 %

Table 1. Tests results

See the file *Test Classification Results.html* for details.

Prediction errors:

The prediction errors occur mostly in the chunks that contain:

- Weak signal.
- Noise between hammer hitting or other tools working.
- Start/end of hammer hitting or other tools working.
- Drill and jackhammer are similar. Sometimes the model is confused between them.

Possible improvements and additions:

- Extending audio files dataset, both training and testing. Increasing number of audio files.
- Adding digital noise filters to reduce noise.
- Adding spectral envelopes classification instead of/combined with time domain classification. It can improve accuracy, but can worsen the performance.
- In order to add extra tool for recognition it is enough to add a new directory with relevant audio files to the directory “Data” and in the file “settings.py” to the variable “CLASSES” to add the name of the new class. The new directory name must be the same as the class name. And train a model again.

Conclusion

The results that I received are enough good. But they are based on small dataset. In order to get more reliable results, the size of the dataset must be increased. Anyway, the proposed method may be used for tools sounds classification.

References

1. Sounds database - Zvukogram: https://zvukogram.com/
2. Sounds database: https://wav-mp3-ogg.net/
3. Conv2D layer: https://keras.io/api/layers/convolution_layers/convolution2d/
4. Lukasik, Ewa, Cong Yang, and Lukasz Kurzawski. "Temporal envelope for audio classification." *Audio Engineering Society Convention 140*. Audio Engineering Society, 2016.
5. Magal: https://magalsecurity.com/solutions/military-grade-vibration-sensor
6. Schwarz, Diemo. “Spectral envelopes in sound analysis and synthesis.” Diss. Universität Stuttgart, Fakultät Informatik, 1998.
7. Jähnel, Tobias, Tom Bäckström, and Benjamin Schubert. "Envelope modeling for speech and audio processing using distribution quantization." *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE, 2015.
8. Altaf, M. Umair Bin, and Biing-Hwang Juang. "Audio signal classification with temporal envelopes." *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011.
9. Liang, H., Sakari Lukkarinen, and Iiro Hartimo. "Heart sound segmentation algorithm based on heart sound envelogram." *Computers in Cardiology 1997*. IEEE, 1997.
10. Thalmayer, Angelika, et al. "A robust and real-time capable envelope-based algorithm for heart sound classification: Validation under different physiological conditions." *Sensors* 20.4 (2020): 972.