JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
**Sebastian Böck**

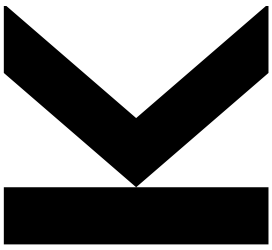
Submitted at
**Department of Computational Perception**

Supervisor and
First Examiner
**Gerhard Widmer**

Second Examiner
**Geoffroy Peeters**

November 2016

# Event Detection in Musical Audio:
## Beyond simple feature design



**Doctoral Thesis**

to obtain the academic degree of

**Doktor der technischen Wissenschaften**

in the Doctoral Program

**Technische Wissenschaften**

**JOHANNES KEPLER**
**UNIVERSITY LINZ**
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

## STATUTORY DECLARATION

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references. This printed thesis is identical with the electronic version submitted.

# ABSTRACT

This thesis is about the automatic detection and classification of sound events (e.g. notes or percussive sounds) in musical audio. It deals with four different sub-aspects, namely (i) the detection of the timing of these events (onset detection), (ii) their position inside the metrical grid (beat and downbeat tracking), (iii) the estimation of the dominant periodicity (tempo estimation), as well as (iv) identifying the frequency of the played notes (note onset transcription).

Historically, beat tracking, tempo estimation, and note transcription systems were built upon onset detection algorithms. Most of them incorporated hand-crafted features, designed specifically for the given task, certain sounds or music styles. Unlike previous approaches, we avoid hand-crafted features almost entirely, but rather learn them directly from audio. We present several algorithms addressing the before mentioned tasks to detect and classify the sound events. All proposed methods perform state-of-the-art in their respective field over a wide range of sounds and music styles, and show the superiority of learned features both in regard to overall performance as well as generalisation capabilities.

Reference implementations of the algorithms developed in this thesis are released as an open-source audio processing and music information retrieval (MIR) library written in Python. Additionally, we make the data used to develop and train the algorithms publicly available, stimulating further research and development in this area.

# ZUSAMMENFASSUNG

Ziel dieser Dissertation ist die automatische Erkennung und Klassifizierung von musikalischen Klangereignissen (z.B. Noten oder perkussiven Klängen). Dabei werden vier Aspekte genauer beleuchtet: (i) die genaue Erkennung der Anfangspositionen dieser Ereignisse, (ii) ob das Ereignis auf einem Beat (Taktschlag) innerhalb des Metrums oder – im Speziellen – auf dem ersten Schlag liegt, (iii) das daraus abzuleitende Tempo, und (iv) – mit Einschränkungen – die Tonhöhe des Klanges.

Frühere System arbeiteten vor allem hierarchisch, d.h. zuerst werden die Anfangspositionen aller Klänge ermittelt und darauf basierend dann entschieden, ob es sich z.B. um Taktschläge handelt, oder mit welcher Tonhöhe ein Klang erklingt. Die meisten Systeme extrahieren dabei aus dem Signal von Hand definierte Merkmale, die oft nur für eine bestimme Musikrichtung oder spezielle Anwendung entworfen wurden und daher in ihrem Anwendungsbereich limitiert sind. Im Gegensatz dazu wird in den hier vorgestellten Algorithmen diese explizite Modellierung der Merkmale weitestgehend vermieden, indem stattdessen mittels maschinellen Lernalgorithmen relevante Merkmale direkt aus dem Musiksignal erlernt werden. Die hier vorgestellten Algorithmen sind in ihrem jeweiligen Bereich führend – unabhängig vom Musikstil.

Zu sämtlichen Algorithmen werden Referenzimplementierungen in Form von freier Software zur Verfügung gestellt, gebündelt in einer einfach zu installierenden und erweiterbaren Softwarebibliothek. Darüber hinaus werden außerdem alle Daten die in dieser Dissertation verwendet wurden frei zugänglich zur Verfügung gestellt.

# ACKNOWLEDGMENTS

First of all, I would like to thank my family for everything. Thanks to my supervisor Gerhard Widmer for making this thesis possible and my second examiner Geoffroy Peeters for reviewing it. Big thanks to all my co-authors and colleagues at the Department of Computational Perception in Linz and at the OFAI in Vienna. It is a great pleasure working with you. Peter Knees deserves explicit mention for acquiring the *GiantSteps* project, allowing me to work on music I also like to listen to. Furthermore, I would like to thank all authors who shared their source code, datasets, annotations, and algorithm outputs or made it publicly available, and last but not least all the anonymous reviewers who helped to improve the papers before being published.

This research was supported by the Austrian Science Funds (FWF) through the P22856-N23 project and the European Union Seventh Framework Programme FP7 / 2007-2013 through the *Phenicx* and *GiantSteps* projects (grant agreements no. 601166 and 610591, respectively).

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

Music information retrieval (MIR) is the research field aiming at extracting musically meaningful features from music – ideally enabling computers to 'listen to' and 'understand' music like humans do. In order to achieve this goal, MIR encompasses many fields such as acoustics, musicology, electrical engineering, computer science, and – recently more and more – artificial intelligence. The features extracted range from low-level descriptors (e.g. the energy of a signal or spectral description of it), to musically more descriptive mid-level representations (e.g. note onsets or beat positions), to high-level characteristics such as the key or genre of a musical piece.

This thesis puts the focus on mid-level features – more specifically, on onsets, beats, downbeats, and tempo – and how they can be derived automatically from musical audio. The process of extracting these features can be formulated as the following MIR tasks:

ONSET DETECTION

Onset detection aims to identify the starting points of all musical events in an audio signal, the onsets. Based on their characteristics, onsets can be broadly categorised into three groups: pitched (e.g. hummed voice, bowed strings, or wind instruments), percussive (e.g. drums or hand claps), and pitched-percussive onsets (e.g. strung instruments such as piano or guitar). The difficulty of this task lies in the fact that these characteristics differ a lot and music comprises all types of onsets which can occur simultaneously, at different volume levels, or in various modified way. Some of the most common modifications are effects, which can be either of artistic nature (e.g. vibrato or tremolo) or generated synthetically by digital audio effects. These effects do not constitute onsets, but are often falsely identified as such. Onset detection is often used as a first processing step towards note transcription, beat tracking, and tempo estimation. These methods can either work on a series of discrete onsets or on a continuous onset detection function.

BEAT AND DOWNBEAT TRACKING

The metrical structure of a musical piece is generally organised in a hierarchical way, with beats defining the most salient level, and downbeats the first beat inside a bar. Every bar consists of a certain number of beats, defined by the meter or time signature. Beats can be captured intuitively by humans. Most people are able to clap their hands or nod their head to the beats.

Downbeats, however, usually require some musical training to be identified correctly. In western music, the downbeats often coincide with harmonic changes, whereas in non-western music the start of a bar is often defined by the boundaries of rhythmic patterns.

TEMPO ESTIMATION

This task deals with the problem of how to accurately determine the tempo of a musical piece. The tempo corresponds to the frequency of the beats. Although the tempo can theoretically be derived directly from the beats, this approach is not as straight forward as it seems, since good beat tracking algorithms in turn rely on a good tempo hypothesis. Thus, a common strategy is to first estimate the periodicities based on the signal's envelope or a continuous onset detection function, often split into multiple frequency bands.

NOTE TRANSCRIPTION

The 'ultimate' goal of this task is to obtain a complete transcription (i.e. score) of a musical piece, however, currently this is infeasible. Research therefore concentrates on smaller subtasks, such as identifying the instruments or notes being played, often limited to monophonic signals or music played by a solo instrument. Still, these steps lead to solving real world problems, such as score following or the discovery of repeated patterns. Within the scope of this thesis we focus on obtaining a transcription of the note onsets for piano music renditions. This requires the simultaneous detection of the onset timings and pitch information.

In the past, most algorithms addressed these tasks using hand-crafted features exploiting special characteristics often present only in certain music styles. To lift this limitation, we avoid hand-crafted features almost entirely, but rather learn the relevant features directly from spectral representations of the audio signals. As training material, we use large datasets comprising very diverse music, leading to algorithms that generalise well on all kinds of music.

The algorithms proposed are not only relevant to the scientific community, but also have practical real-world applications. These include audio software (e.g. digital audio workstations), which can analyse the music automatically to slice it into loop-able parts, align the beats to a fixed grid by dynamically time-stretching it, or sync audio effects to the tempo of the piece. Other common applications are DJ-mixing software and hardware products. These tools aid the DJ in analysing the tempo of the piece or even aligning two musical pieces to be beat- and – more importantly – downbeat-synchronous. Within the *GiantSteps* project,[1] which provides the frame of this thesis,

1 http://www.giantsteps-project.eu

we investigated how users of such tools could benefit from MIR algorithms in creating music or performing live [81]. The demand for this technology is reflected in one typical quote:

> *"Onset detection, beat detection, tempo detection and harmony detection is pretty much all we need."*

To provide these, reference implementations of all methods developed in this thesis are bundled in an open-source software library that can be used by developers, music application hackers, and musicians with a background in music programming to support them in their specific tasks.

## 1.1 OUTLINE

This thesis is a dissertation by publication, i.e. Chapters 2 to 10 have the same content as the published peer-reviewed papers. Minor errors such as typos have been corrected without mentioning. If the paper contained errors, these were addressed accordingly and mentioned explicitly as ERRATA in the preface of the respective chapter. All publications have been reformatted to get a uniform style throughout the thesis, which might imply a rearrangement of table columns and splitting of large tables into multiple smaller ones. All references have been renumbered and collected in the shared bibliography at the end of the thesis.

The individual chapters are grouped into parts, each defined by the task identified in the previous section. Each chapter is succeeded by an ADDENDUM, which outlines further developments of the algorithms and how it was adopted and subsequently used by the scientific community.

## 1.2 CONTRIBUTIONS

The main contributions of this thesis can be summarised as follows:

### DEFINE STATE OF THE ART IN THE RESPECTIVE FIELD

The systems and algorithms proposed in this thesis define the state of the art in the respective field at the time of publication. All systems were constantly improved over time and – except for the piano note transcription system described in Part iv – still define the current state of the art in onset detection (Part i), beat and downbeat tracking (Part ii), and tempo estimation (Part iii).

More specifically, we contributed to these areas by:

1. Proposing an online, real-time onset detection algorithm build upon a recurrent neural network (RNN) capable of reporting the onsets with almost no delay (Chapter 2).

2. Evaluating existing onset detection algorithms regarding their online capabilities and modifying them for online real-time applications, achieving performance comparable to offline algorithms (Chapter 3).
3. Enhancing spectral flux-based onset detection algorithms with an additional vibrato-suppression stage based on a simple, yet very effective trajectory tracking with maximum filters which can reduce the number of false positive detections by up to 60% (Chapter 4).
4. Proposing a beat tracking system which models the temporal context a beat occurs in with an RNN incorporating long short-term memory (LSTM) units to predict beats on a frame-by-frame basis, using autocorrelation to determine the predominant tempo and the phase of the beats (Chapter 5).
5. Extending the system with a multi-model approach which considers heterogeneous music styles and replacing the autocorrelation function with a more powerful dynamic Bayesian network (DBN) for the globally best decoding of the beat sequence (Chapter 6).
6. Further extending the system to report both beats and downbeats. The system features an RNN which models the metrical structure of the music at multiple levels and a DBN being able to model bars of arbitrary length, inferring jointly the tempo, meter, beat and downbeat positions (Chapter 7).
7. Proposing an algorithm which accurately estimates the tempo of a musical piece by using an intermediate beat-level representation obtained by an RNN as input to a bank of resonating comb filters (Chapter 8).
8. Introducing a system for polyphonic piano note transcription utilising an RNN with a regression output layer, lowering the number of erroneous note detections considerably due to its ability to model simultaneously played notes (Chapter 9).

DEFINE A GENERAL-PURPOSE AUDIO FEATURE

We define a universally applicable, well-performing audio signal processing pipeline which can be used either directly (Chapter 3 and 4), or as a pre-processing step for (recurrent) neural networks (Chapter 6, 7, 8, and 9).

PROVIDE A COMPREHENSIVE OPEN-SOURCE SOFTWARE LIBRARY

Reference implementations for all algorithms developed throughout this thesis are bundled in an open-source software library written in Python (Chapter 10).

PROVIDE A LARGE SET OF AUDIO DATA ANNOTATIONS

We do not only provide software, but also release all the data used within this thesis. Due to copyright reasons, we provide only ground-truth annotations for all datasets, not the audio itself. The data is organised in a central Git repository which can be accessed at `http://phd.minimoog.org`.

## 1.3 MAIN PUBLICATIONS

The following peer-reviewed publications constitute the individual chapters of this thesis.

- Sebastian Böck and Markus Schedl. "Enhanced Beat Tracking with Context-Aware Neural Networks". In: *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*. Paris, France, 2011
- Sebastian Böck and Markus Schedl. "Polyphonic Piano Note Transcription with Recurrent Neural Networks". In: *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan, 2012
- Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl. "Online Real-time Onset Detection with Recurrent Neural Networks". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*. York, UK, 2012
- Sebastian Böck, Florian Krebs, and Markus Schedl. "Evaluating the Online Capabilities of Onset Detection Methods". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012
- Sebastian Böck and Gerhard Widmer. "Maximum Filter Vibrato Suppression for Onset Detection". In: *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*. Maynooth, Ireland, 2013
- Sebastian Böck, Florian Krebs, and Gerhard Widmer. "A multi-model approach to beat tracking considering heterogeneous music styles". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014
- Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Accurate Tempo Estimation based on Recurrent Neural Networks and Resonating Comb Filters". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015

- Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Joint Beat and Downbeat Tracking with Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016
- Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. "madmom: a new Python Audio and Music Signal Processing Library". In: *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, 2016

## 1.4 RELATED PUBLICATIONS

In addition to the before mentioned publications, the author contributed to the following peer-reviewed papers as the first author or as a co-author.

- Andreas Arzt, Sebastian Böck, and Gerhard Widmer. "Fast Identification of Piece and Score Position via Symbolic Fingerprinting". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012
- Andreas Arzt, Gerhard Widmer, Sebastian Böck, Reinhard Sonnleitner, and Harald Frostel. "Towards a Complete Classical Music Companion". In: *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*. Montpellier, France, 2012
- Andreas Arzt, Sebastian Böck, Sebastian Flossmann, Harald Frostel, Martin Gasser, and Gerhard Widmer. "The complete classical music companion v0.9". In: *Proceedings of the Audio Engineering Society 53rd International Conference on Semantic Audio*. London, UK, 2014
- Andreas Arzt, Sebastian Böck, Sebastian Flossmann, Harald Frostel, Martin Gasser, Cynthia C. S. Liem, and Gerhard Widmer. "The Piano Music Companion". In: *Proceedings of the 21th European Conference on Artificial Intelligence (ECAI)*. Prague, Czech Republic, 2014
- Sebastian Böck and Gerhard Widmer. "Local Group Delay based Vibrato and Tremolo Suppression for Onset Detection". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013
- Sebastian Böck, Jan Schlüter, and Gerhard Widmer. "Enhanced peak picking for onset detection with recurrent neural networks". In: *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*. Prague, Czech Republic, 2013

- Tom Collins, Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Bridging the Audio-Symbolic Gap: The Discovery of Repeated Note Content Directly from Polyphonic Music Audio". In: *Proceedings of the Audio Engineering Society 53rd International Conference on Semantic Audio*. London, UK, 2014

- Matthew E. P. Davies and Sebastian Böck. "Evaluating the Evaluation Measures for Beat Tracking". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014

- Bruno Di Giorgi, Massimiliano Zanoni, Sebastian Böck, and Augusto Sarti. "Multipath Beat Tracking". In: *Journal of the Audio Engineering Society* 64.7 (2016)

- Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. "Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks". In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, The Netherlands, 2010

- Florian Hörschläger, Richard Vogl, Sebastian Böck, and Peter Knees. "Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from Wikipedia". In: *Proceedings of the 12th Sound and Music Conference (SMC)*. Maynooth, Ireland, 2015

- Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. "On the Potential of Simple Framewise Approaches to Piano Transcription". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016

- Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015

- Filip Korzeniowski, Sebastian Böck, and Gerhard Widmer. "Probabilistic extraction of beat positions from a beat activation function". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014

- Florian Krebs, Sebastian Böck, and Gerhard Widmer. "Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013

- Florian Krebs, Sebastian Böck, and Gerhard Widmer. "An Efficient State Space Model for Joint Tempo and Meter Tracking". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015

- Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. "Downbeat Tracking using Beat-Synchronous Features and Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016

- Bernhard Lehner, Gerhard Widmer, and Sebastian Böck. "A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks". In: *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*. Nice, France, 2015

- Bernhard Niedermayer, Sebastian Böck, and Gerhard Widmer. "On the Importance of "Real" Audio Data for MIR Algorithm Evaluation at the Note-Level - A Comparative Study". In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA, 2011

- Markus Schedl, Peter Knees, and Sebastian Böck. "Investigating the Similarity Space of Music Artists on the Micro-Blogosphere". In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA, 2011

- Jan Schlüter and Sebastian Böck. "Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*. Prague, Czech Republic, 2013

- Jan Schlüter and Sebastian Böck. "Improved Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy, 2014

# Part I

# ONSET DETECTION

# 2

# ONLINE REAL-TIME ONSET DETECTION

### TITLE

Online Real-time Onset Detection with Recurrent Neural Networks.

### AUTHORS

Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl.

### PUBLISHED

In Proceedings of the 15th International Conference on Digital Audio Effects (DAFx), September 2012, York, United Kingdom.

### CONTRIBUTION

Idea, reference implementation, and experiments were conceived and carried out by me. Andreas Arzt contributed a real-time implementation working on live audio signals. Florian Krebs and Markus Schedl provided valuable input.

### ERRATA

The original paper includes an error in Section 2.2.2, second paragraph: The targets of the training examples were shifted one frame (i.e. 10 ms) into the future. Given the fact that the delayed reporting (as outlined in Section 2.2.3) is not needed, this results in an extra shift of roughly 5 ms in Table 2.2. The performance reported in Table 2.1 is not affected. We corrected this error and updated this chapter accordingly.

### ABSTRACT

We present a new onset detection algorithm which operates online in real time with minimal delay. Our method incorporates a recurrent neural network to model the sequence of onsets based solely on causal audio signal information. Comparative performance against existing state-of-the-art online and offline algorithms was evaluated using a very large database. The new method – despite being an online algorithm – shows performance only slightly short of the best existing offline methods while outperforming standard approaches.

## 2.1 INTRODUCTION

Onset detection is the process of locating events in an audio signal (e.g. a singing voice, a played note, or any other sounds). Various methods have been proposed over the years, but most of them work only in offline mode. Bello et al. [6] and Collins [27] give good overviews of standard methods, and Dixon [43] proposes enhancements to several of these. Traditional onset detection methods usually incorporate only spectral and/or phase information of the signal. However, unlike current top-performance algorithms, they neither employ machine learning techniques nor use probabilistic information. For example, the approaches presented in [55, 91] use neural networks and that in [36] a Hidden Markov model. They all have in common, that they usually work only in offline mode because the peak-picking methods used rely on future information to determine the location of an onset.

Only few algorithms have been designed specifically for online scenarios [127], where the aim is to minimise the delay between the occurrence of the onset in the audio signal and its reporting. Instantaneously detected onsets are a prerequisite for all kinds of real-time applications, ranging from beat-tracking and tempo estimation methods to look-ahead compressors for live audio processing.

## 2.2 SYSTEM DESCRIPTION

The proposed system is based on the state-of-the-art onset detection algorithm that won the last two years' MIREX onset detection evaluations.[1][2]. The system was originally proposed by Eyben et al. [55] and has since then been modified and enhanced considerably, as the improvements in the MIREX results show. In the next sections, we present the modifications and enhancements made in order to enable the system to work in real-time online scenarios.

The system is structured as depicted in Figure 2.1 and comprises three main processing steps: signal pre-processing, neural network onset prediction, and peak post-processing. As input, the system takes a discretely sampled audio signal and transfers it to the frequency domain via three parallel short-time Fourier transforms (STFT) with different window lengths. The information obtained is then fed into the recurrent neural network to detect the next occurring onset in the audio stream. Finally, simple post-processing is used to report the onsets instantaneously while minimising the number of false detections.

1 http://nema.lis.illinois.edu/nema_out/mirex2010/results/aod/
2 http://nema.lis.illinois.edu/nema_out/mirex2011/results/aod/

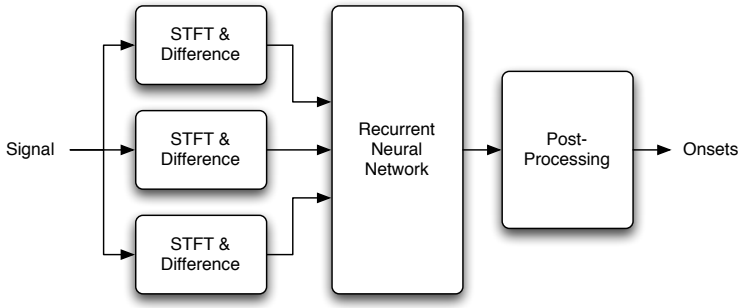



Figure 2.1: Online real-time onset detection system overview.

### 2.2.1 *Audio signal pre-processing*

The system processes the audio signal frame-wise with adjacent frames 10 ms apart (i.e. the resulting frame rate is 100 frames per second). The audio signal is transformed to the frequency domain with the short-time Fourier transform (STFT). Three parallel STFTs with different window lengths are used to capture both very recent and also 'older' information. The right edges of Hann windows are aligned at the current position of the audio signal and the windows are normalised to have equal area (cf. Figure 2.2). The sizes used are 512, 1024, and 2048 samples, which corresponds to periods of 11.61, 23.22, and 46.44 ms, respectively, at a sample rate of 44,100 Hz.

The linear magnitude spectrogram of each STFT is then filtered to obtain a compressed representation. We investigated various strategies to reduce the dimensionality of the input vector for the neural network. Using a filterbank with frequencies aligned to the Bark scale yielded a good compromise between performance and size of the neutral network input vector. The edge frequencies of the bins correspond to the frequencies of the 24 critical bands of the Bark scale [57], and triangular filters (with an area normalised to one) are used to sum the multiple frequency bins of the STFT to a single one. To transform the values to a range better suited to the downstream neural network step, we chose a logarithmic representation of the Bark spectrograms.

Since onsets are characterised by a rise in energy in their attack phase, the differences relative to preceding frames are also included in the input vector. The exact delay $\tau$ for calculating the difference is determined according to the STFT length such that the overlap of the two frames is 0.5. This results in $\tau$ values of 1, 2, and 4 for the STFT window lengths of 512, 1024, and 2048 samples. Although technically speaking it is a quotient because it is calculated using logarithmic representations, we use the term "difference" between two frames. The three parallel Bark-filtered spectrograms and the differences make up the 144-dimensional input vector for the neural network.

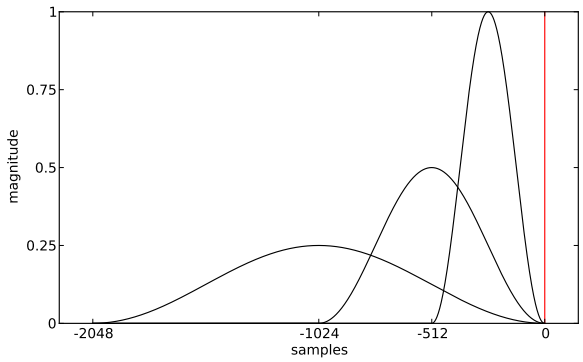



Figure 2.2: Window functions applied to audio signal before STFT, with the current position of the audio signal indicated by a vertical line.

### 2.2.2 *Neural network*

To work in a real-time online scenario, the neural network of the offline approach [55] had to be changed considerably. Since bidirectional neural networks violate causality, they are not suitable for this task and were replaced by a unidirectional one. Also, the Long Short-Term Memory (LSTM) units used in the hidden layer were replaced by standard units with a hyperbolic tangent activation function. This reduces the connections in the recurrent hidden layers by a factor of four, because the standard units do not require the gates of the LSTM units to be connected. Although LSTM units are able to model a wider temporal context, normal units perform similarly well because the temporal context for onset detection is limited to only a few frames. The overall topology of the network, consisting of three fully connected recurrent hidden layers with 20 units each, is retained. The modifications listed reduce the computational complexity of the system and make it suitable for real-time processing.

#### *Network training*

The network was trained as a classifier with supervised learning and early stopping on a 75% portion of the complete dataset described in Section 2.3. Each audio sequence was pre-processed as described above and presented to the network for learning. The network weights were initialised with random values following a Gaussian distribution with mean 0 and standard deviation 0.1. Standard gradient descent with backpropagation of the errors was used to train the network. To avoid over-fitting, the performance was evaluated after each training iteration on a separate validation set (a disjoint 15% of the training set chosen at random). If no improvement was observed for 20 epochs, training was stopped, and the network state with the best performance on the validation set was subsequently used.

When training a neural network to detect an upcoming onset, various strategies for target placement are possible: placing them at the real ground-truth positions and training the classifier as in an offline scenario, or displacing the targets into the "future". Although introducing some delay, training with targets shifted one frame into the future yielded the best classification results.

*Network testing*

The output of the network is an onset activation function with values in the range of $[0 \dots 1]$ which represent the probabilities of onsets at given positions. Figure 2.3 shows a typical onset activation function with clearly visible peaks at the annotated positions.

### 2.2.3 *Post-processing*

Since no future values are available in online mode, the traditional approach of finding local maxima in the thresholded onset activation function cannot be applied here. Instead, the onset is predicted as soon as the activation function exceeding a given threshold, determined on the validation set by 8-fold cross-validation.

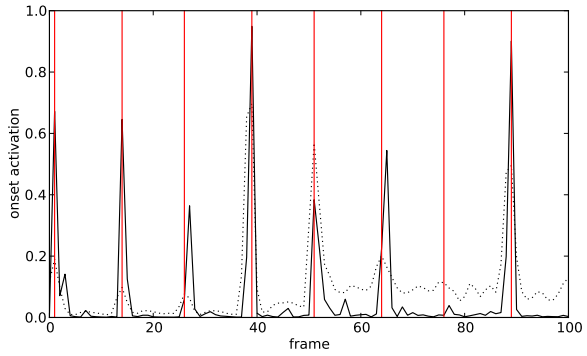



Figure 2.3: Onset activation function (output of the neural network) of the system for a 1-second-excerpt of a pop song shown as a solid black line. Annotated onsets are indicated by vertical lines and the normalised detection function obtained with *spectral flux* is plotted as a black dotted line.

Compared to simple signal-based onset detection methods, the main advantage of using a neural network is that its onset activation function has a very low noise floor with high peaks at the onset positions (see Figure 2.3, solid black line). Thus, a very low threshold can be used to detect the onsets as early as possible without risking many false detections. To prevent repeated reporting of an onset (and thus producing numerous false positive detections), an onset is only reported if no onsets have been detected in the previous two frames (20 ms).

In the rare case of a slowly rising onset activation function (which exceeds the threshold), this peak-picking method could lead to some early false positive detections. To give an estimate of the penalty, we also evaluated our new algorithm with an offline peak picking algorithm which uses only local maxima after thresholding of the onset activation function.

## 2.3 DATA

In online real-time processing, the definition of onsets is crucial. An onset is usually defined as the exact time a note or instrument starts sounding after being played. However, this timing is difficult to determine, and it is therefore impossible to annotate the real onset timing in complex audio recordings with multiple instruments, voices, and effects.

The most commonly used method for onset annotation is marking the earliest time point at which a sound is audible to humans. This instant cannot be defined by pure measures (e.g. minimum increase of volume or sound pressure), but is a complex mixture of various factors. All annotations of the dataset try to match the onset time as accurately as possible. Compared to synthesised sounds generated from MIDI, this generally leads to a delay in the range of a few milliseconds (determined by manual correction of a piece of synthesised piano music to match the style of other annotations).

The annotation process is very time-consuming because it is performed in multiple passes. First, onsets are annotated manually during slowed-down playback. In the second pass, visualisation support is used to refine the onset positions. Spectrograms obtained with different STFT lengths are used together to capture the precise timing of an onset without missing any onsets due to insufficient frequency resolution. This multi-resolution procedure seems to be a good approach since the best onset detection algorithms also use it internally. If multiple onsets are located in close vicinity, they are annotated as multiple onsets. For reference, Figure 2.4 shows a piano chord in which the individual notes were not played perfectly simultaneously with two individual annotations.

The dataset consists of 327 audio excerpts taken from different sources. 87 tracks were taken from the dataset used in [55], 23 from [6], and 92 from [74]. All annotations were manually checked and corrected where needed. The remaining 125 files were newly annotated during the evolutionary process of the offline *OnsetDetector*. The complete set contains 28,067 onsets according to the annotation style outlined above. Although musically correct, the precise annotations do not necessarily represent the human perception of onsets. Thus, all onsets within 30 ms are combined into a single one located at the

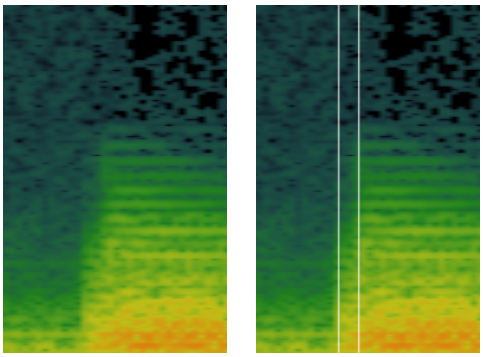


Figure 2.4: Zoomed-in spectrogram of a piano chord with two notes played 4 ms apart. The pictures show a period of 50 ms (identical to the detection window used for evaluation) taken with an STFT length of 512 samples and 86% overlap.

arithmetic mean of the positions, which results in 26,223 combined onsets used for evaluation.

## 2.4 RESULTS AND DISCUSSION

We used 8-fold cross-validation with the standard measures precision, recall, and F-measure to evaluate our approach. An onset is considered correctly detected if there is a ground-truth annotation within $\pm 25$ ms around the predicted position. We refer to this region as *detection window*. We chose this relatively strict evaluation method (also used in [55] and [27] for percussive sounds) because it gives more meaningful results – especially for the task of online onset detection – than the detection window of $\pm 50$ ms used in [6, 43, 127]. All annotated onsets can only be matched once: two detections within the detection window of a single annotated onset are counted as one true positive and one false positive detection.

### 2.4.1 *Performance comparison*

Table 2.1 lists the results for the complete dataset described in Section 2.3. The new online algorithm *OnsetDetector.LL*[3] is compared to its offline variant and also to all detection methods implemented by `aubio`.[4] Our new online algorithm was evaluated by 8-fold cross-validation with disjoint training, validation, and test sets. Since the offline variants of *OnsetDetector* were trained with audio material originating from the same dataset, we took particular care to avoid using the same audio excerpts for both training and testing. The parameters of the `aubio` results were optimised on the complete dataset

3 The onset detector is named after Lucky Luke, the cowboy known to"shoot faster than his shadow", because it is able to detect an onset before a human can hear it.

4 `http://aubio.org/` version 0.3.2

(optimised threshold and shifting of the reported onsets to best match the annotations), and hence represent the maximum achievable performance by traditional algorithms when perfectly adapted to the test set.

It should be noted that the upper part of Table 2.1 lists offline algorithms which incorporate future information for onset detection, while our new algorithm solely relies on past information to detect upcoming onsets.

| OFFLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| OnsetDetector.2010 | 0.857 | 0.796 | 0.826 |
| OnsetDetector.2011 | 0.906 | 0.830 | 0.866 |
| `aubio` default * | 0.718 | 0.690 | 0.704 |
| `aubio` specdiff * | 0.653 | 0.650 | 0.652 |
| `aubio` phase * | 0.516 | 0.600 | 0.555 |
| `aubio` complexdomain * | 0.700 | 0.690 | 0.695 |
| `aubio` hfc * | 0.750 | 0.733 | 0.742 |
| `aubio` energy * | 0.608 | 0.572 | 0.590 |
| `aubio` kl * | 0.672 | 0.670 | 0.671 |
| `aubio` mkl * | 0.647 | 0.599 | 0.622 |
| ONLINE | | | |
| NEW OnsetDetector.LL | 0.850 | 0.787 | 0.817 |
| NEW OnsetDetector.LL † | 0.897 | 0.789 | 0.840 |

Table 2.1: Comparison of performance results using the complete dataset with a detection window of $\pm 25$ ms. All algorithms in the upper part operate offline, while only the new one works in online mode. Asterisks mark results obtained with parameters optimised on the complete dataset. The last result, denoted with † was obtained with an offline peak-picking method.

As expected, the new *OnsetDetector.LL* falls short of the performance of state of the art offline onset detection algorithms (i.e. the offline version of OnsetDetector) but clearly outperforms other onset detection methods such as *Spectral Flux*, *Complex Domain*, *High Frequency Content*, and combinations thereof (as reflected by the result obtained with the `aubio` algorithm), even when they were perfectly adapted to the test set. This shows the strength of the new online onset detection algorithm.

As mentioned in Section 2.2.3, the chosen online peak-picking method can lead to false positive detections. The last line of Table 2.1 shows the performance obtained with an offline peak-picking method, which yields a reduced number of false positive detections as reflected by absolute increase in precision of almost 5%. Comparison

of this result with the offline *OnsetDetector* suggests that using future information is advantageous for onset detection.

### 2.4.2 *Detailed evaluation*

Table 2.2 presents the performance results of the new algorithm evaluated on the complete data set with various detection window sizes. It exhibits remarkable performance down to a window size of ±25 ms around the ground-truth positions of the onsets. Only when evaluated with smaller window sizes, the performance drops considerably. Although a detection window of ±15 ms is close to the accuracy of a manual annotation (which is typically around ±2 ms for percussive sounds and up to ±10 ms for soft onsets generated by instruments such as string or woodwind instruments), the algorithm continues to identify some of onsets correctly.

| WINDOW | PRECISION | RECALL | F-MEASURE | ERROR |
|---|---|---|---|---|
| ±50 ms | 0.885 | 0.786 | 0.833 | 4.9±8.5 ms |
| ±35 ms | 0.839 | 0.753 | 0.794 | 5.0±7.2 ms |
| ±30 ms | 0.808 | 0.725 | 0.765 | 4.9±6.8 ms |
| ±25 ms | 0.738 | 0.662 | 0.698 | 4.6±6.1 ms |
| ±20 ms | 0.586 | 0.526 | 0.554 | 3.9±5.4 ms |
| ±15 ms | 0.368 | 0.331 | 0.348 | 1.8±3.9 ms |

Table 2.2: Performance results of the new algorithm on the complete dataset with different evaluation windows. Additionally, the mean and standard deviation error of all correctly detected onsets relative to their ground-truth annotations are given.

### 2.4.3 *Computational cost*

The system works on a frame-by-frame basis with a hop-size of 10 ms. For each audio frame, pre-processing, computing the output activations of the neural network, and post-processing take a constant amount of time and are easily done in real time on a single core of a 2.26 GHz Intel Core 2 Duo CPU.

## 2.5 CONCLUSIONS

We have presented a new onset detection algorithm specifically designed for real-time online detection of musical onsets in audio signals. It achieves performance close to current state-of-the-art offline onset detection algorithms while introducing a minimal delay between the

audio signal and the reporting of an onset. On modern hardware, the computational processing can easily be achieved in real time.

### ADDENDUM

The signal pre-processing was later adapted to the findings of this thesis. Instead of Bark-filtering (cf. Section 2.2.1), a logarithmically spaced filterbank with 6 bands per octave as in Section 3.2.5 is used. The implementation included in *madmom* (Chapter 10) reflects this change.

The dataset used contains 6 files which were later removed, because they were duplicates (2 files also contained in the set of others) or were near-duplicates (4 files recorded at different attenuation levels). Chapter 3 and 4 use the corrected dataset.

#### *MIREX evaluation*

The system presented was submitted and evaluated during MIREX 2012 and ranked second, after the offline variant *OnsetDetector* (an improved version of the system proposed by Eyben et al. [55]). Although *OnsetDetectorLL* operates in online mode, it outperforms all previously submitted offline algorithms.

The 2015 submission uses the altered signal pre-processing parameters outlined above. Furthermore it uses multiple neural network models trained on a larger training set and averages the predictions of these individual models. This version shows the highest online performance reported ever.

The submission with the highest ever reported score, the *CNNOnsetDetector* refers to the algorithm presented by Schlüter and Böck [119, 120]. It incorporates the knowledge and findings obtained during the course of this thesis and defines the current state-of-the-art in onset detection.

| ONLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| OnsetDetectorLL (2012) | 0.845 | 0.848 | 0.831 |
| OnsetDetectorLL (2015) | 0.874 | 0.856 | 0.851 |
| OFFLINE | | | |
| OnsetDetector (2013) | 0.869 | 0.888 | 0.867 |
| CNNOnsetDetector (2016) | 0.860 | 0.899 | 0.873 |

Table 2.3: Performance of the presented *OnsetDetectorLL* algorithm in MIREX evaluations compared to state-of-the-art online and offline onset detection algorithms. The names reflect the names of the executable programs of the *madmom* package (cf. Section 10.2.2).

# 3

# ONLINE ONSET DETECTION EVALUATION

### TITLE

Evaluating the Online Capabilities of Onset Detection Methods.

### AUTHORS

Sebastian Böck, Florian Krebs, and Markus Schedl.

### PUBLISHED

In Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR), October 2012, Porto, Portugal.

### CONTRIBUTION

Main idea and reference implementation were conceived and carried out by me. Florian Krebs ran most of the experiments and designed the peak detection (cf. Section 3.2.3). Markus Schedl provided valuable input.

### ABSTRACT

In this paper, we evaluate various onset detection algorithms in terms of their online capabilities. Most methods use some kind of normalization over time, which renders them unusable for online tasks. We modified existing methods to enable online application and evaluated their performance on a large dataset consisting of 27,774 annotated onsets. We focus particularly on the incorporated preprocessing and peak detection methods. We show that, with the right choice of parameters, the maximum achievable performance is in the same range as that of offline algorithms, and that preprocessing can improve the results considerably. Furthermore, we propose a new onset detection method based on the common *spectral flux* and a new peak-picking method which outperforms traditional methods both online and offline and works with audio signals of various volume levels.

## 3.1 INTRODUCTION AND RELATED WORK

Onset detection, the task of finding musically meaningful events in audio signals, is fundamental to many applications: Real-time applications such as automatic score followers [31] can be enhanced by incorporating (online) onset detectors that look for note onsets in a live performance, while (offline) onset detection is used increasingly to improve digital audio workstations with a view to event-wise audio processing.

Many different methods of solving this task have been proposed and evaluated over the years. Comprehensive overviews of onset detection methods were presented by Bello et al. [6] and Collins [27] (with special emphasis on psychoacoustically motivated methods in the latter). Dixon [43] proposed enhancements to several of these. All methods were evaluated in an *offline* setting, using a normalisation over the whole length of the signal or applying averaging techniques which require future information.

For *online* onset detection, only few evaluations have been carried out: Brossier et al. [24] compared four onset functions based on spectral features and proposed a method for dynamic thresholding in online scenarios, using a dataset of 1,066 onsets for evaluation. Stowell and Plumbley [127] proposed *adaptive whitening* as an improvement to short-time Fourier transform (STFT) based onset detection methods and evaluated eight detection functions using a dataset of 9,333 onsets. Glover et al. [61] applied linear prediction and sinusoidal modelling to online onset detection, but used a relatively small dataset of approximately 500 onsets for evaluation.

These traditional onset detection methods usually incorporate only spectral and/or phase information of the signal, are easy to implement, and have modest computational cost. In contrast, methods based on machine learning techniques (e.g. neural networks [55, 91]) or on probabilistic information (e.g. hidden Markov models [36]) depend on large datasets for training and are in general computationally more demanding, which makes them unsuited for online processing.

The onset detection process is usually divided into three parts (as shown in Figure 3.1): signal preprocessing, computation of the actual onset detection function (ODF), and peak detection.


Figure 3.1: Basic onset detection workflow.

There are generally two normalisation steps that require special attention in an online context: The first can be found in the prepro-

cessing step where many implementations normalise the audio input prior to further processing.

The second and more widespread use of normalisation is in the peak detection stage, where the whole ODF is normalised before being processed further. An exception to this rule are some machine learning approaches like the neural network-based methods, since their detection function can be considered as a probability function which already has the range $[0..1]$. Furthermore, most offline methods use smoothing or averaging over (future) time to compute dynamic thresholds for the final peak-picking.

This paper is structured as follows: We combine the ODFs described in Section 3.2.2 with different preprocessing methods from Section 3.2.1 and evaluate them on the dataset described in Section 3.3.1 using the peak-picking method given in Section 3.2.3. In Section 3.4 we discuss the results, and we give conclusions in Section 3.5.

## 3.2 COMPARED METHODS

Previously, onset detection algorithms used to work directly with the time signal $x(t)$. However, all current onset detection algorithms use a frequency representation of the signal. We used frames of 23 ms length (2048 samples at a sample rate of 44.1 kHz) that are filtered with a Hann window before transfer into the frequency domain by means of STFT. The hopsize between two consecutive frames was set to 10 ms, which results in a frame rate of 100 frames per second. The resulting spectrogram $X(n,k)$ ($n$ denoting the frame and $k$ the frequency bin number) was then processed further by the individual preprocessing and onset detection algorithms.

### 3.2.1 *Pre-processing*

#### *Filtering*

Scheirer [117] stated that, in onset detection, it is advantageous if the system divides the frequency range into fewer sub-bands as done by the human auditory system. Filtering has been applied by many authors (e.g. [27, 79, 117]), and neural network based approaches also use filter banks to reduce the dimensionality of the STFT spectrogram [55].

#### *Logarithmic magnitude*

Using the logarithmic magnitude instead of the linear representation was found to yield better results in many cases, independently of the ODF used [55, 79]. $\lambda$ is a compression parameter and was adjusted for

each method separately. Adding a constant value of 1 results in only positive values:

$$X^{log}(n,k) = log(\lambda \cdot X(n,k) + 1) \tag{3.1}$$

*Adaptive whitening*

Stowell and Plumbley [127] proposed adaptive whitening, which normalises the magnitudes $|X(n,k)|$ of each frequency bin separately by past peak values. The iterative algorithm (with $r$ being a floor parameter and $m$ the memory coefficient) is given as follows:

$$P_{n,k} = \begin{cases} max(|X(n,k)|, r, m \cdot P_{n-1,k}) & \text{if } n \geq 1 \\ max(|X(n,k)|, r) & \text{otherwise} \end{cases}$$

$$|X(n,k)| \leftarrow \frac{|X(n,k)|}{P_{n,k}} \tag{3.2}$$

### 3.2.2 *Onset detection functions*

We have chose to omit other common methods such as phase deviation (PD) [8], high frequency content (HFC) [98] or rectified complex domain (RCD) [43], since they exhibited inferior performance in our tests.

*Spectral flux*

The spectral flux (SF) [98] describes the temporal evolution of the magnitude spectrogram by computing the difference between two consecutive short-time spectra. This difference is determined separately for each frequency bin, and all positive differences are then summed to yield the detection function.

$$SF(n) = \sum_{k=1}^{k=\frac{N}{2}} H(|X(n,k)| - |X(n-1,k)|) \tag{3.3}$$

with $H(x) = \frac{x+|x|}{2}$ being the half-wave rectifier function. Variants of this method use the $L_2$-norm instead of the $L_1$-norm or the logarithmic magnitude [79] (cf. Section 3.2.1).

*Weighted phase deviation*

Another class of detection function utilises the phase of the signal [8, 43]. The change in the instantaneous frequency (the second order

derivative of the phase $\varphi(n,k)$) is an indicator of a possible onset. [43] proposed an improvement to the phase deviation ODF called weighted phase deviation (WPD). The WPD function weights each frequency bin of the phase deviation function with its magnitude.

$$WPD(n) = \frac{2}{N} \sum_{k=1}^{k=\frac{N}{2}} |X(n,k) \cdot \varphi''(n,k)| \tag{3.4}$$

Another way to incorporate both magnitude and phase information (as in the WPD detection function) was proposed by Duxbury et al. [48]. First, the expected target amplitude and phase $X_T(n,k)$ for the current frame are estimated based on the values of the two previous frames assuming constant amplitude and rate of phase change. The complex domain (CD) ODF is then defined as:

$$CD(n) = \sum_{k=1}^{k=\frac{N}{2}} |X(n,k) - X_T(n,k)| \tag{3.5}$$

### 3.2.3 *Peak detection*

Illustrated in Figure 3.2 and common to all onset detection methods is the final thresholding and peak-picking step to detect the onsets in the ODF. Various methods have been proposed in the literature; we give an overview of the different components and modifications needed to make them suitable for online processing.

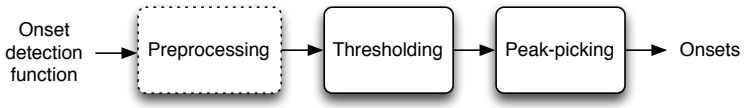



Figure 3.2: Peak detection process.

*Pre-processing*

The preprocessing stage of the peak detection process consists mainly of two components: smoothing of the peaky ODF and normalisation. Both of them cannot be used in an online scenario. Instead, moving average techniques as outlined in Section 3.2.3 are applied to normalise the ODF locally. To prevent detecting many false positives due to a peaky ODF, the effect of smoothing can be approximated by introducing a minimal distance from the last onset $w_5$ as proposed in Section 3.2.3.

*Thresholding*

Before picking the final onsets from the ODF, thresholding is performed to discard the non-onset peaks. Most methods use dynamic thresholding to take into account the loudness variations of a music piece. Mean [43], median[8, 55, 127] or combinations [24, 61] are commonly used to filter the ODF. If only information about the present or past is used, the thresholding function is suitable for online processing.

*Peak-picking*

Two peak-picking methods are commonly used for final detection of onsets. One selects all local maxima in the thresholded detection function as the final onset positions. Since detecting a local maximum requires both past and future information, this method is only applicable to offline processing.

The other method selects all values above the previously calculated threshold as onsets and is also suitable for online processing. The downside of this approach is its relatively high false positive rate because the threshold parameter must be set to a very low level to detect the onsets reliably.

*Proposed peak detection*

We use a modified version of the peak picking method proposed by Dixon [43] to also satisfy the constraints for online onset peak detection. A frame $n$ is selected as an onset if the corresponding $ODF(n)$ fullfils the following three conditions:

1. $ODF(n) = max(ODF(n - w_1 : n + w_2))$
2. $ODF(n) \geq mean(ODF(n - w_3 : n + w_4)) + \delta$
3. $n - n_{last\ onset} > w_5$

where $\delta$ is a fixed threshold and $w_1..w_5$ are tuneable peak-picking parameters. For online detection, we set $w_2 = w_4 = 0$. Our online experiments experiments showed that, on average, onsets are detected one frame earlier than annotated in the dataset (using the values specified in Section 3.3.3). As we want to find the perceptual onset times (as annotated), we report the onset one frame later than detected. Note that this does not mean that we predict the onset, it only means that the onset can be recognised in the signal before it is perceived.

Unlike in previous studies [24, 61, 127] we do not use the same thresholding parameters for all ODFs. This is mainly because some of the ODFs have fewer peaks and hence need less averaging in the thresholding stage than others.

### 3.2.4 *Neural network based methods*

For reference, we compare the presented methods with two state-of-the-art algorithms, the *OnsetDetector* [55] and its online variant *OnsetDetector.LL* [11]:

*OnsetDetector* uses a bidirectional neural network which processes the signal both in a forward and backward manner, making it an offline algorithm. The algorithms showed exceptional performance compared to other algorithms independently of the type of onsets in the audio material, especially in its latest version tested during the 2011 MIREX onset detection evaluation.[1]

*OnsetDetector.LL* incorporates a unidirectional neural network to model the sequence of onsets based solely on causal audio signal information.

Since these methods show very sharp peaks (representing the probability of an onset) at the actual onset positions, the before mentioned peak detection method is not applied, and a simple thresholding is used instead.

### 3.2.5 *New method*

We propose a new onset detection method which is based on the spectral flux (cf. Section 3.2.2), drawing on various other author's ideas.

As a first step, we filter the linear magnitude spectrogram $|X(n,k)|$ with a filterbank. We investigated different types of filterbanks (Mel, Bark, Constant-Q) and found that they all outperform the standard spectral flux. Since they all perform approximately equally well when using a similar number of filter bands, we chose a pseudo Constant-Q, where the frequencies are aligned according to the frequencies of the semitones of the western music scale over the frequency range from 27.5 Hz to 16 kHz, but using a fixed window length for the STFT. Overlapping triangular filters sum all STFT bins belonging to one filter bin (similarly to Mel filtering). The resulting filterbank $F(k,b)$ has $B = 82$ frequency bins with $b$ denoting the bin number of the filter and $k$ the bin number of the linear spectrogram. The filters have not been normalised, resulting in an emphasis of the higher frequencies, similar to the HFC method. The resulting filtered spectrogram $X_{filt}(n,b)$ is given by:

$$X_{filt}(n,b) = |X(n,k)| \cdot F(k,b) \tag{3.6}$$

Applying Equation 3.1 to the filtered linear magnitude spectrogram $X_{filt}(n,b)$ yields the logarithmic filtered spectrogram $X^{log}_{filt}(n,b)$.

1 http://nema.lis.illinois.edu/nema_out/mirex2011/results/aod/

The final ODF $O$ is then given by:

$$O(n) = \sum_{k=1}^{k=\frac{N}{2}} H\left(\left|X_{filt}^{log}(n,b)\right| - \left|X_{filt}^{log}(n-1,b)\right|\right) \tag{3.7}$$

where $H$ is the half-wave rectifier function defined in Section 3.2.2.

## 3.3 EXPERIMENTS

To evaluate the methods described, we conducted three experiments: First, the methods were evaluated under online conditions: no future information was used to decide whether there is an onset at the current time point. Second, the same methods were evaluated under offline conditions (enabling prior data normalisation or computing averages that incorporate future information) to determine the maximum performance achievable by each method. Third, we attenuated the volume of the audio data to an increasing degree to test the online methods' abilities to cope with signals of different volume without access to normalisation.

### 3.3.1 *Dataset*

To evaluate the presented onset detection and peak-picking methods we use a dataset of real world recordings.

An onset is usually defined as the exact time a note or instrument starts sounding after being played. However, this timing is hard to determine, and thus it is impossible to annotate the real onset timing in complex audio recordings with multiple instruments, voices, and effects. Thus, the most commonly used method for onset annotation is marking the earliest time point at which a sound is audible by humans. This instant cannot be defined in pure terms (e.g. minimum increase of volume or sound pressure), but is a rather complex mixture of various factors.

The annotation process is very time-consuming because it is performed in multiple passes. First, onsets are annotated manually during slowed down playback. In the second pass, visualisation support is used to refine the onset positions. Spectrograms obtained with different STFT lengths are used in combination to capture the precise timing of an onset without missing any onset due to insufficient frequency resolution. This multi-resolution procedure seems to be a good approach since the best onset detection algorithms also use this mechanism. If multiple onsets are located in close vicinity, they are annotated as multiple onsets.

The dataset contains 321 audio excerpts taken from various sources. 87 tracks were taken from the dataset used in [55], 23 from [6], and 92 from [74]. All annotations were manually checked and corrected

to match the annotation style outlined above. The remaining 119 files were newly annotated and contain the vast majority of the 27,774 onsets of the complete set.

Although musically correct, the precise annotations (raw onsets) do not necessarily represent human perceptions of onsets. Thus, all onsets within 30 ms were combined into a single one located at the arithmetic mean of the positions,[2] which resulted in 25,966 combined onsets used for evaluation. The dataset can be roughly divided into six main groups (Table 3.1).

| TYPE OF AUDIO | FILES | RAW ONSETS | COMBINED |
|---|---|---|---|
| Complex mixtures | 193 | 21,091 | 19,492 |
| Pitched percussive | 60 | 2,981 | 2,795 |
| Non-pitched percussive | 17 | 1,390 | 1,376 |
| Wind instruments | 25 | 822 | 820 |
| Bowed strings | 23 | 1,180 | 1,177 |
| Vocal | 3 | 310 | 306 |
| ALL | 321 | 27,774 | 25,966 |

Table 3.1: Description of the used dataset: Pitched percussive (e.g. piano or guitar), non-pitched percussive (e.g. percussion), wind instruments (e.g. sax or trumpet), bowed string instruments (e.g. violin or kemence), monophonic vocal music and complex mixtures (e.g. jazz, pop, or classical music).

### 3.3.2 *Measures*

For evaluation, the standard measures precision, recall, and F-measure were used. An onset is considered to be correctly detected if there is a ground truth annotation within ±25 ms around the predicted position. This rather strict evaluation method (also used in [55] and [27] for percussive sounds) was chosen because it gives more meaningful results - especially in online onset detection - than an evaluation window of ±50 ms as used in [6, 43, 127].

An important factor in the evaluation is how false positives and negatives are counted. Let us assume that two onsets are detected inside the detection window around a single annotation. If tolerant counting is used, no false positives are counted. Every single detection is considered a true positive, since there is an annotated onset within the detection window. This is often referred to as *merged* onsets. If

2 To better predict the perceived position of an onset, psychoacoustical knowledge must be applied. Since the masking effects involved depend on both loudness and frequency of an onset, they are not applied here. For the evaluation of onset detection methods as in this paper, the selected method of combination is adequate.

counted in a strict way, all annotated onsets can only be matched once, i.e. two detections within the detection window of a single onset are counted as one true positive and one false positive detection.

Since many papers do not explicitly describe the criteria, it must be assumed that the results were obtained with the first method (usually yielding better results). In this paper, we evaluated the stricter way, but with combined annotated onsets (not to be confused with merged onsets). The combining of onsets leads to less false negative detections if the algorithm reports only a single onset where multiple ones are annotated. Since most of the algorithms are not capable reporting multiple consecutive onsets, this results in a more fair comparison.

### 3.3.3 *Parameter selection*

The peak-picking parameters $w_1 ... w_5$ and the fixed threshold $\delta$ introduced in Section 3.2.3 were optimised by a grid search over the whole set for each method separately. As in [6, 43], we report the best performance for each method using the optimised global parameter set. For online detection ($w_2 = w_4 = 0$), the optimal values for $w_3$ were found to be between 4 and 12, $w_1 = 3$, and $w_5 = 3$. For the offline case, $w_2 = 3$, $w_4 = 1$ and $w_5 = 0$ yielded the best results ($w_1$ and $w_3$ were left unchanged). The adaptive whitening parameters $m = 10$ and $r = 0.005$ were found to be generally good settings and were used for all ODFs in the experiments. The compression parameter $\lambda$ (Section 3.2.1) was chosen to be between 0.01 and 20. The neural networks are trained and evaluated using 8-fold cross validation on disjoint training, validation, and test sets.

All parameters were optimised on the dataset and left unchanged for the unnormalised penalty task.

## 3.4 RESULTS AND DISCUSSION

### 3.4.1 *Comparison of different ODFs*

Table 3.2 lists the results for all algorithms working in online mode on the complete dataset using the peak detection method described in Section 3.2.3. It shows that application of adaptive whitening and use of a logarithmic magnitude both outperform the traditional methods without any preprocessing. Both preprocessing methods compress the magnitude and hence emphasise higher frequency bands that are important for detecting percussive onsets. Furthermore, our proposed method (*SF log filtered*) clearly outperforms all the other methods (apart from the reference *OnsetDetector.LL*). In particular, it is characterised by a high precision value due to the reduced number of false positives compared to the other methods. We believe that the filtering process reduces the spectrum to the most relevant components for onset

detection. This may facilitate better distinction between signal changes that are arising from an onset and spurious, non-onset-related changes.

| ALGORITHM | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| SF | 0.763 | 0.728 | 0.745 |
| SF aw | 0.780 | 0.734 | 0.757 |
| SF log | 0.783 | 0.740 | 0.761 |
| SF log filtered | 0.883 | 0.735 | 0.803 |
| CD | 0.724 | 0.698 | 0.711 |
| CD aw | 0.764 | 0.751 | 0.758 |
| CD log | 0.774 | 0.711 | 0.741 |
| WPD | 0.688 | 0.706 | 0.697 |
| WPD aw | 0.708 | 0.720 | 0.714 |
| WPD log | 0.746 | 0.675 | 0.709 |
| OnsetDetectorLL [11] | 0.850 | 0.787 | 0.817 |

Table 3.2: Precision, Recall, and F-measure of different onset detection algorithms using *online* peak-picking, where *aw* denotes adaptive whitening, *log* denotes the use of a logarithmic magnitude and *SF log filtered* is the method proposed in Section 3.2.5.

Our tests showed that - if the parameters are properly chosen - the offline results are in the same range as the online results.[3] We deem this is a remarkable finding and think that the reasons for this behaviour are the following: First, the audio tracks of the dataset have similar volume levels, which renders the normalisation step less important. Second, when looking only at single independent frames, it seems reasonable that frames after the current onset frame do not carry much additional information. However, the superior results of the offline *OnsetDetector* (F-measure 86.6, Precision 90.6, Recall 83.0) suggest that using both past and future information contained in the magnitude spectrogram can be valuable to detect also the "harder" onsets (as reflected by the much higher recall value of this method).

### 3.4.2 *Unnormalised penalty*

When dealing with unnormalised data, the investigated onset detection methods experience different levels of performance loss. As shown in Figure 3.3, our proposed onset detection method exhibits superior performance at all attenuation levels and is only beaten by the *OnsetDetector.LL*, that is unaffected by any volume changes. This shows the power of machine learning techniques that do not depend

3 We observed an average gain in F-measure of 0.25% in offline mode

on predefined peak-picking thresholds. The methods using adaptive whitening score third, which seems reasonable as these methods include an implicit normalisation using past frames. Computing the difference of two adjacent frames of the logarithmic spectrum (*SF log*) has the effect of dividing the magnitude at frame $n$ by that at frame $n-1$, resulting in the relative magnitude change rather than the absolute difference. This makes the spectral flux obtained with logarithmic magnitudes more robust against absolute volume changes, compared to the standard variant (*SF*).

Finally, methods using the logarithmic magnitude spectrum performed better at lower volume levels when using a high value of the compression parameter $\lambda$.

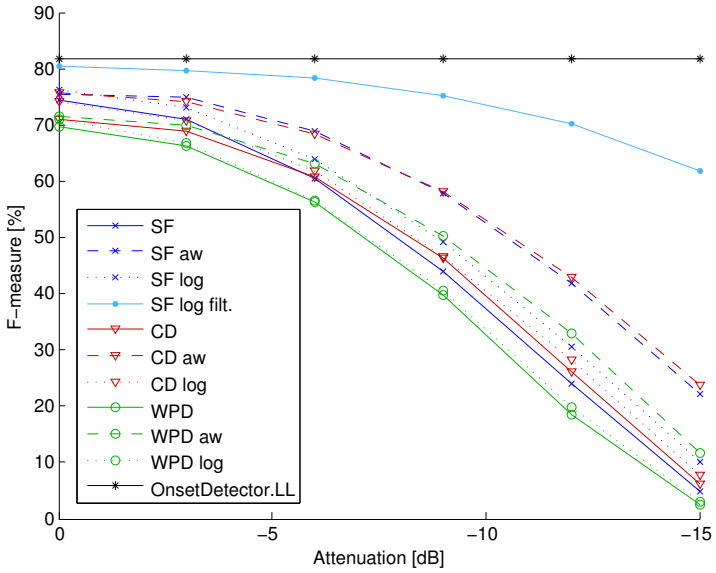



Figure 3.3: Performance of the online methods at different attenuation levels.

### 3.4.3 *Remarks*

In this paper, we give only results for the complete dataset. Results for subsets (organised by author) and a BSD licensed Python implementation of the newly proposed algorithm can be found online at `http://www.cp.jku.at/people/boeck/ISMIR2012.html`.

## 3.5 CONCLUSIONS

In this paper we have evaluated various onset detection algorithms in terms of their suitability for online use, focusing on the preprocessing and peak detection algorithms. We have shown that using logarithmic magnitudes or adaptive whitening as a preprocessing step results in improved performance in all methods investigated. When the pa-

rameters for peak detection are chosen carefully, online methods can achieve results in the same range as those of offline methods.

Further, we have introduced a new algorithm which outperforms other preprocessing methods. It copes better with audio signals of various volume levels, which is of major importance for onset detection in real-time scenarios.

Apart from that, machine learning techniques like neural network based methods are much more robust against volume changes in online scenarios and are the methods of choice if enough training data is available.

### ADDENDUM

The features, i.e. the logarithmically filtered and scaled differences of a magnitude spectrogram, developed in this paper were found to perform well for different MIR tasks and were successfully deployed in many systems, e.g. [73, 87–90].

#### *MIREX evaluation*

The newly introduced onset detection algorithm was submitted and evaluated during MIREX 2012 in both *online* and *offline* setting. In 2015 it was tuned for better offline performance. It shows performance close to the current state-of-the-art without the added complexity of the neural network-based systems.

| ONLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| LogFiltSpecFlux (2012) | 0.817 | 0.857 | 0.822 |
| OnsetDetectorLL (2015) | 0.874 | 0.856 | 0.851 |
| OFFLINE | | | |
| LogFiltSpecFlux (2012) | 0.817 | 0.848 | 0.818 |
| LogFiltSpecFlux (2015) | 0.841 | 0.855 | 0.837 |
| OnsetDetector (2013) | 0.869 | 0.888 | 0.867 |
| CNNOnsetDetector (2016) | 0.860 | 0.899 | 0.873 |

Table 3.3: Performance of the presented *LogFiltSpecFlux* algorithm in MIREX evaluations compared to state-of-the-art online and offline onset detection algorithms. The names reflect the names of the executable programs of the *madmom* package (cf. Section 10.2.2).

# 4

# VIBRATO SUPPRESSION FOR ONSET DETECTION

### TITLE

Maximum Filter Vibrato Suppression for Onset Detection.

### AUTHORS

Sebastian Böck and Gerhard Widmer.

### PUBLISHED

In Proceedings of the 16th International Conference on Digital Audio Effects (DAFx), September 2013, Maynooth, Ireland.

### CONTRIBUTION

All work was carried out by me. Gerhard Widmer provided very valuable input.

### ABSTRACT

We present *SuperFlux* - a new onset detection algorithm with vibrato suppression. It is an enhanced version of the universal spectral flux onset detection algorithm, and reduces the number of false positive detections considerably by tracking spectral trajectories with a maximum filter. Especially for music with heavy use of vibrato (e.g. sung operas or string performances), the number of false positive detections can be reduced by up to 60% without missing any additional events. Algorithm performance was evaluated and compared to state-of-the-art methods on the basis of three different datasets comprising mixed audio material (25,927 onsets), violin recordings (7,677 onsets) and operatic solo voice recordings (1,448 onsets). Due to its causal nature, the algorithm is applicable in both offline and online real-time scenarios.

## 4.1 INTRODUCTION AND RELATED WORK

Onset detection is the process of finding the starting points of all musically relevant events in an audio performance. While the detection of percussive onsets can be considered a solved problem,[1] softer onsets, vibrato and tremolo still constitute major challenges for existing algorithms.

Since soft onsets (e.g. of woodwind or bowed string instruments) have a long attack phase with a slow rise in energy, energy- and magnitude-based approaches are not the best choice for detecting them. To overcome the shortcomings of these approaches, specific algorithms that solve the soft onset problem by additionally incorporating phase [6, 8, 43] or pitch information [28, 118, 140] or a combination thereof [74] have been proposed. However, magnitude-based methods [13] have advanced and perform on par with the above methods and outperform them on all kinds of percussive audio material.

The current state-of-the-art methods for online [11] and offline [55] onset detection are based on a probabilistic model and incorporate a recurrent neural network with the spectral magnitude and its first time derivative as input features. In particular the offline variant *OnsetDetector* shows superior performance on all sorts of signals in the MIREX 2012 onset detection evaluation.[2] Because of its bidirectional architecture, it is able to model the context of an onset in order to both detect hard do discover onsets in complex mixes (e.g. a soft note onset of relatively low volume) and suppress events which are erroneously considered onsets by other algorithms, such as the sound of stopped strings.

Vibrato is an artistic effect commonly used in classical music and can be sung or played by instruments. It reflects a quasi-periodic change in the frequency of a played or sung note. Vibrato is characterised technically by the amount of pitch variation (e.g. $\pm$ a semitone for string instruments and up to a complete tone in operas) and the frequency with which the pitch changes over time (e.g. 6 Hz). It is sometimes used synonymously as a combination with another effect: the tremolo, which describes changes in the volume of a note. As it is technically difficult for a human musician to play pure vibrato or tremolo, both effects are usually performed simultaneously. Because of the resulting fluctuations in loudness and frequency, it is very hard for onset detection algorithms to correctly distinguish between new note onsets and an intended variation of the note.

So far only very few publications have addressed the problem of spuriously detected onsets in vibrato music. Collins [28] uses a vibrato suppression stage in his pitch-based onset detection method that first

1 State of the art detection algorithms achieve F-measure values greater than 0.95 on percussive sounds in the annual MIREX evaluation.

2 http://nema.lis.illinois.edu/nema_out/mirex2012/results/aod/

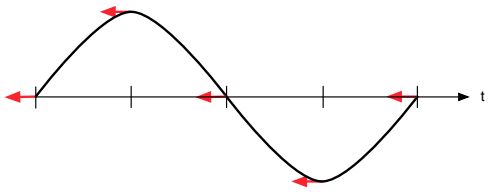
identifies vibrato regions which fluctuate by at most one semitone around the center frequency and collects the extrema in a list. The region is then expanded gradually in time to cover the whole duration of the vibrato. After having identified the complete extent of the vibrato, all values within this window are replaced by the mean of the extrema list. The onset detection function is based on the concept of stable pitches and uses the changes in pitches as cues for new onsets.

Schleusing et al. [118] deploy a system based on the inverse correlation of $N$ consecutive spectral frames centred around the current location. Regions of stable pitch lead to low inverse correlation values, and pitch changes result in peaks in the detection function. To suppress vibrato, they use a warp compensation which cancels out small pitch changes within the window under consideration, leaving the changes due to onsets mostly untouched.

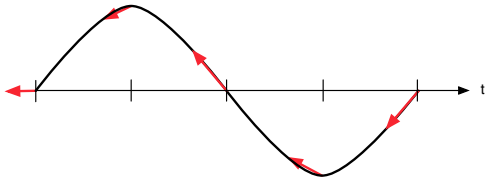
Both systems work only in offline mode because they require future information to reliably detect the vibrato and apply their counter-measures. Furthermore, they can be used only for pitched non-percussive music and are unsuitable for all other kinds of audio material. Glover et al. [61] described a linear prediction post-processing technique that can be applied to existing online onset detection algorithms and is not limited to pitched instruments. Although not designed specifically for vibratos, it is related because it improves mostly the recall performance of the investigated onset detection algorithms.

In this paper, we concentrate on vibrato suppression methods which can be applied both to online (i.e. real-time processing of a continuous audio stream with minimal latency) and offline onset detection scenarios. As a basis for our research we chose the *LogFiltSpecFlux* method proposed by Böck et al. [13], which is the current non-probabilistic state-of-the-art onset detection method.[3] It operates in the spectral domain; more specifically, it only considers the magnitude spectrogram without incorporating any phase information. Like the common *Spectral Flux* algorithm [98] it relies on the detection of positive changes in the energy over time, but instead of calculating the difference from the same bin of a previous frame (see Figure 4.1a), it includes a special trajectory-tracking stage. A general approach to trajectory tracking is shown in Figure 4.1b and illustrates the ability of this method to suppress spurious positive energy fragments (which are falsely detected as new onsets by the spectral flux algorithm) because it calculates the difference along the trajectory path. The new method incorporates a maximum filter (Figure 4.1c) to track the trajectory in a computationally efficient and simple but effective way.

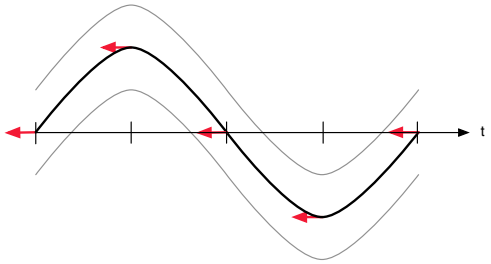
3 http://nema.lis.illinois.edu/nema_out/mirex2012/results/aod/




(a) Classical bin-wise difference calculation




(b) Trajectory tracking-based difference calculation




(c) Maximum filter-based difference calculation

Figure 4.1: (a) the problem of difference calculation in vibrato signals inherent in spectral flux-based methods, (b) a general trajectory tracking-based solution, and (c) the proposed maximum filter-based method. Arrows indicate the positions used for difference calculation, with the tails indicating the positions of the minuends and the heads those of the subtrahends. The grey lines in (c) mark the frequency bounds of the regions which are assigned the same magnitude value via the maximum filter.

## 4.2 PROPOSED METHOD

Our method adds a spectral trajectory-tracking stage to the common spectral flux (SF) [98] algorithm. The system processes the signal in a frame-wise manner. Thus the signal is divided into overlapping chunks of length $N = 2048$ samples, and each frame is weighted with a Hann window of the same length before being transformed to the spectral domain via the discrete Fourier transform (DFT).

The original spectral flux implementation uses the temporal evolution of the magnitude spectrogram $|X(n,k)|$ by calculating the bin-wise

difference between two consecutive short-time spectra and then sums all positive deviations [98]:

$$SF(n) = \sum_{k=1}^{k=\frac{N}{2}} H\left(|X(n,k)| - |X(n-1,k)|\right) \tag{4.1}$$

with $H(x) = \frac{x+|x|}{2}$ being the half-wave rectifier function, $n$ the frame number and $k$ the frequency bin index.

The problem of the difference calculation in signals containing vibrato can be seen in Figure 4.2b. Many spectral peaks appear if the difference between a frequency bin and the same frequency bin $k$ of the previous frame $n-1$ is calculated. The result of our maximum filter-based trajectory-tracking approach is illustrated in Figure 4.1c and described below.

### 4.2.1 *Pre-processing*

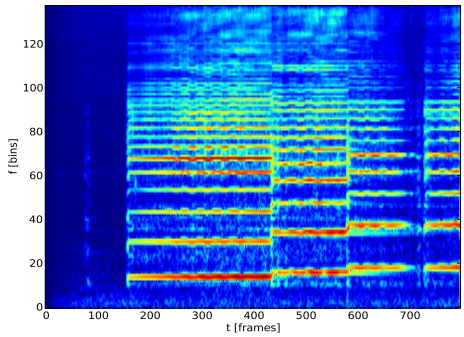
To facilitate trajectory tracking, some pre-processing measures are taken. In general, it is desirable to have a much finer temporal resolution than the standard frame-rate of $f_r = 100\,\text{fps}$ used for onset detection. We thus chose to double the frame-rate so that we can report onsets with 5 ms accuracy. An increased frame rate has the advantage that the quantised magnitude spectrogram features much smoother trajectories, which simplifies tracking. However (in addition to the higher computational cost) is has the disadvantage that the individual differences (on which the onset detection function is based) are much smaller due to greater overlapping of the windows. Thus, instead of calculating the difference between consecutive frames, we use frames that are further apart, the offset determined by the parameter $\mu$:

$$\mu = \max\left(1, \lfloor \left(N/2 - \min\{n | w(n) > r\}\right) / h + 0.5 \rfloor\right) \tag{4.2}$$

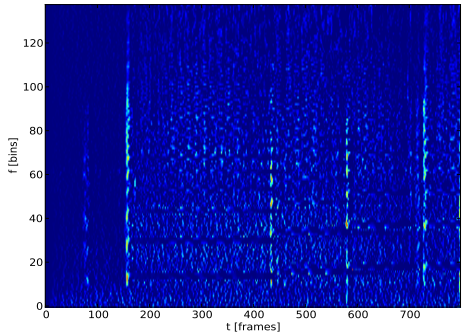
with $r$ being a parameter which defines the height ratio of the window function $w(n)$ with length $N$, and the hop-size $h$ between two frames. The hop-size can be calculated by dividing the sample-rate of the audio signal $f_s$ by the frame-rate $f_r$. The spectral flux onset detection function with the improved difference calculation is given by:

$$SF'(n) = \sum_{k=1}^{k=\frac{N}{2}} H\left(|X(n,k)| - |X(n-\mu,k)|\right) \tag{4.3}$$

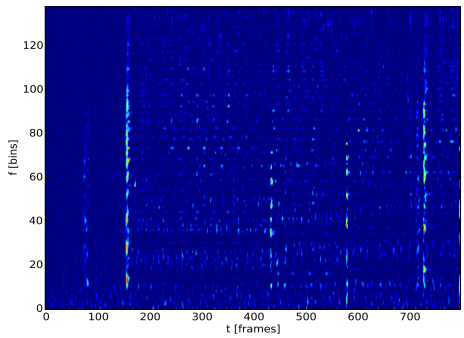
with $\mu \geq 1$. The main advantage of this measure is that the difference values are greater since the overlap of the two windows considered is smaller (because they are located further apart). Values of $r = 0.5$, resulting in $\mu = 2$, were found to yield the best performance for a



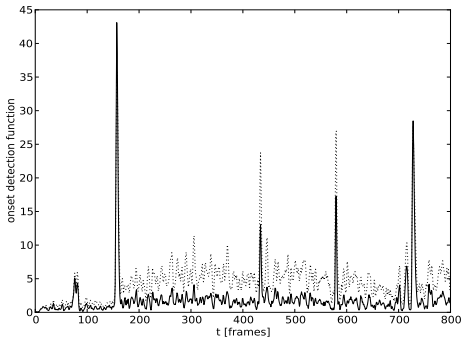

(a) Magnitude spectrogram




(b) Classical bin-wise positive difference




(c) Positive difference with maximum-filtering trajectory tracking




(d) Sum of differences

Figure 4.2: (a) logarithmic magnitude spectrogram of a 4 s violin recording featuring vibrato, and filtered with a quarter-tone filterbank, (b) the positive differences calculated by taking the bin-wise difference between two consecutive spectrogram frames, and (c) with the proposed maximum-filtering trajectory tracking. (d) shows the sum of all positive differences, the dotted line representing the sum of the differences given in (b) (i.e. the method proposed in [13]) and the solid line the sum with the maximum filter applied shown in (c).

frame-rate $f_r = 200\,\text{fps}$ and the standard sample-rate $f_s = 44.1\,\text{kHz}$ of the audio signal. Additionally, the peaks of the resulting onset detection function are much closer to the actual onset positions, which renders lag compensation in the final peak-picking unnecessary.

To simplify trajectory tracking, the linear magnitude spectrogram is filtered with a filterbank $F(k, m)$ with $M = 138$ triangular filters with center frequencies aligned on the western music scale and separated by a quarter-tone from each other, covering a frequency range of $[27.5, 000]\,\text{Hz}$. Operating on a logarithmic frequency scale has the advantage that a constant frequency shift (e.g. by a semi-tone) always results in a shift by the same number of frequency bins (2 if quarter-tone filters are used) independent of the fundamental frequency of a sounding note. Thus, the search range for trajectory tracking is constant, independently of the starting frequency bin $m$.

It has been found to be advantageous (i) to filter the spectrogram first and then take the logarithm of the summed (filtered) magnitude as in [13] (using the same trick of adding 1 before taking the logarithm) and (ii) not to normalise the filters of the filterbank to have equal areas. The logarithmic filtered spectrogram is given by:

$$X_{log,filt}(n, m) = \log_{10}\left(|X(n, k)| \cdot F(k, m) + 1\right) \tag{4.4}$$

with $m$ being the frequency bin index on the quarter-tone frequency scale used.

### 4.2.2 *Trajectory tracking*

The frequency deviation of vibrato in string music is usually $\pm 1$ semitone with an alternation frequency of up to 10 Hz. In operatic singing, the frequency deviation can be much greater, but the alternation frequency is lower, which results in a very similar search space for the tracking of the magnitude trajectories. For the given setting of $f_r = 200\,\text{fps}$ and a quarter-tone filtered spectrogram, a search space of $m = \pm 1$ frequency bins over $\mu = 2$ consecutive time frames covers the expected fluctuations.

Below we present two methods we investigated. They are superseded by our maximum filter-based approach, described subsequently, which performs as well or better but has a much lower computational complexity.

First we investigated an approach which uses the cross-correlation of two frames to determine the shift in frequency needed to achieve the highest similarity between the two frames. Based on this frequency shift, we calculated the bin-wise differences from the $\mu$-th preceding frame shifted by exactly this lag. The method is similar to that used in [118]. There, the correlation between two consecutive frames of the linearly scaled spectrogram is used to formulate a detection function,

but a special warping method is needed to compensate for the greater frequency spreading at higher frequencies. Using a logarithmic frequency scale as described in the previous section and incorporated in the approach in [124] renders warping for proper cross-correlation calculation between two frames unnecessary. Because both methods use the level of correlation directly as a feature (in [118] weighted by the energy of the signal), they only use frequencies up to 8000 Hz [118] and 3000 Hz [124], respectively, to achieve higher correlation values. Since we use the cross-correlation values only to choose the shift needed for maximum correlation, the frequency range need not be limited.

While this method works perfectly for monophonic pitched non-percussive music, it shows inferior performance when used for mixed audio signals where high energy portions of the signal can impede the vibrato detection based on correlation. Thus, we implemented a more universal approach which works on all kinds of musical signals. A simple trajectory tracking approach was chosen which follows the magnitude trajectory of each frequency bin $m$ backwards in time in $\mu$ individual time-steps. The difference for each bin is then calculated with respect to the magnitude along the trajectory path, as can be seen in Figure 4.1b. Since this approach is computationally expensive, methods with lower complexity were investigated.

A very simple method which performs as well as (or better than) the two aforementioned approaches – independently of the audio material used – incorporates a maximum filter. Maximum filters are commonly used in computer vision and set the value at a given position to the maximum value in its neighbourhood, which is defined by the shape of the filter. We chose the filter shape such that the current frequency bin and its direct neighbours on the logarithmically scaled filtered spectrogram $X_{log,filt}(n,m)$ are covered, but limited to the current time frame. The effect of this maximum filter is a widened trajectory (on the frequency axis), and is shown in Figure 4.1c. The maximum filtered spectrogram is then given by:

$$X^{max}_{log,filt}(n,m) = \max\left(X_{log,filt}(n,m-1:m+1)\right) \tag{4.5}$$

In the final *SuperFlux* detection function, the difference is then calculated with respect to this maximum-filtered spectrogram:

$$SF^*(n) = \sum_{m=1}^{m=M} H\left(X_{log,filt}(n,m) - X^{max}_{log,filt}(n-\mu,m)\right) \tag{4.6}$$

The effect of the measures described above is clearly visible in Figure 4.2c, which plots the positive difference (calculated to the next to last frame) with maximum-filtering trajectory tracking of the 4-second recording of a violin played with vibrato shown in Figure 4.2a. Compared to the standard spectral flux difference calculation approach

(Figure 4.2b), it clearly shows fewer positive energy components in the regions played with vibrato. Figure 4.2d plots the sums of the two difference calculation approaches shown above. The solid line represents the *SuperFlux* detection function according to Equation 4.6, the dotted line the standard spectral flux algorithm (applied to the filtered logarithmic spectrogram given in Equation 4.4). This approach is described in [13] and serves as a state-of-the-art spectral flux implementation for evaluation in Section 4.3. Although the peaks of the *SuperFlux* detection function are sometimes a bit lower if the new notes are played slurred (e.g. onsets around frame numbers 430 and 580), the overall detection function has a much lower noise floor caused by vibrato. The remaining ripple is mostly due to variations in loudness, for instance effects intended by the player (e.g. tremolo) or a natural loudness fluctuation while playing vibrato.

### 4.2.3 *Peak-picking*

We use the peak-picking method described in [13] to select the final onsets of the *SuperFlux* detection function. This method is simple and suitable for both offline and online settings. In online mode (i.e. when reading an incoming audio stream) no future information is available, and thus only past information can be used. A frame $n$ of the *SuperFlux* onset detection function $SF^*(n)$ is selected as an onset if it fullfils the following three conditions:

1. $SF^*(n) = \max\left(SF^*(n - pre\_max : n + post\_max)\right)$,
2. $SF^*(n) \geq mean(SF^*(n - pre\_avg : n + post\_avg)) + \delta$,
3. $n - n_{previous\,onset} > combination\_width$,

where $\delta$ is the tuneable threshold. The other parameters were chosen to yield the best performance on the complete dataset. Specifically, $pre\_max = 30\,\text{ms}$, $post\_max = 30\,\text{ms}$, $pre\_avg = 100\,\text{ms}$, $post\_avg = 70\,\text{ms}$, and $combination\_width = 30\,\text{ms}$ achieved good overall results. Parameter values must first be converted into frames depending on the frame-rate $f_r$ used. For peak picking in online mode, *post_max* and *post_avg* are set to 0.

## 4.3 EVALUATION

We used a variety of datasets and settings in our evaluation to maximise comparability with published methods.

### 4.3.1 *Datasets*

The biggest dataset used for evaluation is that described in [13], which consists mostly of mixed audio material covering different types of

musical genres, performed on various instruments. It includes the sets used in [6], [74], and [55]. The 321 files have a total length of approximately 102 minutes and have 27,774 annotated onsets (25,927 if all onsets within 30 ms are combined). The main purpose of this set is to show how the new *SuperFlux* algorithm performs on a general-purpose dataset. This dataset is hereafter referred to as *Böck*. Based on this set, we built a subset covering only the violin and cello recordings played with vibrato. These 16 files have 849 onsets.

To compare with the current state-of-the-art algorithm for pitched non-percussive music presented in [118], we use the authors' dataset. However, not all sound files and annotations could be used for evaluation, since the authors could provide only part of this set. As the available dataset contains 75% of the original dataset (7,677 instead of 9,717 onsets) and an identical distribution of the different playing styles (50% contain vibrato, some staccato etc.), we are confident that the results obtained are nonetheless comparable. We call this the *Wang* dataset.

To investigate our algorithm's ability to suppress the vibrato in operatic singing, a third dataset (called the *Opera* dataset) consisting of solo singing rehearsal recordings of a Haydn opera was used. The recordings were made at the *Ars Electronica Future Lab* in Linz, Austria. The set covers both male and female singers and has a total length of 10 minutes, containing 1,448 onsets.

### 4.3.2 *Performance measures and evaluation settings*

The performance of onset detection methods is commonly evaluated by means of Precision, Recall, and F-measure. If a detected onset is within the evaluation window around an annotated ground truth onset location, it is considered to be correct. However, every detected onset can only match once, and thus any detected onset within the evaluation window of two different annotated onsets counts as one true positive and one false negative (a missed onset). The same applies to annotations: all additionally reported onsets within the evaluation window of an annotation are counted as false positive detections. For better comparability with other results, we match the evaluation parameters as follows:

Our standard setting is that used in [13], which combines all annotated onsets within 30 ms to a single onset and uses an evaluation window of $\pm$ 25 ms to identify correctly detected onsets. Thus, the *combination_width* parameter of our peak-picking is also set to 30 ms.

The second set of parameters (used for the evaluation of the *Wang* dataset) uses the same settings as in [118], where all onsets within 50 ms are combined (i.e. *combination_width* $=$ 50 ms) and an evaluation window of $\pm$ 70 ms is used.

Unless otherwise noted, all results were obtained by swiping the threshold parameter $\delta$ of the peak-picking stage and choosing the value that maximises the F-measure on the respective dataset.

### 4.3.3 *Results and discussion*

In order to demonstrate that the *SuperFlux* algorithm is a good all-round performer which not only suppresses false positive onsets in music with vibrato, but also performs on the same level as state-of-the-art methods, we tested it against various other onset detection algorithms.

#### *Competitors*

For comparison, we chose the four best-performing online and offline onset detection methods among those submitted to MIREX 2012.[4] We consider these submissions the state of the art, since they achieved the highest ever F-measures in the *MIREX* evaluation. *OnsetDetector.2012* is an improved version of the method originally proposed in [55], which shows superior performance in offline scenarios. Together with its online variant, *OnsetDetectorLL* [11], it belongs to the group of probabilistic approaches. Since both were trained on the complete *Böck* dataset (cf. Section 4.3.1), results given for these algorithms were obtained with 8-fold cross-validation and parameter tuning on the training subset. The *LogFiltSpecFlux* [13] algorithm uses no probabilistic information and thus is much less computationally demanding. It can be used both in online and offline scenarios and marks the upper bound of performance "simple" algorithms are able to achieve to date.

#### *Böck set*

The results for the full *Böck* dataset are given in Table 4.1. In online mode, the new *SuperFlux* algorithm clearly outperforms the *LogFiltSpecFlux* method [13] on which it is based, and it closes the gap to the reference *OnsetDetectorLL* neural network-based approach [11].

An important aspect of the results is the shift of the new method towards higher recall values (and thus a more balanced ratio with respect to precision). Although the algorithm does not detect more onsets per se, suppressing spurious onsets has the very favourable side effect of allowing a lower overall threshold to be chosen for the peak-picking stage, which leads, in turn, to a higher recall rate without too many additional false positives.

In offline mode, the overall picture is very similar: all methods performed slightly better than in online mode with the exception of the *OnsetDetector.2012* algorithm, which exhibited superior performance. This is mainly due to the algorithm's ability to model the context

4 http://nema.lis.illinois.edu/nema_out/mirex2012/results/aod/

| ONLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| OnsetDetectorLL [11] | 0.863 | 0.783 | 0.821 |
| LogFiltSpecFlux [13] | 0.854 | 0.753 | 0.801 |
| *SuperFlux* | 0.855 | 0.787 | 0.820 |
| OFFLINE | | | |
| OnsetDetector.2012 [55] | 0.892 | 0.855 | 0.873 |
| LogFiltSpecFlux [13] | 0.877 | 0.756 | 0.812 |
| *SuperFlux* | 0.883 | 0.793 | 0.836 |

Table 4.1: Precision, Recall, and F-measure of different onset detection algorithms using online (upper half) and offline (lower half) settings on the *Böck* dataset. Results for the OnsetDetectorLL [11] and OnsetDetector.2012 [55] algorithms were obtained with 8-fold cross-validation and parameters selected solely on the training set.

of an onset and thus to detect "more difficult" onsets that cannot be found by other methods. Detailed investigations of the remaining false positive detections revealed that *OnsetDetector.2012* recognises the sound of a stopped string and thus does not report an onset in such situations, which results in a higher precision rate. However, this is only possible if future information is available (i.e. only in offline mode) and exploited by the algorithm – which is not the case for the *SuperFlux* since its trajectory tracking is strictly causal, and the offline mode only differs in the peak-picking settings.

Table 4.2 compares *SuperFlux* and *LogFiltSpecFlux* on the basis of the string pieces of the dataset, and highlights the ability of our *SuperFlux* algorithm to successfully suppress false positive detections originating mostly from vibrato. Especially in online mode, the number of false detections decreases from 185 to 118, which is a reduction by 36%. At the same time *SuperFlux* misses fewer notes (263 compared to 294) because of the lower threshold chosen. In offline mode, the number of false positive detections cannot be reduced any further, but a few additional correctly identified onsets lead to slightly improved results compared to the online mode.

For the results in Table 4.2 the parameters were not optimised to give the best F-measure performance on the strings subset; rather, the settings used to obtain the results in Table 4.1 were retained to demonstrate our algorithm's ability to outperform existing approaches on both a general-purpose dataset and string recordings with vibrato without altering settings.

| ONLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| OnsetDetectorLL [11] | 0.822 | 0.676 | 0.742 |
| LogFiltSpecFlux [13] | 0.750 | 0.654 | 0.699 |
| *SuperFlux* | 0.832 | 0.690 | 0.755 |
| OFFLINE | | | |
| OnsetDetector.2012 [55] | 0.834 | 0.820 | 0.827 |
| LogFiltSpecFlux [13] | 0.786 | 0.684 | 0.732 |
| *SuperFlux* | 0.836 | 0.701 | 0.762 |

Table 4.2: Precision, Recall, and F-measure of different onset detection algorithms using online (upper half) and offline (lower half) settings on the strings subset of the *Böck* dataset using the same parameters as used for the results in Table 4.1.

*Wang set*

Table 4.3 shows the performance on violin music on the basis of the *Wang* dataset. The *SuperFlux* method outperforms all other algorithms in terms of false positive detections both in online and offline mode. In comparison to the *LogFiltSpecFlux* method, a reduction in false positives by 61% in online mode and 58% in offline mode can be achieved.

Compared to the algorithm described in [118], which is tuned specifically for pitched non-percussive signals with vibrato, *SuperFlux* is able to achieve the same low level of false positive detections, but increases the number of correctly reported onsets by 3.8%. Since the method of Schleusing et al. [118] works only in offline mode, no results for online mode can be given. Because the results of the *SuperFlux* algorithm performing in online mode are on the same level as this highly specialised algorithm for pitched non-percussive instruments (in offline mode), it can be considered a more universal approach for onset detection.

Interestingly, the methods without any dedicated vibrato suppression (*LogFiltSpecFlux* and *OnsetDetector*) outperform the one proposed by Collins [28], which does include a vibrato suppression stage and is also tuned specifically towards pitched instruments.

Since the recordings in the *Wang* dataset are exclusively solo recordings made in a sound-absorbing room and contain only very few polyphonic parts, we consider the results given in Table 4.2 a much better approximation to real-world examples since they also feature accompanying instruments, which make vibrato tracking and suppression harder. Also, the evaluation criteria chosen are very lax compared to those used for all other results. With the stricter evaluation, the new

| ONLINE | TRUE POSITIVES | FALSE POSITIVES |
|---|---|---|
| OnsetDetectorLL [11] | 92.3% | 20.8% |
| LogFiltSpecFlux [13] | 97.1% | 20.7% |
| *SuperFlux* | 92.7% | 9.6% |
| OFFLINE | | |
| Collins [28] | 62.4% | 24.4% |
| OnsetDetector.2012 [55] | 96.5% | 15.5% |
| LogFiltSpecFlux [13] | 97.0% | 17.8% |
| Schleusing et al. [118] | 91.2% | 9.2% |
| *SuperFlux* | 94.7% | 9.1% |

Table 4.3: True and false positive rates of different onset detection algorithms using online (upper half) and offline (lower half) settings on the Wang dataset. Results for the methods of Collins [28] and Schleusing et al. [118] algorithms were taken from [118]. For evaluation, the same settings as in [118] are used (cf. Section 4.3.2).

*SuperFlux* algorithm achieves true and false positive rates of 89.7% and 22.8% respectively (0.772 Precision, 0.897 Recall, and 0.830 F-measure).

*Opera set*

The last dataset for performance evaluation was the newly created dataset with male and female opera rehearsal recordings. In line with the other results, our method dramatically outperforms the *LogFiltSpecFlux* algorithm and thus closes the gap to probabilistic methods. In the case of online peak picking, the number of false detections decreased from 1198 to 498, which is a reduction by 58%. In offline mode, the false positive rate was reduced by 55%. The recalls of both algorithms are almost identical in both cases.

Note that no opera material was used to train the two neural network-based methods. Only the threshold values for peak picking were adapted to yield the best overall performance. This explains the imbalance of the recall and precision values compared to those of our new method, which exhibits a much better balance.

### 4.3.4 *Runtime*

The new *SuperFlux* algorithm has almost the same low computational complexity as the *LogFiltSpecFlux* method [13] on which it is based. On a single 2.26 GHz core of an Intel Core 2 Duo MacBook Pro, processing of a 60-second audio piece takes 2 seconds (30 times real-time) compared to 1.7 seconds of the same algorithm without any maximum filtering trajectory tracking. This is extremely fast compared to neural

| ONLINE | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| OnsetDetectorLL [11] | 0.588 | 0.744 | 0.657 |
| LogFiltSpecFlux [13] | 0.435 | 0.638 | 0.518 |
| *SuperFlux* | 0.649 | 0.637 | 0.643 |
| OFFLINE | | | |
| OnsetDetector.2012 [55] | 0.576 | 0.777 | 0.662 |
| LogFiltSpecFlux [13] | 0.480 | 0.632 | 0.546 |
| *SuperFlux* | 0.672 | 0.635 | 0.653 |

Table 4.4: Precision, Recall, and F-measure of different onset detection algorithms using online (upper half) and offline (lower half) settings on the Opera dataset.

network-based approaches, which take approximately 14 and 20 seconds (online- and offline-mode). Additionally, they require annotated audio material for training, which takes several hours.

## 4.4 CONCLUSIONS

This paper has presented a new method for vibrato suppression with maximum filtering. Our *SuperFlux* onset detection algorithm is based on the common spectral flux method and is able to reduce the number of false positive detections originating from vibrato by up to 60% compared to current state-of-the-art implementations. It does so without missing any onsets otherwise detected.

In comparison to highly specialised vibrato suppression mechanisms for monophonic pitched music, our method achieves the same precision rate but improves the recall rate by 4%. The same rise in recall rate can be observed on complex polyphonic mixed audio signals. This underlines the universal suitability of the new algorithm.

Since our method's vibrato suppression mechanism is based solely on past information, it can be used in online real-time applications without any fundamental modifications. In online scenarios, it closes the performance gap to the best neural network-based approach but has the advantage of a much lower computational complexity. Because of this low processing demands it can be considered the first choice for a universal onset detection method suitable for all kinds of music. An open-source (BSD-licensed) reference Python implementation of the method can be found at `https://github.com/CPJKU/SuperFlux/`.

### ADDENDUM

The *SuperFlux* algorithm currently acts as the de facto standard for simple onset detection and was adopted by many authors. For instance, it is used in other work as a pre-processing step, as a dedicated onset detection stage, or as the basis for further enhancements [19, 20, 112, 128, 129, 139].

An updated software implementation of this algorithm can be found in the *madmom* package, which is described in Chapter 10.

#### *MIREX evaluation*

The *SuperFlux* onset detection algorithm was submitted and evaluated during MIREX 2013. It was then constantly improved by adjusting the peak-picking parameters based on additionally annotated musical excerpts. These improvements are reflected by the 2014 and 2015 MIREX submissions. With its latest parametrisation, *SuperFlux* shows performance close to the current state-of-the-art without the added complexity of the neural network-based systems.

| ALGORITHM | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| SuperFlux (2013) | 0.883 | 0.787 | 0.812 |
| SuperFlux (2014) | 0.834 | 0.861 | 0.834 |
| SuperFlux (2015) | 0.854 | 0.856 | 0.839 |
| OnsetDetector (2013) | 0.869 | 0.888 | 0.867 |
| CNNOnsetDetector (2016) | 0.860 | 0.899 | 0.873 |

Table 4.5: Performance of the presented *SuperFlux* algorithm in MIREX evaluations compared to state-of-the-art offline onset detection algorithms. The names reflect the names of the executable programs of the *madmom* package (cf. Section 10.2.2).

# Part II

# BEAT AND DOWNBEAT TRACKING

# 5

# BEAT TRACKING

### TITLE

Enhanced Beat Tracking with Context-Aware Neural Networks.

### AUTHORS

Sebastian Böck and Markus Schedl.

### PUBLISHED

In Proceedings of the 14th International Conference on Digital Audio Effects (DAFx), September 2011, Paris, France.

### CONTRIBUTION

All work was carried out by me. Markus Schedl provided valuable input.

### ABSTRACT

We present two new beat tracking algorithms based on the autocorrelation analysis, which showed state-of-the-art performance in the MIREX 2010 beat tracking contest. Unlike the traditional approach of processing a list of onsets, we propose to use a bidirectional Long Short-Term Memory recurrent neural network to perform a frame by frame beat classification of the signal. As inputs to the network the spectral features of the audio signal and their relative differences are used. The network transforms the signal directly into a beat activation function. An autocorrelation function is then used to determine the predominant tempo to eliminate the erroneously detected – or complement the missing – beats. The first algorithm is tuned for music with constant tempo, whereas the second algorithm is further capable to follow changes in tempo and time signature.

## 5.1 INTRODUCTION

For humans, tracking the beat is an almost natural task. We tap our foot or nod our head to the beat of the music. Even if the beat changes, humans can follow it almost instantaneously. Nonetheless, for machines the task of beat tracking is much harder, especially when dealing with varying tempi, as the numerous publications by different authors on this subject suggest.

Locating the beats precisely opens new possibilities for a wide range of music applications, such as automatic manipulation of rhythm, time-stretching of audio loops, beat accurate automatic DJ mixing or self-adapting digital audio effects. Beats are also crucial for analysing the rhythmic structure, and the genre of songs. In addition they help identifying cover songs or estimating the similarity of music pieces.

The remainder of this paper is structured as follows: Section 5.2 gives a short overview over existing methods for beat tracking. Section 5.3 briefly introduces the concept and different types of neural networks with a special emphasis on bidirectional Long Short-Term Memory recurrent neural networks, which are used in the proposed algorithms. Section 5.4 details all aspects of the newly proposed beat tracking algorithms. Results and discussion are given in Section 5.5 and the final section presents conclusions and an outlook to further works.

## 5.2 RELATED WORK

Most methods for beat tracking of audio signals have a working scheme like the one shown in Figure 5.1. After extracting features from the audio signal, they try to determine the periodicity of the signal (the tempo) and the phase of the periodic signal (the beat locations). The features can be for example onsets, chord changes, amplitude envelopes, or spectral features. The choice of a particular feature mostly depends on the subsequent periodicity estimation and phase detection stages. For periodicity estimation, autocorrelation, comb filter, histogram, and multiple agent based induction methods are widely used. Some methods also produce phase information during periodicity estimation, and therefore do not need a phase detection stage to determine the exact position of the beat pulses. Gouyon and Dixon [64] gives a good overview on the subject.

Most of todays top performing beat tracking algorithms rely on onsets as features [35, 50, 106]. Since music signals contain much more onsets than beats, additional processing is needed to locate the beats within the onsets. By transferring this determination of beats into a neural network, less complex post-processing is needed to achieve comparable or better results.

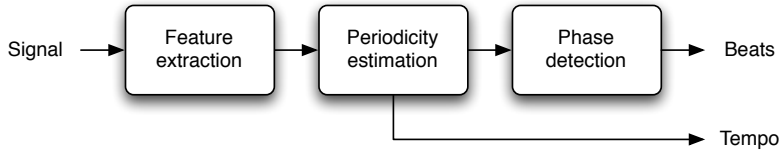



Figure 5.1: Basic workflow of traditional beat tracking methods.

## 5.3 NEURAL NETWORKS

Neural networks have been around for decades and are successfully used for all kind of machine learning tasks.

The most basic approach is the *multilayer perceptron* (MLP) forming a *feed forward neural network* (FNN). It has a minimum of three layers where the input values are fed through one or more hidden layers consisting of neurons with non-linear activation functions. The output values of the last hidden layer are finally gathered in the output nodes. This type of network is a strictly causal one, where the output is calculated directly from the input values.

If cyclic connections in the hidden layers are allowed *recurrent neural networks* (RNN) are formed. They are theoretically able to remember any past value. In practice however, RNNs suffer from the vanishing gradient problem, i.e. input values decay or blow up exponentially over time.

Hochreiter and Schmidhuber [70] proposed the use of *Long Short-Term Memory* (LSTM) units to overcome this problem. Each LSTM unit (depicted in Figure 5.2) has a recurrent connection with weight 1.0 which enables the block to act as a memory cell. Input, output, and forget gates control the content of the memory cell through multiplicative units and are connected to other neurons as usual. If LSTM blocks are used, the network has access to all previous input values.

If not only the past, but also the future context of the input is necessary to determine the output, a number of different strategies can be applied. One is to add a fixed time window to the input, another solution is to add a delay between the input values and the output targets. Both measures have their downsides as they either increase the input vector size considerably or the input values and output targets are displaced from each other.

*Bidirectional recurrent neural networks* (BRNN) [121] offer a more elegant solution to the problem by doubling the number of hidden layers. The input values to the newly created set of hidden layers are presented to the network in reverse temporal order. This offers the advantage that the network not only has access to past input values but can also 'look into the future'.

If bidirectional recurrent networks are used in conjunction with LSTM neurons, a *bidirectional Long Short-Term Memory* (BLSTM) recur-

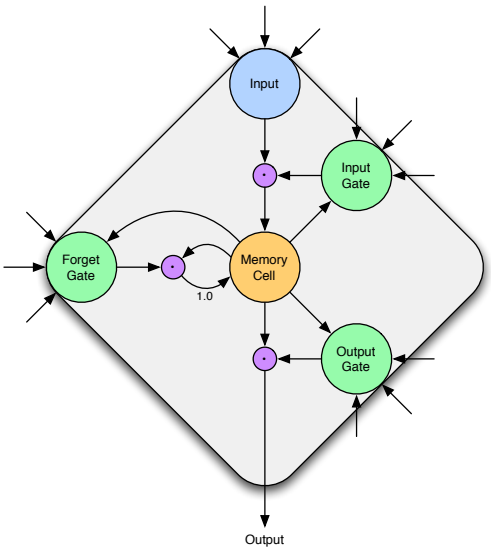



Figure 5.2: LSTM block with memory cell

rent neural network is build. It has the ability to model any temporal context around a given input value. BLSTM networks performed very well in areas like phoneme and handwriting recognition and are described more detailed in [67].

## 5.4 ALGORITHM DESCRIPTION

This section describes our algorithm for beat detection in audio signals. It is based on bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks. Due to their ability to model the temporal context of the input data [67], they perfectly fit into the domain of beat detection. Inspired by the good results for musical onset detection Eyben et al. [55], the approach of this work is used as a basis and extended to suit the needs for audio beat detection by modifying the input representation and adding an advanced peak detection stage.

Figure 5.3 shows the basic signal flow of the proposed system. The audio data is transformed to the frequency domain via three parallel short-time Fourier transforms (STFT) with different window lengths. The obtained magnitude spectra and their first order differences are used as inputs to the BLSTM network, which produces a beat activation function. In the peak detection stage, first the periodicity within this activation function is detected with the autocorrelation function to determine the most dominant tempo. The beats are then aligned according to the previously computed beat interval. We propose two different peak detection algorithms, one tuned for music with constant tempo and beats (*BeatDetector*) and a second one which is able to track tempo changes (*BeatTracker*). The individual blocks are described in more detail in the following sections.

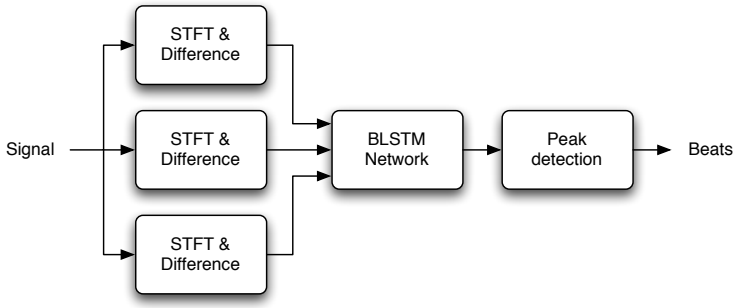



Figure 5.3: Basic signal flow of the presented beat detector / tracker

### 5.4.1 *Feature extraction*

As input, the raw pulse code modulated (PCM) audio signal with a sampling rate of $f_s = 44.1\,\text{kHz}$ is used. To reduce the computational complexity, stereo signals are converted to a monaural signal by averaging both channels. The discrete input audio signal $x(n)$ is segmented into overlapping frames of $W$ samples length. The windows with lengths of 23.2 ms, 46.4 ms, and 92.8 ms (1024, 2048, and 4096 samples respectively) are sampled every 10 ms, resulting in a frame rate $f_r = 100\,\text{fps}$. A standard Hamming window $w(l)$ of the same length is applied to the frames before the STFT is used to compute the complex spectrogram $X(n,k)$

$$X(n,k) = \sum_{l=-\frac{W}{2}}^{\frac{W}{2}-1} w(l) \cdot x(l+nh) \cdot e^{-2\pi jlk/W} \tag{5.1}$$

with $n$ being the frame index, $k$ the frequency bin index, and $h$ the hop size or time shift in samples between adjacent frames. The complex spectrogram is converted to the power spectrogram $S(n,k)$ by omitting the phase portion of the spectrogram by

$$S(n,k) = |X(n,k)|^2 . \tag{5.2}$$

Psychoacoustic knowledge is used to reduce the dimensionality of the resulting magnitude spectra. To this end, a filterbank with 20 triangular filters located equidistantly on the Mel scale is used to transform the spectrogram $S(n,k)$ to the Mel spectrogram $M(n,m)$. To better match the human perception of loudness, a logarithmic representation is chosen (cf. Figure 5.4a):

$$M(n,m) = \log\left(S(n,k) \cdot F(m,k)^T + 1.0\right) \tag{5.3}$$

If large window lengths are used for the STFT, the raise of the magnitude values in the spectrogram occurs early compared to the

actual beat location (cf. Figure 5.4b). Instead of calculating the simple positive first order difference as in [55], a more advanced method is used to overcome this displacement of the actual beat locations compared to the positive first order difference. First a median spectrogram $M_{median}(n,m)$ is obtained according to

$$M_{median}(n,m) = \text{median}\{M(n-l^*,m),\ldots,M(n,m)\} \tag{5.4}$$

with $l^*$ being the length for which the median is calculated. This length depends on the used window size $W$ for the STFT, and is computed as: $l^* = \lfloor W/100 \rfloor$. Both the use of the median and the length of the window were empirically determined during preliminary studies. The positive first order median difference $D^+(n,m)$ is then calculated as

$$D^+(n,m) \quad = \quad H\left(M(n,m) - M_{median}(n,m)\right) \tag{5.5}$$

with $H(x)$ being the half-wave rectifier function $H(x) = \frac{x+|x|}{2}$ (cf. Figure 5.4c). Using only the positive differences as additional inputs to the neural network gave better performance than omitting the differences at all or including both the positive and negative values.

### 5.4.2 *Neural network*

For the neural network stage, a bidirectional recurrent neural network with LSTM units is used. As inputs to the neural network three logarithmic Mel-spectrograms $M_{23}(n,m)$, $M_{46}(n,m)$ and $M_{93}(n,m)$ (computed with window sizes of 23.2 ms, 46.4 ms, and 92.8 ms, respectively) and their corresponding positive first order median differences $D^+_{23}(n,m)$, $D^+_{46}(n,m)$, and $D^+_{93}(n,m)$ are used, resulting in 120 input units. The fully connected network has three hidden layers in each direction, with 25 LSTM units each (6 layers with 150 units in total). The output layer has two units, representing the two classes 'beat' and 'no beat'. Thus the network can be trained as a classifier with the cross entropy error function. The outputs use the softmax activation function, i.e. the output of each unit is mapped to the range $[0,1]$ and their sum is always 1. The output nodes thus represent the probabilities for the two classes.

#### *Network training*

The network is trained as a classifier with supervised learning and early stopping. The used training set consists of 88 audio excerpts taken from the ISMIR 2004 tempo induction contest [1] (also known as the "Ballroom set") with lengths of 10 seconds each, the 26 training

1 http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html

and bonus files from the MIREX 2006 beat tracking evaluation [2] with lengths of 30 seconds, and 6 musical pieces of the set introduced by Bello et al. [6] with lengths from 3 to 15 seconds. Each musical piece is manually beat annotated, marking every quarter note in case of time signature with a denominator of four (i.e. 2/4, 3/4, and 4/4), and the eighth note for all pieces (or parts of pieces) with a time signature of 5/8 or 7/8. The 120 files have a total length of 28.5 minutes and 3,595 annotated beats.

Each audio sequence is preprocessed as described above and presented to the network for learning. The network weights are initialised with random values following a Gaussian distribution with mean 0 and standard deviation 0.1. Standard gradient descent with back-propagation of the errors is used to train the network. To prevent over-fitting, the performance is evaluated after each training iteration on a separate validation set (a 15% randomly chosen disjoint part of the training set). If no improvement is observed for 20 epochs, the training is stopped and the network state with the best performance on the validation set is used onwards.

#### *Network testing*

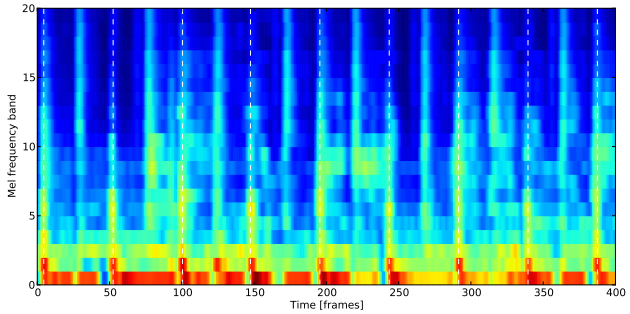
Since the network weights were initialised randomly, five different networks were trained on different sets of the training data. The beat activation functions of the 'beat' output nodes are then averaged and used as input to the following stage (cf. Figure 5.4d). For the evaluation the preprocessed music excerpts are presented to these five previously trained networks.

### 5.4.3 *Peak detection*

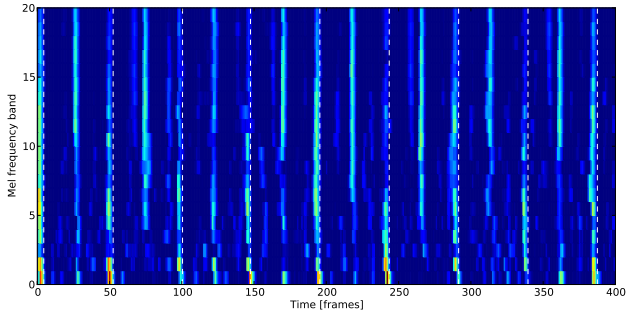
The averaged beat activation function (cf. Figure 5.4d) gives the probability of a beat at each frame. Similar to [55], the function could be used directly to determine the beats by applying a simple threshold. However, a more sophisticated algorithm for peak picking is applied here. It is able to reduce the relatively high number of false positives and negatives even further. This method yields an F-measure value of 0.88 for a 5-fold cross validation on the complete training set, compared to 0.81 achieved using a simple threshold.

If constant tempo is assumed for (a part of) the musical piece, the predominant tempo can be used to eliminate false positive beats, or complement missing false negative ones. The two different proposed peak detection techniques differ only in the length for which a constant tempo is assumed. The *BeatDetector* assumes a constant tempo throughout the whole musical piece, whereas the *BeatTracker* consid-

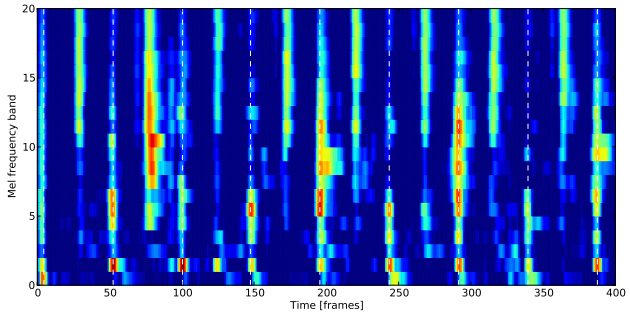
2 http://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking



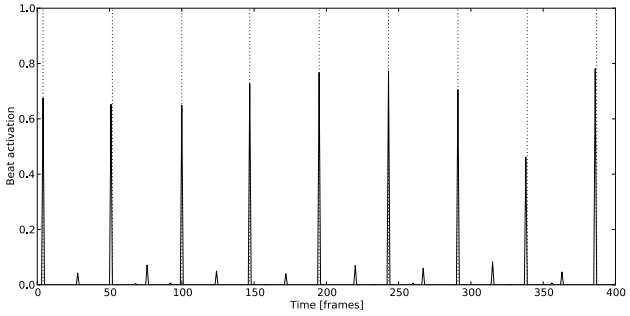

(a) Logarithmic Mel spectrogram with an STFT window of 92.8 ms




(b) Positive first order difference to the preceding frame




(c) Positive first order difference to the median of the last 0.41 s




(d) Beat activation function (output of the neural network stage)

Figure 5.4: Evolution of the signal through the processing steps of the algorithm. It shows a 4 s excerpt from *'Basement Jaxx - Rendez-Vu'*. Beat positions are marked with dashed/dotted vertical lines.

ers only a moving window which covers the next 6 seconds. This modification enables the *BeatTracker* to follow tempo changes.

*Autocorrelation function*

Both proposed algorithms first determine the tempo for the musical piece. The *BeatDetector* uses the entire input signal for calculation, whereas the *BeatTracker* only uses the next 6 seconds relative to the actual starting point. The most dominant beat interval of this segment is used to estimate the tempo. The autocorrelation function (ACF) is calculated on the beat activation function $a_b(n)$ as follows:

$$A(\tau) = \sum_n a_b(n+\tau) \cdot a_b(n) \tag{5.6}$$

The algorithm constrains the possible tempo range of the audio signal from $T_{min} = 40$ to $T_{max} = 220$ given in beats per minute. Thus only values of $A(\tau)$ corresponding to the range from $\tau_{min} = 273\,\text{ms}$ to $\tau_{max} = 1.5\,\text{s}$ are used for calculation. Since music tends to slightly vary in tempo and beats sometimes occur early or late relative to the absolute position of the dominant tempo, the resulting inter beat intervals vary as well. Therefore a smoothing function $s$ is applied to the result of the autocorrelation function $A(\tau)$. A Hamming window with a size of $\tau_t = 150\,\text{ms}$ is used. The size of this window is not crucial, as long as it is wide enough to cover all possible interval fluctuations and remains shorter than the smallest delay $\tau_{min}$ used for the autocorrelation. This results in the smoothed autocorrelation function $A^*(\tau)$:

$$A^*(\tau) = A(\tau) \star s\,(\tau_t) \tag{5.7}$$

*Beat phase detection*

The dominant tempo corresponds to the highest peak of the smoothed autocorrelation function $A^*(\tau)$ at index $\tau^*$. This delay $\tau^*$ is used as the beat interval $i$. The phase of the beat $p^*$ is computed as the highest value of the beat activation function's sum at the possible beat positions for the given interval $i$:

$$p^* = \max_{p=0\ldots i} \sum_k a_b(p + k \cdot i) \tag{5.8}$$

*Peak-picking*

Finally, the beats are represented by the local maxima of the beat activation function. Thus, we use a standard peak search around the locations given by $n_k = p^* + k \cdot i$ calculated with the previously

determined $p^*$. To allow for small timing fluctuations, a deviation factor $d = 0.1 \cdot i$ is introduced and the final beat function $b(n)$ is given by:

$$b(n) = \begin{cases} 1 & \text{for } a_b(n_k - d) \leq a_b(n_k) \geq a_b(n_k + d), \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

The *BeatDetector* determines all beats in this manner. The *BeatTracker* only detects the next beat and moves the beginning of the lookahead window to that beat. Then the dominant tempo estimation and all consecutive steps (Section 5.4.3 to 5.4.3) are performed on the new section of the beat activation function.

## 5.5 EVALUATION

Beat tracking performance was evaluated during the MIREX 2010 beat tracking contest with two different datasets.[3] The first set, the McKinney collection (MCK set) [102] has rather stable tempo. The second collection (MAZ set) consists of Chopin Mazurkas, which are in 3/4 time signature and contain tempo changes.

Both described algorithms outperformed all other contributions on the MCK set. The *BeatDetector* shows a small overall advantage over the *BeatTracker*. Depending on the used performance measure the relative performance gain compared to the next best algorithm is up to 5.7% (F-measure with a detection window of ±70 ms), 6.9% (Cemgil: accuracy based on a Gaussian error function with 40 ms std. dev.), 8.2% (Goto: binary decision based on statistical properties of a beat error sequence), and 4.7 (P-Score: McKinney's impulse train cross-correlation method). Table 5.1 summarizes the results and also includes the best result ever achieved in the MIREX competition by any algorithm as a reference to the state-of-the-art. It can be seen that our *BeatTracker* algorithm performs better or close to it (depending on the used performance measure). This shows the future potential of this approach compared to other signal based ones, given the fact that the actual peak picking algorithm is a rather simple one.

The tempo changes of the MAZ set are the main reason for the *BeatDetector* not performing better (see Table 5.2), as it assumes a constant tempo throughout the whole musical piece. Nonetheless the algorithm performs still reasonably well. As expected, the more flexible *BeatTracker* performs better and ranks second according to F-measure and first according to Cemgil's performance measure. However, the most mentionable aspect is that the neural networks were trained solely on ballroom dance and other kinds of western pop music. Neither a

3 Evaluation measures described at http://www.music-ir.org/mirex/wiki/2010:Audio_Beat_Tracking

| ALGORITHM | F-MEASURE | CEMGIL | GOTO | P-SCORE |
|---|---|---|---|---|
| *BeatTracker* | 0.540 | 0.413 | 0.089 | 0.592 |
| *BeatDetector* | 0.532 | 0.403 | 0.226 | 0.577 |
| GP3 [106] | 0.503 | 0.372 | 0.209 | 0.565 |
| LGG2 [104] | 0.500 | 0.377 | 0.179 | 0.550 |
| TL2 * | 0.420 | 0.299 | 0.025 | 0.506 |
| NW1 [133] | 0.356 | 0.258 | 0.058 | 0.457 |
| MRVCC1 [99] | 0.257 | 0.183 | 0.001 | 0.384 |
| ZTC1 [141] | 0.246 | 0.186 | 0.004 | 0.261 |
| GP1 (2009) [106] | 0.548 | 0.410 | 0.222 | 0.590 |

Table 5.1: Results for the MIREX 2010 beat tracking evaluation on the McKinney (MCK) set. Only the best performing algorithm of other participants are shown; GP1 & GP3: Peeters [106], LGG2: Oliveira et al. [104], TL2: Lee, NW1: Wack and Aylon [133], MRVCC1: Mata-Campos et al. [99], ZTC1: Zhu et al. [141]. Asterisks mark submissions for which no description is available.

classical piece nor piano music was used for training. Furthermore, only one training example actually contained tempo changes. This suggest that even better performance can be expected when trained on music which has properties similar to the MAZ data set.

## 5.6 CONCLUSIONS

This paper presented two novel beat tracking algorithms which perform state-of-the-art although they use a relatively simple and straight forward approach. The *BeatTracker* outperformed all other algorithms in the MIREX 2010 beat tracking contest for the McKinney dataset. Although no classical music was used for training and the training set had less then 3.5 minutes of material with a time signature of 3/4 the new *BeatTracker* performed still reasonably well on the Mazurka test set (all excerpts are in 3/4 time signature). This shows the aptitude of the BLSTM neural network for correctly modelling the temporal context and directly classifying beats. Since the *BeatTracker* shows superior performance over the more simple *BeatDetector* even for musical excerpts with constant tempo, future development will concentrate on this algorithm.

Besides training with a more comprehensive training set, future work should also investigate a possible performance boost by implementing some more advanced beat tracking algorithms in the peak detection stage. Kalman filters [123], particle filters [68], a multiple agents architecture [44] and dynamic programming [50] seem promis-

| ALGORITHM | F-MEASURE | CEMGIL | GOTO | P-SCORE |
|---|---|---|---|---|
| TL2 * | 0.685 | 0.404 | 0.000 | 0.722 |
| *BeatTracker* | 0.587 | 0.518 | 0.000 | 0.579 |
| MRVCC2 [99] | 0.493 | 0.395 | 0.003 | 0.512 |
| GP4 [106] | 0.483 | 0.367 | 0.003 | 0.501 |
| *BeatDetector* | 0.473 | 0.382 | 0.000 | 0.459 |
| LGG2 [104] | 0.415 | 0.307 | 0.000 | 0.435 |
| NW1 [133] | 0.276 | 0.198 | 0.000 | 0.314 |
| ZTC1 [141] | 0.012 | 0.009 | 0.000 | 0.009 |

Table 5.2: Results for the MIREX 2010 beat tracking evaluation on the Mazurka (MAZ) set. Only the best performing algorithm of other participants are shown; TL2: Lee, MRVCC2: Mata-Campos et al. [99], GP4: Peeters [106], LGG2: Oliveira et al. [104], NW1: Wack and Aylon [133], ZTC1: Zhu et al. [141]. Asterisks mark submissions for which no description is available.

ing choices. Another possibility is the inclusion of other input features which haven proven to be effective for identifying beats [66].

### ADDENDUM

All consecutive beat tracking chapters use this work as a basis. However, almost every part has been modified by now, reflecting the findings of this thesis. The signal pre-processing uses a logarithmically spaced filterbank as in Section 3.2.5 instead of Mel-filtering (cf. Equation 5.3). Also, the median filtering in the difference calculation step (Equation 5.4) was removed completely. The output layer of the neural network (as described in Section 5.4.2) is now modelled as a single *sigmoid* unit instead of the *softmax* output layer with two units.[4] All these changes are reflected in the 2013 MIREX submissions. The 2014 submissions replace the old autocorrelation-based tempo model with the new comb filter-based tempo estimation algorithm described in Chapter 8. The 2015 submissions use neural network models trained on a larger training set. The implementation included in *madmom* incorporates all these modifications.

#### *MIREX evaluation*

Table 5.3 lists the results of the presented algorithm in MIREX beat tracking evaluations on the MCK both in its original version and with all the improvements described above. It defines the current state of the art on this test set.

| ALGORITHM | F-MEASURE | CEMGIL | AMLC | AMLT |
|---|---|---|---|---|
| BeatTracker (2010) | 0.545 | 0.413 | 0.317 | 0.493 |
| BeatTracker (2013) | 0.590 | 0.449 | 0.442 | 0.618 |
| BeatTracker (2014) | 0.608 | 0.464 | 0.474 | 0.655 |
| BeatTracker (2015) | 0.639 | 0.488 | 0.538 | 0.705 |
| BeatDetector (2010) | 0.532 | 0.402 | 0.511 | 0.650 |
| BeatDetector (2013) | 0.582 | 0.442 | 0.529 | 0.701 |
| BeatDetector (2014) | 0.613 | 0.467 | 0.525 | 0.716 |
| BeatDetector (2015) | 0.638 | 0.487 | 0.587 | 0.750 |

Table 5.3: Performance of the presented algorithm in MIREX beat tracking evaluations on the MCK set. The results represent the current state of the art in beat tracking on this dataset. The names reflect the names of the executable programs of the *madmom* package (cf. Section 10.2.2).

4 This change does not influence the performance at all, it is just an implementation detail.

# 6

# MULTI-MODEL BEAT TRACKING

### TITLE

A Multi-Model Approach to Beat Tracking Considering Heterogeneous Music Styles.

### AUTHORS

Sebastian Böck, Florian Krebs, and Gerhard Widmer.

### PUBLISHED

In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR), October 2014, Taipei, Taiwan.

### CONTRIBUTION

Main idea, implementation, and experiments were conceived and carried out by me. Florian Krebs contributed the dynamic Bayesian network related part (Section 6.2.4). Gerhard Widmer provided valuable input.

### ABSTRACT

In this paper we present a new beat tracking algorithm which extends an existing state-of-the-art system with a multi-model approach to represent different music styles. The system uses multiple recurrent neural networks, which are specialised on certain musical styles, to estimate possible beat positions. It chooses the model with the most appropriate beat activation function for the input signal and jointly models the tempo and phase of the beats from this activation function with a dynamic Bayesian network. We test our system on three big datasets of various styles and report performance gains of up to 27% over existing state-of-the-art methods. Under certain conditions the system is able to match even human tapping performance.

## 6.1 INTRODUCTION AND RELATED WORK

The automatic inference of the metrical structure in music is a fundamental problem in the music information retrieval field. In this line, *beat tracking* deals with finding the most salient level of this metrical grid, the *beat*. The beat consists of a sequence of regular time instants which usually invokes human reactions like foot tapping. During the last years, beat tracking algorithms have considerably improved in performance. But still they are far from being considered on par with human beat tracking abilities – especially for music styles which do not have simple metrical and rhythmic structures.

Most methods for beat tracking extract some features from the audio signal as a first step. As features, commonly low-level features such as amplitude envelopes [117] or spectral features [17], mid-level features like onsets either in discretised [42, 63] or continuous form [35, 51, 80, 108], chord changes [63, 108] or combinations thereof with higher level features such as rhythmic patterns [87] or metrical relations [60] are used. The feature extraction is usually followed by a stage that determines periodicities within the extracted features sequences. Autocorrelation [17, 49, 63] and comb filters [35, 117] are commonly used techniques for this task. Most systems then determine the most predominant tempo from these periodicities and subsequently determine the beat times using *multiple agents* approaches [42, 63], *dynamic programming* [35, 51], *hidden Markov models (HMM)* [37, 80, 108], or *recurrent neural networks (RNN)* [17]. Other systems operate directly on the input features and jointly determine the tempo and phase of the beats using *dynamic Bayesian networks (DBN)* [25, 68, 87, 135].

One of the most common problems of beat tracking systems are "octave errors", meaning that a system detects beats at double or half the rate of the ground truth tempo. For human tappers this generally does not constitute a problem, as can be seen when comparing beat tracking results at different metrical levels [35]. Hainsworth and Macleod [68] stated that beat tracking systems will have to be style specific in the future in order to improve the state-of-the-art. This is consistent with the finding of Krebs et al. [87] who showed on a dataset of Ballroom music that the beat tracking performance can be improved by incorporating style-specific knowledge, especially by resolving the octave error. While approaches have been proposed which combined multiple existing features for beat tracking [137], no one has so far combined several models specialised on different musical styles to improve the overall performance.

In this paper, we propose a multi-model approach to fuse information of different models that have been specialised on heterogeneous music styles. The model is based on the *recurrent neural network* (RNN) beat tracking system proposed in [17] and can be easily adapted to any music style without further parameter tweaking, only by providing a

corresponding beat-annotated dataset. Further, we propose an additional *dynamic Bayesian network* stage based on the work of Whiteley et al. [135] which jointly infers the tempo and the beat phase from the beat activations of the RNN stage.

## 6.2 PROPOSED METHOD

The new beat tracking algorithm is based on the state-of-the-art approach presented by Böck and Schedl [17]. We extend their system to be able to better deal with heterogeneous music styles and combine it with a dynamic Bayesian network similar to the ones presented in [135] and [87].

The basic structure is depicted in Figure 6.1 and consists of the following elements: first the audio signal is pre-processed and fed into multiple *neural network* beat tracking modules. Each of the modules is trained on different audio material and outputs a different beat activation function when activated with a musical signal. These functions are then fed into a module which chooses the most appropriate model and passes its activation function to a *dynamic Bayesian network* to infer the actual beat positions.

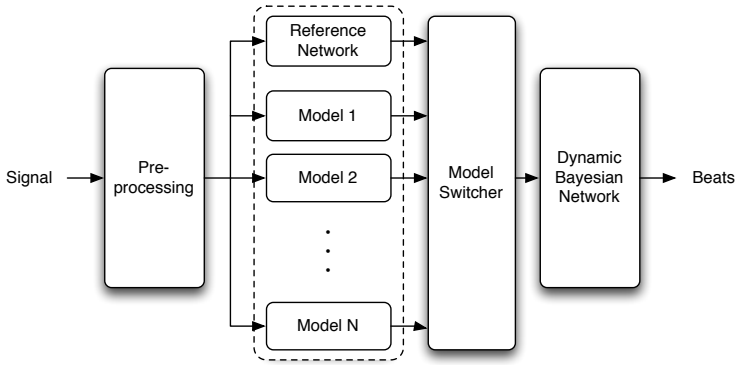



Figure 6.1: Overview of the new multi-model beat tracking system.

Theoretically, a single network large enough should be able to model all the different music styles simultaneously, but unfortunately this optimal solution is hardly achievable. The main reason for this is the difficulty to choose an absolutely balanced training set with an evenly distributed set of beats over all the different dimensions relevant for detecting beats. These include rhythmic patterns [87, 117], harmonic aspects and many other features. To overcome this limitation, we split the available training data into multiple parts. Each part should represent a more homogeneous subset than the whole set so that the networks are able to specialise on the dominant aspects of this subset.

It seems reasonable to assume that humans do something similar when tracking beats [29]. Depending on the style of the music, the rhythmic patterns present, the instrumentation, the timbre, they apply

their musical knowledge to chose one of their "learned" models and then decide which musical events are beats or not. Our approach mimics this behaviour by learning multiple distinct models.

### 6.2.1 *Signal pre-processing*

All neural networks share the same signal pre-processing step, which is very similar to the work in [17]. As inputs to the different neural networks, the logarithmically filtered and scaled spectrograms of three parallel short-time Fourier transforms (STFT) obtained with different window lengths and their positive first order differences are used. The system works with a constant frame rate $f_r$ of 100 frames per second. Window lengths of 23.2 ms, 46.4 ms and 92.9 ms are used and the resulting spectrogram bins of the discrete Fourier transforms are filtered with overlapping triangular filters to have a frequency resolution of three bands per octave. To put all resulting magnitude values into a positive range we add 1 before taking the logarithm.

### 6.2.2 *Multiple parallel neural networks*

At the core of the new approach, multiple neural networks are used to determine possible beat locations in the audio signal. As outlined previously, these networks are trained on material with different music styles to be able to better detect the beats in heterogeneous music styles.

As networks we chose the same *recurrent neural network (RNN)* topology as in [17] with three bidirectional hidden layers with 25 *long short-term memory (LSTM)* units per layer. For training of the networks, standard gradient descent with error backpropagation and a learning rate of $1e^{-4}$ is used. We initialise the network weights with a Gaussian distribution with mean 0 and standard deviation of 0.1. We use early stopping with a disjoint validation set to stop training if no improvement over 20 epochs can be observed.

One reference network is trained on the complete dataset until the stopping criterion is reached for the first time. We use this point during the training phase to diverge the specialised models from the reference network.

Afterwards, all networks are fine-tuned with a reduced learning rate of $1e^{-5}$ on either the complete set or the individual subsets (cf. Section 6.3.1) with the above mentioned stopping criterion. Given $N$ subsets, $N + 1$ models are generated.

The output functions of the network models represent the beat probability at each time frame. Instead of tracking the beats with an autocorrelation function as described in the original work, the beat activation functions of the different models are fed into the next model-selection stage.

### 6.2.3 *Model selection*

The purpose of this stage is to select a model which outputs a better beat activation function than the reference model when activated with a signal. Compared to the reference model, the specialised models produce better predictions on input data which is similar to that used for fine-tuning, but worse predictions on signals dissimilar to the training data. This behaviour can be seen in Figure 6.2, where the specialised model produces higher beat activation values at the beat locations and lower values elsewhere.

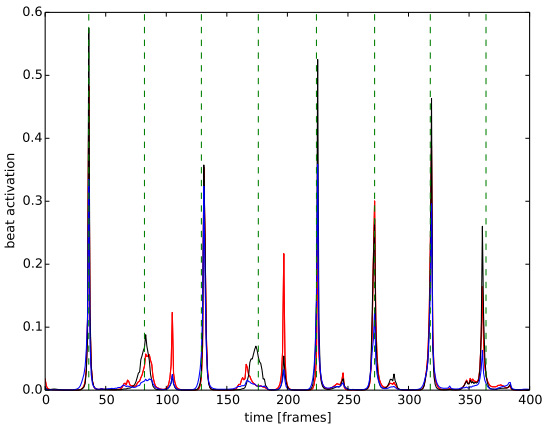



Figure 6.2: Example beat activations for a 4 seconds ballroom snippet. Red is the reference network's activations, black the selected model and blue a discarded one. Green dashed vertical lines denote the annotated beat positions.

Table 6.1 illustrates the impact on the *Ballroom* subset, where the relative gain of the best specialised model compared to the reference model (+1.7%) is lower than the penalties of the other models (−2.3% to −6.3%). The fact that the performance degradation of the unsuitable specialised models is greater than the gain of the most suitable model allows us to use a very simple but effective method to choose the best model.

To select the best performing model, all network outputs of the fine-tuned networks are compared with the output of the reference network (which was trained on the whole training set) and the one yielding the lowest *mean squared difference* is selected as the final one and its output is fed into the final beat tracking stage.

### 6.2.4 *Dynamic Bayesian network*

Independent of whether only one or multiple neural networks are used, the approach of Böck and Schedl [17] has a fundamental shortcoming: the final peak-picking stage does not try to find a global

| | F-MEASURE | CEMGIL | AMLC | AMLT |
|---|---|---|---|---|
| SMC * | 0.834 | 0.807 | 0.664 | 0.767 |
| Hainsworth * | 0.867 | 0.839 | 0.694 | 0.793 |
| Ballroom * | 0.904 | 0.872 | 0.777 | 0.853 |
| Reference | 0.887 | 0.855 | 0.748 | 0.831 |
| Multi-model | 0.897 | 0.866 | 0.759 | 0.841 |

Table 6.1: Performance of differently specialised models (marked with asterisks, fine-tuned on the *SMC*, *Hainsworth* and *Ballroom* subsets, respectively) on the *Ballroom* subset compared to the reference model and the network selected by the multi-model selection stage.

optimum when selecting the final locations of the beats. It rather determines the dominant tempo of the piece (or a segment of certain length) and then aligns the beat positions according to this tempo by simply choosing the best start position and then progressively locating the beats at positions with the highest activation function values in a certain region around the pre-determined position. To allow a greater responsiveness to tempo changes, this chosen region must not be too small. However, this also introduces a weakness to the algorithm, because the tracking stage can easily get distracted by a few misaligned beats and needs some time to recover from this fault. The activation function depicted in Figure 6.2 has two of these spurious detections around frames 100 and 200.

To circumvent this problem, we feed the output of the chosen neural network model into a *dynamic Bayesian network (DBN)* which jointly infers tempo and phase of a beat sequence. Another advantage of this new method is that we are able to model both beat and non-beat states, which was shown to perform superior to the case where only beat states are modelled [37].

The DBN we use is closely related to the one proposed by Whiteley et al. [135], adapted to our specific needs. Instead of modelling whole bars, we only model one beat period which reduces the size of the search space. Additionally we do not model rhythmic patterns explicitly and leave this higher level analysis to the neural networks. This finally leads to a DBN which consists of two hidden variables, the tempo $\omega$ and the position $\phi$ inside a beat period. In order to infer the hidden variables from an audio signal, we have to specify three entities: A *transition model* which describes the transitions between the hidden variables, an *observation model* which takes the beat activations from the neural network and transforms them into probabilities suitable for the DBN, and the *initial distribution* which encodes prior knowledge about the hidden variables. For computational ease we discretise the tempo-beat space to be able to use standard hidden Markov model (HMM) [113] algorithms for inference.

*Transition model*

The beat period is discretised into $\Phi = 640$ equidistant cells and $\phi \in \{1, ..., \Phi\}$. We refer to the unit of the variable $\phi$ (position inside a beat period) as *pib*. $\phi_k$ at audio frame $k$ is then computed by

$$\phi_k = (\phi_{k-1} + \omega_{k-1} - 1) \bmod \Phi + 1. \tag{6.1}$$

The tempo space is discretised into $\Omega = 23$ equidistant cells, which cover the tempo range up to 215 beats per minute (BPM). The unit of the tempo variable $\omega$ is *pib per audio frame*. As we want to restrict $\omega$ to integer values (to stay within the $\phi$ grid at transitions), we need a high resolution of $\phi$ in order to get a high resolution of $\omega$. Based on experiments with the training set, we set the tempo space to $\omega \in \{6, ..., \Omega\}$, where $\omega = 6$ is equivalent to a minimum tempo of $6 \times 60 \times f_r/\Phi \approx 56$ BPM. As in [135] we only allow for three tempo transitions at time frame $k$: It stays constant, it accelerates, or it decelerates.

$$\omega_k = \begin{cases} \omega_{k-1}, & P(\omega_k|\omega_{k-1}) = 1 - p_\omega \\ \omega_{k-1} + 1, & P(\omega_k|\omega_{k-1}) = \frac{p_\omega}{2} \\ \omega_{k-1} - 1, & P(\omega_k|\omega_{k-1}) = \frac{p_\omega}{2} \end{cases} \tag{6.2}$$

Transitions to tempi outside of the allowed range are not allowed by setting the corresponding transition probabilities to zero. The probability of a tempo change $p_\omega$ was set to 0.002.

*Observation model*

Since the beat activation function $a$ produced by the neural network is limited to the range $[0, 1]$ and shows high values at beat positions and low values at non-beat positions, we use the activation function directly as state-conditional observation distributions, similar to [37]. We define the observation likelihood as

$$P(a_k|\phi_k) = \begin{cases} a_k, & 1 \leq \phi_k \leq \frac{\Phi}{\lambda} \\ \frac{1-a_k}{\lambda-1}, & \textit{otherwise}. \end{cases} \tag{6.3}$$

$\lambda \in [\frac{\Phi}{\Phi-1}, \Phi]$ is a parameter that controls the proportion of the beat interval which is considered as beat and non-beat location. Smaller values of $\lambda$ (a higher proportion of beat locations and a smaller proportion of non-beat locations) are especially important for higher tempi, as the DBN visits only a few position states of a beat interval and could possibly miss the beginning of a beat. On the other hand, higher values of $\lambda$ (a smaller proportion of beat locations) lead to less accurate beat tracking, as the activations are blurred in the state domain of the DBN. On our training set we achieved the best results with the value $\lambda = 16$.

*Initial state distribution*

The initial state distribution is normally used to incorporate any prior knowledge about the hidden states, such as tempo distributions. In this paper, we use a uniform distribution over all states, for simplicity and ease of generalisation.

*Inference*

We are interested in the sequence of hidden variables $\phi_{1:K}$ and $\omega_{1:K}$, that maximise the posterior probability of the hidden variables given the observations (activations $a_{1:K}$). Combining the discrete states of $\phi$ and $\omega$ into one state vector $\mathbf{x}_k = [\phi_k, \omega_k]$, we can compute the maximum a-posteriori state sequence $\mathbf{x}^*_{1:K}$ by

$$\mathbf{x}^*_{1:K} = \arg\max_{\mathbf{x}_{1:K}} p(\mathbf{x}_{1:K}|a_{1:K}). \tag{6.4}$$

Equation 6.4 can be computed efficiently using the well-known Viterbi algorithm [113]. Finally the set of beat times $\mathcal{B}$ are determined by the set of time frames $k$ which were assigned to a beat position ($\mathcal{B} = \{k : \phi_k < \phi_{k-1}\}$). In our experiments we found that the beat detection becomes less accurate if the part of the beat interval which is considered as beat-state is too large (i.e. smaller values of $\lambda$). Therefore we determine the final beat times by looking for the highest beat activation value inside the beat-state window $\mathcal{W} = \{k : \phi_k \leq \frac{\Phi}{\lambda}\}$.

## 6.3 EVALUATION

For the development and evaluation of the algorithm we used some well-known datasets. This allows for highest comparability with previously published results of state-of-the-art algorithms.

### 6.3.1 *Datasets*

As training material for our system, the datasets introduced in [65, 68, 72] are used. They are called *Ballroom*, *Hainsworth* and *SMC* respectively. To show the ability of our new algorithm to adapt to various music styles, a very simple approach of splitting the complete dataset into multiple subsets according to the original source was chosen. Although far from optimal – both the *SMC* and *Hainsworth* datasets contain heterogeneous music styles – we still consider this a valid choice, since any "better" splitting would allow the system to adapt even further to heterogeneous styles and in turn lead to better results. At least the three sets have a somehow different focus regarding the music styles present.

### 6.3.2 *Performance measures*

In line with almost all other publications on the topic of beat tracking, we report the following scores:

F-MEASURE
: Counts the number of true positive (correctly located beats within a tolerance window of ±70 ms), false positive and negative detections.

P-SCORE
: Measures the tracking accuracy by the correlation of the detections and the annotations, considering deviations within 20% of the annotated beat interval as correct.

CEMGIL
: Places a Gaussian function with a standard deviation of 40 ms around the annotations and then measures the tracking accuracy by summing up the scores of the detected beats on this function normalising it by the overall length of the annotations or detections, whichever is greater.

CMLC & CMLT
: Measure the longest continuous segment (CMLc) or all correctly tracked beats (CMLt) at the correct metrical level. A beat is considered correct if it is reported within a 17.5% tempo and phase tolerance, and the same applies for the previously detected beat.

AMLC & AMLT
: Same calculation as CMLc and CMLt, but additionally allow offbeat and double/half as well as triple/third tempo variations of the annotated beats.

D & $D_G$
: The information gain (D) and global information gain ($D_g$) are phase agnostic measures comparing the annotations with the detections (and vice-versa) building a error histogram and then calculating the Kullback-Leibler divergence w.r.t. a uniform histogram.

A more detailed description of the evaluation methods can be found in [33]. Since we only investigate offline algorithms, we do not skip the first five seconds for evaluation.

### 6.3.3 *Results and discussion*

Table 6.2 and 6.3 (continuity scores) list the performance results of the reference implementation, Böck's *BeatTracker.2013*, and the various

extensions proposed in this paper for all datasets. All results are obtained with 8-fold cross validation with previously defined splittings, ensuring that no pieces are used both for training or parameter tuning and testing purposes. Additionally, we compare our new approach to published stat-of-the-art results on the *Hainsworth* and *Ballroom* datasets.

| BALLROOM | $F_1$ | P | CEM | D | $D_G$ |
|---|---|---|---|---|---|
| BeatTracker.2013 [17] | 0.887 | 0.863 | 0.855 | 3.404 | 2.596 |
| — Multi-Model | 0.897 | 0.875 | **0.866** | **3.480** | **2.674** |
| — DBN | 0.903 | 0.876 | 0.838 | 3.427 | 2.275 |
| — Multi-Model + DBN | **0.910** | **0.881** | 0.845 | 3.469 | 2.352 |
| Krebs et al. [87] | 0.855 | 0.839 | 0.772 | 2.499 | 1.681 |
| Zapata et al. [137] † | 0.767 | 0.735 | 0.672 | 2.750 | 1.187 |
| HAINSWORTH | | | | | |
| BeatTracker.2013[17] | 0.832 | 0.843 | 0.712 | 2.167 | 1.468 |
| — Multi-Model | 0.832 | 0.847 | **0.716** | 2.171 | **1.490** |
| — DBN | **0.843** | **0.867** | 0.711 | 2.251 | 1.481 |
| — Multi-Model + DBN | 0.840 | 0.865 | 0.707 | **2.268** | 1.466 |
| Zapata et al. [137] | 0.710 | 0.732 | 0.589 | 2.057 | 0.880 |
| SMC | | | | | |
| BeatTracker.2013 [17] | 0.497 | 0.598 | 0.402 | 1.263 | 0.416 |
| — Multi-Model | 0.514 | 0.617 | **0.415** | 1.324 | 0.467 |
| — DBN | 0.516 | 0.622 | 0.404 | 1.426 | 0.504 |
| — Multi-Model + DBN | **0.529** | **0.630** | **0.415** | **1.460** | **0.531** |
| Zapata et al. [137] † | 0.369 | 0.460 | 0.285 | 0.879 | 0.126 |

Table 6.2: Performance of the proposed algorithm on the *Ballroom* [65], *Hainsworth* [68] and *SMC* [72] datasets. *BeatTracker* is the reference implementation our *Multi-Model* and *dynamic Bayesian network (DBN)* extensions are built on. The results marked with † are obtained with Essentia's implementation of the multi-feature beat tracker.[1] ‡ denotes causal (i.e. online) processing, all listed algorithms use non-causal analysis (i.e. offline processing). Best results in bold.

*Multi-model extension*

As can be seen, the use of the *multi-model* extension almost always improves the results over the implementation it is based on, especially on the *SMC* set. The gain in performance on the *Ballroom* set was expected, since Krebs et al. already showed that modelling rhythmic

| BALLROOM | CMLC | CMLT | AMLC | AMLT |
|---|---|---|---|---|
| BeatTracker.2013 [17] | 0.719 | 0.795 | 0.748 | 0.831 |
| — Multi-Model | 0.740 | 0.814 | 0.759 | 0.841 |
| — DBN | 0.792 | 0.825 | 0.873 | 0.915 |
| — Multi-Model + DBN | **0.800** | **0.830** | **0.885** | **0.924** |
| Krebs et al. [87] | 0.745 | 0.786 | 0.818 | 0.865 |
| Zapata et al. [137] † | 0.586 | 0.607 | 0.824 | 0.860 |
| HAINSWORTH | | | | |
| BeatTracker.2013[17] | 0.618 | 0.756 | 0.655 | 0.807 |
| — Multi-Model | 0.617 | 0.761 | 0.652 | 0.809 |
| — DBN | **0.696** | **0.808** | 0.759 | **0.883** |
| — Multi-Model + DBN | **0.696** | 0.803 | **0.760** | 0.881 |
| Zapata et al. [137] | 0.569 | 0.642 | 0.709 | 0.824 |
| Davies and Plumbley [35] | 0.548 | 0.612 | 0.681 | 0.789 |
| Peeters and Papadopoulos [108] | 0.547 | 0.628 | 0.703 | 0.831 |
| Degara et al. [37] | 0.561 | 0.629 | 0.719 | 0.815 |
| Human tapper [35] ‡ | 0.528 | 0.812 | 0.575 | 0.874 |
| SMC | | | | |
| BeatTracker.2013 [17] | 0.238 | 0.360 | 0.279 | 0.436 |
| — Multi-Model | 0.257 | 0.389 | 0.296 | 0.467 |
| — DBN | 0.294 | 0.415 | 0.378 | 0.550 |
| — Multi-Model + DBN | **0.296** | **0.428** | **0.383** | **0.567** |
| Zapata et al. [137] † | 0.115 | 0.158 | 0.239 | 0.397 |

Table 6.3: Continuity performance of the proposed algorithm on the *Ballroom* [65], *Hainsworth* [68] and *SMC* [72] datasets. *BeatTracker* is the reference implementation our *Multi-Model* and *dynamic Bayesian network (DBN)* extensions are built on. The results marked with † are obtained with Essentia's implementation of the multi-feature beat tracker.[1] ‡ denotes causal (i.e. online) processing, all listed algorithms use non-causal analysis (i.e. offline processing). Best results in bold.

patterns helps to increase the overall detection accuracy [87]. Although we did not split the set according to the individual rhythmic patterns, the overall style of ballroom music can be considered unique enough to be distinct from the other music styles present in the other sets and the salient features can be exploited successfully by the multi-model approach.

1 http://essentia.upf.edu, v2.0.1

*Dynamic Bayesian network extension*

As already indicated in the original paper [17] (and described earlier in Section 6.2.4), the original *BeatTracker* can be easily distracted by some misaligned beats and then needs some time to recover from any failure. The newly adapted dynamic Bayesian network beat tracking stage does not suffer from this shortcoming by searching for the globally best beat locations. The use of the DBN boosts the performance on all datasets for almost all evaluation measures. Interestingly, the Cemgil accuracy is degraded by using the DBN stage. This might be explained by the fact that the discretisation grid of the beat period beat positions becomes too coarse for low tempi (cf. Section 6.2.4) and therefore yields inaccurate beat detections, which especially affect the Cemgil accuracy. This is one of the issues that needs to be resolved in the future, especially for lower tempi where the penalty is the highest.

*Comparison with other methods*

Our new system set side by side with other state-of-the-art algorithms draws a clear picture. It outperforms all of them considerably – independently of the dataset and evaluation measure chosen. Especially the high performance boosts of the CMLc and CMLt scores on the *Hainsworth* dataset highlight the ability to track the beats at the correct metrical level significantly more often than any other method.

Davies and Plumbley [35] also list performance results of a human tapper on the same dataset. However it must be noted that these were obtained by online real-time tapping, hence they cannot be compared directly to the system presented. However, the system of Davies and Plumbley [35] can also be switched to causal mode (and thus being comparable to a human tapper). In this mode it achieved performance reduced by approximately 10% [35]. Adding the same amount to the reported tapping results of 0.528 CMLc and 0.575 AMlc suggests that our system is capable of performing as good as humans when continuous tapping is required.

On the *Ballroom* set we achieve higher results than the particularly specialised system of Krebs et al. [87]. Since our DBN approach is a simplified variant of their model, it can be assumed that the relatively low scores of the Cemgil accuracy and the information gain are due to the same reason – the coarse discretisation of the beat or bar states. Nonetheless, comparing the continuity scores (which have higher tolerance thresholds) we can still report an average increase in performance of more than 5%.

## 6.4 CONCLUSIONS

In this paper we have presented a new beat tracking system which is able to improve over existing algorithms by incorporating multiple

models which were trained on different music styles and combining it with a dynamic Bayesian network for the final inference of the beats. The combination of these two extensions yields a performance boost – depending on the dataset and evaluation measures chosen – of up to 27% relative, matching human tapping results under certain conditions. It outperforms other state-of-the-art algorithms in tracking the beats at the correct metrical level by 20%.

We showed that the specialisation on a certain musical style helps to improve the overall performance, although the method for splitting the available data into sets of different styles and then selecting the most appropriate model is rather simple. For the future we will investigate more advanced techniques for the selection of suitable data for the creation of the specialised models, e.g. splitting the datasets according to dance styles as performed by Krebs et al. [87] or applying unsupervised clustering techniques. We also expect better results from more advanced model selection methods. One possible approach could be to feed the individual model activations to the dynamic Bayesian network and let it choose among them.

Finally, the Bayesian network could be tuned towards using a finer beat positions grid and thus reporting the beats at more appropriate times than just selecting the position of the highest activation reported by the neural network model.

### ADDENDUM

The system presented shows its strengths especially on music with hard-to-track beats, like the SMC set. Since it was also trained on this set, the MIREX results for this set are biased and therefore we do not show them here. Non-overfitted results can be found in the previous chapter. It should be noted however, that we used the SMC set for training before it was added as a test set for the MIREX evaluation. We decided to keep it in the training set, since it adds valuable complexity to the training set which is otherwise really scarce and hard to find.

Table 6.4 gives results for the easier-to-track MCK set, together with the best results achieved so far (depending on the evaluation metric used). It can be seen that the presented systems performs state-of-the-art, although its strengths are not highlighted by the dataset. The 2015 submissions utilise the more efficient state space model proposed by Krebs et al. [88] for the dynamic Bayesian network.

| ALGORITHM | F-MEASURE | CEMGIL | AMLC | AMLT |
|---|---|---|---|---|
| MMBeatTracker (2014) | 0.613 | 0.467 | 0.525 | 0.716 |
| MMBeatTracker (2015) | 0.620 | 0.472 | 0.543 | 0.728 |
| DBNBeatTracker (2014) [2] | - | - | - | - |
| DBNBeatTracker (2015) | 0.636 | 0.484 | 0.567 | 0.749 |
| BeatTracker (2015) | 0.639 | 0.488 | 0.538 | 0.705 |
| BeatDetector (2015) | 0.638 | 0.487 | 0.587 | 0.750 |

Table 6.4: Performance of the presented algorithm in MIREX evaluations on the MCK set. The names reflect the names of the executable programs of the *madmom* package (cf. Section 10.2.2).

The system of Krebs et al. [86] uses this beat tracker as its first stage to determine the positions of the beats and to compute the beat-synchronous features for the downbeat tracking stage.

2 Unfortunately, no MIREX evaluation was performed for this submission in 2014.

# 7

# JOINT BEAT AND DOWNBEAT TRACKING

### TITLE

Joint Beat and Downbeat Tracking with Recurrent Neural Networks.

### AUTHORS

Sebastian Böck, Florian Krebs, and Gerhard Widmer.

### PUBLISHED

In Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR), August 2016, New York, USA.

### CONTRIBUTION

Main idea, implementation, and experiments were conceived and carried out by me. Florian Krebs contributed the theoretical background for the dynamic Bayesian network (Section 7.3.3). Gerhard Widmer provided valuable input.

### ABSTRACT

In this paper we present a novel method for jointly extracting beats and downbeats from audio signals. A recurrent neural network operating directly on magnitude spectrograms is used to model the metrical structure of the audio signals at multiple levels and provides an output feature that clearly distinguishes between beats and downbeats. A dynamic Bayesian network is then used to model bars of variable length and align the predicted beat and downbeat positions to the global best solution. We find that the proposed model achieves state-of-the-art performance on a wide range of different musical genres and styles.

## 7.1 INTRODUCTION

Music is generally organised in a hierarchical way. The lower levels of this hierarchy are defined by the *beats* and *downbeats* which define the *metrical structure* of a musical piece. While considerable amount of research focused on finding the *beats* in music, far less effort has been made to track the *downbeats*, although this information is crucial for a lot of higher level tasks such as structural segmentation and music analysis and applications like automated DJ mixing. In western music, the *downbeats* often coincide with chord changes or harmonic cues, whereas in non-western music the start of a *measure* is often defined by the boundaries of rhythmic patterns. Therefore, many algorithms exploit one or both of these features to track the *downbeats*.

## 7.2 RELATED WORK

Klapuri et al. [80] proposed a system which jointly analyses a musical piece at three time scales: the tatum, tactus, and measure level. The signal is split into multiple bands and then combined into four accent bands before being fed into a bank of resonating comb filters. Their temporal evolution and the relation of the different time scales are modelled with a probabilistic framework to report the final position of the downbeats.

The system of Davies and Plumbley [34] first tracks the beats and then calculates the Kullback-Leibler divergence between two consecutive band-limited beat synchronous spectral difference frames to detect the downbeats, exploiting the fact that lower frequency bands are perceptually more important.

Papadopoulos and Peeters [105] jointly track chords and downbeats by decoding a sequence of (pre-computed) beat synchronous chroma vectors with a hidden Markov model (HMM). Two time signatures are modelled. In a later paper, Peeters and Papadopoulos [108] jointly model beat phase and downbeats while the tempo is assumed to be given. Beat and downbeat times are decoded using a HMM from three input features: the correlation of the local energy with a beat-template, chroma vector variation, and the spectral balance between high and low frequency content.

The system proposed by Khadkevich et al. [78] uses impulsive and harmonic components of a reassigned spectrogram together with chroma variations as observation features for a HMM. The system is based on the assumption that downbeats mostly occur at location with harmonic changes.

Hockman et al. [71] present a method designed specifically for hardcore, jungle, and drum and bass music, that often employ breakbeats. The system exploits onset features and periodicity information from a

beat tracking stage, as well as information from a regression model trained on the breakbeats specific to the musical genre.

Durand et al. [47] first estimates the time signature by examining the similarity of the frames at the beat level – with the beat positions given as input. The downbeats are then selected by a linear support vector machine (SVM) model using a bag of complementary features, comprising chord changes, harmonic balance, melodic accents and pattern changes. In consecutive works [45, 46] they lifted the requirement of the beat positions to be given and enhanced their system considerably by replacing the SVM feature selection stage by several deep neural networks which learn higher level representations from which the final downbeat positions are selected by means of Viterbi decoding.

Krebs et al. [87] jointly model bar position, tempo, and rhythmic patterns with a dynamic Bayesian network (DBN) and apply their system to a dataset of ballroom dance music. Based on their work, Holzapfel et al. [73] developed a unified model for metrical analysis of Turkish, Carnatic, and Cretan music. Both models were later refined by using a more sophisticated state space proposed by Krebs et al. [88].

The same state space has also been successfully applied to the beat tracking system proposed by Böck et al. [14]. The system uses a recurrent neural network (RNN) similar to the one proposed in [17] to discriminate between beats an non-beats at a frame level. A DBN then models the tempo and the phase of the beat sequence.

In this paper, we extend the RNN-based beat tracking system in order to jointly track the whole metrical cycle, including beats and downbeats. The proposed model avoids hand-crafted features such as harmonic change detection [45–47, 78, 105], or rhythmic patterns [71, 73, 87], but rather learns the relevant features directly from the spectrogram. We believe that this is an important step towards systems without cultural bias, as postulated by the "Roadmap for Music Information Research" [122].

## 7.3 ALGORITHM DESCRIPTION

The proposed method consists of a recurrent neural network (RNN) similar to the ones proposed in [14, 17], and is trained to jointly detect the beats and downbeats of an audio signal in a supervised classification task. A dynamic Bayesian network is used as a post-processing step to determine the globally best sequence through the state-space by jointly inferring the meter, tempo, and phase of the (down-)beat sequence.

### 7.3.1 *Signal pre-processing*

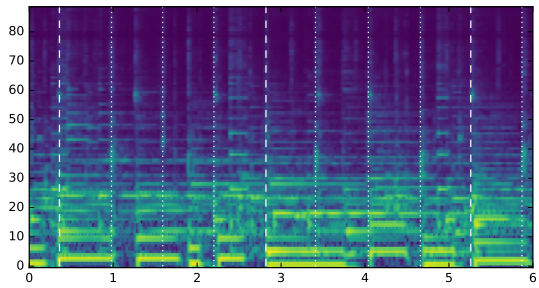
The audio signal is split into overlapping frames and weighted with a Hann window of same length before being transferred to a time-frequency representation with the discrete Fourier transform (DFT). Two adjacent frames are located 10 ms apart, which corresponds to a rate of 100 fps (frames per second). We omit the phase portion of the complex spectrogram and use only the magnitudes for further processing. To enable the network to capture features which are precise both in time and frequency, we use three different magnitude spectrograms with short-time Fourier transform (STFT) lengths of 1024, 2048, and 4096 samples (at a signal sample rate of 44.1 kHz). To reduce the dimensionality of the features, we limit the frequencies range to [30, 17000] Hz and process the spectrograms with logarithmically spaced filters. A filter with 12 bands per octave corresponds to semitone resolution, which is desirable if the harmonic content of the spectrogram should be captured. However, using the same number of bands per octave for all spectrograms would result in an input feature of undesirable size. We therefor use filters with 3, 6, and 12 bands per octave for the three spectrograms obtained with 1024, 2028, and 4096 samples, respectively, accounting for a total of 157 bands. To better match human perception of loudness, we scale the resulting frequency bands logarithmically. To aid the network during training, we add the first order differences of the spectrograms to our input features. Hence, the final input dimension of the neural network is 314. Figure 7.1a shows the part of the input features obtained with 12 bands per octave.

### 7.3.2 *Neural network processing*

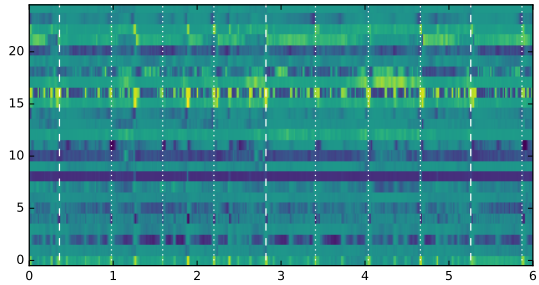
As a network we chose a system similar to the one presented in [17], which is also the basis for the current state-of-the-art in beat tracking [14, 83].

#### *Network topology*

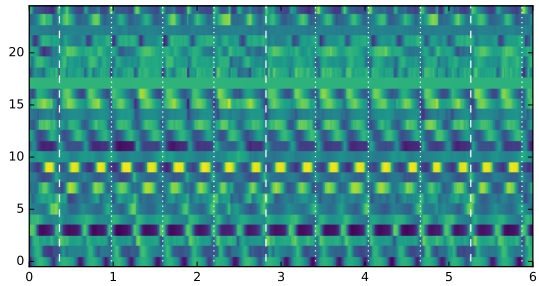
The network consists of three fully connected bidirectional recurrent layers with 25 Long Short-Term Memory (LSTM) units each. Figure 7.1b to 7.1d show the output activations of the forward (i.e. half of the bidirectional) hidden layers. A softmax classification layer with three units is used to model the *beat*, *downbeat*, and *non-beat* classes. A frame can only be classified as downbeat *or* beat but not both at the same time, enabling the following dynamic Bayesian network to infer the meter and downbeat positions more easily. The output of the neural network are three activation functions $b_k$, $d_k$, and $no_k$, which represents the probability of a frame $k$ being a beat but no downbeat,



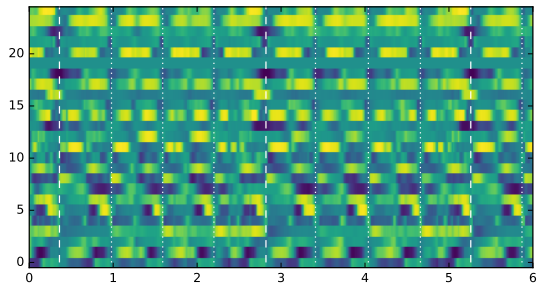

(a) Spectrogram of audio signal



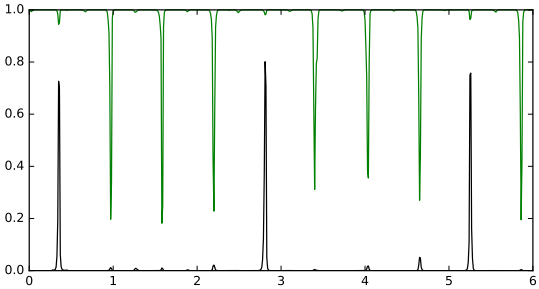

(b) Activations of the first hidden layer




(c) Activations of the second hidden layer




(d) Activations of the third hidden layer




(e) Activations of the softmax output layer

Figure 7.1: Signal propagation of a 6 second song excerpt in 4/4 time signature through the network: (a) part of the neural network input, (b) the first hidden layer shows activations at onset positions, (c) the second hidden layer models mostly faster metrical levels (e.g. 1/8th notes at neuron 3), (d) the third hidden layer models multiple metrical levels (e.g. neuron 8 firing at beat positions and neuron 16 around downbeat positions), (e) the softmax output layer finally models the relation of the different metrical levels resulting in clear downbeat (black) and beat (green, flipped for better visualisation) activations.

downbeat or non-beat position. Figure 7.1e shows $b_k$ and $d_k$ for an audio example.

*Network training*

We train the network on the datasets described in Section 7.4.1 — except the ones marked with an asterisk (*) which are used for testing only — with 8-fold cross validation based on a random splits. We initialise the network weights and biases with a uniform random distribution with range [-0.1, 0.1] and train it with stochastic gradient decent minimising the cross entropy error with a learning rate of $10^{-5}$ and 0.9 momentum. We stop training if no improvement on the validation set can be observed for 20 epochs. We then reduce the learning rate by a factor of ten and retrain the previously best model with the same early stopping criterion.

*Network output thresholding*

We experienced that the very low activations at the beginning and end of a musical excerpt can hurt the tracking performance of the system. This is often the case if a song starts with a (musically irrelevant) intro or has a long fade out at the end. We thus threshold the activations and use only the activations between the first and last time they exceed the threshold. We empirically found a threshold value $\theta = 0.05$ to perform well without harming pieces with overall low activations (e.g. choral works).

### 7.3.3 *Dynamic Bayesian network*

We use the output of the neural network as observations of a *dynamic Bayesian network (DBN)* which jointly infers the meter, tempo, and phase of a (down-)beat sequence. The DBN is very good at dealing with ambiguous RNN observations and finds the global best state sequence given these observations.[1] We use the state-space proposed in [88] to model a whole bar with an arbitrary number of beats per bar. We do not allow meter changes throughout a musical piece, thus we can model different meters with individual, independent state spaces. All parameters of the DBN are tuned to maximise the downbeat tracking performance on the validation set.

*State space*

We divide the state space into discrete states $\mathbf{s}$ to make inference feasible. These states $s(\phi, \dot{\phi}, r)$ lie in a three-dimensional space indexed by the bar position state $\phi \in \{1..\Phi\}$, the tempo state $\dot{\phi} \in \{1..\dot{\Phi}\}$, and the

1 The average performance gain of the DBN compared to simple thresholding and peak-picking of the RNN activations is about 15% F-measure on the validation set.

time signature state $r$ (e.g. $r \in \{3/4, 4/4\}$). States that fall on a *downbeat* position ($\phi = 1$) constitute the set of *downbeat* states $\mathcal{D}$, all states that fall on a *beat* position define the set of *beat* states $\mathcal{B}$. The number of bar-position states of a tempo $\dot{\phi}$ is proportional to its corresponding beat period $1/\dot{\phi}$, and the number of tempo states depends on the tempo ranges that the model accounts for. For generality, we assume equal tempo ranges for all time signatures in this paper but this could easily changed to adapt the model towards specific styles. In line with [88] we find that by distributing the tempo states logarithmically across the beat intervals, the size of the state space can be reduced efficiently without affecting the performance too much. Empirically we found that using $N = 60$ tempo states is a good compromise between computation time and performance.

*Transition model*

Tempo transitions are only allowed at the beats and follow the same exponential distribution proposed in [88]. We investigated "peephole" transitions from the end of every beat back to the beginning of the bar, but found them to harm performance. Thus, we assume that there are no transitions between time signatures in this paper.

*Observation model*

We adapted the observation model of the DBN from [14] to not only predict beats, but also downbeats. Since the activation functions ($d$, $b$) produced by the neural network are limited to the range $[0, 1]$ and show high values at beat/downbeat positions and low values at non-beat positions (cf. Figure 7.1e), the activations can be converted into state-conditional observation distributions $P(o_k|s_k)$ by

$$P(o_k|s_k) = \begin{cases} d_k & s_k \in \mathcal{D} \\ b_k & s_k \in \mathcal{B} \\ \frac{n_k}{\lambda_o - 1}, & otherwise \end{cases} \tag{7.1}$$

where $\mathcal{D}$ and $\mathcal{B}$ are the sets of downbeat and beat states respectively, and the observation lambda $\lambda_o \in [\frac{\Phi}{\Phi-1}, \Phi]$ is a parameter that controls the proportion of the beat/downbeat interval which is considered as beat/downbeat and non-beat locations inside one beat/downbeat period. On our validation set we achieved the best results with the value $\lambda_o = 16$. We found it to be advantageous to use both $b_k$ and $d_k$ as provided by the neural network instead of splitting the probability of $b_k$ among the $N$ beat positions of the transition model.

*Initial state distribution*

The initial state distribution can be used to incorporate any prior knowledge about the hidden states, such as meter and tempo distributions. In this paper, we use a uniform distribution over all states.

*Inference*

We are interested in the sequence of hidden states $\mathbf{s}_{1:K}$, that maximise the posterior probability of the hidden states given the observations (activations of the network). We obtain the maximum a-posteriori state sequence $\mathbf{s}^*_{1:K}$ by

$$\mathbf{s}^*_{1:K} = \arg\max_{\mathbf{s}_{1:K}} p(\mathbf{s}_{1:K}|o_{1:K}) \tag{7.2}$$

which can be computed efficiently using the well-known Viterbi algorithm.

*Beat and downbeat selection*

The sequence of beat **B** and downbeat times **D** are determined by the set of time frames $k$ which were assigned to a beat or downbeat state:

$$\mathbf{B} = \{k : s^*_k \in \mathcal{B}\} \tag{7.3}$$

$$\mathbf{D} = \{k : s^*_k \in \mathcal{D}\} \tag{7.4}$$

After having decided on the sequences of beat and downbeat times we further refine them by looking for the highest beat/downbeat activation value inside a window of size $\Phi/\lambda_o$, i.e. the beat/downbeat range of the whole beat/downbeat period of the observation model (Section 7.3.3).

## 7.4 EVALUATION

In line with almost all other publications on the topic of downbeat tracking, we report the F-measure ($F_1$) with a tolerance window of $\pm 70$ ms.

### 7.4.1 *Datasets*

For training and evaluation we use diverse datasets as shown in Table 7.1. Musical styles range from pop and rock music, over ballroom dances, modern electronic dance music, to classical and non-western music.

We do not report scores for all sets used for training, since comparisons with other works are often not possible due to different

| DOWNBEAT TRACKING DATASETS | # FILES | LENGTH |
|---|---|---|
| Ballroom [65, 87] [2] | 685 | 5 h 57 m |
| Beatles [33] | 180 | 8 h 09 m |
| Hainsworth [68] | 222 | 3 h 19 m |
| HJDB [71] | 235 | 3 h 19 m |
| RWC Popular [62] | 100 | 6 h 47 m |
| Robbie Williams [39] | 65 | 4 h 31 m |
| Rock [26] | 200 | 12 h 53 m |
| Carnatic [126] | 176 | 16 h 38 m |
| Cretan [73] | 42 | 2 h 20 m |
| Turkish [125] | 93 | 1 h 33 m |
| GTZAN [96, 132] * | 999 | 8 h 20 m |
| Klapuri [80] * [3] | 320 | 4 h 54 m |
| BEAT TRACKING DATASETS | | |
| SMC [72] * | 217 | 2 h 25 m |
| Klapuri [80] * [3] | 474 | 7 h 22 m |

Table 7.1: Overview of the datasets used for training and evaluation of the algorithm. Sets marked with asterisks (*) are held-out datasets for testing only.

evaluation metrics and/or datasets. Results for all datasets, including additional metrics can be found online at the supplementary website http://www.cp.jku.at/people/Boeck/ISMIR2016.html which also includes an open source implementation of the algorithm.

### 7.4.2 *Results and Discussion*

Table 7.2 to 7.4 list the results obtained by the proposed method compared to current and previous state-of-the-art algorithms on various datasets. We group the datasets into different tables for clarity, based on whether they are used for testing only, cover western, or non-western music. Since our system jointly tracks beats and downbeats, we compare with both downbeat and beat tracking algorithms.

First of all, we evaluate on completely unseen data. We use the recently published beat and downbeat annotations for the *GTZAN* dataset, the *Klapuri*, and the *SMC* set (built specifically to comprise hard-to-track musical pieces) for evaluation. Results are given in Ta-

2 We removed the 13 duplicates identified by Bob Sturm: http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html

3 The beat and downbeat annotations of this set were made independently, thus they do not necessarily correspond to each other.

ble 7.2. Since these results are directly comparable (the only exception being the results of Durand et al. on the *Klapuri* set[4] and of Böck et al. on the *SMC* set[5]), we perform statistical significance tests on them. We use Wilcoxon's signed-rank test with a p-value of 0.01.

| GTZAN | $F_1$ BEAT | $F_1$ DOWNBEAT |
|---|---|---|
| NEW (bar lengths: 3, 4) * | 0.856 | 0.640 |
| Durand et al. [46] * | - | 0.624 |
| Böck et al. [14] * | 0.864 | - |
| Davies and Plumbley [34] * | 0.806 | 0.462 |
| Klapuri et al. [80] * | 0.706 | 0.309 |
| KLAPURI | | |
| NEW (bar lengths: 3, 4) * | 0.811 | 0.745 |
| Durand et al. [46] ‡ | - | 0.689 |
| Böck et al. [14] * | 0.798 | - |
| Davies and Plumbley [34] * | 0.698 | 0.528 |
| Klapuri et al. [80] ‡ | 0.704 | 0.483 |
| SMC | | |
| NEW (bar lengths: 3, 4) * | 0.516 | n/a |
| Böck et al. [14] § | 0.529 | n/a |
| Davies and Plumbley [34] * | 0.337 | n/a |
| Klapuri et al. [80] * | 0.352 | n/a |

Table 7.2: Beat and downbeat tracking F-measure comparison with state-of-the-art algorithms on the test datasets. ‡ denotes overlapping train and test sets, § cross validation, and * testing only.

Additionally, we report the performance on other sets commonly used in the literature, comprising both western and non-western music. For western music, we give results on the *Ballroom*, *Beatles*, *Hainsworth*, and *RWC Popular* sets in Table 7.3. For non-western music we use the *Carnatic*, *Cretan*, and *Turkish* datasets and group the results in Table 7.4. Since these sets were also used during development and training of our system, we report results obtained with 8-fold cross validation. Please note that the results given in Table 7.3 and 7.4 are not directly comparable because they were either obtained via cross validation, leave-one-dataset-out evaluation, with overlapping train and test sets, or tested on unseen data. However, we still consider them to be a good indicator for the overall performance and capabilities of the systems. For the music with non-western rhythms and meters (e.g. Carnatic art

4 40 out of the 320 tracks were used for training.
5 The complete set was used for training.

music contains 5/4 and 7/4 meters) we compare only with algorithms specialised on this type of music, since other systems typically fail completely on them.

| BALLROOM | $F_1$ BEAT | $F_1$ DOWNBEAT |
|---|---|---|
| NEW (bar lengths: 3, 4) § | 0.938 | 0.863 |
| Durand et al. [46] †/‡ | - | 0.778 / 0.797 |
| Krebs et al. [88] § | 0.919 | - |
| Böck et al. [14] § | 0.910 | - |
| BEATLES | | |
| NEW (bar lengths: 3, 4) § | 0.918 | 0.832 |
| Durand et al. [46] †/‡ | - | 0.815 / 0.842 |
| Böck et al. [14] * | 0.880 | - |
| HAINSWORTH | | |
| NEW (bar lengths: 3, 4) § | 0.867 | 0.684 |
| Durand et al. [46] †/‡ | - | 0.657 / 0.664 |
| Böck et al. [14] § | 0.843 | - |
| Peeters and Papadopoulos [108] | 0.630 | |
| RWC POPULAR | | |
| NEW (bar lengths: 3, 4) § | 0.943 | 0.861 |
| Durand et al. [46] †/‡ | - | 0.860 / 0.879 |
| Böck et al. [14] * | 0.877 | - |
| Peeters and Papadopoulos [108] | 0.840 | 0.800 |

Table 7.3: Beat and downbeat tracking F-measure comparison with state-of-the-art algorithms on western music datasets. † denotes leave-one-set-out evaluation, ‡ overlapping train and test sets, § cross validation, and * testing only.

*Beat tracking*

Compared to the current state-of-the-art [14], the new system performs on par or outperforms this dedicated beat tracking algorithm. It only falls a bit behind on the *GTZAN* and *SMC* sets. However, the results on the latter might be a bit biased, since Böck et al. [14] obtained their results with 8-fold cross validation. Although the new system performs better on the *Klapuri set*, the difference is not statistically significant. All results compared to those of other beat tracking algorithms on the test datasets in Table 7.2 are statistically significant.

Although the new algorithm and [14] have a very similar architecture and were trained on almost the same development sets (the

| CARNATIC | $F_1$ BEAT | $F_1$ DOWNBEAT |
|---|---|---|
| NEW (bar lengths: 3, 4) § | 0.804 | 0.365 |
| —— (bar lengths: 3, 5, 7, 8) § | 0.792 | 0.593 |
| Krebs et al. [88] § | 0.805 | 0.472 |
| CRETAN | | |
| NEW (bar lengths: 3, 4) § | 0.982 | 0.605 |
| —— (bar lengths: 2, 3, 4) § | 0.981 | 0.818 |
| —— (bar lengths: 2) § | 0.980 | 0.909 |
| Krebs et al. [88] § | 0.912 | 0.774 |
| TURKISH | | |
| NEW (bar lengths: 3, 4) § | 0.740 | 0.495 |
| —— (bar lengths: 4, 8, 9, 10) § | 0.777 | 0.631 |
| —— (tempo: 55..300 bpm) § | 0.818 | 0.683 |
| Krebs et al. [88] § | 0.826 | 0.639 |

Table 7.4: Beat and downbeat tracking F-measure comparison with state-of-the-art algorithms on non-western music datasets. —— denotes the same system as the line above with altered parameters in parentheses, § cross validation.

new one plus those sets given in Table 7.1, except the *SMC* dataset), it is hard to conclude whether the new algorithm performs better sometimes because of the additional – more diverse – training material or due to the joint modelling of beats and downbeats. Future investigations with the same training sets should shed some light on this question, but it is safe to conclude that the joint training on beats and downbeats does not harm the beat tracking performance at all.

On non-western music the results are in the same range as the ones obtained by the method of Krebs et al. [88], an enhanced version of the algorithm proposed by Holzapfel et al. [73]. Our system shows almost perfect beat tracking results on the *Cretan* lap dances while performing a bit worse on the *Turkish* music.

*Downbeat tracking*

From Table 7.2 to 7.4, it can be seen that the proposed system not only does well for beat tracking, but also shows state-of-the-art performance in downbeat tracking. We outperform all other methods on all datasets – except *Beatles* and *RWC Popular* when comparing to the overfitted results obtained by the system of Durand et al. [46] – even the systems designed specifically for non-western music. We find this striking, since our new system is not designed specifically for a certain music

style or genre. The results of our method w.r.t. the other systems on the test datasets in Table 7.2 are all statistically significant.

It should be noted however, that the dynamic Bayesian network must model the needed bar lengths for the respective music in order to achieve this performance. Especially when dealing with non-western music, this is crucial. However, we do not consider this a drawback, since the system is able to chose the correct bar length reliably by itself.

*Meter selection*

As mentioned above, for best performance the DBN must model measures with the correct number of beats per bar. Per default, our system works for 3/4 and 4/4 time signatures, but since the parameters of the DBN are not learnt, this can be changed during runtime in order to model any time signature and tempo range.

To investigate the system's ability to automatically decide on which bar length to select, we performed an experiment and limited the DBN to model only bars with lengths of three or four beats, both time signatures simultaneously (the default setting), or bar lengths of up to eight beats.

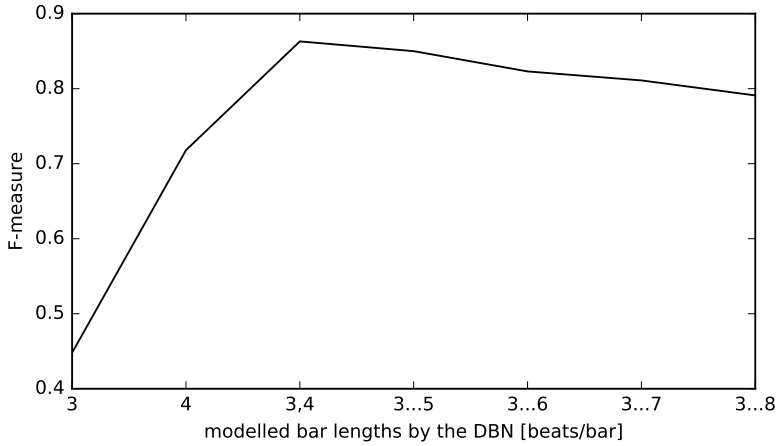



Figure 7.2: Downbeat tracking performance of the new system with different bar lengths on the *Ballroom* set.

Figure 7.2 shows this exemplarily for the *Ballroom* set, which comprises four times as many pieces in 4/4 as in 3/4 time signature. The performance is relatively low if the system is limited to model bars with only three or four beats per bar. When being able to model both time signatures present in the music, the system achieves it's maximum performance. The performance then slightly decreases if the DBN models bars with a length up to eight beats per bar, but remains

on a relatively high performance level. This shows the system's ability to select the correct bar length automatically.

## 7.5 CONCLUSIONS

In this paper we presented a novel method for jointly tracking beats and downbeats with a recurrent neural network (RNN) in conjunction with a dynamic Bayesian network (DBN). The RNN is responsible for modelling the metrical structure of the musical piece at multiple interrelated levels and classifies each audio frame as being either a beat, downbeat, or no beat. The DBN then post-processes the probability functions of the RNN to align the beats and downbeats to the global best solution by jointly inferring the meter, tempo, and phase of the sequence. The system shows state-of-the-art beat and downbeat tracking performance on a wide range of different musical genres and styles. It does so by avoiding hand-crafted features such as harmonic changes, or rhythmic patterns, but rather learns the relevant features directly from audio. We believe that this is an important step towards systems without any cultural bias. We provide a reference implementation of the algorithm as part of the open-source *madmom* [12] framework.

Future work should address the limitation of the system of not being able to perform time signature changes within a musical piece. Due to the large state space needed this is intractable right now, but particle filters as used in [89] should be able to resolve this issue.

### ADDENDUM

Since this algorithm was introduced just recently, no further improvement can be reported so far. However, the results initiated working on an online real-time capable version of the algorithm.

#### *MIREX evaluation*

The system was submitted to the 2016 MIREX downbeat estimation evaluation.[6] It achieved the highest F-measure scores on six out of eight datasets with a margin as high as 19% absolute. It ranks 2nd on the *Beatles* dataset after the system of Durand et al. [46] (0.865 vs. 0.872 F-measure) and the *GTZAN* dataset behind the system of Krebs et al. [86] (0.638 vs. 0.647 F-measure) – which incorporates the beat tracking algorithm described in the previous Chapter. It can thus safely be concluded that it is the current state-of-the-art system for downbeat tracking.

6 http://www.music-ir.org/mirex/wiki/2016:Audio_Downbeat_Estimation_Results

# Part III

# TEMPO ESTIMATION

# 8

# TEMPO ESTIMATION

### TITLE

Accurate Tempo Estimation based on Recurrent Neural Networks and Resonating Comb Filters.

### AUTHORS

Sebastian Böck, Florian Krebs, and Gerhard Widmer.

### PUBLISHED

In Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR), October 2015, Malaga, Spain.

### CONTRIBUTION

Main idea, implementation, and experiments were carried out by me. Florian Krebs and Gerhard Widmer provided valuable input.

### ABSTRACT

In this paper we present a new tempo estimation algorithm which uses a bank of resonating comb filters to determine the dominant periodicity of a musical excerpt. Unlike existing (comb filter based) approaches, we do not use handcrafted features derived from the audio signal, but rather let a recurrent neural network learn an intermediate beat-level representation of the signal and use this information as input to the comb filter bank. While most approaches apply complex post-processing to the output of the comb filter bank like tracking multiple time scales, processing different accent bands, modelling metrical relations, categorising the excerpts into slow / fast or any other advanced processing, we achieve state-of-the-art performance on nine of ten datasets by simply reporting the highest resonator's histogram peak.

## 8.1 INTRODUCTION

Tempo estimation is one of the most fundamental music information retrieval (MIR) tasks. The tempo of music corresponds to the frequency of the beats, i.e. the speed at which humans usually tap to the music.

In this paper, we only deal with global tempo estimation, i.e. report a single tempo estimate for a given musical piece, and do not consider the temporal evolution of tempo. Possible applications for such algorithms include automatic DJ mixing, similarity estimation, music recommendation, playlist generation, and tempo aware audio effects. Finding the correct tempo is also vital for many beat tracking algorithms which use a two-folded approach of first estimating the tempo of the music and then aligning the beats accordingly.

Many different methods for tempo estimation have been proposed in the past. While early approaches estimated the tempo based on discrete time events (e.g. MIDI notes or a sequence of onsets) [42], almost all of the recently proposed algorithms [35, 53, 58, 80, 104, 136] use some kind of continuous input. Generally, they follow this procedure: they transform the audio signal into a down-sampled feature, estimate the periodicities and finally select one of the periodicities as tempo.

As a reduction function, the signal's envelope [117], band pass filters [58, 80, 136], onset detection functions [35, 58, 104, 136] or combinations thereof are commonly used. Popular choices for periodicity detection include fast Fourier transform (FFT) based methods like tempograms [25, 136], autocorrelation [42, 58, 104, 109] or comb filters [35, 80, 117]. Finally, post-processing is applied to chose the most promising periodicity as perceptual tempo estimate. These post-processing methods range from simply selecting the highest periodicity peak to more sophisticated (machine learning) techniques, e.g. hidden Markov models (HMM) [80], Gaussian mixture model (GMM) regression [107] or support vector machines (SVM) [60, 109].

In this paper, we propose to use a neural network to derive a reduction function which makes complex post-processing redundant. By simply selecting the comb filter with the highest summed output, we achieve state-of-the-art performance on nine of ten datasets in the ACCURACY 2 evaluation metric.

## 8.2 RELATED WORK

In the following, we briefly describe some important works in the field of tempo estimation. Gouyon et al. [65] give an overview of the first comparative algorithm evaluation which took place for ISMIR 2004, followed by another study by Zapata and Gómez [138].

The work of Scheirer [117] was the first one to process the audio signal continuously rather than working on a series of discrete time events. He proposed the use of resonating comb filters, which are

one of the main techniques used for periodicity estimation since then. Periodicity analysis is performed on a number of band pass filtered signals and then the outputs of this analysis are combined and a global tempo is reported.

Dixon [42] uses discrete onsets gathered with the spectral flux method to build clusters of inter onset intervals which are in turn processed by a multiple agent system to find the most likely tempo. Oliveira et al. [104] extend this approach to use a continuous input signal instead of discrete time events and modified it to allow causal processing.

Klapuri et al. [80] jointly analyse the musical piece at three time scales: the tatum, tactus (which corresponds to the beat or tempo) and measure level. The signal is split into multiple bands and then combined into four accent bands before being fed into a bank of resonating comb filters similar to [117]. Their temporal evolution and the relation of the different time scales are modelled with a probabilistic framework to report the final position of the beats. The tempo is then calculated as the median of the beat intervals during the second half of the signal.

Instead of a multi-band approach as used in [80, 117], Davies and Plumbley [35] process an autocorrelated version of a complex domain onset detection function with a shift invariant comb filter bank to get the beat period. Although this method uses only a single dimensional input feature, it performs almost as good as the competing algorithms in [65] but has much lower computational complexity.

Gainza and Coyle [58] use a multi-band decomposition to split the audio signal into three frequency bands and then perform a transient / onsets detection (with different onset detection methods). These are transformed via autocorrelation into periodicity density functions, combined, and weighted to extract the final tempo.

Gkiokas et al. [60] utilise harmonic/percussive source separation on top of a constant-Q transformed signal in order to extract chroma features and filter bank energies from the separated signal respectively. Periodicity is estimated for both representations with a bank of resonating comb filters for overlapping windows of 8 seconds length and the resulting features are combined before a metrical level analysis is performed to report the final tempo. In a consecutive work Gkiokas et al. [59] use a support vector machine (SVM) to classify the music into tempo classes to better predict the tempo to be reported.

Elowsson et al. [53] also use harmonic / percussive source separation to model the speed of music. They derive various features like onset densities (for multiple frequency ranges) and strong onset clusters and use a regression model to predict the tempo of the signal.

Percival and Tzanetakis [109] use a "traditional" approach by first generating a spectral flux onset strength signal, followed by a stage which detects the beat period in overlapping windows of approx-

imately 6 seconds length (via generalised autocorrelation with harmonic enhancement) and a final accumulating stage which gathers all these tempo estimates and uses a support vector machine (SVM) to decide which octave the tempo should be in.

Wu and Jang [136] first derive an unaltered and a low pass filtered version of the input signal. Then they obtain a tempogram representation of a complex domain onset detection function for both signals to obtain tempo pairs. A classifier is then used to report the final most salient tempo.

## 8.3 ALGORITHM DESCRIPTION

Scheirer [117] found it beneficial to compute periodicities individually on multiple frequency bands and then subsequently combine them to estimate a single tempo. Klapuri et al. [80] followed this route but Davies and Plumbley [35] argued that is is enough to have a single – musically meaningful – feature to estimate the periodicity of a signal.

Given the fact that beats are the musically most relevant descriptors for the tempo of a musical piece, we take this approach one step further and do not use the pre-processed signal directly – or any representation that is strongly correlated with it, e.g. an onset detection function – as an input for a comb filter, but rather process the signal with a neural network which is trained to predict the positions of beats inside the signal. The resulting beat activation function is then fed into a bank of resonating comb filters to determine the tempo.

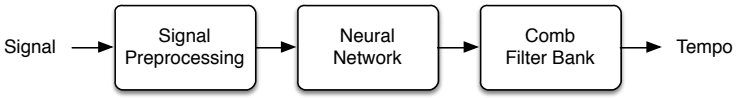
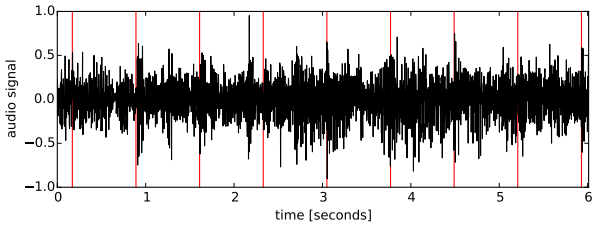



Figure 8.1: Overview of the new tempo estimation system.

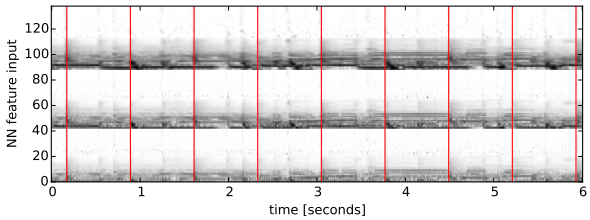
Figure 8.1 gives general overview over the different steps of the tempo estimation system, which are described into more detail in the following sections.

### 8.3.1 *Signal pre-processing*

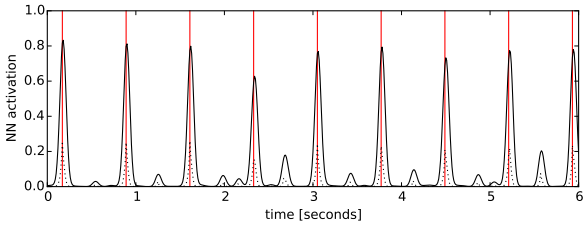
The proposed system processes the signal in a frame-wise manner. Therefore the audio signal is split into overlapping frames and weighted with a Hann window of same length before being transferred to a time-frequency representation by means of the discrete Fourier transform (DFT). Two adjacent frames are located 10 ms apart, which corresponds to a rate of 100 fps (frames per second). We omit the phase portion of the complex spectrogram and use only the magnitudes for further processing. To reduce the dimensionality of the signal, we process it with a logarithmically spaced filter which has



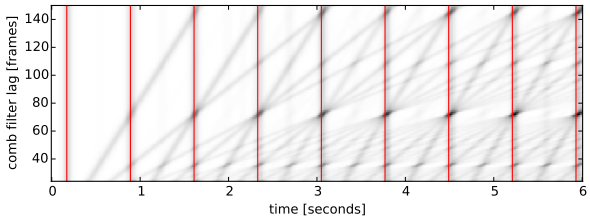

(a) Input audio signal



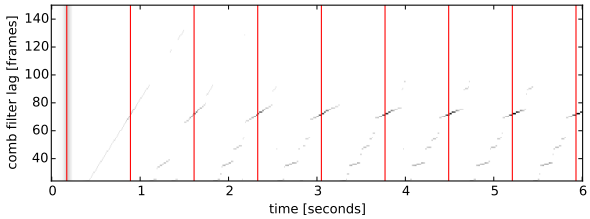

(b) Input to the neural network



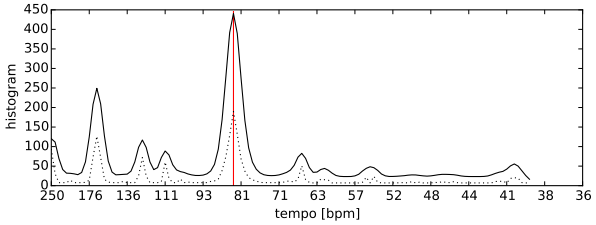

(c) Neural network output (beat activation function)




(d) Resonating comb filter bank output




(e) Maxima of the resonating comb filter bank




(f) Weighted histogram with summed maxima

Figure 8.2: Signal flow of a 6 second pop song excerpt: (a) input audio signal, (b) pre-processed input to the neural network, (c) its raw (dotted) and smoothed (solid) output, (d) corresponding comb filter bank response, (e) the maxima thereof, (f) resulting raw (dotted) and smoothed (solid) weighted histogram of the summed maxima. Ground-truth beat positions and corresponding tempo are marked with vertical red lines.

three bands per octave and is limited to the frequency range $[30, 17000]$ Hz. To better match the human's perception of loudness, we scale the resulting frequency bands logarithmically. As the final input features for the neural network, we stack three spectrograms and their first order difference calculated with different short-time Fourier transform (STFT) sizes of 1024, 2048 and 4096 samples, a visualisation is given Figure 8.2a.

### 8.3.2 *Neural network processing*

As a network we chose the system presented in [17], which is also the basis for the current state-of-the-art in beat tracking [14, 83]. The output of the neural network is a beat activation function, which represents the probability of a frame being a beat position. Instead of processing the beat activation function to extract the positions of the beats, we use it directly as a one-dimensional input to the bank of resonating comb filters.

Using this continuous function instead of discrete beats is advantageous since the detection is never 100% effective und thus introduces errors when inferring the tempo directly from the beats. This is in line with the observation that recent tempo induction algorithms use onset detection functions or other continuously valued inputs rather than discrete time events.

We believe that the learned feature representation (at least to some extent) incorporates information that otherwise would have to be modelled explicitly, either by tracking multiple time scales [80], processing multiple accent bands [117], modelling metrical relations [60], dividing the excerpts into slow / fast categories [53] or any other advanced processing. Figure 8.2c shows an exemplary output of the neural network. It can be seen that the network activation function has strong regular peaks that do not always coincide with high energies in the network's inputs.

#### *Network training*

We train the network on the datasets described in Section 8.4.2 which are marked with an asterisk (*) in an 8-fold cross validation setting based on a random splitting of the datasets. We initialise the network weights and biases with a uniform random distribution with range $[-0.1, 0.1]$ and train it with stochastic gradient decent with a learning rate of $10^{-4}$ and a momentum of 0.9. We stop training if no improvement of the cross entropy error of the validation set can be observed for 20 epochs. All adjustable parameters of the system are tuned to maximise the tempo estimation performance on the validation set.

*Activation function smoothing*

The beat activation function of the neural network reflects the probability that a given frame is a beat position. However, it can happen that the network is not sure about the exact position of the beat if it falls close to the border between two frames and hence splits the reported probability between these two frames. Another aspect to be considered is the fact that the ground truth annotations used as targets for the training are sometimes generated via manual tapping and thus deviate from the real beat position by up to 50 ms. This can result also in blurred peaks in the beat activation function. To reduce the impact of these artefacts, we smooth the activation function before being processed with the filter bank by convolving it with a Hamming window of length 140 ms.[1]

### 8.3.3 *Comb filter periodicity estimation*

We use the output of the neural network stage as input to a bank of resonating comb filters. As outlined previously, comb filters are a common choice to detect periodicities in a signal, e.g. [35, 80, 117]. The advantage of comb filters over autocorrelation lays in the fact that comb filters also resonate at multiples, fractions and simple rationales of the filter lag. This behaviour is in line with the perception of humans, which do not necessarily consider double or half tempi wrong. We use a bank of resonating feed backward comb filters with different time lags ($\tau$), defined as:

$$y(t, \tau) = x(t) + \alpha * y(t - \tau, \tau). \tag{8.1}$$

Each comb filter adds a scaled (by factor $\alpha$) and delayed (with lag $\tau$) version of its own output $y(t)$ to the input signal $x(t)$ with $t$ denoting the time frame index.

*Lag range definition*

For the individual bands of the comb filter bank we use a linear spacing of the lags with the minimum and maximum delays calculated as:

$$\begin{aligned} \tau_{min} &= \lfloor 60 * fps / bpm_{max} \rfloor \\ \tau_{max} &= \lceil 60 * fps / bpm_{min} \rceil \end{aligned} \tag{8.2}$$

with $fps$ representing the frame rate of the system given in frames per second and the minimum and maximum tempi $bpm_{min}$ and $bpm_{max}$

1 Because of this smoothing the beat activations do not reflect probabilities any more (and they may exceed the value of 1), but this does not harm the overall interpretation and usefulness.

given in beats per minute. We found the tempo range of $[40, 250]$ bpm to perform best on the validation set.

*Scaling factor definition*

Scheirer [117] found it beneficial to use different scaling factors $\alpha(\tau)$ for the individual comb filter bands. He defines them such that the individual filters have the same half-energy time. Klapuri et al. [80] also uses filters with exponentially decaying pulse response, but sets the scaling factor such that the response decays to half after a defined time of 3 seconds.

Contrary to these findings, we use a single value for all filter lags, which is set to $\alpha = 0.79$. The reason that a single value works better for this system may lay in the fact that we sum all peaks of the filters. With a fixed scaling factor, the resonance of filters with smaller lags tend to decay faster, but they also produce more peaks, hence leading to a more "balanced" histogram.

*Histogram building*

After smoothing the neural network output and processing it with the comb filter, we build a weighted histogram $H(\tau)$ from the output $y(t, \tau)$ by simply summing the activations of the individual comb filters (over all frames) if this filter produced the highest peak at the given time frame:

$$\begin{aligned} H(\tau) &= \sum_{t=0}^{T} y(t, \tau) * I(\tau, \arg\max_{\tau} y(t, \tau)) \\ I(a, b) &= \begin{cases} 1 & \text{if } a \equiv b \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{8.3}$$

with $t$ denoting the time frame index, $T$ the total number of frames, and $\tau$ the filter delays.

The bins of the weighted histogram correspond to the time lags $\tau$ and the bin heights represent the number of frames where the corresponding filter has a maximum at this delay, weighted by the activations of the comb filter. This weighting has the advantage that it favours filters which resonate at lags which correspond to intervals with highly probable beat positions (i.e. high values of the beat activation function) over those which are less probable. Figure 8.2d illustrates the output of the comb filter bank, Figure 8.2e the weighted maxima which are used to build the weighted histogram shown as the dotted line in Figure 8.2f.

*Histogram smoothing*

Music almost always contains tempo fluctuations – at least with regard to the frame rate of the system. Even stable tempi result in weights being split between two or more histogram bins. Therefore we combine bins before reporting the final tempo.

Our approach simply smooths the histogram by convolving it with a Hamming window with a width of seven bins, similar to [109]. Depending on the bin index (corresponding to the filter lag $\tau$), a fixed width results in different tempo deviations, ranging from $-7\%$ to $+8\%$ for a lag of $\tau = 24$ (corresponding to 250 bpm) to $-2\%$ to $+2.9\%$ for a lag of $\tau = 40$ (i.e. 40 bpm). Although this allows a greater deviation for higher tempi, we found no improvement over choosing the size of the smoothing window as a function of the tempo. Figure 8.2f shows the smoothed histogram as the solid line.

*Peak selection*

The histogram shows peaks at the different tempi of the musical piece. Again, previous works put much effort into this stage to select the peak with the strongest perceptual strength, ranging from simple rules driven by heuristics [109] over GMM regression based solutions [107] to utilising a support vector machine (SVM) [59, 109] or decision trees [109]. In order to keep our approach as simple as possible, we simply select the highest peak of the smoothed histogram as our final tempo.

## 8.4 EVALUATION

To assess the performance of the proposed system we compare it to an autocorrelation based tempo estimation method as described in [17], which operates on the same beat activation function obtained with the neural network described in Section 8.3.2. The algorithms of Gkiokas et al. [60], Percival and Tzanetakis [109], Klapuri et al. [80], Oliveira et al. [104], and Davies and Plumbley [35] were chosen as additional reference systems based on their availability and overall performance.

For a short description of these algorithms, please refer to Section 8.2.

All of the algorithms were used in their default configuration, except the system of Oliveira et al. [104], which we operated in offline mode with an induction length of 100 seconds, because it yielded significantly better results.[2] It should be noted however, that this mode results in a reduced tempo search range of 81-160 bpm, which can lead to biased results in favour of datasets in this tempo range.

Following [138] and [109] we perform statistical tests of our results compared to the others with McNemar's test using a significance value of $p < 0.01$.

2 This corresponds to: `ibt -off -i auto-regen -t 100`

### 8.4.1 *Evaluation metrics*

Since humans perceive tempo and rhythm subjectively, there is no single best tempo estimate. For example, the perceived tempo can be a multiple or fraction of the tempo given by the score of the piece. This is also known as the tempo octave problem. Therefore, two evaluation measures are used in the literature: ACCURACY 1 considers only the single annotated tempo for the evaluation, whereas ACCURACY 2 also includes integer multiples or fractions of the annotated tempo. Since the data that we use also contains music in ternary meter, we do not only add double and half tempo annotations, but also triple and third tempo. In line with most other publications we report accuracy values which denote the algorithms' ability to correctly estimate the tempo of the musical piece with less than 4% deviation form the annotated ground truth.

### 8.4.2 *Datasets*

We use a total of ten datasets to evaluate the performance of our algorithm. Table 8.1 lists some statistics of the datasets. Datasets marked with an asterisk (*) were used to train the neural networks with 8-fold cross validation as described in Section 8.3.2.

For all sets with beat annotations available (Ballroom, Hainsworth, SMC, Beatles, RWC, HJDB), we generated the tempo annotations as the median of the inter beat intervals. For the HJDB set (which is in 4/4 meter), we first derived the beat positions from the downbeat annotations before inferring the tempo ground truth. For all other sets we use the provided tempo annotations and – where applicable – the corrected annotations from [109].

### 8.4.3 *Results and discussion*

Table 8.2 lists the results of the proposed algorithm compared to the reference systems. The results (of our algorithm) reported on the Ballroom, Hainsworth and SMC set are obtained with 8-fold cross-validation, since these datasets were used to train the neural network. Although this is a technically correct evaluation, it can lead to biased results, since the system knows for example about ballroom music and its features in general and thus has an advantage over the other systems. It is thus no surprise that the proposed system outperforms the others on these sets.

Nonetheless, the new system outperforms the autocorrelation based tempo estimation method operating on the very same neural network output in almost all cases. This clearly shows the advantage of the resonating comb filters, which are less prone to single missing or misaligned peaks in the beat activation function, due to their recurrent

| DATASET | # FILES | LENGTH | ANNOTATIONS |
|---|---|---|---|
| Ballroom [65, 87] * [3] | 685 | 5 h 57 m | beats |
| Hainsworth [68] * | 222 | 3 h 19 m | beats |
| SMC [72] * | 217 | 2 h 25 m | beats |
| Klapuri [80] | 474 | 7 h 22 m | beats |
| GTZAN [109, 132] | 999 | 8 h 20 m | tempo |
| Songs [65] | 465 | 2 h 35 m | tempo |
| Beatles [33] | 180 | 8 h 09 m | beats |
| ACM Mirum [95, 107] | 1410 | 15 h 05 m | tempo |
| RWC Popular [62] | 100 | 6 h 47 m | beats |
| HJDB [71] | 235 | 3 h 19 m | downbeats |
| Total | 4987 | 63 h 17 m | |

Table 8.1: Overview of the datasets used for tempo estimation evaluation. Sets marked with asterisks were used for training.

nature and the fact that they also resonate on fractions and multiples of the dominant tempo.

The results for the other datasets reflect the algorithm's ability to estimate the tempo of a completely unknown signal without tuning any of the parameters. It can be seen that no single system performs best on all datasets. Our proposed system performs state-of-the-art (i.e. no other algorithm is statistically significantly better) in all but the HJDB set w.r.t. ACCURACY 2. We even outperform most of the other methods in ACCURACY 1, which highlights the algorithm's ability to not only capture a meaningful tempo, but also choose the correct tempo octave.

An inspection of incorrectly detected tempi in the HJDB set showed that the algorithm's histogram usually has a peak at the correct tempo but that this peak is not the highest. The reason lays in the fact that this set contains music with breakbeats and strong syncopation. Unfortunately, the neural network often identifies these syncopated notes as beats. Contrary to single or infrequently misaligned beats, the comb filter is not able to correct regularly recurring misalignments. e.g. in drum & bass music, where the bass drum usually falls on the offbeat between the third and fourth beat, this leads to additional peaks in the histogram corresponding to 0.5 and 1.5 times the beat interval, and a much lower peak at the correct position. Since we do not perform intelligent clustering of the histogram peaks, often the rate of the downbeats is reported, which results in a tempo which is not covered by the ACCURACY 2 measure any more.

3 We removed the 13 duplicates identified by Bob Sturm: http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html

| ACCURACY 1 | NEW | [17] | [60] | [109] | [80] | [104] | [35] |
|---|---|---|---|---|---|---|---|
| Ballroom | **0.950**† | 0.639†– | 0.625– | 0.653– | 0.642– | 0.651– | 0.709– |
| Hainsworth | **0.847**† | 0.541†– | 0.667– | 0.721– | 0.752– | 0.698– | 0.739– |
| SMC | **0.512**† | 0.442† | 0.346– | 0.267– | 0.189– | 0.166– | 0.152– |
| Klapuri | **0.789** | 0.502– | 0.741 | 0.732 | 0.768 | 0.724– | 0.692– |
| GTZAN | 0.668 | 0.601– | 0.716– | **0.754**+ | 0.704+ | 0.599– | 0.582– |
| Songs | 0.477 | 0.570+ | 0.570+ | **0.611**+ | 0.585+ | 0.486 | 0.424 |
| Beatles | **0.850** | 0.700– | 0.778 | 0.811 | 0.789 | 0.767 | 0.761– |
| ACM Mirum | **0.741** | 0.540– | 0.725 | 0.733 | 0.679– | 0.621– | 0.646– |
| RWC Pop | 0.600 | 0.450 | **0.900**+ | 0.810+ | 0.770 | 0.750 | 0.770+ |
| HJDB | 0.796 | 0.434– | 0.783 | 0.285– | 0.494– | **0.911**+ | 0.706 |
| Dataset avg. | **0.721** | 0.543 | 0.563 | 0.638 | 0.636 | 0.637 | 0.617 |
| Total avg. | **0.734** | 0.560– | 0.685– | 0.677– | 0.658– | 0.623– | 0.618– |
| ACCURACY 2 | | | | | | | |
| Ballroom | **1.000**† | 0.997† | 0.981 | 0.953– | 0.921– | 0.921– | 0.974 |
| Hainsworth | **0.941**† | 0.910† | 0.887 | 0.901 | 0.869 | 0.802– | 0.878 |
| SMC | **0.673**† | 0.599† | 0.512– | 0.438– | 0.438– | 0.359– | 0.415– |
| Klapuri | 0.937 | 0.907– | **0.954** | 0.937 | 0.918 | 0.880– | 0.924 |
| GTZAN | **0.950** | 0.942 | 0.938 | 0.925– | 0.923– | 0.841– | 0.922– |
| Songs | **0.933** | 0.918 | 0.910 | 0.865– | 0.910 | 0.791– | 0.875– |
| Beatles | 0.983 | 0.967 | 0.978 | **0.989** | 0.928 | 0.883 | 0.978 |
| ACM Mirum | 0.976 | 0.958– | **0.979** | 0.972 | 0.967 | 0.915– | 0.975 |
| RWC Pop | 0.950 | 0.940 | **1.000** | **1.000** | 0.990 | 0.980 | **1.000** |
| HJDB | 0.868 | 0.851 | 0.911 | **1.000**+ | 0.864 | 0.991+ | **1.000**+ |
| Dataset avg. | **0.919** | 0.899 | 0.916 | 0.896 | 0.871 | 0.837 | 0.893 |
| Total avg. | **0.946** | 0.929– | 0.935– | 0.923– | 0.909– | 0.861– | 0.923– |

Table 8.2: ACCURACY 1 and ACCURACY 2 results for different datasets (cf. Table 8.1) and algorithms (Böck and Schedl [17], Gkiokas et al. [60], Percival and Tzanetakis [109], Klapuri et al. [80], Oliveira et al. [104], Davies and Plumbley [35]). Best results in bold, + and − mark results statistically significance compared to ours, † denote values obtained with 8-fold cross validation.

### 8.4.4 *MIREX evaluation*

We submitted the algorithm to last year's MIREX evaluation.[4] Performance is tested on a hidden set of 140 files with a total length of 1 hour

4 http://nema.lis.illinois.edu/nema_out/mirex2014/results/ate/

and 10 minutes. The tempo evaluation used for MIREX is different, because for each song the two most dominant tempi are annotated. MIREX uses the following three evaluation metrics: *P-Score* [102] and the percentage of files for which *at least one* or *both* of the annotated tempi was identified correctly within a maximum allowed deviation of ±8% from the ground truth annotations. Since MIREX requires the algorithms to report two tempi with a relative strength, we adapted the peak-picking strategy outlined in Section 8.3.3 to simply report the two highest peaks.

| ALGORITHM | P-SCORE | ≥1 TEMPO | BOTH TEMPI |
|---|---|---|---|
| NEW | **0.876** | **0.993** | 0.629 |
| Elowsson et al. [53] | 0.857 | 0.943 | **0.693** |
| Gkiokas et al. [60] | 0.829 | 0.943 | 0.621 |
| Wu and Jang [136] | 0.826 | 0.957 | 0.550 |
| Lartillot et al. [92] | 0.816 | 0.921 | 0.571 |
| Klapuri et al. [80] | 0.806 | 0.943 | 0.614 |
| Böck and Schedl [17] | 0.798 | 0.957 | 0.564 |
| Davies and Plumbley [35] | 0.776 | 0.929 | 0.457 |

Table 8.3: Tempo estimation results on the McKinney (MCK) test collection used for MIREX evaluation.

Table 8.3 gives an overview of the five best performing algorithms (of different authors) over all years the MIREX tempo estimation task is run, together with results for algorithms also used for evaluation in the previous section.

Our algorithm ranked first in last year's MIREX evaluation [5] and achieved the highest *P-Score* and *at least one tempo reported correctly* performance ever. The best performing algorithm for the *both tempi correct* evaluation was the one submitted by Elowsson et al. [53] in 2013, which explicitly models the speed of the music and thus has a much higher chance to report the two annotated tempi which are inferred from human beat tapping.

## 8.5 CONCLUSIONS

The presented tempo estimation algorithm based on recurrent neural networks and resonating comb filters is able to perform state-of-the-art or outperforms existing algorithms on all but one datasets investigated. Based on the high Accuracy 2 score, which also considers integer multiples and fractions of the annotated ground truth tempo, it can

5 http://nema.lis.illinois.edu/nema_out/mirex2014/results/ate

be concluded that the system is able to capture a meaningful tempo in almost all cases.

Additionally, we outperform many existing algorithms w.r.t. ACCURACY 1 which suggests that it is advantageous to use a musically more meaningful representation than just the onset strength of the signal – even if split into multiple accent bands – as an input for a bank of resonating comb filters.

In future, we want to investigate methods of perceptually clustering the peaks of the histogram to report the most relevant tempo, as this has been identified to be the main problem of the new algorithm when dealing with very syncopated music. We believe that this should increase the ACCURACY 1 performance considerably.

### ADDENDUM

Using this tempo estimation model (instead of an autocorrelation-based one), boosts the beat tracking performance of the method described in Chapter 5 considerably, as can be seen by the MIREX results given in Table 5.3.

#### *MIREX evaluation*

The system presented was first submitted to MIREX 2014 [6] and then consecutively in 2015 [7] and 2016.[8] It always performed best and only falls slightly short in *both tempi correct* performance. The 2015 submission uses neural network models trained on a wider range of musical styles (the same as the beat tracking algorithms described in Chapter 5). The system was submitted unaltered in 2016 and achieved the same performance.

| ALGORITHM | P-SCORE | $\geq$1 TEMPO | BOTH TEMPI |
|---|---|---|---|
| TempoDetector (2014) | 0.876 | 0.993 | 0.629 |
| TempoDetector (2015) | 0.898 | 0.993 | 0.664 |
| Elowsson et al. [53] | 0.857 | 0.943 | 0.693 |

Table 8.4: Performance of the presented algorithm in MIREX evaluations on the MCK set. The results represent the current state-of-the-art in beat tracking on this dataset. *TempoDetector* reflects the name of the executable program of the *madmom* package (cf. Section 10.2.2).

6 http://nema.lis.illinois.edu/nema_out/mirex2014/results/ate/
7 http://nema.lis.illinois.edu/nema_out/mirex2015/results/ate/
8 http://nema.lis.illinois.edu/nema_out/mirex2016/results/ate/

# Part IV

# NOTE TRANSCRIPTION

# 9

# PIANO TRANSCRIPTION

TITLE

Polyphonic Piano Note Transcription with Recurrent Neural Networks.

AUTHORS

Sebastian Böck and Markus Schedl.

PUBLISHED

In Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2012, Kyoto, Japan.

CONTRIBUTION

All work was carried out by me. Markus Schedl provided valuable input.

ABSTRACT

In this paper a new approach for polyphonic piano note onset transcription is presented. It is based on a recurrent neural network to simultaneously detect the onsets and the pitches of the notes from spectral features. Long Short-Term Memory units are used in a bidirectional neural network to model the context of the notes. The use of a single regression output layer instead of the often used one-versus-all classification approach enables the system to significantly lower the number of erroneous note detections. Evaluation is based on common test sets and shows exceptional temporal precision combined with a significant boost in note transcription performance compared to current state-of-the-art approaches. The system is trained jointly with various synthesised piano instruments and real piano recordings and thus generalises much better than existing systems.

## 9.1 INTRODUCTION

Music transcription is the process of converting an audio recording into a musical score or a similar representation. In this paper we concentrate on the transcription of piano notes, especially on the two most important aspects of notes, their pitch and onset times. To detect them as accurately as possible is crucial for a proper transcription of the musical piece. We leave out higher level tasks like determining the length of a note (given either in seconds or in a musical notation like quarter note). Also we do not consider the velocity or intensity. The output of the system is a simplified piano-roll notation of the audio signal.

Traditional music transcription systems are based on a wide range of different technologies, but all have to deal with the subtasks of estimating the fundamental frequencies and the onset locations of the notes. A very basic approach formulated by Dixon [41] solely relies on the spectral peaks of the signal to detect notes; local maxima represent the onsets and the drop of energy below a minimum threshold marks the offset of the note. Bello et al. [7] additionally incorporate time-domain features to predict multiple sounding pitches assuming that the signal can be constructed as a linear sum of individual waveforms based on a database of piano notes. Raphael [114] proposes a probability-based system which uses a hidden Markov model (HMM) to find chord sequences. The states are represented by frames with labels based on the sounding pitches. Ryynänen and Klapuri [115] also use HMMs to model note events based on multiple fundamental frequency features. Transition between notes are controlled via musical knowledge.

Most of today's top performing piano transcription systems rely on machine learning approaches. Marolt [97] describes an elaborate approach based on different neural networks to recognise tones in an audio recording, combined with adaptive oscillators to track partials. Poliner and Ellis [111] use multiple support vector machine (SVM) classifiers trained on spectral features to detect the sounding fundamental frequencies of a frame. Post-processing with HMM is applied to temporally smooth the output. Boogaart and Lienhart [23] use a cascade of boosted classifiers to predict the onsets and the corresponding pitches of each note. All these systems use multiple classifiers and thus can not reliably distinguish whether a sounding pitch is the fundamental frequency of a note or a partial of another one. This results in lots of false note detections. In contrast, our system uses a single regression model and is thus able to distinguish between these states and hence lowers the number of false detections significantly.

## 9.2 SYSTEM DESCRIPTION

Figure 9.1 shows the proposed piano transcription system. It takes a discretely sampled audio signal as its input. The signal is transferred to the frequency domain via two parallel *Short-Time Fourier Transforms* (STFT) with different window lengths. The logarithmic magnitude spectrogram of each STFT is then filtered to obtain a compressed representation with the frequency bins corresponding to the tone scale of a piano with a semitone resolution. This representation is used as input to a bidirectional Long Short-Term Memory (BLSTM) recurrent neural network. The output of the network is a piano-roll like representation of the note onsets for each MIDI note.

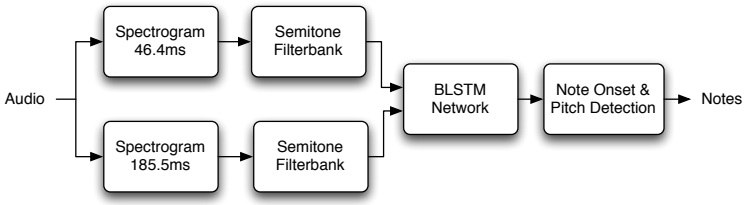



Figure 9.1: Proposed piano transcription system overview.

### 9.2.1 *Feature extraction*

As input, the system takes a monophonic pulse code modulated (PCM) audio signal $x(n)$ with a sampling rate of $f_s = 44.1\,\text{kHz}$ in floating point representation (in the range of $[-1...1]$). The signal is split into overlapping frames with frame lengths of 2048 and 8192 samples (46.4 and 185.8 ms). Different frame lengths have been chosen to achieve both a good temporal precision and a sufficient frequency resolution for the transcription of the notes. Two consecutive frames are located 10 ms apart, resulting in a constant frame rate $f_r = 100\,\text{fps}$. A Hamming window with the same size as the frame is applied before the signals are transferred to the frequency domain with the Short-Time Fourier Transform.

Phase information of the resulting complex spectrograms $X(n,k)$ is omitted, and only the magnitude values are used for all further calculations. A logarithmic representation of the magnitude spectrograms is advantageous, compared to the linear one. To avoid negative values and to put the values in a suitable range for the following neural network stage, the spectrograms are multiplied with a factor 1000 and a fixed value of 1 is added before taking the logarithm. These values yielded the best results during preliminary tests.

To reduce the dimensionality of the input vector of the neural network, the two magnitude spectrograms $S(n,k)$ are filtered with semitone filterbanks $F(m,k)$. The frequencies $m$ are spaced equally

on a logarithmic frequency scale and are aligned according to the pitches of the 88 MIDI notes (i.e. semitone spacing). This spacing is expanded up to the maximum frequency of 16 kHz. Overlapping triangular filters are used to combine multiple spectrogram frequency bins into one. The area of each filter is normalised to 1 to compensate the overemphasis of high frequencies. Finally duplicate filters (which occur if the frequency resolution of the STFT is too coarse for low MIDI pitches) are eliminated. This results in a dimensionality reduction from 5120 values of the two spectrograms down to 183.

The use of a semitone spacing instead of a less granular one (e.g. quarter tone) has two main advantages: first, it reduces the dimensionality of the input vector for the neural network by roughly a factor of two, thus resulting in reduced training time. It also desensitises the whole system against minor tuning variations of different pianos, hence leading to a much better generalisation without the need for a manual adjustment of the piano tuning.

Since the energy of the signal rises during the note attack phase which directly follows the note onset, also the first order differences of the semitone filtered spectrograms are included. For the small window length, the difference is calculated to the preceding frame, whereas for the long window length it is calculated relative to the frame at the index $n - 4$. This measure cancels the delay of the rise in energy relative to the actual note onset position. Although adding the first order differences doubles the input vector size of the neural network from 183 to 366, it increases the overall transcription performance and simultaneously reduces the needed training epochs, since the network converges faster.

### 9.2.2 *Neural network*

For the neural network stage, a bidirectional *recurrent neural network* (RNN) with *Long Short-Term Memory* (LSTM) units is used. Compared to *feed forward neural networks* (FNNs), RNNs have the advantage that they are able to model temporal contexts due to the use of recurrent connections in the hidden layers. Although theoretically able to remember any past values, they suffer from the vanishing gradient problem, i.e. input values decay or blow up exponentially over time, thus limiting their range to a maximum of a few time steps. Hochreiter and Schmidhuber [70] developed a new method called LSTM to overcome this problem. Each LSTM block has a recurrent connection with weight 1.0 which enables the block to act as a memory cell. Input, output, and forget gates control the content of the memory cell through multiplicative units and are connected to other neurons as usual.

A *bidirectional recurrent neural network* (BRNN) [121] doubles the number of hidden layers and presents the input values to the newly

created set of hidden layers in reverse temporal order. This offers the advantage that the network not only has access to past input values but can also 'look into the future'.

If BRNNs are used in conjunction with LSTM neurons, a *bidirectional Long Short-Term Memory* (BLSTM) recurrent neural network is built. It has the ability to model a wider temporal context around a given input value. For the detection of notes this is an essential feature, since the onset is not only characterised by an increase in energy during the attack phase, but also by a special energy envelope during the following decay, sustain, and release phases.

BLSTMs have been successfully implemented in systems for onset detection [55] and beat detection and tracking [17] which both showed state-of-the-art performance in their respective field. In contrast to those implementations, the neural network of this approach uses a regression output layer. The biggest advantage compared to multiple classifier system [23, 97, 111] lies in the ability of the system to correctly identify whether a sounding pitch is the fundamental frequency of a note or a partial of another one, thus reducing the number of false positive and negative note detections significantly.

The used neural network has three bidirectional hidden layers with 88 LSTM units each. The regression output layer has 88 units, each representing one MIDI pitch. The output of these units represent the activation functions for each note.

*Network training*

The network is trained with supervised learning and early stopping. The used training data set is described in Section 9.3. Together with the target values extracted from the MIDI data, each audio sequence is preprocessed as described above and presented to the network for learning. The network weights are initialised with random values following a Gaussian distribution with mean 0 and standard deviation 0.1. Standard gradient descent with backpropagation of the errors is used to train the network. To prevent over-fitting, performance is evaluated after each training iteration on the validation set. If no improvement on the summed squared error is observed for 20 epochs, the training is stopped.

*Network testing*

For the evaluation of the system, the unknown music excerpts of the test set are preprocessed as described in Section 9.2.1 and presented to the previously trained network. The resulting note activation regression matrix of the output nodes is used as input to the following stage.

### 9.2.3 *Note onset and pitch detection*

The notes onset times and pitches are derived directly from the neural network output. The activation values for each pitch are smoothed with a Hamming window of 90 ms length before being thresholded. The length of the window is not crucial as long as it is smaller than the duration between two consecutive notes of the same pitch. The threshold is determined individually per note on the validation set by: $\theta_p = \arg\max_\theta \{TP_\theta - FP_\theta - FN_\theta\}$, TP denoting true positive, FP false positive, and FN false negative detections. A standard local maximum peak picking algorithm is applied to gather the final note onset positions for each pitch.

## 9.3 DATA

Solo piano music has been chosen for training and evaluation of the described system. As a basis, the musical renderings and recordings of the MAPS database [1] introduced by Emiya et al. [54] are used. They consist of 209 pieces rendered by seven different software synthesisers and 60 real piano recordings with an upright Yamaha Disklavier. To expand the dataset, 267 MIDI files from the same source, the Classical Piano Midi Page [2], were synthesised with the freely available Maestro Concert Grand v2 [3] sound font. To compensate the emphasis towards synthesised sounds, the LabROSA Disklavier recordings [4] (used for evaluation in [111]) and real audio recordings of 13 Mozart sonatas played on a Bösendorfer SE290 computer monitored grand piano by the pianist Roland Batik were added to the set. The whole dataset is split into training, validation, and testing examples according to the original splitting in [111], thus maintaining the comparability of the results. Table 9.1 shows the distribution of the dataset.

| DATASET | TRAINING | VALIDATION | TESTING |
|---|---|---|---|
| MAPS (MIDI instruments) | 854,507 | 108,778 | 107,310 |
| MAPS (Disklavier) | 86,026 | 16,495 | 5,675 |
| MIDI (Maestro Concert) | 519,479 | 59,838 | 71,225 |
| Batik (Bösendorfer) | 76,095 | 13,387 | 16,926 |
| LabROSA (Disklavier) | 47,134 | 0 | 23,298 |

Table 9.1: Number of notes in the individual datasets.

1 http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/
2 http://www.piano-midi.de
3 http://www.linuxsampler.org/instruments.html
4 http://labrosa.ee.columbia.edu/projects/piano/

## 9.4 RESULTS

To measure the performance of our system, standard precision, recall, and f-measure scores are used. Another measure is the accuracy, defined by Dixon [41]. Since it counts false detections twice (both the false negative and the false positive detection), error scores as used by Poliner and Ellis [111] are provided additionally; $E_{subs}$ denote note substitutions, $E_{miss}$ missed notes, $E_{fa}$ false additions, and $E_{tot}$ the sum of all errors.

### 9.4.1 *Note onset transcription*

An onset is considered as correctly identified if its pitch is correctly identified and its location is within a certain window around the ground-truth position. For onset detection usually a 100 ms window is used, and the results given by Poliner and Ellis [111] are based on this window length as well. Boogaart and Lienhart [23] use a window of 68.25 ms. Although penalising our system, we give results only for a detection window of 50 ms.

| | ACC | $E_{SUBS}$ | $E_{MISS}$ | $E_{FA}$ | $E_{TOT}$ |
|---|---|---|---|---|---|
| MAPS (MIDI) | 0.840 | 0.029 | 0.068 | 0.056 | 0.153 |
| MAPS (Disklavier) | 0.687 | 0.066 | 0.170 | 0.090 | 0.326 |
| MIDI | 0.889 | 0.020 | 0.040 | 0.039 | 0.099 |
| Batik | 0.901 | 0.003 | 0.064 | 0.032 | 0.099 |
| LabROSA | 0.627 | 0.096 | 0.178 | 0.119 | 0.393 |
| complete | 0.856 | 0.025 | 0.061 | 0.051 | 0.137 |
| complete (w/o octave) | 0.897 | 0.025 | 0.053 | 0.014 | 0.092 |
| Poliner and Ellis [111] | 0.623 | 0.045 | 0.164 | 0.224 | 0.432 |
| Boogaart and Lienhart [23] | 0.874 | - | - | - | - |

Table 9.2: Note onset transcription accuracy and error rates for the partial and complete test sets. $E_{subs}$ denote note substitution, $E_{miss}$ missed notes, $E_{fa}$ false additions, and $E_{tot}$ the sum of all errors.

Table 9.2 shows the accuracy and error rates for the different test sets compared to other state-of-the-art systems. The new approach clearly outperforms the system of Poliner and Ellis [111] not only in case of the complete test set, but even for the most difficult partial test set (the LabROSA Disklavier recordings). This does not only show the good performance of our approach, but also highlights its good generalisation capability. Concurrent with the rise in accuracy, all error score are significantly lower. This demonstrates the ability of the

system to detect even difficult notes without adding a high number of false detections.

If only a single instrument is evaluated (i.e. the MIDI test set), the systems also performs better than the one of Boogaart and Lienhart [23], which was trained with a single MIDI instrument. This is remarkable, since our system is not trained specifically for a single instrument. Trained solely on the MIDI dataset, our system achieves an accuracy of 93.6%, exceeding their results even further.

The much better result for the Batik test set can be explained by the lower musical complexity of the Mozart sonatas compared to other musical pieces of the test set.

Tonal misalignments can have different impacts on human perception. A pitch error of an octave does usually not harm the overall impression very much, whereas other transcription errors can spoil a musical piece completely. Therefore Table 9.2 also adds results if octave errors are not considered. Since the number of errors are almost evenly distributed across all test sets, only one result for the complete test set is given. Not counted are errors occurring due to notes being added exactly one octave below or above the correct pitch, or notes that are missed if there is a detection exactly one octave apart. It can be seen that more than 70% of the spuriously added notes are pitched exactly one octave aside and hence do not harm the musical perception much.

### 9.4.2 *Temporal resolution*

According to Handel [69], 5 ms is the threshold of perceptual difference for musical performances. For piano transcription it is therefore highly desirable to achieve the maximum possible temporal precision. Table 9.3 shows the precision, recall, f-measure and accuracy results for different detection window sizes on the whole test set. The system achieves roughly the same performance for all detection windows down to a length of 50 ms. If the window size is reduced to 30 ms (which corresponds to a maximum deviation of the detection by a single frame in both directions at the used frame rate), the performance starts to decrease. Even if only the annotated frame is used for evaluation, the system is still performing decently, highlighting the exceptional temporal precision. It should be noted that the Disklavier recordings sometimes have annotation inaccuracies of up to 15 ms, which explains the much lower result if only one frame is considered.

The use of multiple spectrograms enables the algorithm to achieve such a good temporal performance. Inspection of the internal states of the neural network shows that the network gathers timing information almost exclusively from the spectrogram and the differences obtained with the shorter STFT window length, while the information needed

| WINDOW | PRECISION | RECALL | F-MEASURE | ACCURACY |
|---|---|---|---|---|
| ±50 ms | 0.936 | 0.917 | 0.927 | 0.863 |
| ±35 ms | 0.935 | 0.917 | 0.926 | 0.862 |
| ±25 ms | 0.933 | 0.915 | 0.924 | 0.859 |
| ±15 ms | 0.924 | 0.906 | 0.915 | 0.844 |
| ±5 ms | 0.738 | 0.723 | 0.730 | 0.575 |

Table 9.3: Note onset transcription results for the complete test set measured with different window detection sizes.

to determine the pitch of a note (especially the lower pitched ones) is mostly obtained from the spectrogram with the longer window.

## 9.5 CONCLUSIONS

In this paper we presented a new piano transcription system which is a significant step towards real audio-to-MIDI transcription. It gives exceptional temporal precision paired with state-of-the-art note onset and pitch transcription performance.

The evaluation on publicly available test sets shows that our approach greatly reduces both the number of false positive and negative note detections. The reduction is mainly due to the use of a single regression output layer to simultaneously detect note onsets and pitches compared to the one-versus-all classification approaches. Only holistic systems can decide whether a sounding frequency is the real fundamental frequency or a harmonic overtone of another note.

Furthermore our system generalises very well over a wide range of various pianos, resulting in transcription results previously only achieved by systems tuned specifically for a single instrument.

### ADDENDUM

The presented work is used as a transcription step in *'The Complete Classical Piano Music Companion'* developed by Arzt et al. [1, 2, 4]. This system is able to identify the piece being played, and determine the position within the piece almost instantly and then follows the score in real-time accordingly as the performer continues playing.

In Collins et al. [30] it is used to bridge the "audio-symbolic gap" to discover repeated musical patterns directly from polyphonic audio – instead of inferring these patterns from symbolic data.

*MIREX evaluation*

The presented algorithm was submitted to MIREX 2012.[5] It ranked first wrt. average onset transcription F-Measure (given in Table 9.4). A simplified version of this system tuned for real-time performance (also included in the *madmom* package, cf. Chapter 10) was submitted to MIREX 2014.[6] It uses *tanh* units instead of the proposed *LSTM* units. This reduces the number of network weights (roughly by a factor of 4), yielding even a slightly better overall performance.

| ALGORITHM | PRECISION | RECALL | F-MEASURE |
|---|---|---|---|
| PianoTranscriptor (2012) | 0.595 | 0.754 | 0.664 |
| PianoTranscriptor (2014) | 0.640 | 0.728 | 0.680 |
| BW3 [9, 10] (2015) | 0.704 | 0.683 | 0.692 |
| MM1 [97] (2016) | 0.794 | 0.722 | 0.754 |
| EF1 [52] (2014) | 0.845 | 0.764 | 0.802 |
| DT1 [130] (2016) | 0.912 | 0.756 | 0.820 |

Table 9.4: Performance of the presented algorithm in MIREX evaluations compared to state-of-the-art piano transcription algorithms. The *PianoTranscriptor* reflects the name of the executable program included in the *madmom* package (cf. Section 10.2.2).

The 2014 EF1 submission of Elowsson and Friberg [52] and 2016 DT1 submission of Troxel [130] show outstanding performance. They achieve this high performance at the cost of a very high runtime of 23400 seconds (EF1)[7] for 300 seconds of audio material (78×real-time) as compared to 180 seconds of our 2014 submission (0.6×real-time) on the test set.

Table 9.4 list the best results reported so far for this MIREX task, showing that our approach is still performing well, especially considering that it is the only one (of those listed) that achieves better than real-time execution times.[8]

5 http://www.music-ir.org/mirex/wiki/2012:Multiple_Fundamental_Frequency_Estimation_%26_Tracking_Results

6 http://www.music-ir.org/mirex/wiki/2014:Multiple_Fundamental_Frequency_Estimation_%26_Tracking_Results

7 The runtimes of DT1 and MM1 are not given, but we also assume high runtimes, since both submissions use either a large convolutional neural networks (DT1) or multiple neural networks together with a complicated partial tracking technique (MM1).

8 Our system is one of two submissions ever achieving better than real-time execution times.

# Part V

# SOFTWARE

# 10

# MADMOM

## TITLE

madmom: a new Python Audio and Music Signal Processing Library.

## AUTHORS

Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer.

## PUBLISHED

In Proceedings of the 24th ACM International Conference on Multimedia, October 2016, Amsterdam, The Netherlands.

## CONTRIBUTION

The work on this library started on my initiative, and most source code and documentation was contributed by me.

## ABSTRACT

In this paper, we present *madmom*, an open-source audio processing and music information retrieval (MIR) library written in Python. *madmom* features a concise, *NumPy*-compatible, object oriented design with simple calling conventions and sensible default values for all parameters, which facilitates fast prototyping of MIR applications. Prototypes can be seamlessly converted into callable processing pipelines through *madmom*'s concept of *Processors*, callable objects that run transparently on multiple cores. *Processors* can also be serialised, saved, and re-run to allow results to be easily reproduced anywhere.

Apart from low-level audio processing, *madmom* puts emphasis on musically meaningful high-level features. Many of these incorporate machine learning techniques and *madmom* provides a module that implements some methods commonly used in MIR such as hidden Markov models and neural networks. Additionally, *madmom* comes with several state-of-the-art MIR algorithms for onset detection, beat, downbeat and meter tracking, tempo estimation, and chord recognition. These can easily be incorporated into bigger MIR systems or run as stand-alone programs.

## 10.1 INTRODUCTION

Music information retrieval (MIR) has become an emerging research area over the last 15 years. Especially audio-based MIR has become more and more important, since the amount of available audio data in the last years exploded beyond being manageable manually.

Most state-of-the-art audio-based MIR algorithms consist of two components: first, low-level features are extracted from the audio signal (*feature extraction* stage), and then the features are analysed (*feature analysis* stage) to retrieve the desired information. Most current MIR systems incorporate machine learning algorithms in the feature analysis stage, with neural networks currently being the most popular and successful ones [14, 15, 83, 119, 120].

Numerous software libraries have been proposed over the years to facilitate research and development of applications in MIR. Some libraries concentrate on low-level feature extraction from audio signals, such as *Marsyas* [131], *YAAFE* [100] and *openSMILE* [56]. Others also include higher level feature extraction such as onset and beat detection as for example in the *MIRtoolbox* [93], *Essentia* [22] and *LibROSA* [101]. However, to our knowledge, there exist no library that also includes machine learning components (except *Marsyas* [131], which contains two classifiers), although machine learning components are crucial in current MIR applications.

Therefore, we propose *madmom*, a library that incorporates low-level feature extraction *and* high-level feature analysis based on machine learning methods. This allows the construction of the full processing chain within a single software framework, making it possible to build standalone programs without any dependency on other machine learning frameworks. Moreover, *madmom* comes with several state-of-the-art systems including their trained models, for example for onset detection [55, 119, 120], tempo estimation [15], beat [14, 83] and downbeat tracking [16, 87], and chord recognition [84, 85].

*madmom* is written in Python, which has become the language of choice for scientific computing for many people due to its free availability and its simplicity of use. The code is released under BSD license and pre-trained models are released under the CC BY-NC-SA 4.0 license.

### 10.1.1 *Design and functionality*

#### *Object-oriented programming*

*madmom* follows an object-oriented programming (OOP) approach. We encapsulate everything in objects that are often designed as subclasses of *NumPy*'s *ndarray*, offering all array handling routines inherited from *NumPy* [134] with additional functionality. This compactly bundles

data and meta-data (e.g. a *Spectrogram* and its *frame rate*) and simplifies meta-data handling for the user.

*Rapid prototyping*

*madmom* aims at minimising the turnaround time from a research idea to a software prototype. Thus, object instantiation is made as simple as possible: e.g. a *Spectrogram* object can be instantiated with a single line of code by only providing the path to an audio file. *madmom* automatically creates all objects in between using sensible default values.

*Simple conversion into runnable programs*

Once an audio processing algorithm is prototyped, the complete workflow should be easily transformable into a runnable standalone program with a consistent calling interface. This is implemented using *madmom*'s concept of *Processors*.

*Machine learning integration*

We aim at a seamless integration of machine learning methods without the need of any third party modules. We limit ourselves to testing capabilities (applying pre-trained models), since it is impossible to keep up with newly emerging training methods in the various machine learning domains. Models that have been trained in an external library should be easily be convertible to an internal *madmom* model format.

*State-of-the-art features*

Many existing libraries provide a huge variety of low-level features, but few musically meaningful high-level features. *madmom* tries to close this gap by offering high-quality state-of-the-art feature extractors for onsets, beats, downbeats, chords, tempo, etc.

*Reproducible research*

In order to foster reproducible research, we want to be able to save and load the specific settings used to obtain the results for a certain experiment. In *madmom* this is implemented using Python's own *pickle* functionality which allows to save an entire processing chain (including all settings) to a file.

*Few dependencies*

*madmom* is built on top of three excellent and wide-spread libraries: *NumPy* [134] provides all the array handling subroutines for *madmom*'s *data classes*. *SciPy* [76] provides optimised routines for the fast Fourier

transform (FFT), linear algebra operations and sparse matrix representations. Finally, *Cython* [5] is used to speed up time critical parts of the library by automatically generating C code from a Python-like syntax and then compiling and linking it into extensions which can be transparently used from within Python. These libraries are the only installation and runtime dependencies of *madmom* besides the *Python* standard library itself, supported in version 2.7 as well as 3.3 and newer.

*Multi-core capability*

We designed *madmom* to be able to exploit the multi-core capabilities of modern computer architectures, by providing functionality to run several programs or *Processors* in parallel.

*Extensive documentation*

All source code files contain thorough documentation following the *NumPy* format. The complete API reference, instruction on how to build and install the library, as well as interactive *Jupyter* [110] notebooks can be found online at `http://madmom.readthedocs.io`. The documentation is built automatically with *Sphinx*.[1]

*Open development process*

We follow an open development process and the source code and documentation of our project is publicly available on GitHub: `http://github.com/CPJKU/madmom`. To maintain high code quality, we use continuous integration testing via TravisCI[2], code quality tests via QuantifiedCode[3], and test coverage via Coveralls.[4]

## 10.2 LIBRARY DESCRIPTION

In this section, we will describe the overall architecture of *madmom*, its packages as well as the provided standalone programs.

*madmom*'s main API is composed of classes, but much of the functionality is implemented as functions (in turn used internally by the classes). This way, *madmom* offers the 'best of both worlds': concise interfaces exposed through classes, and detailed access to functionality through functions. In general, the classes can be split in two different types: the so called *data classes* and *processor classes*.

### DATA CLASSES

Represent data entities such as audio signals or spectrograms.

1 `http://www.sphinx-doc.org`
2 `http://www.travis-ci.org`
3 `http://www.quantifiedcode.com`
4 `http://www.coveralls.io`

They are implemented as subclasses of *NumPy*'s *ndarray*, and thus offer all array handling routines inherited directly from *NumPy* (e.g. transposing or saving the data to file in either binary or human readable format). These classes are enriched by additional attributes and expose additional functionality via methods.

PROCESSOR CLASSES
: Exclusively store information on how to process data, i.e. how to transform one data class into another (e.g. from a *Signal* into a *Spectrogram*). In order to build chains of transformations, each data class has its corresponding processor class, which implements this transformation. This enables a simple and fast conversion of algorithm prototypes to callable processing pipelines.

### 10.2.1 *Packages*

The library is split into several packages, grouped by functionality. For a detailed description including examples of usage please refer to the library's documentation.

#### *madmom.audio*

The *madmom.audio* package includes basic audio signal processing and "low-level" functionality.

The *Signal* and *FramedSignal* classes are used to load an audio signal and split it into (overlapping) frames. Following *madmom*'s automatic instantiation approach, both classes can be instantiated from any object up the instantiation hierarchy – including a simple file name. *madmom* supports almost a wide range of audio and video formats – provided *ffmpeg*[5] is installed – and transparently converts sample rates and number of channels if needed.

*Signal* is a subclass of *ndarray* with additional attributes like *sample rate* or *number of channels*. *FramedSignal* supports float hop sizes, making it possible to build systems with an arbitrary frame rate – independently of the signal's sample rate – and ensures that all frames are temporally aligned, even if computed with different frame sizes.

The *ShortTimeFourierTransform* and *Spectrogram* classes represent the complex valued STFT and magnitudes respectively. They are the key classes for spectral audio analysis and provide windowing, automatic circular shifting (for correct phase) and zero-padding. Both are independent of the data type (integer or float) of the underlying *Signal*, resulting in spectrograms of the same value range. A *Spectrogram* can be filtered with a *Filterbank* (e.g. Mel, Bark, logarithmic), which in turn can be parametrised to reduce the dimensionality or transform the spectrogram into a logarithmically spaced pitch representation closely

5 http://www.ffmpeg.org

following the auditory model of the human ear. *madmom* also provides standard *MFCC* and *Chroma* features.

Listing 1 shows an example of how a standard spectral flux onset detection function can be prototyped with *madmom* in a few lines of code.

```
from madmom.audio import (Spectrogram,
                          SpectrogramDifference)

# compute spectral flux
spec = Spectrogram('sample.wav')
diff = SpectrogramDifference(spec,
                             positive_diffs=True)
sf = np.mean(diff, axis=1)
```

Listing 1: Rapid prototyping of the spectral flux onset detection function using *madmom*'s data classes.

*madmom.processors*

*Processors* are one of the fundamental building blocks of *madmom*. Each *Processor* accepts a number of processing parameters and must provide a *process* method, which takes the data to be processed as its only argument and defines the processing functionality of the *Processor*. An *OutputProcessor* extends this scheme by accepting a second argument which defines the output and can thus be used to write the output of an algorithm to a file. All *Processors* are callable, making it easy to use them interchangeably with normal functions. Furthermore, the *Processor* class provides methods for saving and loading any *Processor* to a file – including all parameters – using Python's own *pickle* library. This facilitates the reproducibility of an experiment.

Multiple *Processors* can be combined into a processing chain using either a *SequentialProcessor* or a *ParallelProcessor*, which execute the chain sequentially or in parallel, using multiple CPU cores if available.

```
from madmom.audio import (
    SpectrogramProcessor,
    SpectrogramDifferenceProcessor)
from madmom.processors import SequentialProcessor
from functools import partial

# define spectral flux processing chain
spec = SpectrogramProcessor()
diff = SpectrogramDifferenceProcessor(
    positive_diffs=True)
mean = partial(np.mean, axis=1)
# wrap everything in a SequentialProcessor
sf_proc = SequentialProcessor([spec, diff, mean])

# process an audio file by calling the processor
sf = sf_proc('sample.wav')
```

Listing 2: Spectral flux onset detection implemented as a callable *Processor*.

Listing 2 shows the conversion of the prototyped algorithm in Listing 1 into a callable *Processor* by simply replacing the used *data classes* with their respective *processor classes* and wrapping them into a *SequentialProcessor*.

*madmom.features*

The *madmom.features* package includes "high-level" functionality which are related to certain MIR tasks, such as onset detection or beat tracking. *madmom*'s focus is on providing musically meaningful and descriptive features rather than a vast number of low to mid-level features. At the time of writing, *madmom* contains state-of-the-art features for onset detection, beat and downbeat tracking, rhythm pattern analysis, tempo estimation and chord recognition.

All features are implemented as *Processors* without a corresponding *data class*. Users can thus use the provided functionality and build algorithms on top of these features. For most of the features, *madmom* also provides stand-alone programs with a consistent calling interface to process audio files (see Section 10.2.2).

```
from madmom.features.beats import (
    RNNBeatProcessor, DBNBeatTrackingProcessor)
from madmom.processors import SequentialProcessor

# define beat tracking processor
rnn = RNNBeatProcessor()
dbn = DBNBeatTrackingProcessor(
    min_bpm=50, max_bpm=200)
tracker = SequentialProcessor([rnn, dbn])

# track the beats by calling the processor
beats = tracker('sample.wav')
```

Listing 3: Beat tracking with a recurrent neural network (RNN) and dynamic Bayesian network (DBN) in *madmom* using provided *Processors*.

*madmom.evaluation*

All features come with code for evaluation. The implemented metrics are those commonly found in the literature of the respective field.

*madmom.ml*

Most of today's top performing music analysis algorithms incorporate machine learning, with neural networks being the most universal and successful ones at the moment. *madmom* includes Python implementations of commonly used machine learning techniques, namely *Gaussian Mixture Models*, *Hidden Markov Models*, linear-chain *Conditional Random Fields*, and different types of *neural networks*, including feed forward, convolutional, batch normalisation, and recurrent layers, various activation functions and special purpose *long short-term memory* and gated recurrent units.

*madmom* provides functionality to use these techniques without any dependencies on third-party modules, but does not contain training algorithms. This decision was made on purpose since the library's main focus is on applying machine learning techniques to MIR, rather than providing an extensive set of learning techniques. However, trained models can be easily converted to be compatible with *madmom*, since neural network layers usually are simply defined as a set of weights, biases and an activation function they apply to the input data. Listing 4 shows the code necessary to convert a simple neural network trained using the *Lasagne* library [40].

```
import lasagne
from madmom import ml

# ... here comes the Lasagne training code
# 'net' is the Lasagne network handle
p = lasagne.layers.get_all_param_values(net)
# create a NeuralNetwork in madmom
nn = ml.nn.NeuralNetwork([
    ml.nn.layers.FeedForwardLayer(
        p[0], p[1], ml.nn.activations.relu),
    ml.nn.layers.FeedForwardLayer(
        p[2], p[3], ml.nn.activations.relu),
    ml.nn.layers.FeedForwardLayer(
        p[4], p[5], ml.nn.activations.relu),
    ml.nn.FeedForwardLayer(
        p[6], p[7], ml.nn.activations.sigmoid)
])
# save the network (can be used as a processor)
nn.dump('neural_net.pkl')
```

Listing 4: Converting a deep neural network with three hidden layers from *Lasagne* to *madmom*.

*madmom.models*

*madmom* comes with a set of pre-trained models which are distributed under a Creative Commons attribution non-commercial share-alike license, i.e. they can be freely used for research purposes as long as derivative works are distributed under the same license. *madmom* uses the exact same mechanism to save and load the models it uses for *Processors* to be pickled.

### 10.2.2 *Standalone programs*

*madmom* comes with a set of standalone programs, covering many areas of MIR. Table 10.1 lists selected programs included in the library with the performance achieved at the annual *Music Information Retrieval Evaluation eXchange* (MIREX) [6], where MIR algorithms are compared on test datasets. We aggregated the results of all years (2006-2016), i.e. a rank 1 means that the algorithm is the best performing one of all submissions from 2006 until present. The outstanding results in Table 10.1 highlight the state-of-the-art features *madmom* provides.

These programs are simple wrappers around the functionality provided by the *madmom.features* package, and provide a simple and easy to use command line interface. They are implemented as *Processors* and can operate either in *single* or *batch* mode, processing single or multiple input files, respectively. Additionally all programs can be

6 http://www.music-ir.org/mirex/wiki/

| PROGRAM | TASK | YEAR | RANK |
|---|---|---|---|
| CNNOnsetDetector [119] | onset | 2016 | 1 |
| OnsetDetector [55] | onset | 2013 | 2 |
| BeatTracker [17] | beat MCK | 2015 | 1 |
| DBNBeatTracker [14] | beat SMC | 2015 | 1 |
| CRFBeatDetector [83] | beat MAZ | 2015 | 1 |
| DBNDownBeatTracker [16] | downbeat | 2016 | 1 |
| TempoDetector [15] | tempo | 2015 | 1 |
| CNNChordRecognition [84] | chord | 2016 | 1 |

Table 10.1: Ranks of the programs included in *madmom* for the MIREX evaluations, results aggregated over all years (2006-2016).

*pickled*, serialising all parameters in a way that the program can be executed later on with the exact same settings.

```
#!/usr/bin/env python

from madmom.processors import (
    IOProcessor, io_arguments)
from madmom.features.beats import (
    RNNBeatProcessor, DBNBeatTrackingProcessor)
from madmom.utils import write_events
from argparse import ArgumentParser

if __name__ == '__main__':
    # define parser
    p = ArgumentParser()
    # add I/O related arguments and define
    # single/batch/pickle processing mode
    io_arguments(p, output_suffix='.beats.txt')
    # parse arguments
    args = p.parse_args()
    # define beat tracking processor
    rnn = RNNBeatProcessor()
    dbn = DBNBeatTrackingProcessor()
    out = write_events
    # wrap as IOProcessor
    processor = IOProcessor([rnn, dbn], out)
    # call the processor in single/batch/pickle
    # mode defined by 'args.func'
    args.func(processor, **vars(args))
```

Listing 5: Beat tracking algorithm of Listing 3 converted into a runnable program.

Listing 5 shows the beat tracking algorithm of Listing 3 converted into a runnable program. The usage of an *IOProcessor* instead of a

*SequentialProcessor* enables the program to output the detected beats into a file or to `STDOUT`. `io_arguments` provides all the handling needed for the different operation modes.

## 10.3 CONCLUSIONS

This paper gave a short introduction to *madmom*, its design principles and library structure. Up-to-date information on functionality can be found in the project's online documentation at https://madmom.readthedocs.io and source code repository at https://github.com/CPJKU/madmom.

Future work aims at including a streaming mode, i.e. providing online real-time processing of audio signals in a memory efficient way instead of processing whole audio files at a time. In addition, we will gradually extend the set of features and algorithms, as well as add tools to automatically convert models that have been trained with machine learning libraries.

# BIBLIOGRAPHY


[1] Andreas Arzt, Sebastian Böck, Sebastian Flossmann, Harald Frostel, Martin Gasser, Cynthia C. S. Liem, and Gerhard Widmer. "The Piano Music Companion". In: *Proceedings of the 21th European Conference on Artificial Intelligence (ECAI)*. Prague, Czech Republic, 2014, pp. 1221–1222.

[2] Andreas Arzt, Sebastian Böck, Sebastian Flossmann, Harald Frostel, Martin Gasser, and Gerhard Widmer. "The complete classical music companion v0.9". In: *Proceedings of the Audio Engineering Society 53rd International Conference on Semantic Audio*. London, UK, 2014, pp. 18–20.

[3] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. "Fast Identification of Piece and Score Position via Symbolic Fingerprinting". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012, pp. 433–438.

[4] Andreas Arzt, Gerhard Widmer, Sebastian Böck, Reinhard Sonnleitner, and Harald Frostel. "Towards a Complete Classical Music Companion". In: *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*. Montpellier, France, 2012, pp. 67–72.

[5] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. "Cython: The Best of Both Worlds". In: *Computing in Science Engineering* 13.2 (2011), pp. 31–39. ISSN: 1521-9615.

[6] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Matthew E. P. Davies, and Mark Sandler. "A Tutorial on Onset Detection in Music Signals". In: *IEEE Transactions on Speech and Audio Processing* 13.5 (2005), pp. 1035–1047. ISSN: 1063-6676.

[7] Juan Pablo Bello, Laurent Daudet, and Mark B. Sandler. "Automatic Piano Transcription Using Frequency and Time-Domain Information". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.6 (2006), pp. 2242–2251. ISSN: 1558-7916.

[8] Juan Pablo Bello, Chris Duxbury, Matthew Davies, and Mark Sandler. "On the use of phase and energy for musical onset detection in the complex domain". In: *IEEE Signal Processing Letters* 11.6 (2004), pp. 553–556. ISSN: 1070-9908.

[9] Emmanouil Benetos and Tillman Weyde. "An efficient temporally-constrained probabilistic model for multiple-instrument music transcription". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, pp. 701–707.

[10] Emmanouil Benetos and Tillman Weyde. "Polyphonic Transcription with Deep Layered Learning". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2015.

[11] Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl. "Online Real-time Onset Detection with Recurrent Neural Networks". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*. York, UK, 2012, pp. 301–304.

[12] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. "madmom: a new Python Audio and Music Signal Processing Library". In: *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[13] Sebastian Böck, Florian Krebs, and Markus Schedl. "Evaluating the Online Capabilities of Onset Detection Methods". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012, pp. 49–54.

[14] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "A multi-model approach to beat tracking considering heterogeneous music styles". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, pp. 603–608.

[15] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Accurate Tempo Estimation based on Recurrent Neural Networks and Resonating Comb Filters". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, pp. 625–631.

[16] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Joint Beat and Downbeat Tracking with Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016, pp. 255–261.

[17] Sebastian Böck and Markus Schedl. "Enhanced Beat Tracking with Context-Aware Neural Networks". In: *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*. Paris, France, 2011, pp. 135–139.

[18] Sebastian Böck and Markus Schedl. "Polyphonic Piano Note Transcription with Recurrent Neural Networks". In: *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan, 2012, pp. 121–124.

[19] Sebastian Böck, Jan Schlüter, and Gerhard Widmer. "Enhanced peak picking for onset detection with recurrent neural networks". In: *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*. Prague, Czech Republic, 2013, pp. 15–18.

[20] Sebastian Böck and Gerhard Widmer. "Local Group Delay based Vibrato and Tremolo Suppression for Onset Detection". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013, pp. 589–594.

[21] Sebastian Böck and Gerhard Widmer. "Maximum Filter Vibrato Suppression for Onset Detection". In: *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*. Maynooth, Ireland, 2013, pp. 55–61.

[22] Dmitry Bogdanov, Nicholas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R. Zapata, and Xavier Serra. "ESSENTIA: an Audio Analysis Library for Music Information Retrieval". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013, pp. 493–498.

[23] Gregor van den Boogaart and Rainer Lienhart. "Note onset detection for the transcription of polyphonic piano music". In: *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*. 2009, pp. 446–449.

[24] Paul Brossier, Juan Pablo Bello, and Mark D. Plumbley. "Real-time temporal segmentation of note objects in music signals". In: *Proceedings of the International Computer Music Conference (ICMC)*. Miami, FL, USA, 2004.

[25] Ali Taylan Cemgil, Bert Kappen, Peter Desain, and Henkjan Honing. "On Tempo Tracking: Tempogram Representation and Kalman Filtering". In: *Journal of New Music Research* 28:4 (2001), pp. 259–273.

[26] Trevor de Clercq and David Temperley. "A corpus analysis of rock harmony". In: *Popular Music* 30.1 (1 2011), pp. 47–70. ISSN: 1474-0095.

[27] Nick Collins. "A Comparison of Sound Onset Detection Algorithms with Emphasis on Psychoacoustically Motivated Detection Functions". In: *Proceedings of the 118th AES Convention*. 2005, pp. 28–31.

[28] Nick Collins. "Using a Pitch Detector for Onset Detection". In: *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*. London, UK, 2005, pp. 100–106.

[29] Nick Collins. "Towards a style-specific basis for computational beat tracking". In: *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*. Bologna, Italy, 2006, pp. 461–467.

[30] Tom Collins, Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Bridging the Audio-Symbolic Gap: The Discovery of Repeated Note Content Directly from Polyphonic Music Audio". In: *Proceedings of the Audio Engineering Society 53rd International Conference on Semantic Audio*. London, UK, 2014.

[31] Roger B. Dannenberg. "An On-Line Algorithm for Real-Time Accompaniment". In: *Proceedings of the 1984 International Computer Music Conference*. 1984, pp. 193–198.

[32] Matthew E. P. Davies and Sebastian Böck. "Evaluating the Evaluation Measures for Beat Tracking". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, pp. 637–642.

[33] Matthew E. P. Davies, Norberto Degara, and Mark D. Plumbley. *Evaluation Methods for Musical Audio Beat Tracking Algorithms*. Tech. rep. C4DM-TR-09-06. Centre for Digital Music, Queen Mary University of London, 2009.

[34] Matthew E. P. Davies and Mark D. Plumbley. "A Spectral Difference Approach to Downbeat Extraction in Musical Audio". In: *Proceedings of the 14th European Signal Processing Conference (EUSIPCO)*. Florence, Italy, 2006, pp. 121–124.

[35] Matthew E. P. Davies and Mark D. Plumbley. "Context-Dependent Beat Tracking of Musical Audio". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 1009–1020. ISSN: 1558-7916.

[36] Noberto Degara, Matthew E. P. Davies, Antonio Pena, and Mark Plumbley. "Onset Event Decoding Exploiting the Rhythmic Structure of Polyphonic Music". In: *IEEE Journal of Selected Topics in Signal Processing* 5.6 (2011), pp. 1228–1239. ISSN: 1932-4553.

[37] Norberto Degara, Enrique Argones-Rúa, Antonio Pena, Soledad Torres-Guijarro, Matthew E. P. Davies, and Mark D. Plumbley. "Reliability-Informed Beat Tracking of Musical Signals." In: *IEEE Transactions on Audio, Speech and Language Processing* 20.1 (2012), pp. 290–301.

[38] Bruno Di Giorgi, Massimiliano Zanoni, Sebastian Böck, and Augusto Sarti. "Multipath Beat Tracking". In: *Journal of the Audio Engineering Society* 64.7 (2016), pp. 493–502.

[39] Bruno Di Giorgi, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. "Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony". In: *8th International Workshop on Multidimensional Systems (nDS)*. Erlangen, Germany, 2013, pp. 145–150.

[40] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Eric Battenberg, Aäron van den Oord, et al. *Lasagne: First release.* 2015.

[41] Simon Dixon. "On the Computer Recognition of Solo Piano Music". In: *Proceedings of the Australasian Computer Music Conference*. 2000, pp. 31–37.

[42] Simon Dixon. "Automatic Extraction of Tempo and Beat from Expressive Performances". In: *Journal of New Music Research* 30 (2001), pp. 39–58.

[43] Simon Dixon. "Onset Detection Revisited". In: *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx)*. Montréal, Canada, 2006, pp. 133–137.

[44] Simon Dixon. "Evaluation of the Audio Beat Tracking System BeatRoot". In: *Journal of New Music Research* 36.1 (2007), pp. 39–50.

[45] Simon Durand, Juan Pablo Bello, Bertrand David, and Gaël Richard. "Downbeat tracking with multiple features and deep neural networks". In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia, 2015, pp. 409–413.

[46] Simon Durand, Juan Pablo Bello, Bertrand David, and Gaël Richard. "Feature Adapted Convolutional Neural Networks for Downbeat Tracking". In: *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, 2016, pp. 296–300.

[47] Simon Durand, Bertrand David, and Gaël Richard. "Enhancing downbeat detection when facing different music styles". In: *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy, 2014, pp. 3132–3136.

[48] Chris Duxbury, Juan Pablo Bello, Mike Davies, and Mark B. Sandler. "Complex Domain Onset Detection for Musical Signals". In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx)*. London, UK, 2003.

[49] Douglas Eck. "Beat Tracking using an Autocorrelation Phase Matrix". In: *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Honolulu, HI, USA, 2007, pp. 1313–1316.

[50] Daniel P. W. Ellis. "Beat tracking by dynamic programming". In: *Journal of New Music Research* (2007), pp. 51–60.

[51] Daniel P. W. Ellis and Graham E. Poliner. "Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking". In: *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Honolulu, HI, USA, 2007, pp. 1429–1432.

[52] Anders Elowsson and Anders Friberg. "Polyphonic Transcription with Deep Layered Learning". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2014.

[53] Anders Elowsson, Anders Friberg, Guy Madison, and Johan Paulin. "Modelling the Speed of Music Using Features from Harmonic/Percussive Separated Audio". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013, pp. 481–486.

[54] Valentin Emiya, Roland Badeau, and Bertrand David. "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18 (6 2010), pp. 1643–1654. ISSN: 1063-6676.

[55] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. "Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks". In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, The Netherlands, 2010, pp. 589–594.

[56] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. "Recent Developments in openSMILE, the Munich Open-Source Multimedia Feature Extractor". In: *Proceedings of ACM International Conference on Multimedia*. Barcelona, Spain, 2013, pp. 835–838.

[57] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and Models*. Springer Series in Information Sciences. Springer, 2007. ISBN: 9783642517655.

[58] Mikel Gainza and Eugene Coyle. "Tempo Detection Using a Hybrid Multiband Approach". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.1 (2011), pp. 57–68.

[59] Aggelos Gkiokas, Vassilios Katsouros, and George Carayannis. "Reducing Tempo Octave Errors by Periodicity Vector Coding And SVM Learning". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012, pp. 301–306.

[60] Aggelos Gkiokas, Vassilios Katsouros, George Carayannis, and Themos Stafylakis. "Music tempo estimation and beat tracking by applying source separation and metrical relations". In: *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan, 2012, pp. 421–424.

[61] John Glover, Victor Lazzarini, and Joseph Timoney. "Real-time detection of musical onsets with linear prediction and sinusoidal modeling". In: *EURASIP Journal on Advances in Signal Processing* 68.1 (2011). ISSN: 1687-6180.

[62] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. "RWC Music Database: Popular, Classical, and Jazz Music Databases". In: *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*. Paris, France, 2002, pp. 287–288.

[63] Masataka Goto and Yoichi Muraoka. "Beat Tracking based on Multiple-agent Architecture A Real-time Beat Tracking System for Audio Signals". In: *Proceedings of the International Conference on Multiagent Systems*. 1996, pp. 103–110.

[64] Fabien Gouyon and Simon Dixon. "A Review of Automatic Rhythm Description Systems". In: *Computer Music Journal* 29 (1 2005), pp. 34–54. ISSN: 0148-9267.

[65] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. "An experimental comparison of audio tempo induction algorithms". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.5 (2006), pp. 1832–1844. ISSN: 1558-7916.

[66] Fabien Gouyon, Gerhard Widmer, Xavier Serra, and Arthur Flexer. "Acoustic Cues to Beat Induction: A Machine Learning Perspective". In: *Music Percepcion. UCPress* 24 (2006), pp. 177–188.

[67] Alex Graves. "Supervised Sequence Labelling with Recurrent Neural Networks". PhD thesis. Technische Universität München, 2008.

[68] Stephen Hainsworth and Malcolm Macleod. "Particle filtering applied to musical tempo tracking". In: *EURASIP Journal on Applied Signal Processing* 15 (2004), pp. 2385–2395. ISSN: 1110–8657.

[69] Stephen Handel. *Listening: An Introduction to the Perception of Auditory Events*. MIT Press, 1989. ISBN: 9780262081795.

[70] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computing* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667.

[71] Jason Hockman, Matthew E. P. Davies, and Ichiro Fujinaga. "One in the Jungle: Downbeat Detection in Hardcore, Jungle, and Drum and Bass." In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal, 2012, pp. 169–174.

[72] André Holzapfel, Matthew E. P. Davies, José R. Zapata, João L. Oliveira, and Fabien Gouyon. "Selective Sampling for Beat Tracking Evaluation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.9 (2012), pp. 2539–2548. ISSN: 1558-7916.

[73] André Holzapfel, Florian Krebs, and Ajay Srinivasamurthy. "Tracking the "odd": meter inference in a culturally diverse music corpus". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, pp. 425–430.

[74] André Holzapfel, Yannis Stylianou, Ali C. Gedik, and Barış Bozkurt. "Three dimensions of pitched instrument onset detection". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.6 (2010), pp. 1517–1527.

[75] Florian Hörschläger, Richard Vogl, Sebastian Böck, and Peter Knees. "Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from Wikipedia". In: *Proceedings of the 12th Sound and Music Conference (SMC)*. Maynooth, Ireland, 2015.

[76] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2016-03-11]. 2001–.

[77] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. "On the Potential of Simple Framewise Approaches to Piano Transcription". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016, pp. 475–481.

[78] Maksim Khadkevich, Thomas Fillon, Gaël Richard, and Maurizio Omologo. "A probabilistic approach to simultaneous extraction of beats and downbeats". In: *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan, 2012, pp. 445–448.

[79] Anssi P. Klapuri. "Sound onset detection by applying psychoacoustic knowledge". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 6. Phoenix, AZ, USA, 1999, pp. 3089–3092.

[80] Anssi P. Klapuri, Antti J. Eronen, and Jaakko T. Astola. "Analysis of the meter of acoustic musical signals". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2006), pp. 342–355. ISSN: 1558-7916.

[81] Peter Knees, Kristina Andersen, Sergi Jordà, Michael Hlatky, Andres Bucci, Wulf Gaebele, and Roman Kaurson. "The GiantSteps project: A second-year intermediate report". In: *Proceedings of the 42nd International Computer Music Conference (ICMC)*. Utrecht, The Netherlands, 2016, pp. 363–368.

[82] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, pp. 364–370.

[83] Filip Korzeniowski, Sebastian Böck, and Gerhard Widmer. "Probabilistic extraction of beat positions from a beat activation function". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, pp. 513–518.

[84] Filip Korzeniowski and Gerhard Widmer. "A Fully Convolutional Deep Auditory Model for Musical Chord Recognition". In: *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Salerno, Italy, 2016.

[85] Filip Korzeniowski and Gerhard Widmer. "Feature Learning for Chord Recognition: The Deep Chroma Extractor". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016, pp. 37–43.

[86] Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. "Downbeat Tracking using Beat-Synchronous Features and Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York, NY, USA, 2016, pp. 129–135.

[87] Florian Krebs, Sebastian Böck, and Gerhard Widmer. "Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil, 2013, pp. 227–232.

[88] Florian Krebs, Sebastian Böck, and Gerhard Widmer. "An Efficient State Space Model for Joint Tempo and Meter Tracking". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Malaga, Spain, 2015, pp. 72–78.

[89] Florian Krebs, André Holzapfel, Ali Taylan Cemgil, and Gerhard Widmer. "Inferring Metrical Structure in Music Using Particle Filters". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.5 (2015), pp. 817–827. ISSN: 2329-9290.

[90] Florian Krebs, Filip Korzeniowski, Maarten Grachten, and Gerhard Widmer. "Unsupervised Learning and Refinement of Rhythmic Patterns for Beat and Downbeat Tracking". In: *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*. Lisboa, Portugal, 2014, pp. 611–615.

[91] Alexandre Lacoste and Douglas Eck. "A supervised classification algorithm for note onset detection". In: *EURASIP Journal on Applied Signal Processing* 1 (2007), pp. 153–153. ISSN: 1110-8657.

[92] Olivier Lartillot, Donato Cereghetti, Kim Eliard, Wiebke J. Trost, Marc-André Rappaz, and Didier Grandjean. "Estimating tempo and metrical features by tracking the whole metrical hierarchy". In: *Proceedings of the 3rd International Conference on Music & Emotion (ICME)*. Jyväskylä, Finland, 2013.

[93] Olivier Lartillot and Petri Toiviainen. "A Matlab toolbox for musical feature extraction from audio". In: *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx)*. Bordeaux, France, 2007, pp. 237–244.

[94] Bernhard Lehner, Gerhard Widmer, and Sebastian Böck. "A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks". In: *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*. Nice, France, 2015, pp. 21–25.

[95] Mark Levy. "Improving Perceptual Tempo Estimation with Crowd-Sourced Annotations." In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA, 2011, pp. 317–322.

[96] Ugo Marchand and Geoffroy Peeters. "Swing Ratio Estimation". In: *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx)*. Trondheim, Norway, 2015, pp. 423–428.

[97] Matija Marolt. "A connectionist approach to automatic transcription of polyphonic piano music". In: *IEEE Transactions on Multimedia* 6 (2004), pp. 439–449.

[98] Paul Masri. "Computer Modeling of Sound for Transformation and Synthesis of Musical Signals". PhD thesis. UK: University of Bristol, 1996.

[99] Raul Mata-Campos, Francisco Jose Rodriguez-Serrano, Pedro Vera-Candeas, Julio José Carabias-Orti, and Francisco Jesus Canadas Quesada. "BEAT TRACKING IMPROVED BY AM SINUSOIDAL BEAT TRACKING IMPROVED BY AM SINUSOIDAL MODELED ONSETS". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2010.

[100] Benoit Mathieu, Slim Essid, Thomas Fillon, and Jacques Prado. "YAAFE, an easy to use and efficient audio feature extraction software". In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, The Netherlands, 2010, pp. 441–446.

[101] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and Music Signal Analysis in Python". In: *Proceedings of the 14th Python in Science Conference (SCIPY)*. 2015.

[102] Martin F. McKinney, Dirk Moelants, Matthew E. P. Davies, and Anssi P. Klapuri. "Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms". In: *Journal of New Music Research* 36.1 (2007), pp. 1–16.

[103] Bernhard Niedermayer, Sebastian Böck, and Gerhard Widmer. "On the Importance of "Real" Audio Data for MIR Algorithm Evaluation at the Note-Level - A Comparative Study". In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA, 2011, pp. 543–548.

[104] João L. Oliveira, Fabien Gouyon, Luis G. Martins, and Luis Paulo Reis. "IBT: a Real-Time Tempo and Beat Tracking System". In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, The Netherlands, 2010, pp. 291–296.

[105] Hélène Papadopoulos and Geoffroy Peeters. "Joint Estimation of Chords and Downbeats From an Audio Signal". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.1 (2011), pp. 138–152. ISSN: 1558-7916.

[106] Geoffroy Peeters. "Beat-Marker Location Using a Probabilistic Framework and Linear Discriminant Analysis". In: *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx)*. Como, Italy, 2009, pp. 313–320.

[107] Geoffroy Peeters and Joachim Flocon-Cholet. "Perceptual tempo estimation using GMM-regression". In: *Proceedings of the 2nd international ACM workshop on Music information retrieval with user-centered and multimodal strategies (MIRUM)*. Nara, Japan, 2012, pp. 45–50.

[108] Geoffroy Peeters and Hélène Papadopoulos. "Simultaneous Beat and Downbeat-Tracking Using a Probabilistic Framework: Theory and Large-Scale Evaluation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.6 (2011), pp. 1754–1769. ISSN: 1558-7916.

[109] Graham Percival and George Tzanetakis. "Streamlined Tempo Estimation Based on Autocorrelation and Cross-correlation With Pulses". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.12 (2014), pp. 1765–1776. ISSN: 2329-9290.

[110] Fernando Pérez and Brian E. Granger. "IPython: A System for Interactive Scientific Computing". In: *Computing in Science Engineering* 9.3 (2007), pp. 21–29. ISSN: 1521-9615.

[111] Graham E. Poliner and Daniel P. W. Ellis. "A discriminative model for polyphonic piano transcription". In: *EURASIP Journal on Applied Signal Processing* 2007 (1 2007), pp. 154–154. ISSN: 1110-8657.

[112] Elio Quinton, Mark Sandler, and Simon Dixon. "Estimation of the reliability of multiple rhythm features extraction from a single descriptor". In: *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, 2016, pp. 256–260.

[113] Lawrence R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[114] Christopher Raphael. "Automatic Transcription of Piano Music". In: *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*. Paris, France, 2002.

[115] Matti P. Ryynänen and Anssi P. Klapuri. "Polyphonic music transcription using note event modeling". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2005, pp. 319–322.

[116] Markus Schedl, Peter Knees, and Sebastian Böck. "Investigating the Similarity Space of Music Artists on the Micro-Blogosphere". In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA, 2011, pp. 323–328.

[117] Eric D. Scheirer. "Tempo and beat analysis of acoustic musical signals". In: *The Journal of the Acoustical Society of America* 103.1 (1998), pp. 588–601.

[118] Olaf Schleusing, Bingjun Zhang, and Ye Wang. "Onset detection in pitched non-percussive music using warping-compensated correlation". In: *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada, 2013, pp. 117–120.

[119] Jan Schlüter and Sebastian Böck. "Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*. Prague, Czech Republic, 2013, pp. 79–82.

[120] Jan Schlüter and Sebastian Böck. "Improved Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy, 2014, pp. 6979–6983.

[121] Mike Schuster and Kuldip K. Paliwal. "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[122] Xavier Serra et al. *Roadmap for Music Information ReSearch*. Creative Commons BY-NC-ND 3.0 license, ISBN: 978-2-9540351-1-6, 2013.

[123] Yu Shiu, Namgook Cho, Pei-Chen Chang, and Jay C.-C. Kuo. "Robust on-line beat tracking with kalman filtering and probabilistic data association (KF-PDA)". In: *IEEE Transactions on Consumer Electronics* 54.3 (2008), pp. 1369–1377. ISSN: 0098-3063.

[124] Reinhard Sonnleitner, Bernhard Niedermayer, Gerhard Widmer, and Jan Schlüter. "A Simple and Effective Spectral Feature for Speech Detection in Mixed Audio Signals". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*. York, UK, 2012, pp. 377–383.

[125] Ajay Srinivasamurthy, André Holzapfel, and Xavier Serra. "In Search of Automatic Rhythm Analysis Methods for Turkish and Indian Art Music". In: *Journal of New Music Research* 43.1 (2014), pp. 94–114.

[126] Ajay Srinivasamurthy and Xavier Serra. "A Supervised Approach to Hierarchical Metrical Cycle Tracking from Audio Music Recordings". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy, 2014, pp. 5217–5221.

[127] Dan Stowell and Mark D. Plumbley. "Adaptive Whitening For Improved Real-time Audio Onset Detection". In: *Proceedings of the International Computer Music Conference (ICMC)*. 2007.

[128] Mi Tian, György Fazekas, Dawn A. A. Black, and Mark B. Sandler. "Design And Evaluation of Onset Detectors using Different Fusion Policies". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan, 2014, pp. 631–636.

[129] Mi Tian, György Fazekas, Dawn AA Black, and Mark Sandler. "On the use of the tempogram to describe audio content and its application to Music structural segmentation". In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia, 2015, pp. 419–423.

[130] Daylin Troxel. "Music Transcription with a Convolutional Neural Network". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2016.

[131] George Tzanetakis and Perry Cook. "MARSYAS: a framework for audio analysis". In: *Organised Sound* 4 (03 2000), pp. 169–175. ISSN: 1469-8153.

[132] George Tzanetakis and Perry Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on Speech and Audio Processing* 10.5 (2002), pp. 293–302. ISSN: 1063-6676.

[133] Nicholas Wack and Eduard Aylon. "Beat Detection using PLP". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2010.

[134] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2 (2011), pp. 22–30. ISSN: 1521-9615.

[135] Nick Whiteley, Ali Taylan Cemgil, and Simon Godsill. "Bayesian Modelling of Temporal Structure in Musical Audio". In: *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*. Victoria, Canada, 2006, pp. 29–34.

[136] Fu-Hai Frank Wu and Jyh-Shing Roger Jang. "A supervised learning method for tempo estimation of musical audio". In: *22nd Mediterranean Conference of Control and Automation (MED)*. Palermo, Italy, 2014, pp. 599–604.

[137] José R. Zapata, Matthew E. P. Davies, and Emilia Gómez. "Multi-feature beat tracking". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.4 (2014), pp. 816–825. ISSN: 2329-9290.

[138] José R. Zapata and Emilia Gómez. "Comparative evaluation and combination of audio tempo estimation approaches". In: *Proceedings of the AES 42nd International Conference on Semantic Audio*. Ilmenau, Germany, 2011. ISBN: 978-0-937803-81-3.

[139] Zhengchen Zhang, Dong-yan Huang, Renbo Zhao, and Minghui Dong. "Onset detection based on fusion of Simpls and SuperFlux". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2013.

[140] Ruohua Zhou, Marco Mattavelli, and Giorgio Zoia. "Music Onset Detection Based on Resonator Time Frequency Image". In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.8 (2008), pp. 1685–1695. ISSN: 1558-7916.

[141] Yongwei Zhu, Hui Li Tan, and Lekha Chaisorn. "Audio Beat Tracking Algorithm from Institute for Infocomm Research". In: *Music Information Retrieval Evaluation eXchange (MIREX) Abstracts*. 2010.

# CURRICULUM VITÆ

## PERSONAL DATA

Dipl.-Ing. Univ. Sebastian Böck

born 1977, Munich

http://phd.minimoog.org

## EDUCATION

| | |
|---|---|
| 2010–2016 | Doctoral studies in computer science, Johannes Kepler University Linz |
| 2002–2010 | Diploma studies in electrical and computer engineering, Technical University Munich |

## WORK

| | |
|---|---|
| 2010–2016 | Researcher at the Department of Computational Perception, Johannes Kepler University Linz |
| 1996–2010 | Sound and recording engineer |

## SCIENTIFIC SERVICES

| | |
|---|---|
| PC member | Society for Music Information Retrieval Conference (ISMIR) |
| Reviewer | Society for Music Information Retrieval Conference (ISMIR), Conference on Digital Audio Effects (DAFx), Sound and Music Conference (SMC), IEEE International Conference on Multimedia and Expo (ICME), ACM International Conference on Multimedia (ACMMM), Journal of new music research (JNMR), Transactions on Audio, Speech, and Language Processing (TASLP), Transactions on Signal Processing (TSP), Journal of Multimedia Information Retrieval (MMIR) |