Turbo convolutron 2000

Alexandre Lacoste

August 31, 2007

1 Introduction

MIREX provides us with a framework to evaluate our music information retrieval (MIR) algorithms. It does so by running them on a previously unknown dataset and comparing the performance of different algorithms. There are many different tasks available : Music Classification, Music Similarity, Music Transcription ... In 2005, We have participated to the Onset Detection task and our algorithms [1] has been classified as first and second position for the accuracy of detection. However, it was classified as last and second last for the speed of detection. In real life applications, such a computational power requirement is unacceptable. Therefore, we did some modifications to the algorithm to greatly enhance the speed.

This paper describe the new algorithm. First we give an overview of the algorithm. Next, we explain some details of the convolutional neural network (CNN) [2]. We also describe how we extract the the onset from the output of the CNN. Finally, we have a little discussion about the speed of the algorithm.

2 Algorithm Description

The biggest time bottleneck of the old algorithm was matlab. We therefore have translated our algorithm into the c programming language. The second time consuming element was the spectrogram. We were applying our neural network on the full Short Time Fourier Transform (STFT) spectrogram. Now, we first transform the spectrogram to the mel scale, keeping only 20 frequencies. One other important modification we made is to use a CNN instead of a Feedforward Neural Network (FNN). This also account for an important speed improvement and also a significant performance gain.

The algorithm is simply a sequence of functions that progressively transform the original signal into a curve that makes the onset as obvious as possible for the peek extractor. It goes as follow :

1. Resample the signal to the appropriate sampling rate. We chose 22050 Hz.
2. Build a spectro-temporal representation of the signal
   (a) Extract the STFT of the signal. We use a hamming window of 768 samples and an overlapping of 621 samples
   (b) Keep only the magnitude part of the spectrogram

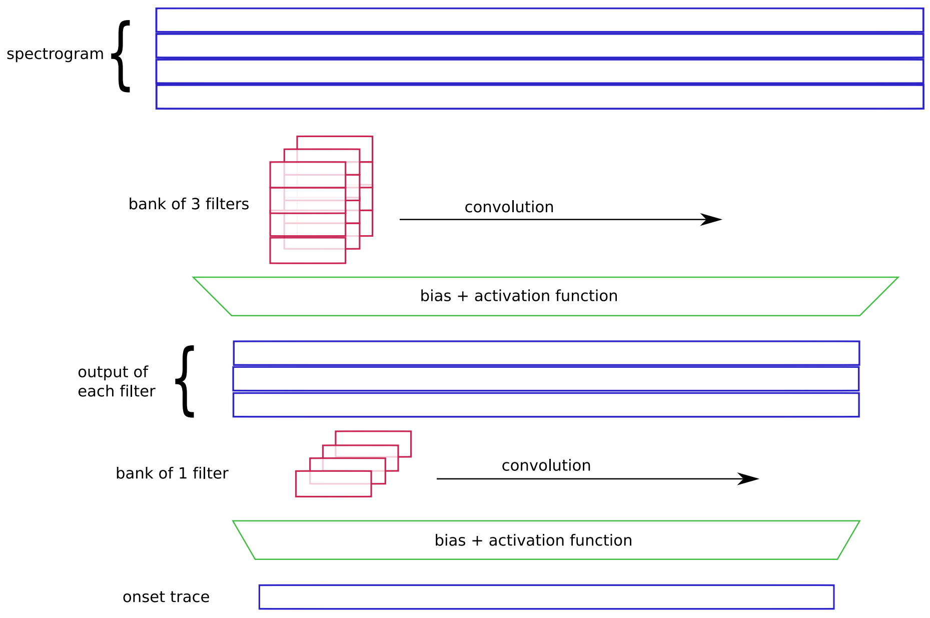

Figure 1: This figure shows a 1 dimensional version of the CNN.

(c) Transform into the mel scale. We keep only 20 frequencies.

(d) Take the logarithm of the magnitude. Here we add a small value of 0.01 before taking the magnitude. This avoid $log(0)$ and prevent from getting too much of the background noise.

3. Apply the convolutional neural network. See section 3 for more details

4. Extract the location peaks

3 Learning the filters

A CNN is simply a sequence of layers where each layer correspond to a Bank of filters that is convolved over it input. For each layer, we also have a bias and a non linear activation function like in conventional FNN. Figure 1 shows a graphical representation of the CNN. For the purpose of our algorithm, we don't apply any translation in the frequency axis and there is no subsampling. Thus, it resemble more to a Time Delayed Neural Network (TDNN) but as we first tried it as full CNN, we still call it like that.

The filters are learned using the Conjugate Gradient algorithm by minimizing the Root Mean Square Error (RMSE) between the output of the CNN and a target trace. The target trace is built by calculating a thin Gaussian around the location of each onset. Figure 2 shows the representation of the onset trace and the corresponding output of the CNN.

For our algorithm, we have used only 2 layers. The first layer corresponding to 6 filters of size 20 x 8 where 20 corresponds to the number of mel frequencies and 8 is the width of each filter. The second layer only need one filter as there

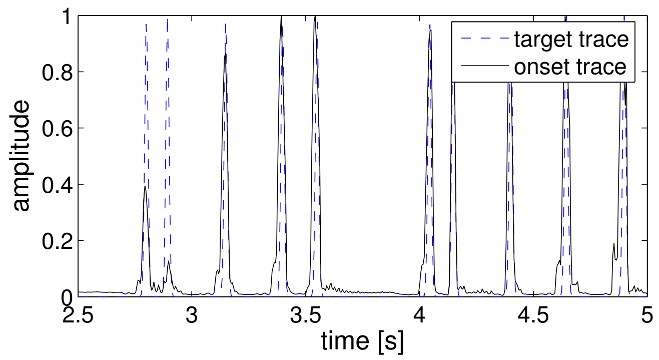

Figure 2: The blue dotted curve corresponds to the target trace, where a thin Gaussian lye at the onset location. The black solid curve represents the output of the CNN after the learning.

is only one output. The size of it is 6 x 30. The height of 6 corresponds to the number of filter in the previous layer. The first activation function was the tanh but for the last layer, we used a logical sigmoid, to have a value between 0 and 1, meaning background or onset respectively.

To learn the filters, we used a dataset composed of 60 hand labelled segments of 10 seconds. This list of songs is a subset of the ballroom dataset which was used during ISMIR 2004 contests. It was hand labelled using a Matlab GUI, providing different visual and auditory hints about onsets. It is also available here `http://www.iro.umontreal.ca/~lacostea/index.php?page=OnsetDetection.html`

4 Peak extraction

For extracting the peaks, we simply used a threshold. When the onset trace is positively crossing the threshold, this correspond to the start of an onset. When it is negatively crossing it, it corresponds to the end of the onset. To calculate the location of the onset, we simply took the center of mass of the peak.

The value of the threshold is also learned. It was selected to maximize the performance on our training dataset. This was done by calculating the performance for many different value of the threshold and selecting the most appropriate.

5 Speed Concern

Our c implementation and mel scale transform greatly improve the speed but the CNN is also held responsible for speed improvement. To determine the presence of an onset, we need to look approximately 100ms before and after it position. The old algorithm, using a FNN, had to deal with this time window with it first layer, yielding huge weight matrix and almost nothing to do for subsequent layers. In the case of the CNN, the work can be shared amongst the different layers, by integrating over $50ms$ for the first layer and over $200ms$ for the second layer. The second layer, having less data, makes it faster to integrate over the $200ms$. Also, we managed to break the convolutions into matrix multiplications, calculated by the gemm function of ATLAS, which uses SIMD instructions for high speed multiplications.

Those modifications makes the CNN step around 10 time faster than the time required to calculate the STFT. For this step, we use the fftw3 library (Fastest Fourier Transform in the West). Using the flag `FFTW_MEASURE` instead of `FFTW_ESTIMATE` would certainly help a bit. However, it is also possible to reduce the window width, the sampling rate and the overlap ratio to enhance the speed at the expense of a small reduction of the accuracy. Another approach could be to implement the spectrogram using complex wavelet transforms. Discrete Wavelets can transform a signal in $O(n)$ instead of $O(n \log k)$ where n is the length of the signal and k, the length of the window.

Finally, on a P4 2.4GHz, we are able to analyze 1 second of audio in approximately $10ms$. This makes a speed improvement factor of around 50 times compared to the old algorithm.

References

[1] A. Lacoste and D. Eck. A supervised classification algorithm for note onset detection. *EURASIP Journal on Applied Signal Processing*, 2006.

[2] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, Mass., 1995.