Beat Detection Challenge 2022

Laurenz Hundgeburth - Team: NeuraBeats

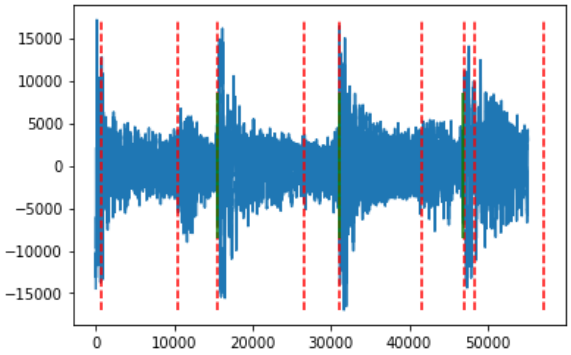
This report describes my final approach and highlights some experiments I did during this challenge. The code of my submission is available on Github (https://github.com/LaurenzBeck/music-processing-challenge), where I also documented the "technical" experiments I did to get to know some libraries in the `notebooks` directory.

First Plans

Adapting the model-based approach from the onset detection challenge would have been a very easy/lazy approach. For the beat detection challenge, I wanted to deviate from my comfort zone and try a creative rule-based approach.

From the lecture, I most vividly recalled the autocorrelation and pulse-train approach, but I also liked the agent-based approach.

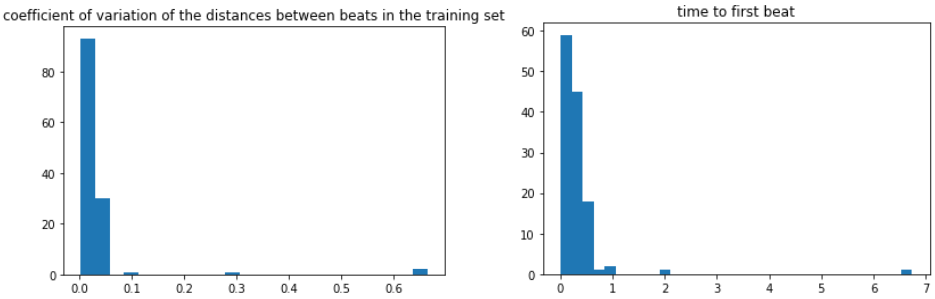
Because I wanted to play around with the periodic nature of beats, I decided to utilize my onset detector from the last challenge and distinguish between onsets and beats based on periodicity assumptions.



My first rough plan was to tackle tempo estimation and beat tracking simultaneously by sweeping over a range of bpm values and offsets and finding the combination with the strongest correlation to the offsets.

Assumption Verification

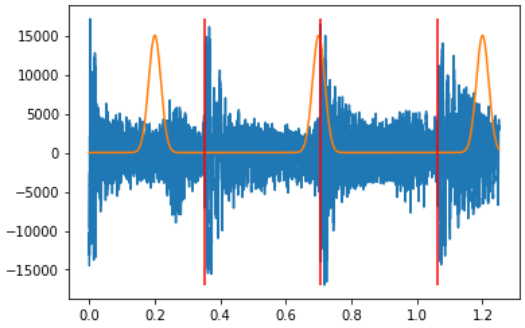
Even though I recall either the professor or the challenge description mentioning the rather constant tempo inside the training data, I wanted to be sure before I base my submission on these assumptions. A histogram of the coefficient of variation of the differences between the beats in the training set revealed a rather constant behaviour. I also plotted a histogram of the time it took until the first beat, which I needed for my planned approach.



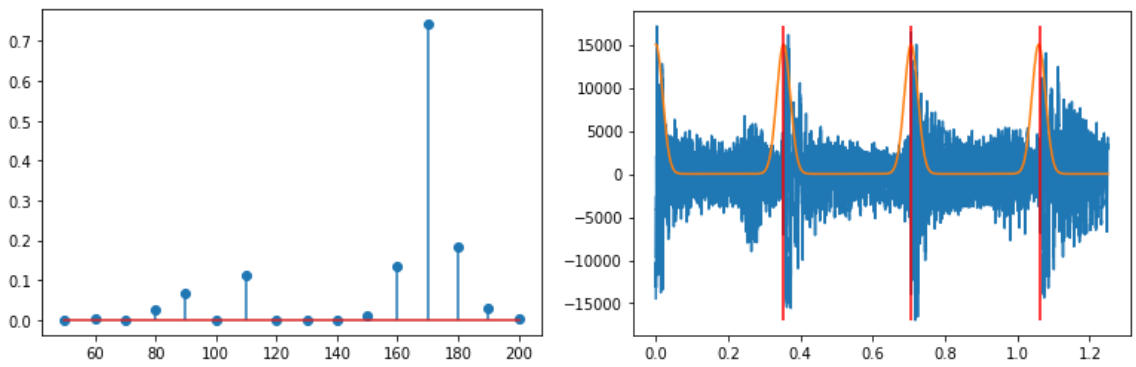
Measuring Periodicity via Gaussian Trains

When starting the project, I had the idea, that a signal with the correct BPM tempo should have a strong correlation with the onsets, given they contain a regular beat with the same bpm. I thought that the uncertainty and offset of my onset predictions might cause a major problem and utilizing gaussian pdf functions to model this uncertainty might be a promising direction. At that time I was feeling quite creative and willing to fail, so I wanted to follow my ideas instead of following a safer route.

After some prototyping, I had my own (horribly space inefficient) way of measuring the periodicity of my onset signal by sampling my signal of regularly spaced Gaussians (which I call gaussian train instead of pulse train).



For the following search of the best fit for a given bpm and offset combination, I divided the sampled signal by the number of possible beats (the Gaussian peaks) to avoid higher bpm values to have better scores. A first search using this periodicity score of the onsets using this gaussian train approach yielded promising results:



➔ The signal with a bpm of 170 (offset 0 for now) had the strongest response, and the gaussian peaks line up nicely with the beats. Notice

that in the above plots, only the beats are visualized, but the signal also contains onsets. So this method was able to find the correct bpm value even with the presence of other onsets.

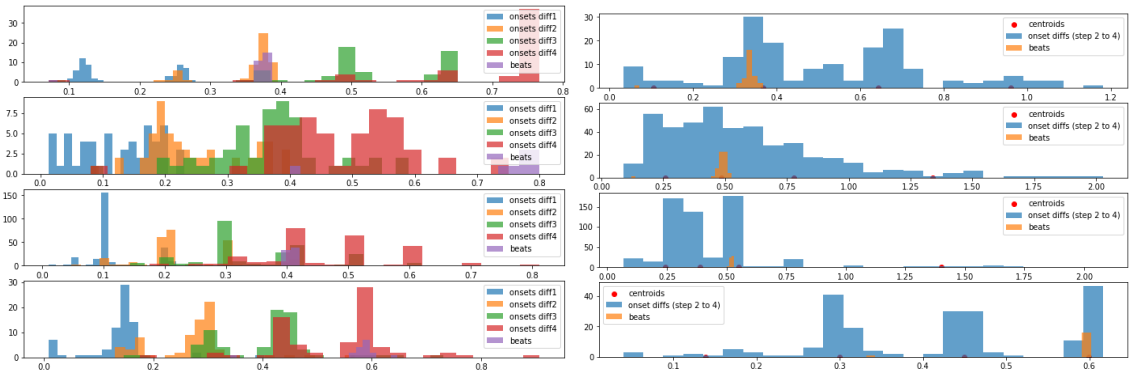
To my surprise, this search for the best bpm and offset fit ran way longer than my DL-based onset detector. I even ran into memory issues due to the poor implementation of my prototype.

I tried parallelizing the search with joblib's parallel pipeline processor and spent a long time finding a solution that would require minimal code changes. Sadly, my beloved alive_progress bar is not serializable and therefore not conveniently usable from multiple threads. I also tried using the numba JIT compiler, which should have been quite fitting for my loop and NumPy intensive workload, but the library didn't deliver on its promises and ultimately after a long error-intensive session, after which I only compiled pure NumPy functions, the performance gains were negligible.

Reduction of Search Space

After going through the script again, I got the idea, that I could speed up my search by smartly reducing the search space. The predictions would not get better, but at least the program would finish in a more feasible amount of time.

I decided to go for a histogram-based clustering approach to the inter-onset-intervals. After some experimentation I documented in the accompanying notebooks, I decided to calculate the Ioi's by calculating the difference of the onsets with step sizes of 2, 3 and 4. This assumes, that there are only 3 onsets between each beat, but a short validation showed that this assumption holds for every sample I checked.



➔ It's hard to see from these plots, but the histogram of the beat intervals lines up nicely with different Ioi's (different step sizes for different sound samples). On the right plot, we see the cluster centres as small red points on the x-axis. For every peak in the beat intervals, there is a cluster centre nearby, which indicates that this reduction of the search space should lead to reasonable results.

Final test results: **0.184** F1 score