Classification of tools sounds using machine learning methods

Student: Rakhlevski Ilia (ID: 316932847)

Course: Machine Learning in Speech Processing Technologies.

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

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

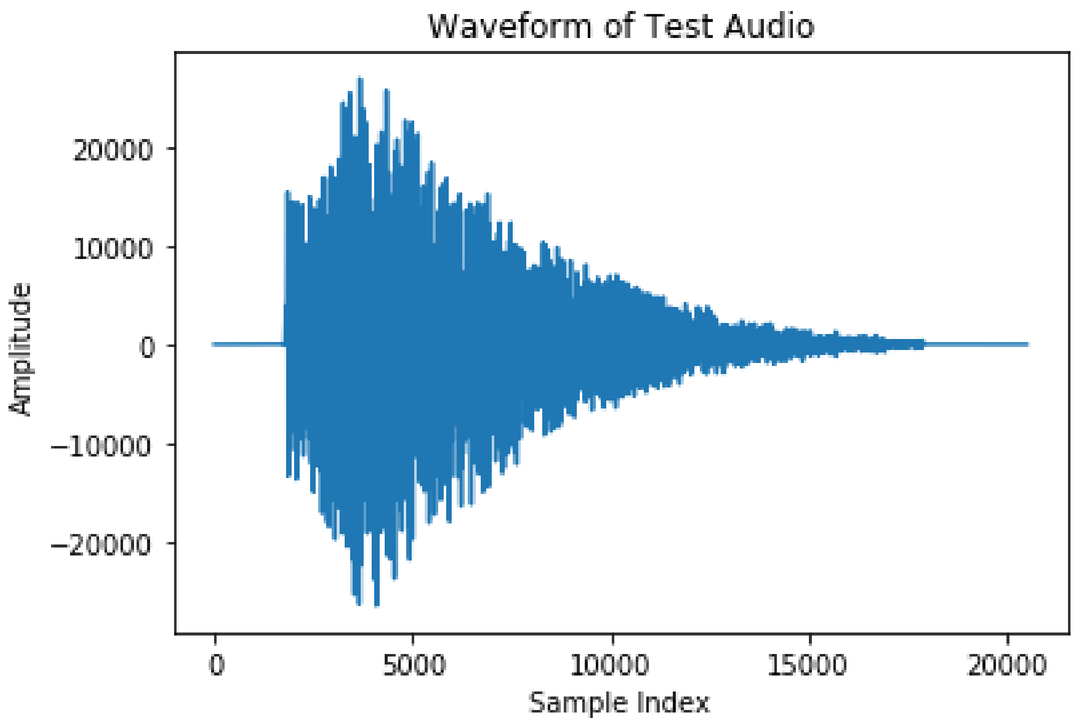
Data: WAV files containing sounds of tools [1], [2].

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.

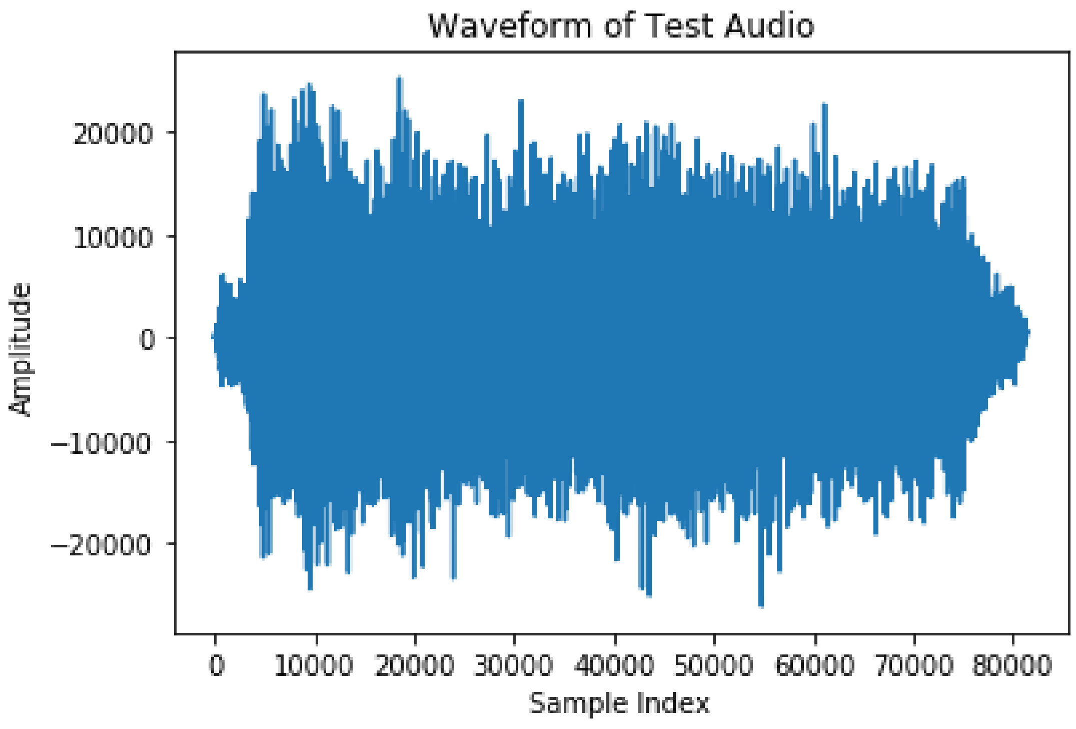
Tools

The tools that are used in the project:
Hammer, jackhammer, drill.

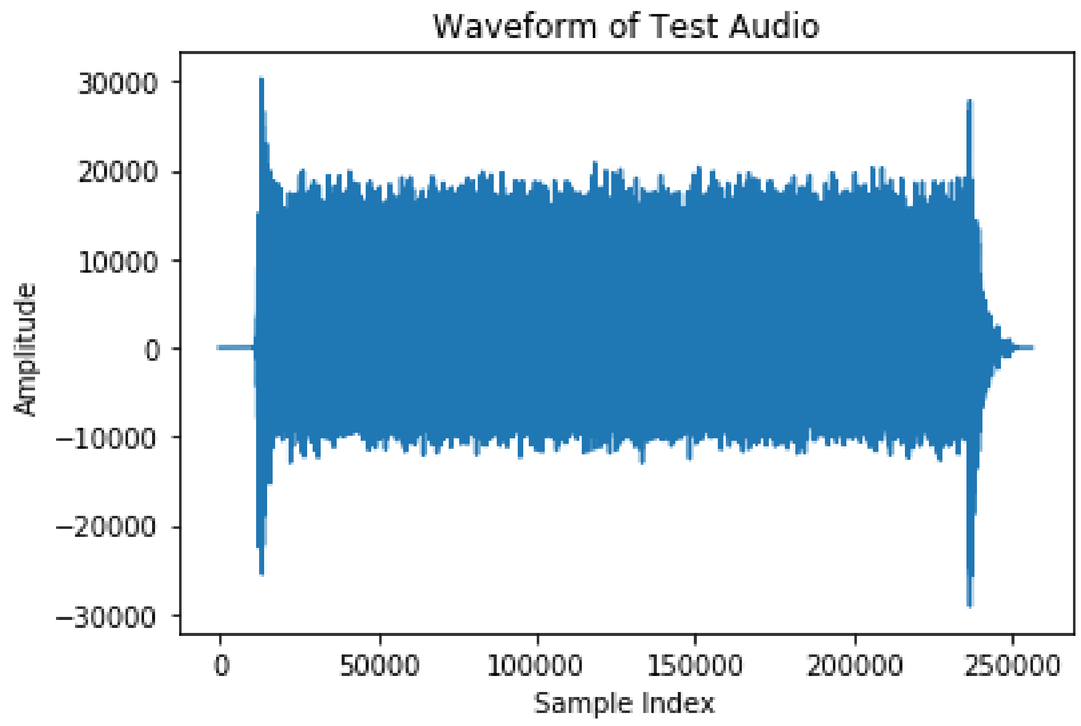
Hammer:



Jackhammer:



Drill:

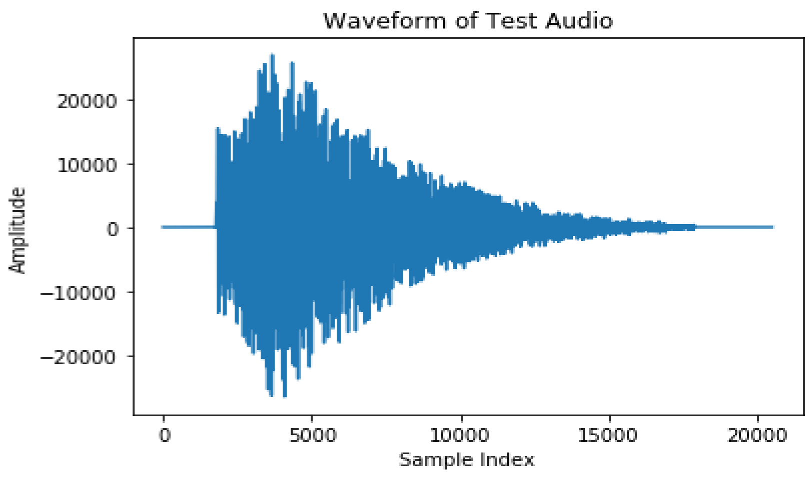


Method

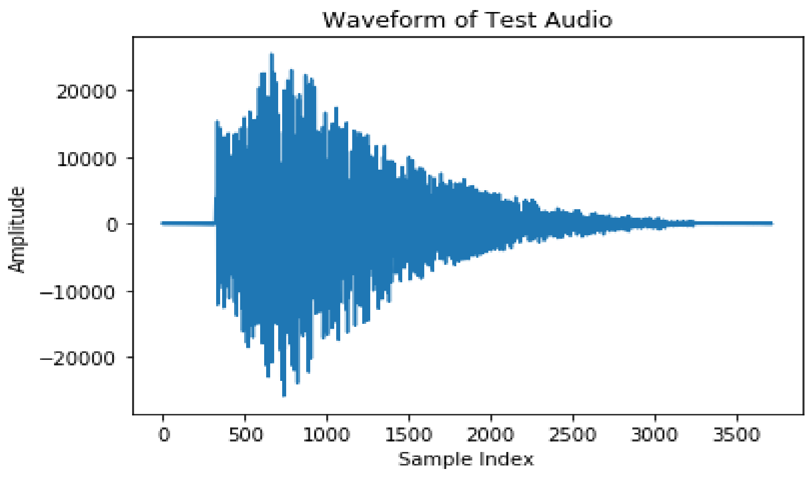
Data preparation

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.
   Example of an audio file resampling:

Original data:

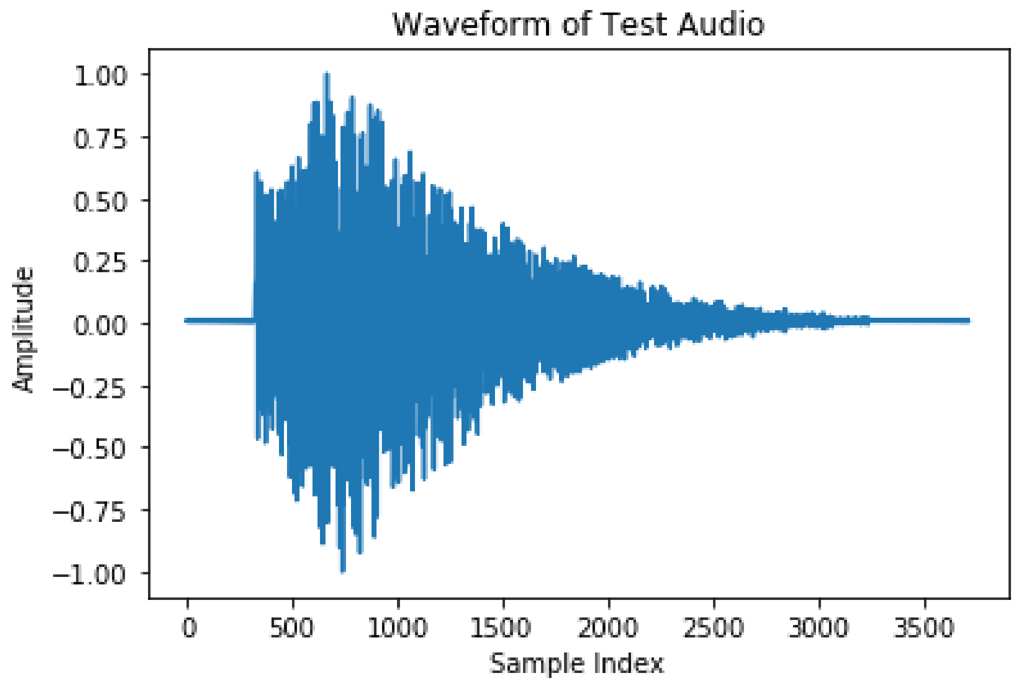


Resampled data:

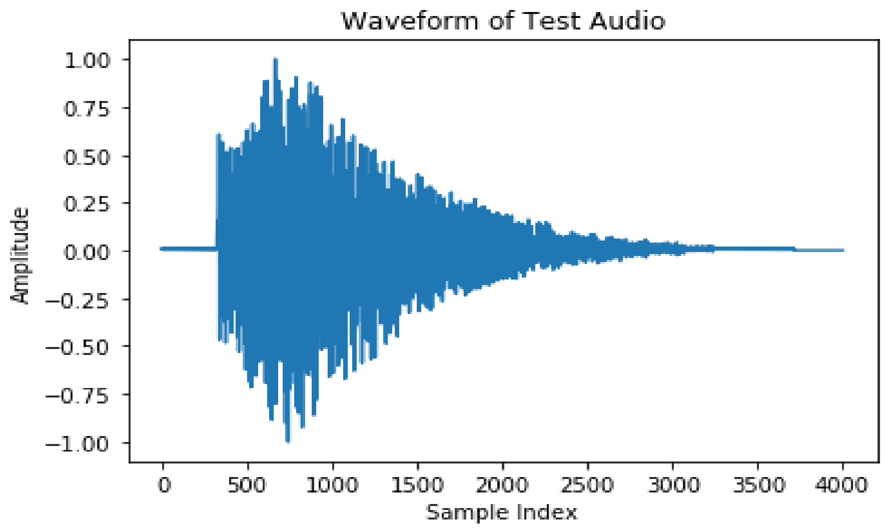


2) Normalization: normalize the data to range [-1, 1].
Audio data can have different amplitudes. In order to train or to classify it we need to unify it. So, we bring it to the range [-1,1].

Example:



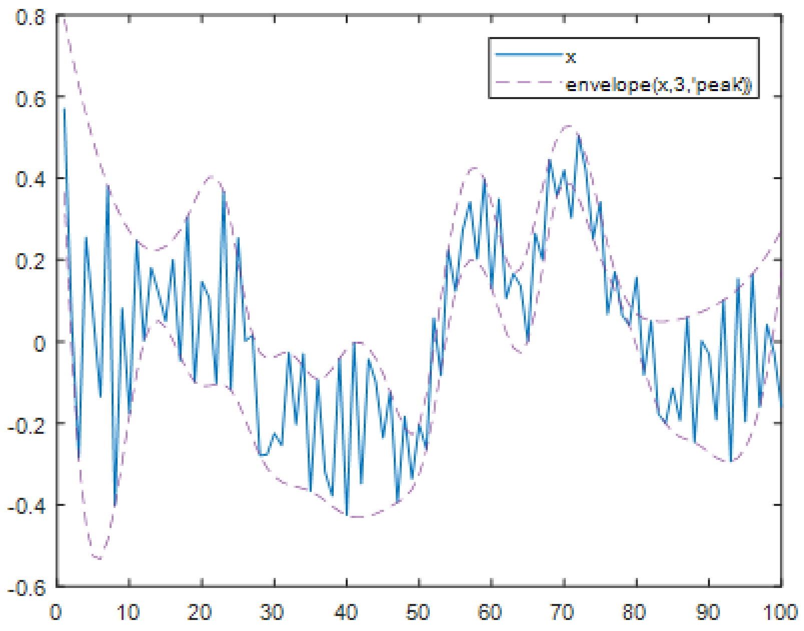
3) Fixing number of samples: adjust number of samples to the window size (4000 samples).
   If the current samples number is grater, then cut them. If it is less than the window size then zeros is added (padding).
   Example:



4) Data peaks envelopes:

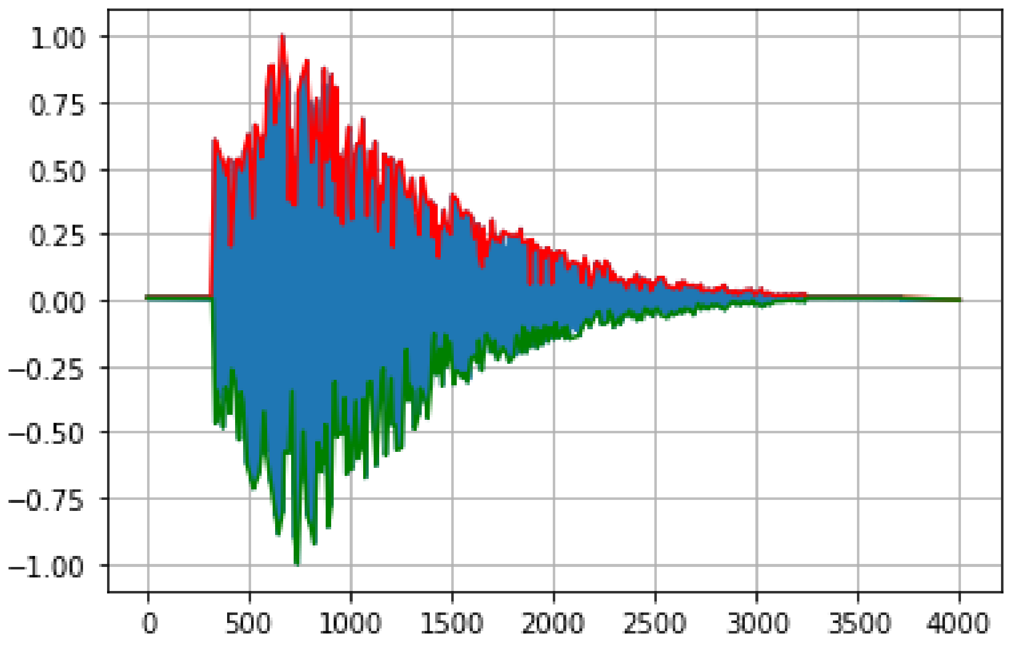
At this stage we calculate peaks envelope for the data from the previous stage.

Examples of peaks envelope:



This process can be repeated several times. It means to receive envelopes on previously created envelopes.
In this project I perform this process twice.

Audio data peaks envelope:



Audio data classification

For audio data classification is used a neural network is based on Conv2D layer [3].

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

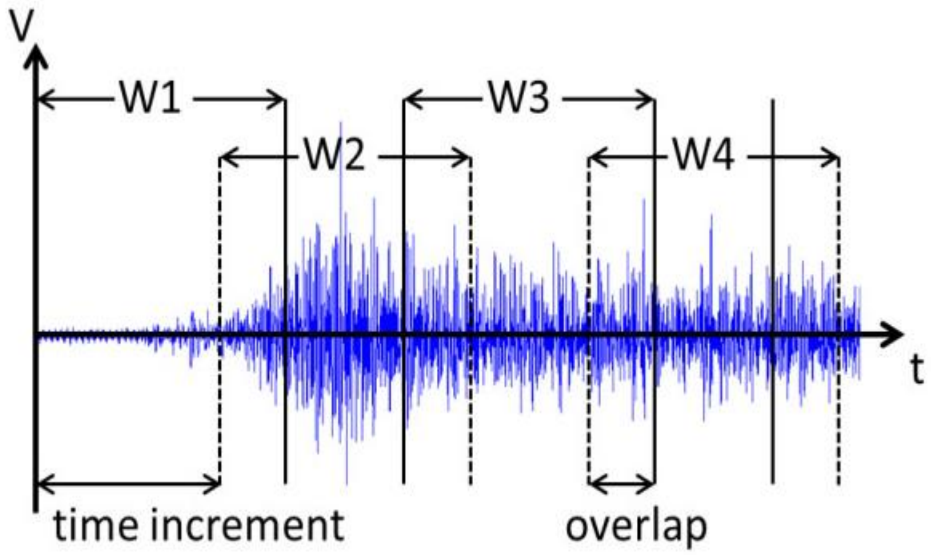
Output: tool class (hammer, drill or jackhammer).

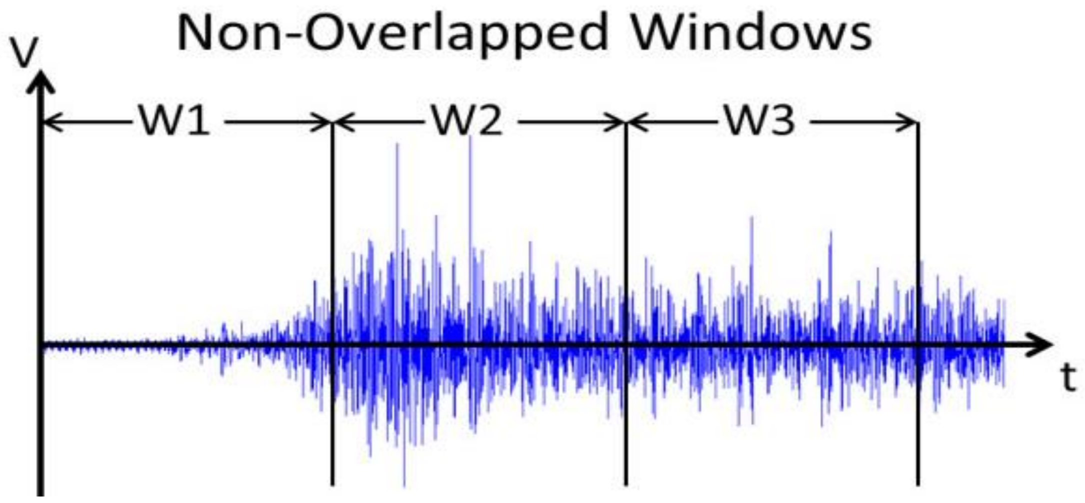
Audio files classification

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. 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).

Examples:





Algorithm:

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 6.

5) Classify the chunk under the window. Update the results array according to the recognized class.

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

7) 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.

<u>Audio data</u>:

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.

<u>Results</u>:

At this moment only 5 audio files were tested. All the files were recognized correct. See *Test Classification Results.html* for details. The best results are shown for drill tool, good for jackhammer, satisfied for hammer.

The 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.
- 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.

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/