Contents

1 Introduction

- A fundamental research topic in music information retrieval is the automatic extraction of beat locations from music signals.
- The aim of a beat tracker is to recover a sequence of time instants from a musical input that are consitent with the times when a human might tap their foot. [1]
- Beat tracking is an important initioal step in computer emulation of human musc understanding, since beats are fundamental to the perception of (Western) music.
- The goal of beat tracking is to construct a computational algorithm capable of extracting a symbolic representation which correspond to the phenomenal experience of beat or pulse in a human listener.
- The primary information required for beat tracking is the onset times of musical events, i.e., musical notes and percussive sounds, and this is sufficient for music of low complexity and litle variation in tempo.
- The interpretation of beat is on of the most fundamental aspects of musical intelligence.
- A naive approach to describe the rhythm of musical data is to specify an exhaustive and accurate list of onset times, maybe together with some other musical features characterising those event, e.g., durations, pitches and intensitues. However, such a representations lacks abstraction. Diverse media used for rhyth transmission suffer a trade-off between the level of abstraction and the comprehensiveness of the representation. Standard (Western) music notation provites an accepted method for communication a composition to a performer, but it has little vale in representing the interpretation of a work as played in a caonter. On the other hand, an acoustic signal implicitly contains all rhythmic aspects but provides no abstraction whatsoever [2].
- The main goal in automatic rhythm desciption is the parsing of acoustic events that occur in time into the more abstract notions of metrical structure, tempo and timing. A major difficulty is the inherent ambiguity of rhythm, as discussed in section 1.
- How well does the proposed model gneralise to different musical styles?
- The pattern recognition task is a nontrivial problem due to the wide variability of rhythm in music.
- It could be tackled using handcrafted rules or heristics for distinguishing on with time instant a beat is base on the properties of the audio signal. In practice such an approach leads to a profileration of rules and of exeptions to the rules and so on, and ivariably gives poor results. Far better results can be obtained by

adopting a machine learning approach in which a large set of N tracks $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ called a training set is used to tune the parameters of an adaptive model. The beat instants of the tracks are known in advance, typically by inspecting them indiviualy and hand-labelling them. So for every track $\mathbf{x}$ there is a target vector $\mathbf{t}$. The result of the machine learning algorithm can be expressed as a function $\mathbf{y}(\mathbf{x})$ which takes a new track $\mathbf{x}$ as input and that generates an output vector $\mathbf{y}$, encoded in the same way as the target vectors. The precice form of the function $\mathbf{y}(\mathbf{x})$ is determind during the training phase, on the basis of the training data. Once the model is trained it can then determine the identity of new tracks, which are said to comprise a test set. The ability to categorize correctly new examples thet differ from those used for training is known as generalization. In practical applications, the variability of the input vectors will be such that the trainig dta can comprise only a tiny fraction of all possible input vectors, and so generalization is a central goul in pattern recognition.

- Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as supervised learning problems.

Difficulties of Beat Tracking It is difficult to reliably extract high-level rhythm related features from musical excerpt having properties such as [Quinton2016]

- The principal reason that beat tracjing is intrinsically difficult is that is is the problem of inferring an original beat strucutre that is not expressed explicitly. The degree of beat tracking difficulty is therefore not determided simply by the number of musical instruents performing a musical piexe it depends on how explicitly the beat structure is expressed in the piece. However, it is very difficult to measure its explicitness because it is influenced from various aspects of the songs.

- The larger the number of syncopation, the greater the difficulty of beat tracking (quantitative measure: power-difference measure [3])

- The main reason that different tendencies with regard to the explicitness with which their beat structure is indicated.

- There is not necessarily a specific sound that directly indicates tho position of beats. In fact, a musical beat may not directly correspond to a real sound, there may even be no signal on a beat.

- absence of a clear rhythmic structure (Classical music)

- soft onsets

- blurred note transitions (e.g. classigal music dominated by string instruments)

- heavy syncopation

- expressive timing (e.g. rubato playing)
- the problem of 'octave errors' (detecting double or half time the rate of the ground truth)
- acoustiv signals consit of sound of various kinds of instruments
- the onsets of notes are difficult to obtain, unlike the case of MIDI signals, where there is no such problem.
- a beat may not directly correnspond to a real sound. It is a perceptal concept that human feels in music.
- Multiple interpretations of beats are possible at any given time
- There is no simple relationship beteen polyphonic complexity - the number and timbres of notes played at a single time - in apiece of music, and its rhythmic complexity or pulse complexity [4]. There are pieces and sryles of music which are texturally and timbrally complex, but have straighforward, perceptual simple rhythms; and ther also exist music which deal in less complex textures but are more difficult to rhythmically understand an describe.
- The complexity of grouping harmonic partials together to form notes, and determining the onst times of those.
- For this reason Cemgil et al. [5] define music transcription as the extraction of an *acceptable* music notation.
- Ther is no canonical form for representing rhythm, and lacking this ground truth, it is difficult, if not impossible, to provide a meaningful quantitative comparison of the variuos computer systems. Therefore we need a common database of test music labelled with the ground truth on which the systems are tested.

Applications of Beat Tracking Estimating the beats of an musical audio opens new possibilities for a wide range of applications.

- Beat tracking can be used to automate the time-consuming tasks that must be completed in order to synchronize events with music.
- Video and audio editing (visual track can be automatically synchronized with an audio track using beat tracking)
- Stage light control. In live performances, beat tracking is useful in the control of stage lighting by a computer. For instance, various properties of lighting such as color, brightness, direction, and effect can be changed in time to the music.
- Visualization for example time-grid in audio sequencers.

- Musical interaction systems [6]
- Content-based audio effects, for multimedia or interactive performances or studio post-production
- Beat-driven reat-time computer graphics.
- Temoral segmentation to higher level MIR tasks such as chord estimation for harmonic description [7].
- Long-term structural segmentaion of audio [8]
- performance analysis investigates the interpretation of musical works, e.g., the performer's choice of tempo and expressive timing.
- In audio content analysis beat tracking is important for automatic indexing and content-based retrieval of audio data, such as in multimedia databases and libraries.
- Automatic transcription and score extraction from performance data
- Music similarity
- Time-streching of audio loops
- Beat tracking can provide computers the ability to participate intelligently in live performances in real time and join the ensemble.
- synchronisation of a musical performance with computers or other devices
- Commercial devices already exist that attept to extract a MIDI clock from an audio signal, indication both the tempo and the actual location of the beat. Such MIDI clocks can the be used to synchronize other devices such as drum machines or audio effects, enabling a new range of beat-synchronized audio processing.
- Automatic playlist generation, where a computer is given the task to choose a series of audio tracks from a track database in a way similar to what a human deejay would do. The track tempo is a very important selection criterion in this context, as deejays will tend to string tracks with similar tempi back to back. Furthermore, deejays also tend to perform beat-synchronous crossfading between successive track manualy, slowing down or speeding up one of the tracks so that the beat in the two tracks line up exactly during the crossfade. This can easily be done automatically once the beats are located in the two tracks.

2 Related Work

This chapter is an overview of related work that fostered the development of beat tracking in musical audio. The task of automatic rhythm detection has been well established over the last thirty-five years and beat tracking algorithms have constantly improved in performance. A chronology with the most influential work is shown in Fig. 1.

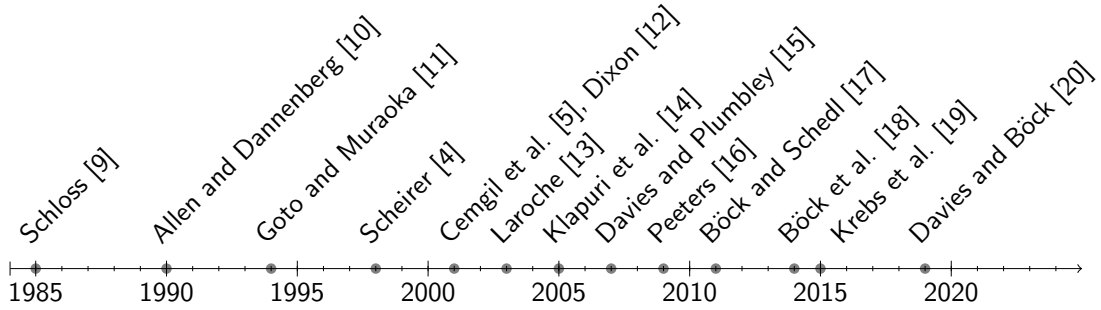

Figure 1: Time-line with the history of beat tracking.

The traditional approach of beat tracking consists of extracting features from the audio signal to obtain a feature list. These features range from note onsets, as time, duration and pitch, to frequency-based signal features and they convey predominant information relevant to rhythmic analysis. The feature extraction is usually followed by a periodicity detection stage and the estimated periodicities subsequently determine the beat times using a temporal decoding strategy. Nevertheless, the recent trend in beat tracking is a shift away from purely signal processing approaches to data-driven approaches incorporating machine learning. In the following, the approaches of worth mentioning works are represented chronologically.

Schloss [9] presented one of the earliest work on automatic extraction of rhythmic content from audio in his percussion transcription system. Onsets are detected as peaks in the slope of the amplitude envelope, where the envelope is defined to be equal to the maximum amplitude in each period of the high-pass filtered signal, and the period defined as the inverse of the lowest frequency expected to be present in the signal. The main limitation of the system is that it requires parameters to be set interactively.

Allen and Dannenberg [10] extended the musical concept of beat to include two aspects, namely period and phase. Based on that concept, they built a method that uses real-time beam search to allow the beat tracker to consider several possible stages at once. They use a credibility measure so that at any given time there is a set of active states that represent the most credible interpretations for the performance encountered so far. However, the system's reliance on MIDI limited the input source to electronic instruments, and moreover limited its application.

Goto and Muraoka [11] introduce the first worth mentioning beat tracking system which could process music played on ensembles of a variety of instruments. However, they restricted their system to rock and pop music in which drums maintain the beats.

The system leverages the fact that for a large class of popular music, a bass drum and a snare drum usually occur on the strong and weak beats, respectively. It manages multiple agents that track beats according to different strategies in order to examine multiple hypothesis in parallel. All hypotheses are gathered and the most reliable one is selected as the output. This enables the system to follow beats without losing track if them, even if some hypothesis became wrong. Assumptions were made; between 65 and 185 BPM, time-signature is 4/4, tempo stays almost constant. In following developments, Goto present a beat tracking system for both music with and without drum-sounds [21]. It uses frequency-domain analysis to detect chord changes, which are assumed to occur in metrically strong positions. This is the fist system to demonstrate the use of high level information in directing the lower-level beat tracking process. The high level information is specific to the musical style, which is a major limitation of the system.

Scheirer [4] figured out from psychoacoustic demonstration on beat perception, that amplitude envelopes from a small number of broad frequency channels are sufficient information to rhythmically analyze musical content. He concludes that a rhythmic processing algorithm should treat frequency bands separately, combining results at the end, rather than attempting to perform beat tracking on the sum of filter-bank outputs. This leads him to the use of a small number of bandpass filters and banks of parallel comb filters, which function as tuned resonators, to perform periodicity analysis. In the next processing step, the phase of the musical signal is extracted by examining the internal state of the delays of the comb filters. Finally, the phase and the period is used to estimate the beat times as far into the future as desired. One problem with the system is that in order to track tempo changes, the system must repeatedly change its choice of filter, which implies the filters must be closely spaced to be able to smoothly track tempo variations. However, the system applies no continuity constraint when switching between filters.

Dixon [12] processes a sequence of note onset times either extracted from an audio signal or from a symbolic representation within a multi-agent system. Likely tempo hypotheses are derived from clustering inter-onset intervals (IOI), thus encoding aspects of the metrical hierarchy. The hypothesis are used to form multiple beat agents using a paradigm, where each agent has a state consisting of the period and the phase of the beat. The sequence of beat times with the best score to date is selected by the agent. The observations are only processed if they occur around the predicted beat locations, i.e., within a window whose width depends on the pulse period. The algorithm is designed to track beats in expressively performed music.

Cemgil et al. [5] formulate beat tracking in a Bayesian framework where tempo and beat is modeled as a stochastic dynamical system. The system is defined with two hidden state variables, namely the period and the phase of beat. To this deterministic model, they add a Gaussian random vector whose covariance matrix models the likely tempo variations. State transitions are defined by a simple set of equations that describe how state variables evolve with time. Because all noises are assumed to be Gaussian and all relationships between variables are linear, the covariance matrix can be efficiently

estimated by a Kalman filter. They also define the tempogram representation which includes a probability distribution over the period and phase given a list of onset. This probability distribution is proportional to the likelihood of the observed onsets under given period and phase hypotheses, weighted by prior distribution, which is equally distributed, as they consider all tempi to be initially equiprobable. For given periods and phases, the likelihood is computed as the integral over all onsets of the product of a constant pulse track and a continuous representation of the onsets. This implements the assuption that a good pulse track is one which matches all onsets well. The tempogram marginal probability distribution $p(\omega|t)$ provides a 1-D representation of periodicities resembling those aforementioned.

Laroche [13] initially finds salient features like note onsets, note changes, and percussion hists by calculating the Fourier transform of an audio signal. A nonlinear monotonic compression function is applied to the amplitude spectrum, so high-frequency components are not masked by higher amplitude low-frequency components. To locate fast variations in the frequency domain contents, a first-order difference is calculated. All bins are summed together, and the result is half-wave rectified to obtain a positive energy flux signal. A least-squares approach is used to determine the best candidates for the tempo and beat locations. The final step consist of going through the successive tempo analysis frames and finding in each frame the best candidates. To that effect a dynamic programming technique is used. This entails continuity and non-syncopation constraints.

Klapuri et al. [14] expand upon Scheirer's amplitude envelope and comb filter model. They adopt a more robust registral accent signal across four parallel analysis bands as the input to their system and use comb filterbanks within a probabilistic framework to simultaniously track three metrical levels. These correspond to the tatum, tactus and measure, which are explained in section 3. Analysis can be performed causally an noncausally, and is not restricted to any particular genre, tempo or time-signature. The robustness of the analysis model is due to the probabilistic modelling of the temporal evolution and interaction between each og the three metrical levels analysed. In a study of audio temoo induction algorhithms [22], this approach was shown to be most accurate.

Davies and Plumbley [15] adopt a simpler and more efficient, heuristic approach than the system of Klapuri by embedding context-dependent information directly into the beat period and alignment estimation processes. They use two state model; the first state performs tempo induction and tracks tempo changes, while the second maintains contextual continuity within a single tempo hypotheseis. The first state, also called general state, operates in a memoryless fashion, extracting the beat period and beat alignment through a process of repeated induction. In this manner, the two-state model can explicitly model tempo discontinuities while smoothing out odd beat errors during each consistent beat period hypothesis.

Peeters [16] approach is based on a probabilistic framework, in which the beat tracking problem is formulated in a hidden Markov model, that can be efficiantly solved with the Viterbi algorithm [23]. An onset-energy-function, time-variable tempo, and meter serves as an input to the system. Beat times are decoded over beat-numbers according to ob-

servation and transition probabilities. A beat-template is used to derive the observation probabilities from the signal. For this purpose, a linear discriminant analysis finds the most discriminative beat-template.

Böck and Schedl [17] present the first beat tracking system which is based on artificial neural networks. The network transforms the signal directly into a beat activation function, which represents the probability of a beat at each frame. As network architecture they use a bidirectional recurrent neural network (RNN) with long short term memory (LSMT) units. The approach is inspired by the good results for musical onset detection [24] and extended to suit the needs for audio beat tracking by modifying the input representation and adding a peak detection stage. As input to the network, three filtered magnitude spectra with different window lengths and their first order differences are used. In a peak detection stage, first the periodicity within the activation function is detected with the autocorrelation function to deternine the most dominant tempo. The beats are then aligned according to the previously computed beat interval. In this way, erroneously detected beats are eliminated or missing beats are complemented.

Böck et al. [18] extend the previous beat tracking system of Böck and Schedl with a multi-model approach to represent different music styles. For this purpose, they use multiple recurrent neural networks, which are trained on certain heterogeneous music styles. The system chooses the model with the most appropriate beat activation function for the input signal and jointly models tempo and phase of the beats with a dynamic Bayesian network. Compared to a reference model, which was trained on the whole trainig set, the specialised models produce better predictions on input data which is similar to that used for training, but worse predictions on signals dissimilar to the training data.

Krebs et al. [19] propose a modified state-space discretisation and tempo transition model for the temporal decoding stage with dynamic Bayesian networks. The modification increases beat tracking accuracy and also reduces time and memory complexity. To be consistent with human tempo sensitivity, they propose to make the number of discrete bar positions dependent on the tempo and distribute the tempo states logarithmically across the range of beat intervals.

Davies and Böck [20] suggest to use a convolutional neural network in form of a temporal convolutional network (TCN) [25]. In comparison to the recurrent model of Böck et al. [18], the TCN can be trained more efficiently on very large datasets due to parallelisation. It requires a smaller number of trainable parameters while achieving state-of-the-art performance.

3 Rhythm

Rhythmic organization is an inhernt part of all human activity. In this chapter, basic principles and definitions about rhythm in music are explained.

3.1 Terminology

Rhythm In music and music theory, there are many different definitions of rhythm. Generally, rhythm means a movement marked by the regulated succession of strong and weak elements, or of opposite or different conditions [26]. It is regarded as the way in which accented and non-accented notes are grouped in a time unit [27]. The definition of Lester [28] considers the patterns of duration between musical events and has the advantage that events pertaining to various musical qualities, giving rise to the idea that more than one rhythm can be defined for a musical piece. Whereas London [29] defines rhythm as the sequential pattern of durations relatively independent of meter or phrase structure. In general, the experience of rhythm involves movement, regularity, grouping, and yet accentutation and differentiation [30].

Onset An onset refers to the beginning of a musical note or other sound. Any rhythmic event is basically characterised by an onset time and a salience. They represent the most basic unit of rhythmic information, from which all beat and tempo information is derived. The concept of onsets is related to the concept of transients, but differs in the way that all musical notes have an onset, but do not necessarily include an initial transient. The more salient events are the more likely to correnspond to beat times than the less salient ones. This tendency for events with greater perceptual salience to occur in stronger metrical positions has been noted by various authors [31, 32, 33]. Lerdahl and Jackenhoff [31] classify musical accents into three types: phenomenal accents, which come from physical attributes of the signal such as amplitude and frequency; structural accents, which arise from perceived points of arrival and departure such as cadences; and metrical accents, points in time which are perceived as accented due to their metrical position.

Beat The term beat, or more technically the *tactus*, refers to the perceived pulses which are approximately equally spaced and define the rate at which notes in a piece of music are played [30]. Intuitively, it is often defined as the rhythm listeners would tap their foot to when listening to a piece of music, or the numbers a musician counts while performing. Therefore, the beat is most often designated as a crotchet or quarter note in western notation. In beat tracking, the period of a beat is the time duration between two successive beats, i.e. the reciprocal of the tempo. Whereas the phase determines where a beat occurs with respect to performance time.

Tempo Given a metrical structure, tempo is defined as the rate of beats at a given metrical level. Thus, it corresponds to the frequency of the primary pulse in a rhythmic

musical signal. The tempo is commonly expressed as a number of beats per minute (BPM). In order to represent changing tempi, various approaches can be used. If tempo is considered as an instantaneous value, it can be calculated as the inter-beat interval (IOI) measured between each pair of successive beat. A more preceptual plausible approach is to take an average tempo measured over a longer period of time. A measure of central tendency of tempo over a complete musical excerpt is called the basic tempo, which is the implied tempo around which the expressive tempo varies [34]. The value of tempo as a function of time is called a tempo curve, and can be visualized in a tempogram [5].

Meter The term meter refers to the regularly recurring patterns and accents such as beats and bars and provides an underlying time frame. Unlike rhythm, meter is a perceptual concept which is inducted from the phenomenally accented points of musical surface [35]. The metrical structure is hierarchical, i.e., it involves a ratio relationship between at least two time levels, namely the referent time level, the beat, and a higher order period based on a fixed number of beat periods, the measure [36]. Meter is regular and stable, and serves as a kind of enhanced temporal grid, which helps to shape expectations about the future and thus be able to anticipate and predict events in time [37]. In order to establish a meter, some regularity has to be manifesed in the acoustic signal in the first place. Once meter has established, all other events are perceived with refernce to this regular pattern. In Lerdahl and Jackendoff's *A Generative Theory of Tonal Music* (GTTM) [31], the rhythmic structure in the tonal music of the Western tradtition consits of two independent elements, grouping and meter. Grouping is the manner in which music is segmented at a whole variety of levels from groups of a few notes up to large-scale from of a piece of music. Whereas meter is described as the regular alteration of strong and waek elements in music. The metrical structure deals with durationless points in time, e.g. the beats, which obey some well-defined rules. In the GTTM, meter perception is described as the progress of finding periodicities in the phenomenal and structural accents in a piece of music. It also propose a set of metrical preference rules, based on musical intuitions, which are assumed to guide the listener to plausible interpretations of rhythms. Nonetheless, a major weakness of the GTTM is that it does not deal with the departures from strict metrical timing which are apparent in almost all styles of music. Thus, it is only suitable for representing the timing structures of musical scores, or as an abstract representation of a performance, where expressive timing is not represented.

3.2 Rhythm Perception

More psychologically or cognitively motivated definitions associate rhythm to the perceived patterns generated by recurring events and how they interact and categorized by listeners.

Perception of beat is a prerequisite to rhythm perception, which in turn is a fundamental part of music perception.

There is no ground thuth for rhythm to be found in simple measurement of an acoustic

signal. The only ground truth is what human listeners agree to be the rhythmic aspects of the musical contetent of that signal

The perception of tempo exhibits a degree of variability. It is not always correct to assume that the denominator of the time signature corresponds to the "foot-tapping" rate, nor to the actual "physical tempo" that would be an inherent property of audio flows [38].

Studies of Povel and Essnes [32] have demonstrated that beat perception may be explianed with a model in which a perceptual clock is aligned with the accent structure of the input. The model relies heavily on structural qualities of the input, such as a sophisticated model of temporal accent, to function. They propose a model of perception of temporal patterns, based on the idea that a listener tries to induce an internal clock which matches the distribution of accents in the stimulis and allows the pattern to be expressed in the simplest possible terms. They use patterns of identical tone bursts at precise multiples of 200 ms apart to test their theory.

Each pair of events in a rhythmic sequence initially contributes to the salience of a single pulse sensation (emphasis occurs), and later that pulse sensations can enhance the salience of other consonant pulse sensations [33]. One may understand the initially above as an indication not to implement influential schemes beteen metrical levels in the induction process, but imdeed to do it in the tracjing process, which is also in agreement with the Dynamic Attending Theory [39].

Drake and Bertrand [40] advocate a universal predisposition toward simple duration ratio, and claim that humans thend to hear a time interval as twice as long or short as previous intervals. The Dynamic Attending Theory [39] proposes that humans spontaneously focus on a reference level of periodicity, and they can later switch to other levels to track events occuring at differnt time spans, e.g., longer span harmony changes, or a particular shorter-span fast motive.

Jones et al. [39] propose that perceived rhythm is the result of differnet attending modes (future-orianted and analytic) which involve anticipatory behaciors to coherent temporal events and their durational patterns. They list the following types of phenomenal accent, which they consider incomplete: note onsets, sforzandi, sudden dynamic or timbral changes, long notes, melodic leaps and harmonic changes. However, they give no indication as to how these factors might be compared or combined, either quantitatively (absolute values) or quanlitatively (relative strengths). Longer notes tend to be perceived as accented.

Human pitch recognition is only sensitive to signal phase under certain unusual conditions. Rhthmic response is crucially a phased phenomenon - tapping on the beat is not at all the same as tapping against the beat. or slightly ahead of or behind the beat, even ig the frequency of tapping is accurate [4]

From psychoacoustic demonstration on beat perception it can be shown that certain kinds of signal manipulations and simplifications can be performed without affecting the perceived pulse content of a musical signal. An amplitude-modulated noise constructed by vocoding a white noise signal with the subband envelopes of a musical signal is sufficient to extract pulse and meter. The simplified signal is created by performing a

frequency analysis of the original signal by processing it through a filterbank of bandpass filters, or grouping FFT bins together. Thus it seems that separating the signal into subbands and maintaining the subband envelopes separately is necessary to do accurate rhythmic processing. This fact leads to the hypotheses that some sort of cross-band ryhtmic integration, not simply summation across frequency bands, is performed by the auditory system to perceive rhythm [4].

Although in the brains of performers music is temporally organized according to its hieaechical beat structure, this structure is not explicitly expressed in music; it is implied in the relations among variuos musical elements which are not fixed and which are dependent on musical genres or poeces.

Expressive timing is generated from perforers' understanding of the musical structure an general knowledge of music theory and musical style [41]. However, there is no precise mathematical model of expressive timing, and the complexity of musical structure fro which timing is derived, coupled with the individuality of each performer and performance, makes it impossible to capture musical nuance in the form of rules [12].

Expressive reductions in tempo are more common and more extreme than tempo increases [34].

Differences in human perception of tempo depend on age, musical training, musical preferences and general listening context, e.g., tempo of a previously heard sequence, listener's activity, instant of the day [42, 43, 44]

Differences in tempo perception are nevertheless far from random. They most often correspond to a focus on a different metrical level, e.g., differences of half or twice the inter-beat interval or one-third or three times the inter-beat interval.

Perception research has shown that with up to 40 ms difference in onset times, two tones are heard as synchronous, and for more than two tnoes, the threshold is up to 70 ms [30]. Rhythmic information is provided by IOIs in the range of approcimately 50 ms to 2 s [30].

Experiments on the tempo sensitivity on humans have shown that the ability to notice tempo changes is proportional to the tempo, with the JND (just noticable difference) beeing around 2-5% [42].

A listener who cannot identify chord names can nevertheless percieve chord changes.

4 Machine Learning

Machine learning is subset of artificial intelligence (AI) and deals with algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is essentially a form of applied statistics with increase emphasis on the use of computers to estimate complicated functions and a decreased emphasis on providing confidence intervals around these functions [45]. On the other hand, pattern recognition is concerned with the automatic discovery of regularities in data, and to take actions such as classifying the data into different categories. Both fields, machine learning and pattern recognition are strongly connected with each other, and together they have undergone a substantial development over the past twenty-five years [46].

Most machine learning algorithms can be divided into the categories of *supervised learning* and *unsupervised learning*. Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. Supervised learning algorithms include classification and regression. Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points.

Machine learning tasks are usually described in terms of how the machine learning system should process an example $\mathbf{x} \in \mathbb{R}^n$, where an example is a collection of n features x_i that have been quantitatively measured from some object or event. Many kinds of tasks can be solved with machine learning, e.g. classification, regression, transcription, machine translation, synthesis, and sampling just to name a few.

In this thesis, the focus is on supervised sequence modeling and classification. In the following, basic concepts of machine learning are described.

4.1 Sequence Modeling

In machine learning, the term sequence modeling encompasses all tasks where sequences of data are transcribed with sequences of discrete labels. Supervised sequence modeling refers specifically to those cases where a set of hand-transcribed sequences is provided for algorithm training. What distinguishes such problems from the traditional framework of supervised pattern classification is that the individual data points cannot be assumed to be independent. Instead, both the inputs and the labels form strongly correlated sequences [47].

Given an input sequence $\mathbf{x}_{1:T} = \mathbf{x}_1, \ldots, \mathbf{x}_T$ with sequence length T, a sequence model is any function $f : \mathcal{X}^T \to \mathcal{Y}^T$, such that

$$\mathbf{y}_{1:T} = \mathbf{y}_1, \ldots, \mathbf{y}_T = f(\mathbf{x}_1, \ldots, \mathbf{x}_T) \tag{1}$$

where vector $\mathbf{x}_t \in \mathbb{R}^n$ is the input at time step t, and $\mathbf{y}_t \in \mathbb{R}^m$ should only depend on $\mathbf{x}_{1:t}$ and not on $\mathbf{x}_{t+1:T}$, i.e., no leakage of information from the future. This causality constraint is essential for autoregressive modeling.

The goal of learning in the sequence modeling setting is to find a network f that minimizes some expected loss between the actual outputs and the predictions,

$$L\big(\mathbf{y}_1, \ldots, \mathbf{y}_T,\, f(\mathbf{x}_1, \ldots, \mathbf{x}_T)\big) \stackrel{!}{=} \min \tag{2}$$

where the sequences and outputs are drawn according some distribution $p_{\text{data}}(\mathbf{x}, \mathbf{y})$.

For most deep learning practitioners, sequence modeling is synonymous with recurrent networks. For example, the sequence modeling chapter in a standard reference on deep learning is titled "Sequence Modeling: Recurrent and Recursive Nets" [45] capturing the common association of sequence modeling and recurrent architectures. Recent results indicate that convolutional architectures can outperform recurrent networks. Concluding from an empirical evaluation of generic convolutional and recurrent networks for sequence modeling, Bai et al. [25] assert, that the common association between sequence modeling and recurrent networks should be reconsidered, and convolutional networks should be regarded as a natural starting point for sequence modeling tasks.

4.2 Feature Extraction

For most practical applications, the original input variables are typically preprocessed to transform them into a new space of variables, where it is hoped the pattern recognition problem will be easier to solve. This preprocessing stage is sometimes also called feature extraction. New test data must be preprocessed using the same steps as the training data. Preprocessing might also be performed in order to speed up the computation, for example if the goal is a real-time application the algorithm should be computationally feasible. Yet, the aim of feature selection is to find features in the input which preserve useful discriminatory information. Usually, the number of features is smaller than the number of input variables, thus a preprocessing is a form of dimensionality reduction.

4.3 Performance Measure

To evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually the performance measure is specific to the task being carried out by the system. For the task of classification, the accuracy of the model is often measured. Accuracy is the proportion of examples for which the model produces the correct output.

The choice of performance measure may seem straightforward and objective, but it is often difficult to choose a performance measure that corresponds well to the desired behavior of the system.

Loss Function Cross entropy is defined for two probability distributions p and q as

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x). \tag{3}$$

In machine learning cross entropy can be used as a loss funciton to measures the performance of a classification model. The probability p_i is the true label (binary indicator 0 or 1), where as the distribution q_i is the predicted value of the current model.

4.4 Model Selection

In pracial applications, we determine the values of the model's parameters, and the principle objective in doing so is usually to achieve the best predictive performance on new data. Furthermore, as well as finding the appropriate values for complexity parameters within the model, we wish to consider a range of different types of model in order to finde the best one for our particular application.

The performance on the training set is not a good indicator of predictive performance on unseen data due to the problem of over-fitting. If data is plentiful, then one approach is simply to use some of the availible data to train a range of models, or a given model with a range of values for its complexity parameters, and then to compare them on indepedet data, also called the validation set, and select the one having the best predictive performance. If the model design is iterated many times using a limited dataset, then some over-fitting to the validation data an occur and so it may be necessary to keep aside a third test set on which the performance of the selceted model is finally evaluated.

- Given a sequence modeling task or dataset, which architecture should one use?
- Bai et al. [25] show that a simple convolutional architectures outperforms canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets, while demonstrating longer effective memory. Is it also the case in beat tracking?

4.5 Optimization

Most machine learning algorithms involve optimization which refers to the task of minimzing an objective function $f(\mathbf{x})$ by altering the arguments $\mathbf{x}$. The goal of optimization is to find an optimal $\mathbf{x}^*$, such that

$$\mathbf{x}^* = \arg\min f(\mathbf{x}) \tag{4}$$

Given the function $y = f(x)$, where both x and y are real numbers, the derivative $f'(x)$ is usefull for minimizing this function because it indicates how to change x in order to make a small improvement in y. Thus, by moving x in small steps with the opposite sign of the derivative, $f(x)$ is reduced. This technique is called gradient decent [48]. When $f'(x) = 0$, the derivative provides no more information about which direction to move and the function reaches a stationary point. Stationary points are classified into three kinds. A local minimum is a point where $f(x)$ is lower than at all neighboring points, and higher for a local maximum respectively. The third kind is known as saddle points where the derivative of the function has the same sign on both sides of the stadionary point.

In the context of machine learning, the objective function has many local minima that are not optimal and many saddle points surrounded by very flat regions. All of this makes

optimization difficult, especially when the input to the function is multidimensional. Therefore, it is usually sufficient to find a value of f that is very low but not necessarily minimal in any formal sense. Thus, in gradient decent we decrease f by moving in the direction of the negative gradient to propose a new point

$$\mathbf{x}' = \mathbf{x} - \mu \nabla_{\mathbf{x}} f(\mathbf{x}) \tag{5}$$

where μ is the learning rate, a positive scalar determining the size of the step, and $\nabla_{\mathbf{x}}$ refers to the gradient with respect to the input $\mathbf{x}$.

Stochastic Gradient Decent Stochastic gradient descent (SGD) is an extension of the basic gradient descent algorithm. A recurring problem in machine learning is that large training sets are necessary for good generailzytion. At the same time, large learning sets are also more computationally expensive. Instead of using the whole learning set, a small set called minibatch is used to estimate the gradient

$$\mathbf{g} = \frac{1}{N} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{M} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \tag{6}$$

where L is the per-example loss depending on the input $\mathbf{x}$, the target y and the model parameters $\boldsymbol{\theta}$. The minibatch size M is typically chosen to be a relatively small number of examples, ranging from one to a few hundred [45]. Larger batches provide a more accurate estimate of the gradient, but with less than linear return. Multicore architectures are usually underutilized by extremely small batches. This motivatates using some absolute minimum batch size, below which there is no reduction in the time to precess a minibatch. Small batches can offer a regularization effect [49], due to the noise they add to the learning process. Generalization error is often best for a batch size 1. Training with such a small batch size might require a small learning rate to maintain stability because of the high variance in the estimate of the gradient. The total runtime can ba very high as a result of the need to make more steps, both because of the reduced learning rate and because it takes more steps to observe the entire training set.

Back-Propagation When a feedforward neural network takes an input $\mathbf{x}$ and produce an output $\hat{\mathbf{y}}$, information flows forward through the network, which is called forward propagation. During training, forward propagation can continue onward until it produces a scalar loss $L(\boldsymbol{\theta})$. The back-propagation algorithm [50] allows the information from the loss to then flow backward through the network in order to compute the gradient. Computing an analytical expression for the gradient is straightforward, but numerically evaluationg such an expression can be computationally expensive. The back-propagation algorithm does so using a simpe and inexpensive procedure. To obtain an algebraic expression for the gradient of a scalar it recursively applies the chain rule with respect to any node in the comultational graph that produced that scalar.

Adam Adam [51] is another adaptive learning rate optimization algorithm. The name Adam derives from the phrase adaptive moments. It can be seen as a combination of RMSProp and momentum with a few important distinctions. First, in Adam, momentum is incorporated directly as an estimate of the first-order moment (with exponential weighting) of the gradient

Batch normalization

4.6 Regularization

Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization. Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. There is no best machine learning algorithm, and in particular, no best form of regularization [52]. Instead the a form of regularization has to be chosen that is well suited to a particular task.

Weight Decay

Early Stopping Estimate generalization error with a validation set during training. Stop training when the error on the validation set rises (early stopping).

Dropout

4.7 Validation

Usually the interest of a machine learning algorithm is in how well it performs on data that has not seen before, since this determines how well it will work when deployed in the real world. The ability to perform well on previously unobserved inputs is called generalization. The performance measure of the model is thus evaluated using a test set of data that is separated from the data used for training. What separates machine learning from optimization is that the generalization error, also called the test error, should be low as well. Formally, the generalization error is defined as the expected value of the error on a new input, where the expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.

There are two central challenges in machine learning, namely underfitting and overfitting. Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. Overfitting occurs when the gap between the training error and test error is too large.

If the supply of data for training and testing will be limited, and in order to build good models, we wish to use as much of the acailable data as possible for training. However, if the calidation set is small, it will give a relatively noisy estimate of predictive performance. One solution to this delemma is to use cross-validation.

Cross-validation The k-fold cross-validation alows a proportion $(k-1)/k$ of the availible date to be used for training while making use of all the data to assess performance. When data is particularly scarce, it may be appropiate to consider the case $k = N$, where $N$is the total number of data points, which gives the leave-one-out technique. One major drawback of cross-validation is that the number of training runs that must be performed is incresed by a factor of k, and this can prove problematic for models in which the training is itself computationally expensive.

Test set method Split the observarions into to disjunct subsets

Typically, one uses about 80 percent of thr training data for training and 20 percent for validation [45].

4.8 Hyperparameters

Most machine learning algorithms have settings called hyperparameters, which control the behavior but are not adapted by the learning algorithm itself. Thus, hyperparameters must be determined outside the learning algorithm. One way to tune hyperparameters is a nested learning procedure in which one learning algorithm learns the best hyperparameters for another learning algorithm.

Sometimes a setting is chosen to be a hyperparameter that the learning algorithm does not learn because the setting is gradient-free and thus difficult to optimize.

- optimzie hyperparameters with nevergrad [53].
- instrumentation: turn a piece of code with parameters into a function defined on an n-dimensional continous data space

5 Deep Neural Networks

Deep neural networks are called networks because they are typically represented by composing together many different functions,

$$f(\mathbf{x}) = (f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)})(\mathbf{x}) \tag{7}$$

where $f^{(1)}$ is called the first layer, $f^{(2)}$ is called the second layer, and so on. The overal length L of the chain gives the depth of the model. The name deep learning arose from this terminology. The final layer is called output layer. The dimensionality of the hidden layers determines the width of the model.

5.1 Feedforward Neural Networks

Feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models. The MLP was the first and simplest type of artificial neural network devised [54]. In this network, the information moves in only one direction, forward, from the input nodes, to the output nodes. Thus, there are no cycles or loops in the network [55].

The goal of a feedforward network is to approximate some function f^*. A feedforward network defines a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and learns the value of the parameters $\boldsymbol{\theta}$ that result in the best funtion approximation.

The model is associated with a directed acyclic graph describing how the functions are composed together.

Feedforward networks are of extreme importance to machine learning practicioners, because they form the bases of many important commercial applications.

5.2 Convolutional Neural Networks

A convolutional neural network (CNN) [56] is a class of deep neural networks specialized for processing data that has a grid-like topology, such as one-dimensional time series or two-dimensional data like raster graphics. CNNs are regularized versions of feedforward neural networks that simply use convolution in place of general matrix multiplication in at least one of their layers [45]. Usually, the operation used in a convolutional neural network does not correspond precisely to the mathematical definition of convolution. Instead, many neural network libraries implement a cross-correlation, which is the same as convolution but flipping the second argument. In the following this convention is obeyed and the convolution is defined as

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(a)\, w(t+\tau) \tag{8}$$

where x is referred to as the input, w as the filter or kernel and the output s is referred to as the feature map. Discrete convolution can be viewed as multiplication by a matrix, but the matrix has several entries constrained to be equalt to other entries. In addition,

convolution usually corresponds to a very sparse matrix. This is because the filter is usually much smaller than the input.

A typical layer of a convolutional neural network consists of three stages, as shown in Fig. 2. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage wich is sometimes called detector stage, each linear activation is run through a nonlinear activation function. In the third stage, a pooling function is used to modify the output of the layer further.

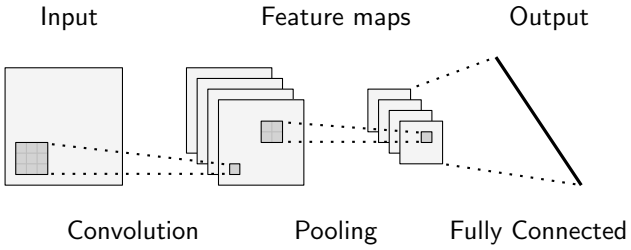

Figure 2: Typical convolutional neural network.

A pooling function replaces the output of the net a a certain location with a summary statistic of the nearby outputs. For example, the max pooling [57] operation reports the maximum output within a rectangular neighborhood. In this way, pooling helps to make the representation approximately invariant to small translations of the input. Invariance to translation means that when a input is translated by a small amount, the values of most of the pooled outputs do not change.

5.3 Recurrent Neural Networks

In deep learning, a recurrent neural network (RNN) [50] is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. Unlike feedforward neural networks, RNNs contain cycles and use an internal state memory $\mathbf{h}$ to process sequences of inputs [58]. A universal recurrent neural network is decribed by the propagation equations,

$$\mathbf{h}_t = \text{Activation}(\mathbf{U}\,\mathbf{x}_t + \mathbf{W}\,\mathbf{h}_{t-1} + \mathbf{b}) \tag{9}$$

$$\mathbf{o}_t = \mathbf{V}\,\mathbf{h}_t + \mathbf{c} \tag{10}$$

where the parameters are the bias vectors $\mathbf{b}$ and $\mathbf{c}$ along with the weight matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$, respectively, for input-to-hidden, hidden-to-output and hidden-to-hidden connections. The recurrent network maps an input sequence $\mathbf{x}_{1:T}$ of length T to an output sequence $\mathbf{o}_{1:T}$ of the same length. The computational graph and its unfolded version is shown in Fig. 3.

Computing the gradinets involves performing a forward propagation pass through the unrolled graph followed by a backward propagation pass. The runtime is $O(T)$ and cannot be reduced by parallelization because the forward propagation graph is inherently

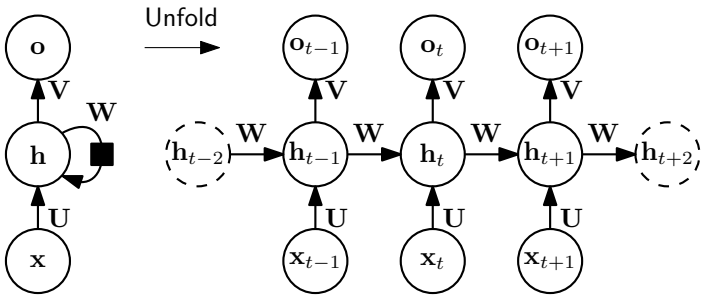

Figure 3: The computational graph on an recurrent neural network and its unfolded version.

sequential, i.e., each time step may be computed only after the previous one. Therefore, back-propagation for recurrent model is called back-propagartion through time (BPTT). Recurrent models construct very deep computational graphs by repeatedly applying the same operation at each time step of a long temporal sequence. This gives rise to the vanishing and exloding gradient problem and makes it notoriously difficult to train RNNs. To prevend these difficulties more elaborate recurrent architectures were developed, such as the LSTM [59] and the GRU [60]. These families of architectures have gained tremendois popularity due to prominent applications to language modeling and machine translation.

LSTM The long short-term memory (LSTM) [59] is an recurrent neural network (RNN) architecture used in the field of deep learning. It comprises leaky units to allow the network to accumulate information over a long duration. However, once that information has been used it might be useful for the network to forget the old state. Instead of manually deciding when to clear the state, the neural network learns to decide when to do it. The time scale of integration can be changed dynamically by making the weights gated, i.e., controllable by another hidden unit.

GRU The gated recurrent unit (GRU) [60] is a gating mechanism in recurrent neural networks. The GRU is similar to an LSTM with forget gate but has fewer parameters than an LSTM, as it lacks an output gate. GRUs have been shown to exhibit even better performance on certain smaller datasets [61]. However, the LSTM is strictly stronger than the GRU as it can easily perform unbounded counting, while the GRU cannot [62].

5.4 Temporal Convolutional Networks

A temporal convolutional network (TCN) [25] represents a special kind of convolutional neural network and is informed by recent convolutional architectures for sequential data. It is designed from first principles and combines simplicity, autoregressive prediction, and very long memory. In comparison to WaveNet [63], the TCN does not employ skip connections across layers (no conditioning, context stacking, or gated activations).

The TCN is based upon two principles: 1) the convolutions are casual, i.e, no information leakage from future to past; 2) the architecture can take a sequence of any length and map it to an output sequence of the same length just as with an RNN. To achieve the first point, the TCN uses a 1D fully-convolutional network architecture [64], where each hidden layer is the same length as the input layer. To accomplish the second point, the TCN uses causal convolutions, i.e., convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer.

Simple causal convolutions have the disadvantage to only look back at history with size linear in the depth of the network. To circumvent this fact, the architecture employs dilated convolutions that enable an exponentially large receptive field. More formally, for a 1-D sequence input $\mathbf{x} \in \mathbb{R}^T$ and a filter $f : \{0, \dots, k-1\} \to \mathbb{R}$, the delated convolution operation F on element s of the sequence is defined as

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i)\, \mathbf{x}_{s-d\cdot i} \tag{11}$$

where $d = 2^\nu$ is the dilation factor, with ν the level of the network, and k is the filter size. The term $s - d \cdot i$ accounts for the direction of the past. Dilation is equivalent to introducing a fixed step between every two adjacent filter taps, as it can be seen in Fig. 4. Using larger dilation enables an output at the top level to represent a wider range of inputs, thus effectivly expanding the receptive field of a CNN. There are two ways to increase the receptive field of a TCN: choosing lager filter sizes k and increasing the dilation factor d, since the effective history of one layer is $(k-1)\, d$.

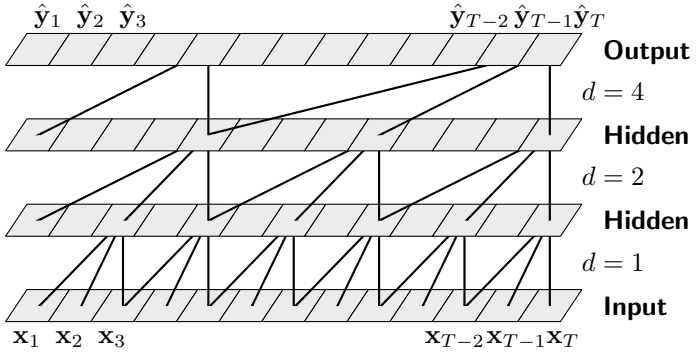

Figure 4: A dilated casual convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$.

Another architectural element of a TCN are residual connections. In place of a convolutional layer, TCNs employ a generic residual module. Each residual block contains a branch leading out to a series of transformations $\mathcal{F}$, whose outputs are added to the input $\mathbf{x}$ of the block,

$$o = \text{Activation}\big(\mathbf{x} + \mathcal{F}(\mathbf{x})\big). \tag{12}$$

This effectively allows layers to learn modifications to the identity mapping rather than the entire transformation, which has been shown to benefit deep neural networks [65]. Especially for very deep networks stabilization becomes important, for example, in the case where the prediction depends on a large history size ($> 2^{12}$) with a high-dimensional input sequence.

A residual block has two layers of dilated causal convolutions and recfied linear units (ReLU) as non-linearity, shown in Fig. 5. For normalization, weight normalization [66] is applied to the convolutional filters. In addition, a spatial dropout [67] is added after each dilated convolution for regularization, i.e., at each training step, a whole channel is zeroed out.

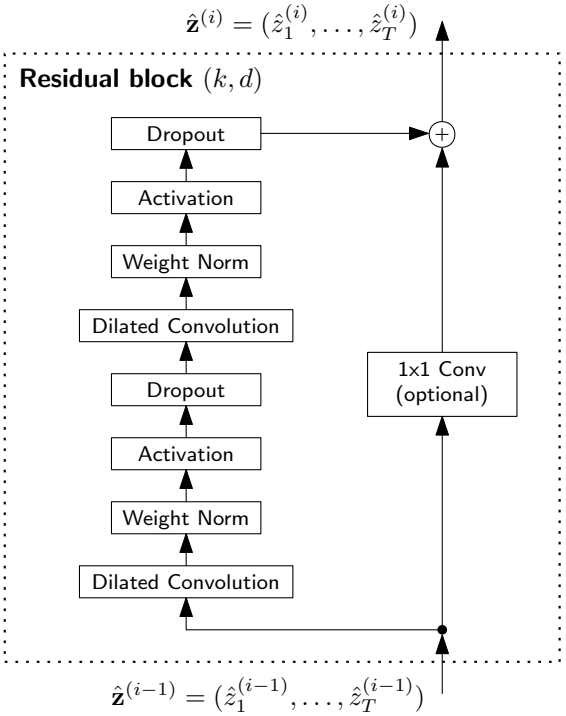

Figure 5: TCN residual block.

- the convolutions are casual, that means no information leakage from future to past (compare to bidirectional RNN).
- long effective history sizes, i.e., the ability for networks to look very far into the past to make a prediction (receptive field?), by using a combination of very deep networks (augmented with residual layers) and dilated convolutions
- this allows layers to learn modifications to the identity mapping rather than the entire transformation, which has been shown to benefit deep neural networks

(where?)

- within a residual block there exist two layers of dilated causal convolution and a rectified linear unit (ReLU) [68, Nair2010] as a non-linearity
- for normalization a weight normalization [66, Salimans2016] is applied to the convolutional filters
- additionaly spatioal dropout [67, Srivastava2014] is added after each dilated convolution for regularization (at each training step, a whole channel is zeroed out)
- The TCN architecture appers not inly more accurate than canonical recurrent networks such as LSTMs and GRUs, but also simpler and clearer.

5.5 General-Purpose Computing on GPUs

In 2004, it was shown by K. S. Oh and K. Jung that standard neural networks can be greatly accelerated on GPUs. Their implementation was 20 times faster than an equivalent implementation on CPU [69]. In 2005, another paper also emphasised the value of GPGPU for machine learning.[42]

While GPUs operate at lower frequencies, they typically have many times the number of cores. Thus, GPUs can process far more pictures and graphical data per second than a traditional CPU. Migrating data into graphical form and then using the GPU to scan and analyze it can create a large speedup.

6 Method

From an overall perspective, the proposed automatic beat tracking system comprises three major stages, as shown in Fig. 6. In the first stage the original audio is preprocessed. Data preprocessing refers to all transformations on the raw data before it is feed to the machine learning algorithm. It includes different methods such as normalization, transformation and feature extraction.

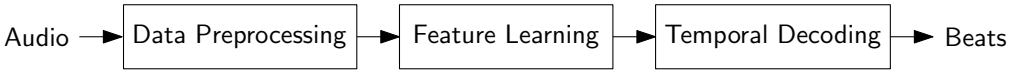

Figure 6: Beat tracking system signal flow

6.1 Dataset

For training and evaluation we use the datasets listed in Table 1.

Table 1: Datasets used for training.

Dataset	**files**	**length**
Ballroom [22, 70]	685	5 h 57 m
GTZAN [71, 72]	1000	8 h 20 m
Hainsworth [73]	222	3 h 19 m
SMC [74]	217	2 h 25 m
Total		h m

Ballroom The Ballroom dataset [22, 70] contains 685 audio files[1] of ballroom dancing music. As genres the files cover Cha Cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, and Slow Waltz. Each file is 30 s long, mono and sampled at 44.1 kHz with 16-bit resolution.

GTZAN The GTZAN [71] dataset was originally proposed for music genre classification problem and later extended with beat annotations [72]. The dataset comprises 1000 excerpts of 10 different genres. The genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. Each file is 30 s long, mono and sampled at 22050 Hz with 16-bit resolution. The audio content of the GTZAN dataset is representative of the real commercial music of various music genre. The dataset has a good balancing between tracks with swing (Blues and Jazz music) and without swing.

[1] After removing the 13 duplicates which are pointed out by Bob Sturm [75].

Hainsworth The Hainsworth dataset [73] contains 222 musical audio files, with the following genre breakdown: Rock/Pop (68), Dance (40), Jazz (40); Classical (30), Folk (22), and Choral (22). Each file is between 30 and 60 s in length, mono and sampled at 44.1 kHz with 16-bit resolution.

SMC The SMC dataset [74] contains 217 musical audio files. Each file is 40 s in length, mono and sampled at 44.1 kHz with 16-bit resolution.

6.2 Data Preprocessing

The training dataset contains raw pulse code modulated (PCM) audio signals stored as WAV files. For the sake of consistency and also to reduce computational complexity every audio signal is resampled at a sampling rate $f_s = 44.1\,\text{kHz}$ with 16-bit resolution and converted to a monaural signal by averaging both stereo channels.

In complex polyphonic mixtures of music, simultaneously occurring events of high intensities lead to masking effects that prevent any observation of an energy increase of a low intensity onset [76]. To circumvent these masking effects, the signal is analyzed in a band-wise fashion to extract transients occurring in certain frequency regions of the signal. Therefore, a filtered frequency spectrum is chosen to serve as the input for the neural network.

Input The discrete audio signal $x(n)$ is segmented into overlapping frames of $N = 2048$ samples, which corresponds to a length of 46.4 ms. The frames are sampled every 10 ms, resulting in a frame rate $f_r = 100\,\text{fps}$. A standard Hamming window $w(n)$ of the same length is applied to the frames before the short-time Fourier transform (STFT) is used to compute the complex spectrogram

$$X(t,\omega) = \sum_{n=1}^{N} w(n)\, x(n + t\,h)\, e^{-2\pi j\omega/N} \tag{13}$$

where t refers to as the frame index, ω the frequency bin index, and $h = 441$ the hop size, i.e., the time shift in samples between adjacent frames. The complex spectrogram $X(t,\omega)$ is converted to the power spectrogram $|X(t,\omega)|^2$ by omitting the phase portion of the spectrogram. The power spectrogram is filtered with a bank of overlapping triangular filters $F(t,\omega)$ with 12 bands per octave covering a frequency range of 30 to 17,000 Hz. To better match the human perception of loudness, a logarithmic representation is chosen,

$$S(t,\omega) = \log\left(|X(t,\omega)|^2 \cdot F(t,\omega)^T + 1\right) \tag{14}$$

At every time instant t the input $\mathbf{x}_t \in \mathbb{R}^n$ to the neural network corresponds to the frequency column of the filtered log power spectrogram $S(t,\omega)$

$$\mathbf{x}_t = S(t,\omega), \quad \forall\, t = 1,\ldots,T \tag{15}$$

and has dimensionality $n = 88$.

Labels The beat tracking task requires annotations in the form of time instants of beats from a musical excerpt. To this end, the beat tracking problem is considered as a binary classification problem, where annotated beat instants are first quantized to the temporal resolution of the input representation, and then represented as training targets $y_{1:T}$. Following the strategy of onset detection [77] the temporal activation region around the annotations is widen by means of including two adjacent temporal frames on either side of each quantized beat location and weight them with a value of 0.5 during training.

Training Set After preprocessing the audio data, we obtain the training set $O = \left\{\mathbf{x}^{(\alpha)}, y^{(\alpha)}\right\}_{\alpha=1}^{p}$, with p samples in total.

Chroma Representation

- Chords are more likely to change in beat times than on other positions. Chords are more likely to change at the beginnings of measures than at other positions of the beat.
- A chromagram comprises a time-series of chroma vectors. which represent harmonic content at a specific time in the audio.
- Chromagrams are concise descriptors of harmony because they encode tone quality and neglect tone height.
- CRP (chroma DCT-reduced log-pitch) features have significant amount of robustness to changes in timbre an instrumentation [78, Mueller2010]
- Employ data-driven approach to extract chromagrams that specifically encode content relevant to harmony
- Chroma features are noisy in their basic formulation be- cause they are affected by various interferences: musical instruments produce overtones in addition to the funda- mental frequency; percussive instruments pollute the spec- trogram with broadband frequency activations (e.g. snare drums) and/or pitch-like sounds (tom-toms, bass drums); different combinations of instruments (and different, possibly genre-dependent mixing techniques) create different timbres and thus increase variance [7,20] [79]

6.3 Feature Learning

From an overall perspective, the neural network architecture consist of two main blocks, as it is shown in Fig. 7. While the filtered log power spectrum could be passed directly to the TCN, the network first seeks to learn some compact intermediate representation, by implementing a preceding convolutional block. To capture the sequential structure, a TCN finally transforms the intermediate representation $\mathbf{z}_{1:T}$ directly into a sequence of beat activations $a_{1:T}$.

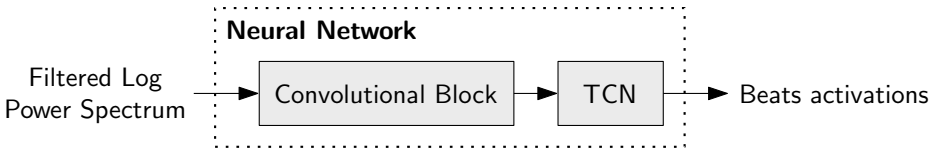

Figure 7: Neural network

The convolutional block is a set of convolution and max pooling layers and reduce the dimensionality both in time and frequency. The convolutional layers contain 16 filters each, with kernel sizes of 3×3 for the first two, and 1×8 for the last layer, as it is shown in Fig 8. The intermediate max pooling layers apply pooling only in the frequency direction over 3 frequency bins. A dropout [80] rate of 0.1 is used with the exponential linear (ELU) [81] as activation function.

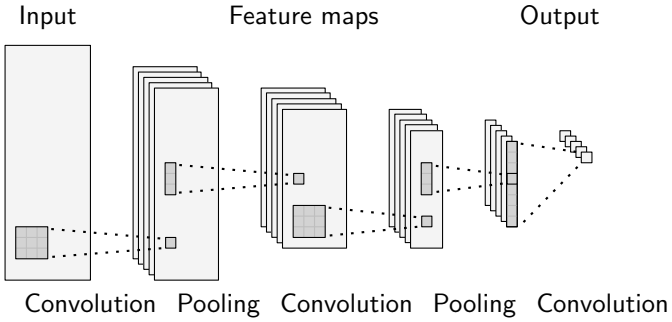

Figure 8: Convolutional Block

The output of the convolutional block $\mathbf{z}_t$ is a 16-dimensional feature vector and serves as the input to the TCN. The TCN is constructed as stacked residual blocks, as shown in Fig. 5, and contains 11 layers with 16 filters of size 5 and geometrically spaced dilations ranging from 2^0 up to 2^{10} time frames. Thus, the resulting receptive field is approximately 81.5 s. A spatial dropout with rate 0.1 and the ELU activation function is used. The output layer of the TCN has two units, representing the two classes "beat" and "no beat". Thus, the network can be trained as a binary classifier with the cross entropy error function. The outputs use the log softmax activation function,

$$\log \mathrm{softmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right) \tag{16}$$

which is added to the last layer of the neural network. Therefore, the output $\hat{\mathbf{y}}_t = (\hat{y}_{1_t}, \hat{y}_{2_t})^T$ represent the log-probabilities for the two classes at time t. Finally, the beat activation a_t can be calculated by

$$a_t = \exp(\hat{y}_{1_t}), \quad \forall\, t = 1, \ldots, T \tag{17}$$

which represent the probability of a beat at each frame.

The final model is trained with the ADAM optimizer [51] with a batch size $M = 20$ and a learn rate of $\mu = 1\,e^{-3}$. The whole system has only 30818 trainable parameters.

6.4 Temporal Decoding

The metrical structure of music builds upon a hierarchy of appriximatey regular pulses, as elaborated in Section 3. To explot this seqential structure, a probabilistic dynamic model is used to result in a more robust estimation of the beat instants and to correctly operate under rhythmic fluctuations, such as ritardando and accelarando or metric modulations.

As a postprocessing stage following the feature learning, a dynamic Bayesian network (DBN) is employed which jointly infers tempo and the phase of the beat. Dynamic Bayesian networks are popular frameworks for meter tracking in music becuase they are able to incorporate prior knowledge about the dynamics of rhythmic parameters. In this DBN, which is a hidden Markov model (HMM) precisely, a sequence of hidden variables that represent the meter of an audio piece is inferred from a sequence of observed variables. The probabilistic state space consist of two hidden variables, the position within a bar and the tempo.

The probabilistic model is based upon the bar pointer model of Whiteley et al. [82] which enables joint inference of rhythmic parameters from a piece of music. Mutual dependencies between these parameters are exploited, which increases the computational complexity of the models. A bar pointer is defined as being a hypothetical hidden object located in state space consiting of the period of a latent rhythmical pattern. The velocitiy of the bar pointer is defined to be proportional to tempo. The advantage of this approach is that the bar pointer continues to move whether or not a beat activation is observed. This explicitly models the concept that meter is a latent process and provides robustness against rest in the music which might otherwise be wrongly interpreted as local variations in tempo. In order to make inference computationally tractable, the state-space is usually divided into discrete cells [19].

Given a sequence of observed data $a_{1:T}$, we wish to identify the most probable hidden state trajectory $\mathbf{s}_{1:T}$. At each time frame t, the bar pointer of the hidden state space is referred to as $\mathbf{s}_t = [\phi_t, \nu_t]$, with $\phi_t \in \{1, 2, \ldots, K\}$ denoting the position within a bar, and $\nu_t \in \{K_{\min}, K_{\min} + 1, \ldots, K_{\max}\}$ referring to as the tempo. The number of discrete bar positions K is dependent on the tempo by using exactly one bar position state per audio frame and thus per observation feature value. The number of observations per bar (four beats) at a tempo Θ in beats per minute (BPM) is

$$K(\Theta) = \left\lfloor \frac{4 \times 60}{\Theta\,\Delta} \right\rfloor \tag{18}$$

where Δ is reffered to as the audio frame length. With Equation. 18, one can compute the number of bar positions of the tempo limits $K_{\min} = K(\Theta_{\max})$ and $K_{\max} = K(\Theta_{\min})$. Using a tempo range of $\Theta = [55, 215]$ the postion whithin a bar has at most 82 tempo states.

With this state space discretisation, the most likely hidden state sequence $\mathbf{s}^*_{1:T} = \mathbf{s}^*_1, \ldots, \mathbf{s}^*_T$ given a sequence of observations $a_{1:T} = a_1, \ldots, a_T$ is computed by

$$\mathbf{s}^*_{1:T} = \arg\max_{\mathbf{s}_{1:T}} P(\mathbf{s}_{1:T} \,|\, a_{1:T}) \tag{19}$$

with

$$P(\mathbf{s}_{1:T} \,|\, a_{1:T}) \propto P(\mathbf{s}_1) \prod_{t=2}^{T} P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})\, P(a_t \,|\, \mathbf{s}_t) \tag{20}$$

where $P(\mathbf{s}_1)$ is the initial state distribution, $P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})$ is the transition model, and $P(\mathbf{a}_t \,|\, \mathbf{s}_t)$ is the observation model. The most likely hidden state sequence can be solved using the well-known Viterbi algorithm [23]. Finally, the set of beat instant $\mathcal{B}$ can be extracted from the sequence of bar positions as

$$\mathcal{B} = \{t : \phi^*_t = 1\} \tag{21}$$

In order to infer the hidden variables from an audio signal, the three entities are specified. The rransition model describes the transitions between the hidden variables. The observation model takes the beat activations from the neural network and initial distribution encodes prior knowledge about the hidden variables.

Initial Distribution Any prior knowledge about tempo distributions can be incorporated into the model. For example, if the music to be tracked from one genre, a specific tempo distribution can be used. To make it genre independend, a uniform distribution is used in this theses.

Transition Model The transition model $P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})$ can be further decomposed into a distribution for each of the two hidden variables ϕ_t and ν_t, this is

$$P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) = P(\phi_t \,|\, \phi_{t-1}, \nu_{t-1})\, P(\nu_t \,|\, \nu_{t-1}) \tag{22}$$

where the first factor is

$$P(\phi_t \,|\, \phi_{t-1}, \nu_{t-1}) = \mathbf{1}_A \tag{23}$$

with the indicator fuction $\mathbf{1}_A$ that equals one if $\phi_t = (\phi_{t-1} + \nu_{t-1} - 1) \mod M + 1$. The modulo operator makes the bar postion cyclic. If $\phi_t \in \mathcal{B}$, the second factor is defined by

$$P(\nu_t \,|\, \nu_{t-1}) = \exp\left(-\lambda \left| \frac{\nu_t}{\nu_{t-1}} - 1 \right| \right) \tag{24}$$

otherwise it is

$$P(\nu_t \,|\, \nu_{t-1}) = \begin{cases} 1, & \nu_t = \nu_{t-1} \\ 0, & \text{else.} \end{cases} \tag{25}$$

The parameter $\lambda \in \mathbb{Z}_{\geq 0}$ determines the steepness of the distribution and models the probability for a tempo change. A value of $\lambda = 0$ means that transitions to all tempi are equally probable. In practice, for music with roughly constant tempo, we set $\lambda \in [1, 300]$. The probability of tempo change is heuristically set to $p_\omega = 0.002$. Higher-level or domain specific knowledge could be used to set this parameter. For example in rock or pop music, the beat is usually quite steady, so a samll value for p_ω would be quite appropriatre, while for classical music, particularly styles including many tempo changes, a higher value would be more optimal.

Observation Model The beat activation function 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. Thus, the activation function is used directly as state-conditional observation distributions [83]. The observation likelihood is defined as

$$P(a_t \,|\, \phi_t) = \begin{cases} a_t, & 1 \leq \phi_t \leq \frac{\lambda}{\Lambda} \\ \frac{1-a_t}{\lambda-1}, & \text{else.} \end{cases} \tag{26}$$

7 Evaluation

- annotations tapped at different metrical levels in beat tracking
- a evaluation method should adequately contend with the inherent uncertainty and/or ambiguity while providing a measurement of performance which is both meaningful and easy to interpret. [84]
- objective methods for beat tracking evaluation compare the output beat times from a beat tracking algorithm against one or more sequences of ground truth annotated beat times. [84]
- We must question the imprtance of strict continuity when evaluating a beat tracking system. A single misplaced beat, in the context of a sequence of otherwise accurate beats, is a far less disturbing error than the accuracy calculated with the ontinuity requiremet would suggest.

7.1 Evaluation Measure

Set of annotated beat times $\mathcal{A} = \{a_1, \ldots, a_A\}$
Set of predicted beat times $\mathcal{B} = \{b_1, \ldots, b_B\}$

True positive if beat is in ± 70 ms rage of annotated beat
tp = number of true positives
fp = number of false positives (extra detections)
fn = number of false negatives (missed detections)

Precision and recall:

$$\text{precision} = \frac{tp}{tp + fp}\,, \qquad \text{recall} = \frac{tp}{tp + fn}$$

F-measure:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\,tp}{2\,tp + fp + fn}$$

P-score: Define two sequences T_a and T_b as

$$T_a(n) = \begin{cases} 1, & \text{if } n \in \mathcal{A} \\ 0, & \text{otherwise} \end{cases}\,, \qquad T_b(n) = \begin{cases} 1, & \text{if } n \in \mathcal{B} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{P-score} = \frac{\sum_{m=-w}^{w} \sum_{n} T_a(n)\, T_b(n+m)}{\max(A, B)}\,, \qquad \text{where } w = 0.2\,\text{median}(\Delta_a)$$

7.2 Saliency Maps

- How do the filters of the ConvNet look like?
- Artificial neural networks often give good results, but it is difficult to understand what they learned, or on which basis they generate their output.
- compute saliency maps using guided back-propagation [25]

7.3 Network Training

- To achieve better results, we could use DNN ensembles instead of a single DNN. We could ensure that the network sees data for which its pre- dictions are wrong more often during training, or similarly, we could simulate a more balanced dataset by showing the net super-frames of rare chords more often [79]

7.4 Labeling

- In an editor (Sonic Visualizer) that enables a user to mark beat positions in a digitalized audio signal while listening to the audio and watching its waveform.
- The positions can be finely adjusted by playing back the audio with click tones at beat times.
- The audio excerpt in time domain and its spectrogram can be visualised using tools like Sonic Visualizer. Beat locations can first be obtained by recording the tap locations the musical excerpt and then manually correcting these locations for exact onsets. Annotators can also follow a semi-automatic process, for example in [2], where beats and downbeats were first obtained through the ircambeat software and then errors were manually corrected by human annotators.

8 Conclusion

- Theoretically, a single network large enough should be able to model all differnt music styles simultaneously but unfortunately this optional solution is hardly achievable. The main reason for this is the diffuculty 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, harmonic aspects and many other features
- We have presented an efficient audio based beat tracking algorithm able to produce equivalante results to the current state-of-the-art.
- Limitations in both the amount of training data and the variable quality of the annotations. Perform data augmentation.
- Dataset size: "*[...] a rough rule of thumb is that a supervised deep learning algorithm will gernerally achieve acceptable performance with around 5000 labeled examples per category and will match or exceed human performance when trained with a dataset containing at least 10 million labeled examples.*" [45, Goodfellow2016]
- The most mentionable aspect is that the neural networks were trained solely on ballroom dance and other kinds of western pop music. The networks accuracy depends on the trainings set. Selective bias.
- Leave higher level analysis to the neural network
- System operates causaly, thus an real-time implementation is possible.
- Data augmentation (transformation or adding noise) $\Rightarrow$ prediction gets robust against transformation and noisy signals
- Additionally, the tempo, metrical level of the beat and downbeat positions can also be annotated for related tasks like downbeat tracking/ meter tracking/ tempo tracking etc (multi-task learning) [Böck2019]

General notes

- Goal of supervised learning: predict not yet observed input
- Motivation: "*It is probable that no universal beat tracking model exist which does not utilise a switching model to recognize style and context prior to application.*" [85, Collins2006]
- Instead of calculating the STFT as for input featuress discover a good set of features by representation learning. ("*Learned representations often result in much better performance than can be obtained with hand-designed representations.*" [45, Goodfellow2016])
- Common association between sequence modeling and recurrent networks: "*Given a new sequence modeling task or dataset, wich architecture should one use?*" [25, Bai2018]
- Certain convolutional architectures can reach state-of-the-art accuracy in audio synthesis (e.g. Google WaveNet [63, Oord2016])
- Do TCNs outperform LSTM architectures in this particular sequence modeling task?
- Contrast of the two statements:
  - "*Feature extraction is an essential step towards efective and accurate beat/downbeat positions extraction.*" [86, Khadkevich2012]
  - "*The system shows state-of-the-art beat and downbeat tracking performance on a wide range of different musical genres and sryles. It does so ny avoiding hand-crafted features such as harmonic changes, or rhythmic patterns, but rather leans the relevant features directly from audio.*" [87, Boeck2016b]
- Hainsworth and Macleaod state that beat tracking systems will have to be style specific in the future in order to improve the state-of-the-art [73].
- The no free lunch therorem fo machine learning [52] states that, averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserverd points. This implies that we must design our machine learning algorithms to perform well on a specific task.
- The method for obtaining ground truth annotations depends on whether the aim is to identify descriptive beat locations or to replicate a human tapping response. In the former case, an initial estimate of the beat locations can be obtained by recording tap times for a given musical excerpt and iteratively modifying and auditing the obtained beat positions while in the latter case, the ground truth can be completely defined by the tapping response [84].

- Due to uncertainties in the annotation process, for many types of inpit signals (especially multi-instriment exerpts) it may not possible to determine onset locations with greater precision than 50 ms. [88]
- The ideal outcome of the annotation is an unambigous representation of the start points of musical events

Appendix

Hyperparameter

- window function (e.g. Hamming window)

References

[1] Daniel P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, mar 2007.

[2] Fabien Gouyon and Simon Dixon. A review of automatic rhythm description systems. *Computer music journal*, 29(1):34–54, 2005.

[3] Masataka Goto and Yoichi Muraoka. Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, 27(3-4):311–335, 1999.

[4] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[5] A. T. Cemgil, H. J. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 28:4:259–273, 2001.

[6] Andrew Robertson and Mark D. Plumbley. B-keeper: A beat-tracker for live performance. In *NIME*, 2007.

[7] Juan Pablo Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *ISMIR*, volume 5, pages 304–311. Citeseer, 2005.

[8] Mark A Bartsch and Gregory H Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on multimedia*, 7(1):96–104, 2005.

[9] W. Andrew Schloss. *On the Automatic Transcription of Percussive Music - From Acoustic Signal to High-Level Analysis*. PhD thesis, Stanford University, 05 1985.

[10] Paul E Allen and Roger B Dannenberg. Tracking musical beats in real time. In *ICMC*, 1990.

[11] Masataka Goto and Yoichi Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the second ACM international conference on Multimedia*, pages 365–372. ACM, 1994.

[12] Simon E. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 08 2001.

[13] Jean Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4):226–233, 2003.

[14] Anssi P Klapuri, Antti J Eronen, and Jaakko T Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342–355, 2005.

[15] Matthew E. P. Davies and Mark D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, March 2007.

[16] Geoffroy Peeters. Beat-tracking unsing a probabilistic framework and linear discriminant analysis. *12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.

[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-11*, 2011.

[18] Sebastian Böck, Florian Krebs, and Gerhard Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *ISMIR*, 2014.

[19] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *ISMIR*, pages 72–78, 2015.

[20] Matthwe E. P. Davies and Sebastian Böck. Temporal convolutional networks for musical audio beat tracking. *European Association for Signal Processing*, 2019.

[21] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[22] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[23] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1967.

[24] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. Universal onset detection with bidirectional long-short term memory neural networks. *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[25] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[26] Oxford English Dictionary. New york: Oxford university press. *Compact Edition)*, 1971.

[27] G.W. Cooper and L.B. Meyer. *The Rhythmic Structure of Music*. Phoenix books. University of Chicago Press, 1966.

[28] Joel Lester. *The rhythms of tonal music*. Pendragon Press, 1986.

[29] Justin London. Rhythm. *The new Grove dictionary of music and musicians*, 21:277–309, 2001.

[30] Stephen Handel. Listening. *An introduction to the perception of auditory events, Cambridge, MA*, 1989.

[31] Fred Lerdahl and Ray S Jackendoff. *A generative theory of tonal music*. MIT press, 1985.

[32] Dirk-Jan Povel and Peter Essens. Perception of temporal patterns. *Music Perception: An Interdisciplinary Journal*, 2(4):411–440, 1985.

[33] Richard Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception: An Interdisciplinary Journal*, 11(4):409–464, 1994.

[34] Bruno H Repp. On determining the basic tempo of an expressive music performance. *Psychology of Music*, 22(2):157–167, 1994.

[35] J London. Hearing in time: Psychological aspects of musical meter. new york, ny, us: Oxford university press, 2004.

[36] Maury Yeston. The stratification of musical rhythm. 1976.

[37] David Brian Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.

[38] Carolyn Drake, L Gros, and Amandine Penel. How fast is that music? the relation between physical and perceived tempo. *Music, mind, and science*, pages 190–203, 1999.

[39] Mari R Jones and Marilyn Boltz. Dynamic attending and responses to time. *Psychological review*, 96(3):459, 1989.

[40] Carolyn Drake and Daisy Bertrand. The quest for universals in temporal processing in music. *PsychoL Sci*, 13:71–4, 2001.

[41] Eric F Clarke. Rhythm and timing in music. In *The psychology of music*, pages 473–500. Elsevier, 1999.

[42] Carolyn Drake and Marie-Claire Botte. Tempo sensitivity in auditory sequences: Evidence for a multiple-look model. *Perception & Psychophysics*, 54(3):277–286, 1993.

[43] Carolyn Drake, Mari Riess Jones, and Clarisse Baruch. The development of rhythmic attending in auditory sequences: attunement, referent period, focal attending. *Cognition*, 77(3):251–288, 2000.

[44] Carolyn Drake, Amandine Penel, and Emmanuel Bigand. Why musicians tap slower than nonmusicians. *Rhythm perception and production*, pages 245–248, 2000.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[46] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[47] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

[48] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[49] D Randall Wilson and Tony R Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks*, 16(10):1429–1451, 2003.

[50] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[51] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[52] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[53] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

[54] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[55] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.

[56] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[57] Y-T Zhou, Rama Chellappa, Aseem Vaid, and B Keith Jenkins. Image restoration using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1141–1151, 1988.

[58] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[61] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[62] Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision rnns for language recognition. *arXiv preprint arXiv:1805.04908*, 2018.

[63] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[64] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[66] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[68] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[69] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[70] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *ISMIR*, pages 227–232, 2013.

[71] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[72] Ugo Marchand and Geoffroy Peeters. Swing ratio estimation. pages 423–428, 2015.

[73] Stephen W Hainsworth and Malcolm D Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Advances in Signal Processing*, 2004(15):927847, 2004.

[74] A. Holzapfel, M. Davies, J. R. Zapata, J. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech and Language Processing*, 20(9), 2012.

[75] Bob Sturm. Faults in the ballroom dataset. `http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html`, 2014. Access: 22.08.2019.

[76] Peter Grosche and Meinard Muller. Extracting predominant local pulse information from music recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1688–1701, 2010.

[77] Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 6979–6983. IEEE, 2014.

[78] Meinard Muller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662, 2010.

[79] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. *arXiv preprint arXiv:1612.05065*, 2016.

[80] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[81] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[82] Nick Whiteley, Ali Taylan Cemgil, and Simon J Godsill. Bayesian modelling of temporal structure in musical audio. In *ISMIR*, pages 29–34. Citeseer, 2006.

[83] N. Degara, E. A. Rua, A. Pena, S. Torres-Guijarro, M. E. P. Davies, and M. D. Plumbley. Reliability-informed beat tracking of musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):290–301, Jan 2012.

[84] Norberto Degara Matthew E. P. Davies and Mark D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical report, Centre for Digital Music, Queen Mary University of London, 2009.

[85] Nick Collins. Towards a style-specific basis for computational beat tracking. 2006.

[86] Maksim Khadkevich, Thomas Fillon, Gaël Richard, and Maurizio Omologo. A probabilistic approach to simultaneous extraction of beats and downbeats. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 445–448. IEEE, 2012.

[87] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, pages 255–261, 2016.

[88] Pierre Leveau and Laurent Daudet. Methodology and tools for the evaluation of automatic onset detection algorithms in music. In *In Proc. Int. Symp. on Music Information Retrieval*. Citeseer, 2004.