# Testing Setup

All the implemented algorithms and functions were tested with all the given audio files. We did this because we believe that they cover a variety of different genres and challenges for each of the tasks. We didn’t synthesize additional Audio because we believe that with our current skill in synthesizing the synthesized files would be either significantly easier or practically impossible to process, additionally we hope that by not synthesizing extra files we could avoid “overfitting” of our Parameters.

All the parameters used were found either through (pseudo-) random search or carefully selected by hand.

# Onset detection Variation 1: Superflux

We chose Superflux as the first onset detection algorithm because it had the overall best performance according to the lecture slides. While implementing the algorithm we discovered that with the parameters that we were using for FFT the theory didn’t match what we were actually doing. This was primarily because subtracting spectra that overlap (FFTSIZE !=HOPSIZE) would not result in properly calculating the spectral flux. We tried to compensate that fact by assuming that the first half of the first spectrum would contain ¼ of the energy of the overall first spectrum. This assumption is based on the assumption that during the relatively small timeframe the composition of the spectrum doesn’t change, and that the energy contained would behave approximately linear. These assumptions seemed correct as pretty much every audio file has a fade in (to avoid unpleasant pops in the beginning). Using this assumption, we could calculate “overlap-free” spectra. Sadly, due to time limitations we were not able to validate the accuracy of these assumptions but since we knew another group would use standard superflux to compare to we decided to roll with it anyway.

We also tried multiple different peak picking methods but the one proposed in the lecture that was used with Superflux worked the best. One of the peak picking methods had a, what we thought, really nice detection function for tempo estimation but it turned out that it didn’t work.

After optimising the parameters with random search our algorithm performed better then presented in the slides and we achieved an fmeasure of 89.59 with a recall of 87.18% and 92.12% precision.

# Onset detection 2: High Frequency Content + Weighted Phase Deviation

For the second onset detection, we started with the high frequency content (HFC) algorithm, because in the table on the lecture slides, it has very good fmeasure values for most of the cases. Because of several reasons, the HFC algorithm could not find a lot of onsets and therefore it was way too bad.

Because of that, we decided to mix it with the weighted phase deviation (WPD) algorithm, which should perform better in that cases, where HFC is bad (according to the table in the lecture slides).

The first problem which appeared by combining them is, how to combine them? First, we tried to add the different onset detection functions together, and perform peak picking on the combined function. We also tried to weight the summation with different weights. For optimizing all the parameters (not only the weight, also things like thresholds, window sizes, …) we applied an evolutionary Monte Carlo search.

For the second way of combining the onset detection functions, we performed the peak picking on each onset detection function separately, and combined the resulting onsets by removing double onsets (which means two onsets, which are very near to each other, given by a window). We also applied the same Monte Carlo search for optimizing all parameters and found out, that this way of combining the two algorithms does a much better job.

After optimizing each of the two combination techniques, the second one had nearly 10% more fmeasure, and the recall and precision values where much more together, while with the first technique, one of them was always very good, while the other one was very bad.

For the peak picking itself, we applied an algorithm, which finds all local maxima which could be an onset, and then checks, if this maximum is higher than the median in the current window and if there is no other local maximum higher than the current one. All of these values where parameters of our search.

After nights of searching for the best parameters and periodically optimizing the search space we end up with an fmeasure of approx. 77.16% over all the given training data.

# Beat detection: Autocorrelation

For the beat detection, we applied the autocorrelation as described in the lecture slides on the onset detection function of the first onset detection (superflux), because it delivers way better results than the second one.

The result of the autocorrelation has some peaks, where one is the right one which describes, how many time between two beats is. If we know this time and the first beat, we can always estimate, where the next beat should be, and then we look in a small window around this time point for an onset, which is chosen as beat. If there is no onset in range, we assume, that there is a beat on the estimated time point and move on, because sometimes, there is no tone (and therefore no onset) on a beat.

The first problem we got with this technique was the peak picking on the autocorrelation result. Because in a lot of the cases, the highest peak in this graph describes the tempo and not the time between two beats. So, we optimized our peak picking in a way, that it collects all local maxima, takes the higher ones of this list (here we use the arithmetic mean as a threshold) and then we choose that one with the lowest time (and therefore highest tempo). This gave us a lot of very good results.

The second problem we figured out was, that the first onset is not always the first beat and so we estimate nearly every beat wrong in such a case. To fix this problem, we create a pulse train with the periodicity of the estimated beat time and a pulse length of 3 frames in the audio signal, cross correlate this signal with the first 1.7 seconds of our audio signal and get a graph, where the highest peak is very often the time offset for the first beat. Of course, all these parameters where found by searching.

At the end, we came up with a 74.49% fmeasure over all given training sets for beat detection. The biggest problem we have at the end is, that sometimes the estimated tempo between to beats is the half or double of the real time.

# Tempo estimation: Autocorrelation

For Tempo estimation, we used autocorrelation just as with Beat detection. Even though it was developed completely separately. The main problem with tempo estimation is not finding a suitable tempo but finding the strongest perceived tempo. Initially we assumed that the highest peak in the autocorrelation function would represent that tempo. Sadly, this was only the case in about one fifth of the files. After looking at the autocorrelation function we found out that the problem is that the tatum often was valued higher than the tactus, so we tried to include periodicity in the autocorrelation function. Initially we tried this with the help of a cosine. but this didn’t work out as planned and the resulting BPM tended to be way to low. Alternatively, we used a pulse train. This resulted in better solutions but still not quite good enough. Dampening higher pulses made the solution even better. Strangely enough the most common error then was that the BPM was a third of what it should be. We corrected this with a check if the 2nd and 3rd multiple of the most prevalent beat had a relation that was not what it should have been. This measure helped us achieve a 75% fmeasure on the test files. Although we are not sure how well this would hold up in real tests.