BEAT TRACKING WITH A CEPSTROID INVARIANT NEURAL NETWORK

Anders Elowsson
KTH Royal Institute of Technology
elov@kth.se

ABSTRACT

We present a novel rhythm tracking architecture that learns how to track tempo and beats through layered learning. A basic assumption of the system is that humans understand rhythm by letting salient periodicities in the music act as a framework, upon which the rhythmical structure is interpreted. Therefore, the system estimates the cepstroid (the most salient periodicity of the music), and uses a neural network that is invariant with regards to the cepstroid length. The input of the network consists mainly of features that capture onset characteristics along time, such as spectral differences. The invariant properties of the network are achieved by subsampling the input vectors with a hop size derived from a musically relevant subdivision of the computed cepstroid of each song. The output is filtered to detect relevant periodicities and then used in conjunction with two additional networks, which estimates the speed and tempo of the music, to predict the final beat positions. We show that the architecture has a high performance on music with public annotations.

1. INTRODUCTION

The beats of a musical piece are salient positions in the rhythmic structure, and generally the pulse scale that a human listener would tap their foot or hand to in conjunction with the music. As such, beat positions are an emergent perceptual property of the musical sound, but in various cases also dictated by conventional methods of notating different musical styles. Beat tracking is a popular subject of research within the Music Information Retrieval (MIR) community. At the heart of human perception of beat are the onsets of the music. Therefore, onset detection functions are commonly used as a front end for beat tracking. The most basic property that characterize these onsets is an increase in energy in some frequency bands. Extracted onsets can either be used in a discretized manner as in [9, 18, 19], or continuous features of the onset detection functions can be utilized [8, 23, 28]. As information in the pitch domain of music is important, chord changes can also be used to guide the beat tracking [26].

After relevant onset functions have been extracted, the periodicities of the music are usually determined by e.g. comb filters [28], the autocorrelation function [10, 19], or by calculating the cepstroid vector [11]. Other ways to understand rhythm are to explicitly model the rhythmic patterns [24], or to combine several different models to get better generalization capabilities [4]. To estimate the beat positions, hidden Markov models [23] or dynamic Bayesian networks (DBNs) have been used [25, 30].

Although onset detection functions often are computed by the spectral flux (SF) of the audio, it has become more common to learn onset detection functions with a neural network (NN) [3, 29]. Given the success of these networks it is not surprising that the same framework has been successfully used also for detecting beat positions [2]. When these network try to predict beat positions, they must understand how different rhythmical elements are connected; this is a very complex task.

1.1 Invariant properties of rhythm

When trying to understand a new piece of music, the listener must form a framework onto which the elements of the music can be deciphered. For example, we use scales and harmony to understand pitch in western music. The tones of a musical piece are not classified by their fundamental frequency, but by their fundamental frequency in relation to the other tones in the piece. In the same way, for the time dimension of music, the listener builds a framework, or *grid*, across time to understand how the different sounds or onsets relate to each other. This framework need not initially be at the beat level. In fact, in various music pieces, beat positions are not the first perceptually emergent timing property of the music. In some pieces, we may first get a strong sense of repetition at downbeat positions, or at subdivisions of the beat. In either of these cases, we identify beat positions after an initial framework of rhythm has been established. If we could establish such a correct framework for a learning algorithm, it would be able to build better representations of the rhythmical structure, as the input features would be deciphered within an underlying metrical structure. In this study we try to use this idea to improve beat tracking.

2. METHOD

In the proposed system we use multiple neural networks that each try to model different aspects related to rhythm, as shown in Figure 1. First we process the audio with harmonic/percussive source separation (HP-separation) and multiple fundamental frequency (MF_0) estimation. From the processed audio, features are calculated that capture onset characteristics along time, such as the SF and the pitch flux (PF). Then we try to find the most salient periodicity of the music (which we call the *cepstroid*), by analyzing histograms of the previously calculated onset characteristics in a NN (Cep Network). We use the cepstroid to subsample the flux vectors with a hop size derived from a subdivision of the computed cepstroid. The subsampled vectors are used as input features in our cepstroid invariant neural network (CINN). The CINN can track beat positions in complex rhythmic patterns, because the previous processing has made the input vectors invariant with regards to the cepstroid of the music. This means that the same neural activation patterns can be used for MEs of different tempi. In addition, the speed of the music is estimated with an ensemble of neural networks, using global features for onset characteristics as input. As the last learning step, the tempo is estimated. This is done by letting an ensemble of neural networks evaluate different plausible tempo candidates. Finally, the phase of the beat is determined by filtering the output of the CINN in conjunction with the tempo estimate; and beat positions are estimated.

An overview of the system is given in Figure 1. In Sections 2.1-2.4 we describe the steps to calculate the input features of our NNs and in Section 2.5 we give an overview of the NNs. In Section 2.6-2.9 we describe the different NNs, and in Section 2.10, we describe how the phase of the beat is calculated.

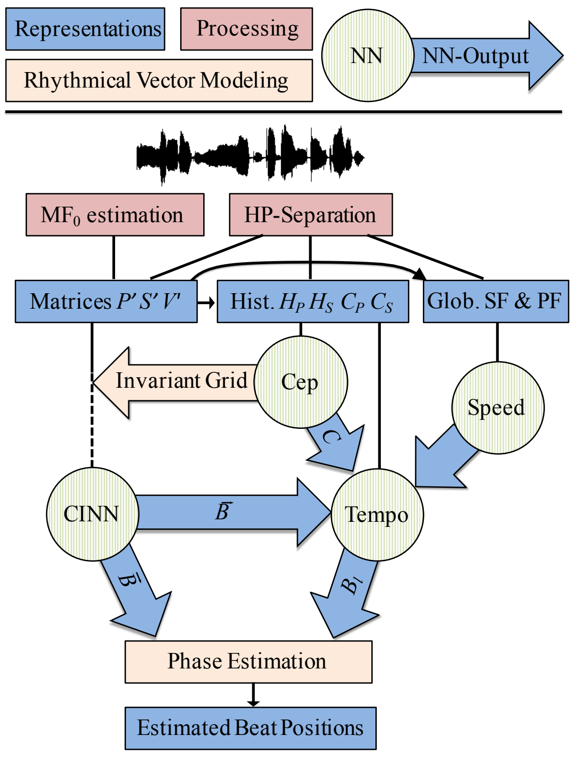

Figure 1. Overview of the proposed system. The audio is first processed with MF_0 estimation and HP-separation. Raw input features for the neural networks are computed and the outputs of the neural networks are combined to build a model of tempo and beats in each song.

2.1 Audio Processing

The audio waveform was converted to a sampling frequency of 44.1 kHz. Then, as a first step, HP-separation was applied. This is a common strategy (e.g. [16]), used to isolate the percussive instruments, so that subsequent learning algorithms can accurately analyze their rhythmic patterns. The source separation of our implementation is based on the method described in [15]. With a median filter across each frame in the frequency direction of a spectrogram, harmonic sounds are detected as outliers, and with a median filter across each frequency bin in the time direction, percussive sounds are detected as outliers. We use these filters to extract a percussive waveform P_1 and a harmonic waveform H_1, from the original waveform O. We further suppress harmonic sounds in P_1 (such as traces of the vocals or the bass guitar) by applying a median filter in the frequency direction of the Constant-Q transform (CQT), as described in [11, 13]. This additional filtering produces a clean percussive waveform P_2, and a harmonic waveform H_2 consisting of the traces of pitched sounds filtered out from P_1.

The task of tracking MF_0s of the audio is usually performed by polyphonic transcription algorithms (e.g. [1]). From several of these algorithms, the frame-wise MF_0s can be extracted at the semi-tone level. We used a frame-wise estimate from [14], extracted at a hop size of 5.8 ms (256 samples).

2.2 Calculating Flux Matrices P', S' and V'

Three types of flux matrices (P', S' and V') were calculated, all extracted at a hop size of 5.8 ms.

2.2.1 Calculating P'

Two spectral flux matrices (P'_1 and P'_2) were calculated from the percussive waveforms P_1 and P_2. The short time Fourier transform (STFT) was applied to P_1 and P_2 with a window size of 2048 samples and the spectral flux of the resulting spectrograms was computed. Let $X_{i,j}$ represent the magnitude at the ith frequency bin of the jth frame of the spectrograms. The SF for each bin is then given by

$$P'_{i,j} = X_{i,j} - X_{i,j-s} \qquad (1)$$

In this implementation we used a step size s of 7 (40 ms).

2.2.2 Calculating V'

The vibrato suppressed SF was computed for waveforms containing instruments with harmonics (H_1, H_2 and O), giving the flux matrices (V'_{H_1}, V'_{H_2} and V'_O). We used the algorithm for vibrato suppression first described in [12] (p. 4), but changed the resolution of the CQT to 36 bins per octave (down from 60) to get a better time resolution.

First, the spectrogram is computed with the CQT. Then, shifts of a peak by one bin, without an increase in sound level, are suppressed by subtracting the sound level of each bin of the new frame, with the maximum sound level of the adjacent bins in the old frame. This means that for the vibrato suppressed SF (V'), Eqn (1) is changed by including adjacent bins and calculating the maximum value before applying the subtraction.

$$V'_{i,j} = X_{i,j} - \max(X_{i-1,j-s}, X_{i,j-s}, X_{i+1,j-s}) \quad (2)$$

2.2.3 Calculating S'

When listening to a melody, we use pitch in conjunction with onset positions to infer the rhythmical structure. Therefore, it seems beneficial to utilize the pitch dimension of music in the beat tracking as well. We calculated the PF by applying the same function as described for the SF in Eqn (1) to the "*semigram*" – the estimated MF$_0$s in a *pitchogram*, interpolated to a resolution of one semitone per bin. The output is the rate of change in the *semigram*, covering pitches between midi pitch 26 and 104, and we will denote this feature matrix as S'.

2.3 Calculating Histograms H_P, H_S, C_P, and C_S

Next we compute two periodicity histograms H_P and H_S from the flux matrices P'_1 and S', and then transform them into the cepstroid vectors C_P and C_S.

The processing is based on a method recently introduced in [11]. In this method, a periodicity histogram of inter-*onset* intervals (IOIs) is computed, with the contribution of each onset-pair determined by their spectral similarity and their perceptual impact. The basic idea is that the IOI of two strong onsets with similar spectra (such as two snare hits) should constitute a relevant level of periodicity in the music. In our implementation we instead apply the processing frame-wise on P'_1 and S', using the spectral similarity and perceptual impact at each inter-*frame* interval. We use the same notion of spectral similarity and perceptual impact as in [11] when computing H_P from P'_1, but when we compute H_S from S', the notion of *spectral distance* is replaced with *tonal distance*. First we smooth S' in the pitch direction with a Hann window of size 13 (approximately an octave). We then build a histogram of tonal distances for each frame, letting n represent the nth semitone of S' and k the kth frame, giving us the tonal distance at all histogram positions a

$$\forall a \in \{1, \cdots, 1900\} \sum_{i=-50,-45,\cdots,50} \sum_{n=26}^{104} |S'^{n}_{k+i} - S'^{n}_{k+i+a}| \quad (3)$$

By using the grid defined by i in Eqn (3), we try to capture similarities in a few consecutive tones. The grid stretches over 100 frames, which corresponds to roughly 0.5 seconds. The idea is that repetitions of small motives occurs at musically relevant periods.

To get the cepstroid vector from a histogram, the discrete cosine transform (DCT) is first applied. The resulting spectrum unveils periodically recurring peaks of the histogram. In this spectral representation, frequency represents the period length and magnitude corresponds to salience in the metrical structure. We then interpolate back to the time domain by inserting spectral magnitudes at the position corresponding to their wavelength. Finally, the Hadamard product of the original histogram and the transformed version is computed to reduce noise. The result is a cepstroid vector (C_P, C_S). The name *cepstroid* (derived from *period*) was chosen based on similarities to how the *cepstrum* is computed from the *spectrum*.

2.4 Calculating Global SF and PF

Global features for the SF and PF were calculated for our speed estimation. We extracted features from the feature matrices of Section 2.2. The matrices were divided into log-spaced frequency bands over the entire spectrum by applying triangular filters as specified in Table 1.

Feature Matrices	P'_1	P'_2	S'	V'_O	V'_{H_1}	V'_{H_2}
Number of bands	3	3	1,2,4	3	3	3

Table 1. The feature matrices are divided into bands.

After the filtering stage we have 22 feature vectors, and each feature vector X is converted into 12 global features. We compute the means $\bar{X}$, $\overline{X^{0.2}}$ and $\overline{X^{0.5}}$, where 0.2 and 0.5 represents the element-wise power (3 features). Also, X is sorted based on magnitude into percentiles, and Hann windows of widths {41, 61}, centered at percentiles {31, 41} are applied (4 features). We finally extract the percentiles at values {20, 30, 35, 40, 50} (5 features).

2.5 Neural Network Settings

Here we define the settings for all neural networks. In the subsequent Sections 2.6-2.9, further details are provided for each individual NN. All networks were standard feed-forward neural networks with one to three hidden layers.

2.5.1 Ensemble Learning

We employed ensemble learning by creating multiple instances of a network and averaging their predictions. The central idea behind ensemble learning is to use different models that are better than random and more or less uncorrelated. The average of these models can then be expected to provide a better prediction than randomly choosing one of them [27]. For the Tempo and Speed networks, we created an ensemble by randomly selecting a subset of the features for the training of 20 networks (Tempo) or 60 networks (Speed). For the CINN, only 3 networks were used in the ensemble due to time constraints, and all features were used in each network.

2.5.2 Target values

The target values in the networks are defined as:

- **Cep** - Classifying if a frame represents a correct (1) or an incorrect cepstroid (0). The beat interval, downbeat interval, and duple octaves above the downbeat or below the beat were defined as correct.

- **CINN** - Classifying if the current frame is at a beat position (1), or if it is not at a beat position (0).
- **Speed** - Fitting to the log of the global beat length.
- **Tempo** - Classifying which of two tempo candidates that is correct (1) and which is incorrect (0).

2.5.3 Settings of the Neural Networks

We use scaled conjugate descent to train the networks. In Table 2, settings of the neural networks are defined.

	Hidden	***Epoch***	***EaSt***	***EnLe***	***OL***
Cep	{20, 20, 20}	600	100	-	*LoSi*
CINN	{25}	1000		3_{-}	*LoSi*
Speed	{6, 6, 6}	20	4	60_{40}	*Li*
Tempo	{20, 20}	100		20_{60}	*LoSi*

Table 2. The settings for the neural networks of the system. *Hidden*, denotes the size of the hidden layers and *Epoch* is the maximum number of epochs we ran the network. *EaSt* defines how many epochs without an increase in performance that were allowed for the internal validation set of the neural networks. *EnLe* is specified as $\mathrm{NE}_{\mathrm{NF}}$, where NE is the number of NNs and NF is the number of randomly drawn features for each NN. *OL* specifies if a logistic activation function (*LoSi*) or a linear summation (*Li*) was used for the output layer.

The activation function of the first hidden layer was always a hyperbolic tangent (tanh) unit, and for subsequent hidden layers it was always a rectified linear unit (ReLU). The use of a mixture of tanh units and ReLUs may seem unconventional but can be motivated. The success of ReLUs is often attributed to their propensity to alleviate the problem of vanishing gradients [17]. Vanishing gradients are often introduced by sigmoid and tanh units when those units are placed in the later layers, because gradients flow backwards through the network during training. With tanh units in the first layer, only gradients for one layer of weight and bias values will be affected. At the same time, the network will be allowed to make use of the smoother non-linearities of the tanh units.

2.6 Cepstroid Neural Network (Cep)

In the first NN we compute the most salient periodicity of the music. To do this we use the cepstroid vectors (C_P and C_S) previously computed in Section 2.3. First, two additional vectors are created from both cepstroid vectors by filtering the vectors with a Gaussian $\sigma = 7.5$, and a Laplacian of a Gaussian $\sigma = 7.5$. Then we include octave versions, by interpolating to a time resolution given by

$$\left(\frac{1}{2}\right)^n, \qquad \left(\frac{1}{2}\right)^n \times \left(\frac{1}{3}\right), \qquad \forall n \in \{-2,-1,0,1,2\} \tag{4}$$

Finally, much like one layer and one receptive field of a convolutional neural network, we go frame by frame through the vectors, trying to classify each histogram frame as correct or incorrect, depending on if that particular time position corresponds to a correct cepstroid. The input features are the magnitude values of the vectors at each frame. As true targets, the beat interval *and* the downbeat interval, as well as duple octaves above the downbeat and duple octaves below the beat are used. The output of the network is our final cepstroid vector (C) and the highest peak is used as our cepstroid ($\hat{C}$).

2.7 Cepstroid Invariant Neural Network (CINN)

After the cepstroid has been computed, we use it to derive the hop size h for our grid in each ME, at which we will subsample the input vectors of the network. By setting h to an appropriate multiple of the cepstroid, the input vectors of songs with different tempo (but potentially a similar rhythmical structure) will be synchronized; and the network can therefore make use of the same neural activation patterns for MEs of different tempi. This enables the CINN to easily identify distinct rhythmical patterns (similar to the ability of a human listener). We want a hop size between approximately 50-100 ms, and therefore compute which duple ratio of 70 ms that is closest to the current cepstroid

$$\min_{n=\cdots,-2,-1,0,1,2,\cdots} \left|\log_2 \frac{70}{\hat{C}/2^n}\right| \tag{5}$$

The value of n, which minimizes the function above, is then used to calculate the hop size h of the ME by

$$h = \frac{\hat{C}}{2^n} \tag{6}$$

The rather coarse hop size (50-100 ms) is used as we wish to include features from several seconds of audio, without the input layer becoming too large. However, to make the network aware of peaks that slips through the coarse grid, we perform a peak picking on the vector $\overrightarrow{P_1'}$, which we have first computed by summing P_1' across frequency. For each grid position, we write the magnitude of the closest peak, the absolute distance to the closest peak, as well as the sign of the computed distance to three feature vectors that we will denote by $\hat{P}$.

Just as for the speed features described in Section 2.4, we filter the feature matrices P_1', S' and V_O' with triangular filters to extract feature vectors. In summary, for each grid position, we extract features by interpolating over the 16 feature vectors defined in Table 3.

Feature	P'_1	$\hat{P}$	S'	V_O
Number of bands/features	6	3	6	1

Table 3. Feature vectors that are interpolated to the grid defined by the cepstroid.

For each frame we try to estimate if it corresponds to a beat (1) or not (0). We include 38 grid-points in each direction from the current frame position, resulting in a time window of $2 \cdot h \cdot 38$ seconds. At $h = 70$ ms, the time window is approximately 5.3 seconds. The computed beat activation from the CINN will be denoted as the beat vector $\vec{B}$ in the subsequent processing.

2.8 Speed Neural Network

Octave errors are a prevalent problem in tempo estimation and beat tracking, and different methods for choosing the correct tempo octave have previously been proposed [13]. It was recently shown that a *continuous* measure of the speed of the music can be very effective at alleviating octave errors [11]. We therefore compute a continuous speed estimate, which guides our tempo estimation, using the input features described in Section 2.4. The ground truth annotation of speed A_s, is derived from the logarithm of the annotated beat length AB_l

$$A_s = \log_2 AB_l \tag{7}$$

Eqn (7) is motivated by our logarithmic perception of tempo [6]. As we have very few annotations (1 per ME), we increase the generalization capabilities with ensemble learning. We also use an inner cross validation (5-fold) for the training set. If this is not done, the subsequent tempo network will overestimate the relevance of the computed speed, rendering a decrease in test performance.

2.9 Tempo Neural Network

The tempo is estimated by finding tempo candidates, and letting the neural network perform a classification between extracted candidates to pick the most likely tempo. First, the candidates are extracted by creating a histogram $H_{\vec{B}}$ of the beat vector $\vec{B}$ (that we previously extracted with the CINN). The energy at each histogram bin is computed as the sum of the product of the magnitudes of the frames of $\vec{B}$ at the frame offset given by a

$$\forall a \in \{1, \cdots, 1900\} \qquad \sum_i \vec{B}_i \cdot \vec{B}_{i+a} \tag{8}$$

We process the histogram to extract a cepstroid vector $C_{\vec{B}}$, by using the same processing scheme as described for C_P in Section 2.3. Peaks are then extracted in both $H_{\vec{B}}$ and $C_{\vec{B}}$, and the corresponding beat length of the histogram peaks are used as tempo candidates.

The neural network is not directly trained to classify if a tempo candidate is correct or incorrect. Instead, to create training data, each possible pair of tempo candidates are examined, and the network is trained to classify which of the two candidates in the pair that correspond to the correct tempo (using only pairs with one correct candidate for the training data).

For testing, the tempo candidate that receives the highest probability in its match-ups against the other candidates is picked as the tempo estimate. This idea was first described in [11] (in that case without using any preceding beat tracking and using a logistic regression without ensemble learning). Input features are defined for both tempo candidates in the pair by their corresponding beat length B_l. We compute:

- The magnitude at B_l in $H_{\vec{B}}$, $C_{\vec{B}}$ and in the feature vectors used for the Cep NN (see Section 2.6). We include octave ratios as defined in Eqn (4).
- We compute $x = \log_2 B_l - Speed$. Then $\operatorname{sgn}(x)$ and $|x|$ are used as features.
- A Boolean vector for all musically relevant ratios defined in Eqn (4), where the corresponding index is 1 if the pair of tempo candidates have that ratio.

We constrain possible tempo candidates to the range 23-270 BPM. This range is a bit excessive for the given datasets, but will allow the system to generalize better to other types of music with more extreme tempi.

2.10 Phase Estimation

At the final stage, we detect the phase of the beat vector and estimate the beat positions. The tempo often drifts slightly in music, for example during performances by live musicians. To model this in a robust way, we compute the CQT of the beat vector. The result is a spectrogram where each frequency corresponds to a particular tempo, the magnitude corresponds to beat strength, and where the phase corresponds to the phase of the beat at specific time positions. The beat vector is upsampled (100 times higher resolution) prior to applying the CQT, and we use 60 bins per octave. We filter the spectrogram with a Hann window of width one tempo octave (60 bins), centered at the frequency that corresponds to the previously computed tempo. As a result, any magnitudes outside of the correct tempo octave are set to 0 in the spectrogram. When the inverse CQT (ICQT) is finally applied to the filtered spectrogram, the result is a beat vector which resembles a sinusoid, where the peaks correspond to tentative beat positions. With this processing technique we have jointly estimated the phase and drift, using a fast transform which seems to be suitable for beat tracking.

The beat estimations are finally refined slightly by comparing the peaks of the computed sinusoidal beat vector with the peaks of the original beat vector from the CINN. Let us define a grid i, consisting of 100 points, onto which we interpolate phase deviations that are within $\pm$ 40 % of the estimated beat length. We then create a "driftogram" M by evaluating each estimated beat position j, adding 1 to each drift position $M_{i,j}$ where a peak was found in the original beat vector. The driftogram is smoothed with a Hann window of size 17 across the beat direction and size 27 across the drift direction. To adjust the beat position, we use the the maximum value for each beat frame of M.

3. EVALUATION

3.1 Datasets

We used the three datasets defined in Table 4 to evaluate our system. The Ballroom datasets consist of ballroom dance music and was annotated by [20, 24]. The Hainsworth dataset [21] is comprised of varying genres, and the SMC dataset [22] consists of MEs that were chosen based on their difficulty and ambiguity. Tempo annotations were computed by picking the highest peak of a smoothed histogram of the annotated inter-beat intervals.

Dataset	**Number of MEs**	**Length**
Ballroom	698	6h 4m
Hainsworth	222	3h 20m
SMC	217	2h 25m

Table 4. Datasets used for evaluation, and their size.

3.2 Evaluation Metrics

There are several different metrics for beat tracking, all trying to capture different relevant aspects of the performance. For an extensive review of different evaluation metrics, we refer the reader to [7].

F-measure is calculated from *Recall* and *Accuracy*, using a limit of $\pm$ 70 ms for the beat positions. **P-Score** measures the correlation between annotations and detections. **CMLc** is derived by finding the longest *Correct Metrical Level* with *continuity* required and **CMLt** is similar to CMLc but does not require continuity. **AMLc** is derived by finding the longest *Allowed Metrical Level* with continuity required. This measure allows for several different metrical levels and off-beats. **AMLt** is Similar as AMLc but does not require continuity. The standard tempo estimation metric **Acc$_1$** was computed from the output of the Tempo Network. It corresponds the fraction of MEs that are within 8 % of the annotated tempo.

3.3 Evaluation procedure

We used a 5-fold cross validation to evaluate the system on the Ballroom dataset. More specifically, the training fold was used to train all the different neural networks of the system. After all networks were trained, the test fold was evaluated on the complete system and the results returned. Then the procedure was repeated with the next train/test-split. The Hainsworth and SMC datasets were evaluated by running the MEs on a system previously trained on the complete Ballroom dataset.

As a benchmark for our cross-fold validation results on the Ballroom dataset, we use the cross-fold validation results of the state-of-the-art systems for tempo estimation [5], and beat tracking [25]. The systems were evaluated on a song-by-song basis with data provided by the authors. To make statistical tests we use bootstrapping for paired samples, with a significance level of $p < 0.01$. For the Hainsworth and SMC dataset, benchmarking is most appropriate with systems that were trained on separate training sets. We use [16] as a benchmark for tempo estimation, and [8] as a benchmark for beat tracking.

4. RESULTS

4.1 Tempo

The tempo estimation results (Acc$_1$), are shown in Table 5, together with the results of the benchmarks.

(Acc$_1$)	**Ballroom**	**Hainsworth**	**SMC**
Proposed	**0.973***	**0.802**	**0.332**
Böck [5]	**0.947***	0.865*	0.576*
Gkiokas [16]	0.625	**0.667**	**0.346**

Table 5. The results for our tempo estimation system in comparison with the benchmarks. Results marked with (*) were obtained from cross-fold validation. Results in bold are most relevant to compare. Statistical significance for systems with song-by-song data in comparison with the proposed system is underlined.

4.2 Beat tracking

Table 6 shows the performance of the system, evaluated as described in Section 3.2.

Ballroom	**F-Me**	**P-Sc**	**CMLc**	**CMLt**	**AMLc**	**AMLt**
Proposed	**92.5***	**92.2***	**86.8***	**90.3***	**89.4***	**93.2***
Krebs [25]	**91.6***	**88.8***	**83.6***	**85.1***	**90.4***	**92.2***
Hainsworth						
Proposed	**74.2**	**77.7**	**57.6**	**67.6**	**65.0**	**79.2**
Davies [8]	**-**	**-**	**54.8**	**61.2**	**68.1**	**78.9**
SMC						
Proposed	37.5	49.4	14.9	22.5	20.9	33.2

Table 6. The results for our proposed system in comparison with the benchmarks. Results marked with (*) were obtained from a cross-fold validation. Statistical significance for systems with song-by-song data in comparison with the proposed system is underlined.

5. SUMMARY & CONCLUSIONS

We have presented a novel beat tracking and tempo estimation system that uses a cepstroid invariant neural network. The many connected networks make it possible to explicitly capture different aspects of rhythm. With a Cep network we compute a salient level of repetition of the music. The invariant representations that were computed by subsampling the feature vectors allowed us to obtain an accurate beat vector in a CINN. By applying the CQT to the beat vector, and then filtering the spectrogram to keep only magnitudes that corresponds to the estimated tempo before applying the ICQT, we computed the phase of the beat. Alternative post processing strategies, such as applying a DBN on the beat vector, could potentially improve the performance. The results are comparable to the benchmarks both for tempo estimation and beat tracking. This indicates that the ideas put forward in this paper are important, and we hope that they can inspire new network architectures for MIR. Tests on hidden datasets for the relevant MIREX tasks would be useful to draw further conclusion regarding the performance.

6. ACKNOWLEDGEMENTS

Thanks to Anders Friberg for helpful discussions as well as proofreading. This work was supported by the Swedish Research Council, Grant Nr. 2012-4685.

7. REFERENCES

[1] E. Benetos: "Automatic transcription of polyphonic music exploiting temporal evolution," *Dissertation*. Queen Mary, University of London, 2012.

[2] S. Böck and M. Schedl: "Enhanced beat tracking with context aware neural networks," In *Proc. of DAFx,* 2011.

[3] S. Böck, A. Arzt, F. Krebs, and M. Sched: "Online real-time onset detection with recurrent neural networks," In *Proc. of DAFx,* 2012.

[4] S. Böck, F. Krebs, and G. Widmer: "A Multi-model Approach to Beat Tracking Considering Heterogeneous Music Styles," In *Proc. of ISMIR*, 2014.

[5] S. Böck, F. Krebs, and G. Widmer: "Accurate tempo estimation based on recurrent neural networks and resonating comb filters," In *Proc. of ISMI*R, pp. 625-631, 2015.

[6] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing: "On tempo tracking: Tempogram Representation and Kalman filtering," *J. New Music Research*, Vol. 29, No. 4, pp. 259-273, 2000.

[7] M. E. P. Davies, N. Degara, and M. D. Plumbley: "Evaluation methods for musical audio beat tracking algorithms," Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06, 2009.

[8] M. Davies and M. Plumbley: "Context-dependent beat tracking of musical audio," *IEEE Trans on Audio, Speech and Language Processing*, Vol. 15, No. 3, pp. 1009–1020, 2007.

[9] S. Dixon: "Evaluation of audio beat tracking system beatroot," *J. of New Music Research,* Vol. 36, No. 1, pp. 39–51, 2007.

[10] D. Eck. "Beat tracking using an autocorrelation phase matrix," In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007),* Vol. 4, pp. 1313–1316, 2007.

[11] A. Elowsson and A. Friberg: "Modeling the perception of tempo," *J. of Acoustical Society of America,* Vol. 137, No. 6, pp. 3163-3177, 2015.

[12] A. Elowsson and A. Friberg: "Modelling perception of speed in music audio," *Proc. of SMC*, pp. 735-741, 2013.

[13] A. Elowsson, A. Friberg, G. Madison, and J. Paulin: "Modelling the speed of music using features from harmonic/percussive separated audio," *Proc. of ISMIR,* pp. 481-486, 2013.

[14] A. Elowsson and A. Friberg: "Polyphonic Transcription with Deep Layered Learning," MIREX Multiple Fundamental Frequency Estimation & Tracking, 2 pages, 2014.

[15] D. FitzGerald: "Harmonic/percussive separation using median filtering," *Proc. of DAFx-10,* 4 pages, 2010.

[16] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stafylakis: "Music tempo estimation and beat tracking by applying source separation and metrical relations," In Proc. of ICASSP, pp. 421–424, 2012.

[17] X. Glorot, Xavier, A. Bordes, and Y. Bengio: "Deep sparse rectifier neural networks," *International Conference on Artificial Intelligence and Statistics,* 2011.

[18] M. Goto and Y. Muraoka: "Music understanding at the beat level real-time beat tracking for audio signals," in *Proc. of IJCAI (Int. Joint Conf. on AI) / Workshop on CASA*, pp. 68–75, 1995.

[19] M. Goto and Y. Muraoka: "Beat tracking based on multiple agent architecture a real-time beat tracking system for audio signals," In *Proc. of the International Conference on Multiagent Systems*, pp. 103–110, 1996.

[20] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano: "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. on Audio, Speech and Language Processing,* Vol. 14, No. 5, pp. 1832-1844, 2006.

[21] S. Hainsworth and M. Macleod: "Particle filtering applied to musical tempo tracking," *EURASIP J. on Applied Signal Processing*, Vol. 15, pp 2385–2395, 2004.

[22] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon: "Selective sampling for beat tracking evaluation," *IEEE Trans. on Audio, Speech, and Language Processing*, Vol. 20, No. 9, pp. 2539–2548, 2012.

[23] A. Klapuri, A. Eronen, and J. Astola: "Analysis of the meter of acoustic musical signals," *IEEE Trans. on Audio, Speech and Language Processing,* Vol. 14, No. 1, pp. 342–355, 2006.

[24] F. Krebs, S. Böck, and G. Widmer: "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," In *Proc. of ISMIR*, pp. 227–232, Curitiba, Brazil, November 2013.

[25] F. Krebs, S. Böck, and G. Widmer: "An Efficient State-Space Model for Joint Tempo and Meter Tracking," In *Proc. of ISMI*R, pp. 72-78, 2015.

[26] G. Peeters and H. Papadopoulos: "Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation," *IEEE Trans. on Audio, Speech, and Language Processing,* Vol. 19, No. 6, pp. 1754–1769, 2011.

[27] R. Polikar: "Ensemble based systems in decision making," *Circuits and Systems Magazine, IEEE,* Vol. 6, No. 3, pp. 21-45, 2006.

[28] E. Scheirer: "Tempo and beat analysis of acoustic musical signals," *J. Acoust. Soc. Am.,* Vol. 103, No. 1, pp. 588–601, 1998.

[29] J. Schlüter, and S. Böck: "Musical onset detection with convolutional neural networks," In *6th International Workshop on Machine Learning and Music (MML)*, Prague, Czech Republic. 2013.

[30] N. Whiteley, A. Cemgil, and S. Godsill: "Bayesian modelling of temporal structure in musical audio," In *Proc. of ISMIR,* pp. 29–34, 2006.