Report Audio and Music Processing

Tobias Halmdienst, Jakob Seibezeder, Theresa König

June 2025

Onset Detection

We first tried Spectral Flux approach and got some decent results, but then focused on a machine learning approach.

We used the default and extra data available, a datasplit of 80% training and 20% for validation. We converted the labeled onset times into a binary vector matching the length of the audio signal (in frames). This vector contains 1s at onset frames and 0s everywhere else. Then we defined a Convolutional Neural Network with the architecture shown in table 1.

Note, we choose the padding and kernel sizes such, that the length of the signal stays the same, while the height of the audio signal is reduced. We end up with an output of size 1xN, where N is the original signal length. This is also our detection function output. As for the Loss function for training, we chose the Binary Cross Entropy loss with Logits Loss. Since the labeled data vector consisting on 0s and 1s is unbalanced, because there are more frames without an onset than with an onset, we gave the loss a positive weight of 24(We calculated the mean ration of all labels). A possible output of the detection function looked the following way. The blue line is the output of the model with the max(0, x) operator applied and the red lines are the true onsets.

Layer	**Parameters**
Conv2d	$1 \to 32$, $k = (3, 5)$, pad="same"
BatchNorm2d	32
GELU	–
MaxPool2d	$k = (2, 1)$
Conv2d	$32 \to 128$, $k = (3, 5)$, pad="same"
BatchNorm2d	128
GELU	–
MaxPool2d	$k = (4, 1)$
Conv2d	$128 \to 256$, $k = (3, 5)$, pad="same"
BatchNorm2d	256
GELU	–
MaxPool2d	$k = (2, 1)$
Conv2d	$256 \to 1$, $k = (5, 1)$, pad=$(0, 0)$

Figure 1: Architecture

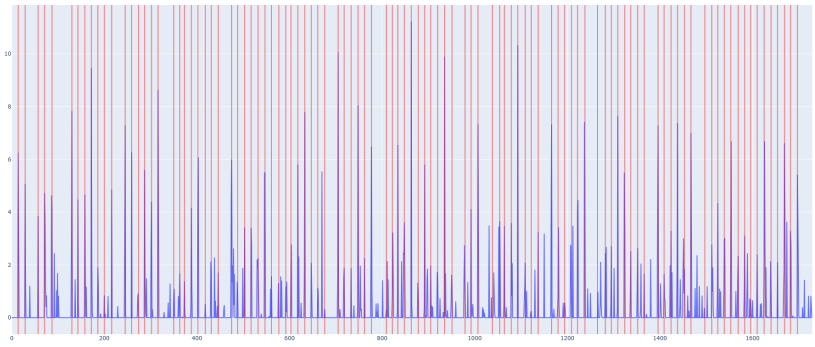

Figure 2: Onset detection function

Since the detection function is already good, the next challenge was the peak-picking function. We tried several approaches: constant threshold, constant threshold + smoothing and adaptive threshold + smoothing. We got the best result with constant threshold + smoothing with the threshold 0.4 combined with a moving-average smoother (window size = 3).

The final predictions look like this, where the blue line is the output of the model, the red lines are the true onsets and the green lines are the predicted onsets.

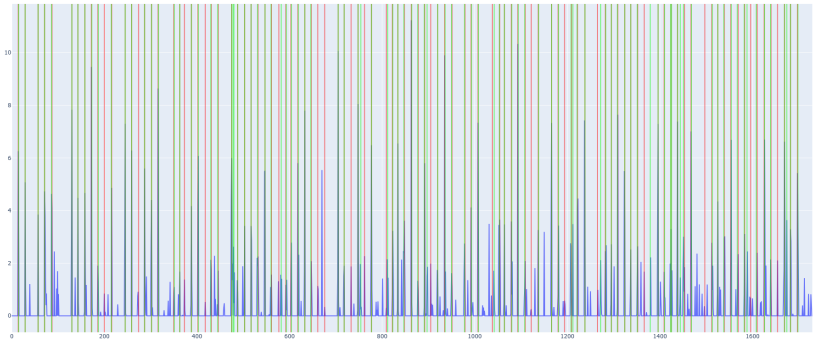

Figure 3: Predicterd onsets

Tempo Detection

Our first intuition for the Tempo Estimation was to estimate tempo from mel-spectrograms by computing the onset envelope, applying smoothing, and using autocorrelation to identify peaks. We used peak picking to identify candidates and limited them to a certain BPM range. Sadly this approach did not perform too well so we adapted.

In our second approach we calculated the autocorrelation on the learned onset detection function. Then we simply selected the lag corresponding to the maximum autocorrelation value and converted it to Tempo. This ultimately yielded good results.

Beat Detection

Multiple agents

We tried two different approaches for beat detection. Our first approach was to try the Beat Tracking Algorithm by Simon Dixon [1] that is also referenced in the lecture slides.

The greatest problem we faced was finding a good way to determine the goodness of an agent. Some of our ideas included using the variance of interbeat intervals, or the overlap between predicted beats and onsets.

Unfortunately, this approach did not yield satisfying results. With these experiments, the best result we got was an f-score below 0.5. This result could have been improved with better tempo estimations or a better salience function.

CNN model

In our final submission we used the exact same CNN model architecture that we used for onset detection and train it on beat data instead. The idea was that most beats are more pronounced onsets. We used a balanced loss function where we set the weight for positive labels to 35, which represents the ratio of 0s to 1s. Training the CNN on the given data (incl. extra data) and without any post-processing, we already got an f-score of 0.68.

We tried to improve this result further with different post-processing methods, such as another multiple agent function that accounts for the periodicity of the beats. Although, since we were struggling to find the correct tempo, this did not improve our results.

Code

All our implementations can be found in github.com/pandahd03/challenge.

References

[1] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.