Style-Specific Beat Tracking with Deep Neural Networks

Julius Richter

Audio Communication Group
Technische Universität Berlin

This thesis is submitted for the degree of
Master of Science

First Supervisor: Prof. Dr. Stefan Weinzierl
Second Supervisor: Prof. Dr. Klaus-Robert Müller

September 7, 2019

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt gegenüber der Fakultät I sowie der Fakultät IV der Technischen Universität Berlin, dass die vorliegende, dieser Erklärung angefügte Arbeit selbstständig und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen und Hilfsmittel angefertigt wurde. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind kenntlich gemacht. Ich reiche die Arbeit erstmals als Prüfungsleistung ein. Ich versichere, dass diese Arbeit oder wesentliche Teile dieser Arbeit nicht bereits dem Leistungserwerb in einer anderen Lehrveranstaltung zugrunde lagen.

Titel der schrifltichen Arbeit

Style-Specific Beat Tracking with Deep Neural Networks

Verfasser

Julius Marius Richter (Matrikelnummer: 342740)

Betreuende Dozenten

Prof. Dr. Stefan Weinzierl
Prof. Dr. Klaus-Robert Müller
Athanasios Lykartsis
Thomas Schnake

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden. Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

Berlin, den 7. September 2019

.....

Acknowledgments

I would like to thank

- Athanasios Lykartsis
- Thomas Schnake
- Sebastian Böck
- Matthew Davies
- my brother, Lorenz Richter, for teaching me mathematical insights in statistical learning theory, and for having nice discussions about recent trends in deep learning.
- Bernd Keul, for encouraging me to decide the topic, which I always dreamed of working on.
- Dida for using their Google cloud service, on which I trained my machine learning model using 8 GPUs in parallel.
- Michael Flamm
- My parents

Abstract

In this thesis, a computational method for extracting the beats from audio signals is presented. The proposed beat tracking system is based on convolutional neural networks which capture the sequential structure of audio input. A dynamic Bayesian network is used to model beat periods of various lengths and align the predicted beat positions to the global best solution. The system is evaluated on four datasets of various styles and achieves state-of-the-art performance, while working causally and requiring considerably less learnable parameters.

Zusammenfassung

In dieser Arbeit wird eine Rechenmethode zum Extrahieren der Beats aus Audiosignalen vorgestellt. Das Beat-Tracking-System basiert auf Convolutional Neural Networks, welche die sequentielle Struktur der Audiosignale erfassen. Ein dynamisches Bayes'sches Netz wird verwendet, um Beat Perioden unterschiedlicher Länge zu modellieren und die vorhergesagten Beat-Positionen an der globalen besten Lösung auszurichten. Das System wird anhand von vier Datensätzen unterschiedlicher, musikalischer Stile ausgewertet und erreicht eine Performance vergleichbar mit dem neuesten Stand der Technik, und dies obwohl es kausal funktioniert und erheblich weniger lernbare Parameter benötigt.

Contents

1 Introduction

Rhythm is indispensable for music and builds its central, hierarchical organizing structure. It represents one of the fundamental dimensions in the perception of music and orders the movement of musical patterns in time. Therefore, rhythm analysis is an essential part of understanding music, and represents an innate cognitive ability of humans even without musical education. Especially, the interpretation of meter is a fundamental aspect of musical intelligence, and thus detecting the beat, defined as the most prominent layer of rhythm, is an important initial step in the computer emulation of human music understanding.

In music information retrieval (MIR), automatic analysis of the temporal structure in an audio piece is an ongoing research topic since the 1980s. Despite its apparent intuitiveness and simplicity compared to other parts of music perception, beat tracking has remained a difficult task to implement on a computer. The basic goal of beat tracking is to construct a computational method capable of extracting a symbolic representation which corresponds to the phenomenal experience of the perceived pulse of a human listener. Intuitively speaking, beat tracking consists of recovering a sequence of time instants from a musical input that are consistent with the times when a human might tap their foot. The primary information required for this task are the onset times of musical events, i.e., musical notes and percussive sounds. As a consequence, a naive approach to describe the rhythm of a musical piece is to specify an accurate list of onset times, together with some other musical features characterizing those events, e.g., durations, pitches and intensities. From this list, the predominant periodicity can be estimated and beat times are subsequently determined using a temporal decoding strategy. However, such a method lacks abstraction. Diverse media which hold information about a piece of music suffer a trade-off between the level of abstraction and the comprehensiveness of the representation. Standard music notation provides an accepted method for communication a composition to a performer, but it has little value in representing the interpretation of a work as played in a concert. On the other hand, an acoustic signal implicitly contains all rhythmic aspects but provides no abstraction [1].

As mentioned above, beat tracking can be accomplished using handcrafted rules or heuristics based on the properties of the audio signal. In practice, such an approach leads to a proliferation of rules and exceptions, and invariably gives poor results. Far better results can be obtained by adopting a machine learning approach in which a large training set of various musical pieces is used to tune the parameters of an adaptive model. The beat instants of the pieces are known in advance, typically by inspecting them individually and hand-labeling them, i.e., for every track $\mathbf{x}$ there is a target vector $\mathbf{y}$ with annotated beat times. The result of the machine learning algorithm can be expressed as a function $\hat{\mathbf{y}} = f(\mathbf{x})$ which takes a new track $\mathbf{x}$ as input and generates an output vector $\hat{\mathbf{y}}$, encoded in the same way as the target vectors. The precise form of the function $f(\mathbf{x})$ is determined during the training phase, on the basis of the training data. Once the model is trained it can determine the beats of new tracks, which were unseen to the model during the training process. The ability to correctly detect the beat in new

examples that differ from those used for training is known as generalization. In practice, the variability of the input will be such that the training data can comprise only a tiny fraction of all possible input tracks, which is the central challenge in pattern recognition. This is mainly because beat tracking is a nontrivial problem due to the wide variability of rhythm in music.

The style of music and its associated learned representational scheme underlies the human faculty of beat tracking. This fact is also applicable to computational beat tracking systems that utilize supervised machine learning techniques. With the selection of training data the learned model can be specialized to certain musical styles. For instance, a beat tracking system which should cope with classical music should be ideally trained on classical music only. Nevertheless, the aim of building a universal beat tracking model, which can provide good accuracy across a large corpus of styles, demands a certain model complexity and also requires a large training set which is balanced across different styles.

1.1 Motivation

Musicians are often surprised when mentioning the difficulty of programming a computer method that is able to follow the beat of the music. To any adult participant in a given culture, moving in time with music is so natural that one easily forgets that the ability of beat tracking is not given in early development, but usually becomes established by the age of four [2]. This observation can be seen as evidence, that the perception of rhythm is learned inductively, i.e., learned from examples, instead of obeying some well-defined rules.

Difficulties of Beat Tracking The principal reason that beat tracking is intrinsically difficult is that it is the problem of inferring an original beat structure that is not expressed explicitly. The degree of beat tracking difficulty is therefore not determined simply by the number of musical instruments performing a musical piece, it depends on how explicitly the beat structure is expressed in the piece. Nevertheless, it is very difficult to measure this explicitness because it is influenced from various aspects of the musical piece [3].

Usually, audio signals consist of sound of various kinds of instruments and other sound sources. There is not necessarily a specific sound that directly indicates the position of a beat. In fact, a musical beat may not directly correspond to a real sound, there may even be no signal on a beat. Furthermore, it is difficult to reliably extract rhythmical high-level features from musical excerpts having properties such as soft onsets, heavy syncopation or making use of expressive timing, e.g., playing rubato. This is often aggravated by blurred note transitions, and the absence of a clear rhythmic structure, e.g., in classical music which is dominated by string instruments. In addition to the various properties of an musical piece which make beat tracking difficult, there is no canonical form for representing rhythm. A beat may not directly correspond to a real sound, it rather is a perceptual concept that human feels in music. Therefore, multiple

interpretations of the beats structure are possible at any given time. Lacking this ground truth, it is difficult, if not impossible, to provide a completely quantitative evaluation of the beat tracking task. However, various beat tracking evaluation methods are developed which involve this ambiguity in rhythm perception, e.g., by circumvent the problem of octave errors by accepting double or half time the rate of the annotated beats. Still, to compare various computational beat tracking systems, a common database of test music labeled with the ground truth is required on which the system is tested.

Applications of Beat Tracking The automatic analysis of the metrical structure in audio is a long-standing, ongoing endeavor, since a good underlying beat tracking system is fundamental for various tasks and opens new possibilities for a wide range of applications. In general, beat tracking can be used to automate time-consuming tasks that must be completed in order to synchronize events with music. For instance in video editing, the visual track can be automatically synchronized with an audio track; or stage light control in live performances, where beat tracking is useful in the control of stage lighting by a computer, e.g., various properties of lighting such as color, brightness, direction, and effect can be changed in time to the music. Furthermore, beat-driven real-time computer graphics or content-based audio effects, for multimedia or interactive performances and studio post-production need synchronization with the beat in music.

In audio content analysis, beat tracking is important for the automatic indexing and content-based retrieval of audio data, such as in multimedia databases and libraries. Temporal segmentation can be useful for higher level MIR tasks such as chord estimation, harmonic description, automatic transcription and score extraction from performance data [4]. In performance analysis, beat tracking could be used to investigate the interpretation of musical works, e.g., the performer's choice of tempo and expressive timing. In addition, rhythm information is required for 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 DJ would do. The track tempo is a very important selection criterion in this context, as DJs will tend to string tracks with similar tempi back to back. Furthermore, DJs also tend to perform beat-synchronous cross-fading between successive track manually, slowing down or speeding up one of the tracks so that the beat in the two tracks line up exactly during the cross-fade. This can easily be done automatically once the beats are located in the two tracks.

Beat tracking is also used in musical interaction systems. It can provide computers the ability to participate intelligently in live performances in real time and join the ensemble. Commercial devices already exist that attempt 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 [5].

1.2 Goals of the Thesis

Do TCNs outperform LSTM architectures in this particular sequence modeling task?

Overview This thesis is structured as follows: Chapter 2 introduces related work which influenced the development of beat tracking. In Chapter 3, basic principles of rhythm and rhythm perception are explained. Chapter 4 deals with the foundations of machine learning, and Chapter 5 continues with the introduction of deep neural networks. In Chapter 6, the proposed beat tracking method is described. In Chapter 7, the experimental set-up for the evaluation is explained and the results are finally discussed in Chapter 8.

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.

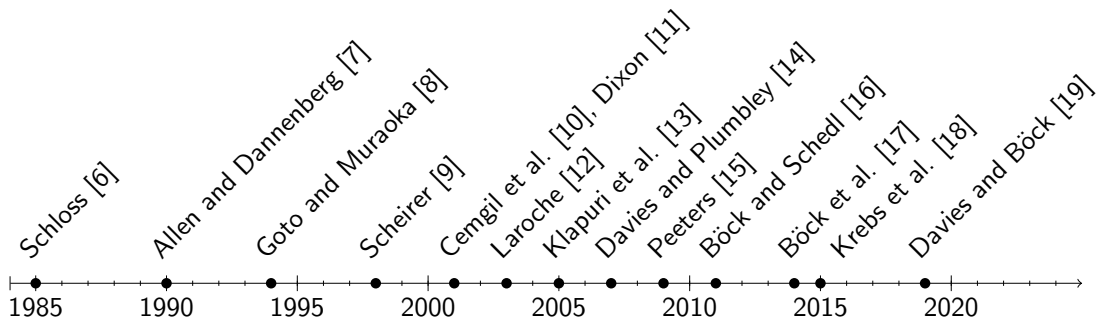


Figure 1: Time-line with the history of beat tracking.

The traditional approach of beat tracking consists of extracting features from an 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 towards data-driven approaches incorporating machine learning. In the following, most influential approaches are represented chronologically.

Schloss [6] proposed 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. The period is 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 [7] extended the musical concept of beat by including 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 [8] 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. In their algorithm, multiple agents estimate beat times 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 of them, even if some hypothesis become wrong. Assumptions were made; tempo between 65 and 185 BPM, time-signature is 4/4, tempo stays almost constant. In following developments, Goto [20] presents a beat tracking system for both music with and without drum-sounds. It uses frequency-domain analysis to detect chord changes, which are assumed to occur in metrically strong positions. This is the first 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 [9] concludes from psychoacoustic studies on beat perception, that amplitude envelopes from a small number of broad frequency channels are sufficient information to rhythmically analyze musical content. He infers 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 [11] 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. [10] formulate beat tracking in a probabilistic Bayesian framework where tempo and beat is modeled as a stochastic dynamical system. The system is defined with two hidden state variables, 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 develop 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's marginal probability distribution provides a 1-D representation of periodicities resembling those aforementioned.

Laroche [12] 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 component is not masked by higher amplitude low-frequency components. To locate fast variations in the frequency domain content, 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. [13] 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 filter-banks within a probabilistic framework to simultaneously track three metrical levels. These correspond to the tatum, tactus and measure. Analysis can be performed causally an non-causally, and is not restricted to any particular genre, tempo or time-signature. The robustness of the analysis model is due to the probabilistic modeling of the temporal evolution and interaction between each og the three metrical levels analyzed. In a study of audio tempo induction algorithms [21], this approach was shown to be most accurate.

Davies and Plumbley [14] 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 hypotheses. 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 [15] approach is based on a probabilistic framework, in which the beat tracking problem is formulated in a hidden Markov model, that can be efficiently solved with the Viterbi algorithm [22]. 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 observation 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 [16] 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 [23] 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 determine 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. [17] 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 training set, the specialized 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. [18] propose a modified state-space discretization 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 [19] suggest to use a convolutional neural network in form of a temporal convolutional network (TCN) [24]. In comparison to the recurrent model of Böck et al. [17], the TCN can be trained more efficiently on very large datasets due to parallelization. It requires a smaller number of trainable parameters while achieving state-of-the-art performance.

3 Rhythm

Rhythmic organization is an inherent part of all human activity. As the time structure of music, rhythm is composed of distinct temporal components such as pattern, meter, and tempo. In this chapter, basic principles and definitions of those components of music are explained, and various aspects of human rhythm perception are examined.

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 [25]. It is regarded as the way in which accented and non-accented notes are grouped in a time unit [26]. The definition of Lester [27] 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 [28] defines rhythm as the sequential pattern of durations relatively independent of meter or phrase structure. In general, the perception of rhythm involves movement, regularity, grouping, as well as accentuation and differentiation [29].

Onset An onset refers to the beginning of a musical note or other sound. Any rhythmic event is basically characterized 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 correspond 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 [30, 31, 32]. Lerdahl and Jackendoff [30] 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 [29]. 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 refers to the
the position within
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 beats. A more perceptual 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 [33]. The value of tempo as a function of time is called a tempo curve, and can be visualized in a tempogram [10].

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 [34]. 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 [35]. 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 [36]. 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) [30], 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. While 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 rhythm. 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

The perception of music requires the ability to build a temporally ordered architecture of sound sequences in rapid succession. The complex processes underlying this ability have attracted accelerating research in ethology, developmental cognitive sciences, experimental psychology, musicology, and behavioural neurology.

The perception of beat is a prerequisite to rhythm perception, which in turn is a fundamental part of music 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. Nevertheless, there is no ground truth 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 content of the signal.

Studies of Povel and Essnes [31] have demonstrated that beat perception may be explained 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. 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 stimulus 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.

Jones et al. [37] propose that perceived rhythm is the result of different attending modes, future-oriented and analytic attending. Future-oriented attending involves anticipatory behaviors to coherent temporal events and their durational patterns. They list the following types of phenomenal accent, which they consider incomplete: note onsets, sforzando, 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 qualitatively (relative strengths). Each pair of events in a rhythmic sequence initially contributes to the salience of a single pulse sensation, an emphasis occurs, and later that pulse sensations can enhance the salience of other consonant pulse sensations [32]. One may understand the "initially" above as an indication not to implement influential schemes between metrical levels in the induction process, but imdeed to do it in the tracking process. This is also in agreement with the *Dynamic Attending Theory* [37], which proposes that humans spontaneously focus on a reference level of periodicity, and they can later switch to other levels to track events occurring at different time spans, e.g., longer span harmony changes, or a particular shorter-span fast motive. Drake and Bertrand [38] advocate a universal predisposition toward simple duration ratio, and claim that humans tend to hear a time interval as twice as long or short as previous intervals.

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 sub-band 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 filter-bank of bandpass filters, or grouping fast Fourier transform (FFT) bins together. Thus, it seems that separating the signal into sub-bands and maintaining the sub-band envelopes separately is necessary to do accurate rhythmic processing. This fact leads to the hypotheses that some sort of cross-band rhythmic integration, not simply summation across frequency

bands, is performed by the auditory system to perceive rhythm [9].

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 [39]. 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 [2, 40, 41]. Nevertheless, these differences are 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 notes, the threshold is up to 70 ms [29]. Rhythmic information is provided by IOIs in the range of approximately 50 ms to 2 s [29]. Experiments on the tempo sensitivity on humans have shown that the ability to notice tempo changes is proportional to the tempo, with the just noticeable difference beeing around 2-5% [40].

Rhythmic response is crucially a phased phenomenon, in contrast to human pitch recognition, which is only sensitive to signal phase under certain unusual conditions. Tapping on the beat is not at all the same as tapping against the beat or slightly ahead of or behind the beat, even if the frequency of tapping is accurate [9]. Although in the brains of performers music is temporally organized according to its hierarchical beat structure, this structure is not explicitly expressed in music; it is implied in the relations among various musical elements which are not fixed and which are dependent on musical genres or pieces. Expressive timing is generated from performers' understanding of the musical structure and general knowledge of music theory and musical style [42]. However, there is no precise mathematical model of expressive timing, and the complexity of musical structure from 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 [11]. Nevertheless, expressive reductions in tempo are more common and more extreme than tempo increases [33].

4 Machine Learning

Machine learning is a scientific study, often associated with 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 increased emphasis on the use of computers to estimate complicated functions and a decreased emphasis on providing confidence intervals around these functions [43]. On the other hand, pattern recognition is concerned with the automatic discovery of regularities in data, and with taking 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 [44].

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 clustering of data points.

Machine learning tasks are usually described in terms of how the machine learning system should process a data point $\mathbf{x}$, where a data point can be 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 [45].

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 function f that minimizes some expected loss between the actual outputs and the predictions,

$$L = \mathbb{E}_p \left[l(\mathbf{y}_{1:T},\, f(\mathbf{x}_{1:T})) \right] \stackrel{!}{=} \min \tag{2}$$

where l is the per-example loss and the input and outputs sequences are drawn according some distribution $p(\mathbf{x}, \mathbf{y})$.

For most deep learning practitioners, sequence modeling is often associated with recurrent networks. For example, the sequence modeling chapter in a standard reference on deep learning is titled "Sequence Modeling: Recurrent and Recursive Nets" [43] capturing the common association of sequence modeling and recurrent architectures. Recent results, however, 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. [24] 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

In the context of an optimization algorithm, the function used to evaluate a candidate solution is termed the objective function. Typically, with neural networks, the objective function is referred to as the loss function L and the value calculated by the loss function is referred to as loss. The loss function reduces all the various good and bad aspects of a possibly complex model down to a single scalar value, which allows candidate solutions to be ranked and compared.

Usually, the loss function is specific to the task being carried out by the system. In the case of a classification problem, cross entropy can be used to measures the performance of the model. In information theory, cross entropy is defined for two discrete probability distributions p and q as

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

where the sum goes over all possible elements x of the set elements $\mathcal{X}$. Transfered to the problem of a binary classification problem, the probability $p(x)$ corresponds to the true label $y_i \in \{0, 1\}$, i.e, representing a binary indicator 0 or 1, where as the distribution $q(x)$ refers to the predicted probability $\hat{y}_i \in [0, 1]$ for the two classes. As a consequence, the cross entropy loss between true label $\mathbf{y} = (y_0, y_1)^T$ and the predicted value $\hat{\mathbf{y}} = (\hat{y}_0, \hat{y}_1)^T$ is defined as

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -w_0\, y_0 \log(\hat{y}_0) - w_1\, y_1 \log(\hat{y}_1) \tag{4}$$

where w_0 and w_1 are optional constants that rescale the loss for the both classes, useful for the case in which both classes occur unbalanced.

4.4 Model Selection

In practical applications, the values of the model's parameters are determined by learning, 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, one wishes to consider a range of different types of models in order to find the best one for a 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 available 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 independent 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 overfitting to the validation data can occur and so it may be necessary to keep aside a third test set on which the performance of the selected model is finally evaluated.

- Given a sequence modeling task or dataset, which architecture should one use?
- Bai et al. [24] 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 minimizing an objective function $L(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ by altering the parameters $\boldsymbol{\theta}$. The goal of optimization is to find an optimal $\boldsymbol{\theta}^*$, such that

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \tag{5}$$

In basic algebra, given the function $y = f(x)$, where both x and y are real numbers, the derivative $f'(x)$ is useful 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 [46]. 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 stationary point.

Back to the context of machine learning, the objective function has many local minima that are not optimal and possibly 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 L that is very low but not necessarily minimal in any formal sense. Thus, in gradient decent we decrease L by moving in the direction of the negative gradient to propose a new set of parameters

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \mu \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{6}$$

where μ is the learning rate, a positive scalar determining the size of the step, and $\nabla_{\boldsymbol{\theta}}$ refers to the gradient with respect to all parameters $\boldsymbol{\theta}$.

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 generalization. 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

$$\mathbf{g} = \frac{1}{M} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{M} L(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \boldsymbol{\theta}) \tag{7}$$

where L is the per-example loss depending on the input $\mathbf{x}$, the target $\mathbf{y}$, 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 [43]. 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 motivates 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 [47], 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 be 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 [48] 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 evaluating such an expression can be computationally expensive. The back-propagation algorithm does so using a simple 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 computational graph that produced that scalar.

Adam Adam [49] 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

4.6 Regularization

Regularization is an important concept in machine learning used to improve the generalizability of an adaptive model, i.e, optimizing the model with respect to the given training set while avoiding to over-fit to the data. Another definition by Goodfellow [43] states that regularization is any modification to a learning algorithm that is intended to reduce its generalization error but not its training error. Some methods put extra constraints to a machine learning model, such as adding restrictions on the parameter values; others add extra terms to the objective function that can be thought of as corresponding to a soft constraint on the parameter values. In general, there is no best machine learning algorithm, and in particular, no best form of regularization. Instead, a form of regularization has to be chosen that it is well suited to a particular task [50].

In the following, several basic regularization techniques are reviewed, which are partially used in this work to enhance the ability of the machine learning model to better generalize to unseen data.

Weight Decay One of the most common form of regularization is termed Tikhonov regularization [51], also known as ridge regression or weight decay. It is particularly useful to mitigate the problem of multicollinearity in linear regression, which often occurs in models with large numbers of parameters. Many regularization approaches try to limit the capacity of a model, and weight decay is doing so by adding a parameter norm penalty to the objective function. For the sake of readability, parameters $\boldsymbol{\theta}$ at the moment are assumed to have no bias, only weights $\mathbf{w}$. Then, the regularized objective function with weight decay is defined as

$$\tilde{L}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = L(\mathbf{x}, \mathbf{y}; \mathbf{w}) + \alpha \, ||\mathbf{w}||^2 \tag{8}$$

where α is a hyperparameter that weights the relative contribution of the norm penalty term. This has the effect that in every training step the weight vector multiplicatively shrinks by a constant factor, before performing the usual gradient update.

Early Stopping During the training of models with sufficient representational capacity to overfit a given task, the training error usually decreases steadily over time, while the validation error initially does the same but eventually begins to rise again. That means, returning to the parameter setting at the point in time with the lowest validation error, one obtains a model that generalizes better, at least with respect to the validation set. In machine learning, this strategy is called early stopping and is referred to as a form of regularization. It also provides guidance as to how many iterations can be run before the model begins to overfit.

Dropout Another regularization approach is termed dropout [52], which provides a computationally inexpensive but powerful method applicable to a broad family of models. With dropout an ensemble consisting of all subnetworks that can be formed by removing any elementary units except output units from an underlying base network. Therefore, dropout is regarded as an approximation to training and evaluating a bagged ensemble of exponentially many networks, where bagging involves training multiple models and evaluating multiple models on each test example. The regularization effect of dropout can be explained, that with randomly omitting units from the hidden layers during training, rare dependencies are excluded in model [53].

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 available data as possible for training. However, if the validation set is small, it will give a relatively noisy estimate of predictive performance. One solution to this dilemma is to use cross-validation.

Cross-validation The k-fold cross-validation allows a proportion $(k-1)/k$ of the available data to be used for training while making use of all the data to assess performance. When data is particularly scarce, it may be appropriate 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 increased 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 observations into to disjunct subsets.
- Typically, one uses about 80 percent of the training data for training and 20 percent for validation [43].

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.

- hyperparameters are optimized with Nevergrad [54].
- instrumentation: turn a piece of code with parameters into a function defined on an n-dimensional continuous data space

5 Deep Neural Networks

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

$$f(\mathbf{x}) = (f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)})(\mathbf{x}) \tag{9}$$

where $f^{(1)}$ is called the first layer, $f^{(2)}$ is called the second layer, and so on. Typically, the hidden layers have the form $f^{(i)}(\mathbf{x}) = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$. The overall length L of the chain gives the depth of the model. 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 [55]. 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 [56].

The goal of a feedforward network is to approximate some function f^*. A feedforward network defines a mapping $\hat{\mathbf{y}} = f(\mathbf{x}; \boldsymbol{\theta})$ and learns the value of the parameters $\boldsymbol{\theta}$ that result in the best function 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 practitioners, because they form the bases of many important commercial applications.

5.2 Convolutional Neural Networks

A convolutional neural network (CNN) [57] 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 [43]. 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

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(a)\, w(t+\tau) \tag{10}$$

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 equal 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 which 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.

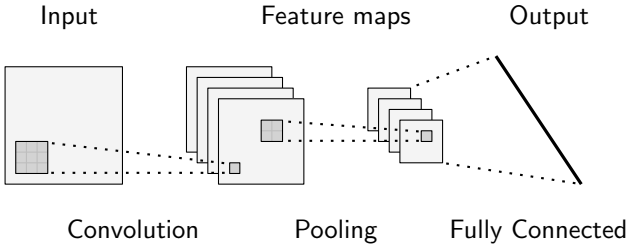


Figure 2: Typical convolutional neural network.

A pooling function replaces the output of the network at a certain location with a summary statistic of the nearby outputs. For example, the max pooling [58] 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) [48] 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 [59]. A universal recurrent neural network is described by the propagation equations,

$$\mathbf{h}_t = \text{Activation}(\mathbf{U}\,\mathbf{x}_t + \mathbf{W}\,\mathbf{h}_{t-1} + \mathbf{b}) \tag{11}$$

$$\mathbf{o}_t = \mathbf{V}\,\mathbf{h}_t + \mathbf{c} \tag{12}$$

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 gradients 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

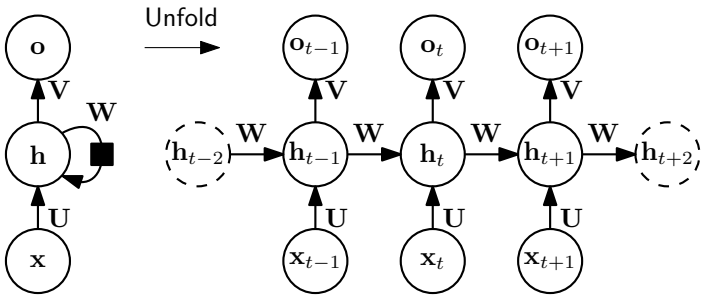


Figure 3: The computational graph on an recurrent neural network and its unfolded version.

sequential, i.e., each time step may be computed only after the previous one. Therefore, back-propagation for recurrent model is called back-propagation 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 exploding gradient problem and makes it notoriously difficult to train RNNs. To prevent these difficulties more elaborate recurrent architectures were developed, such as the LSTM [60] and the GRU [61]. These families of architectures have gained tremendous popularity due to prominent applications to language modeling and machine translation.

LSTM The long short-term memory (LSTM) [60] 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) [61] 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 [62]. However, the LSTM is strictly stronger than the GRU as it can easily perform unbounded counting, while the GRU cannot [63].

5.4 Temporal Convolutional Networks

A temporal convolutional network (TCN) [24] represents a special architecture 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 [64], 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 [65], 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 input sequence $\mathbf{x} \in \mathbb{R}^T$ and a filter $f : \{0, \dots, k-1\} \to \mathbb{R}$, the dilated convolution operation F on element s of the sequence is defined as

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i)\, \mathbf{x}_{s-d \cdot i} \tag{13}$$

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 effectively 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 $(k-1)\, d$.

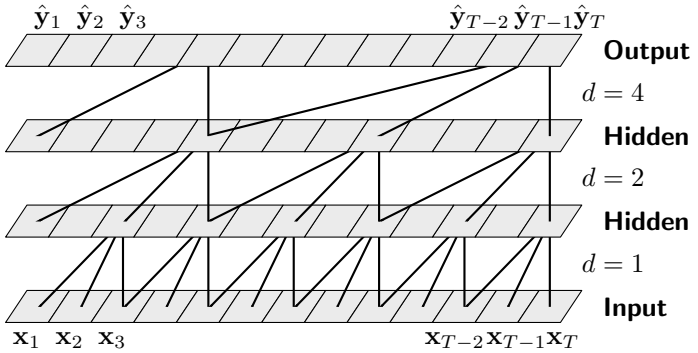


Figure 4: A dilated casual convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$.

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

$$o = \text{Activation}\big(\mathbf{x} + \mathcal{F}(\mathbf{x})\big) \tag{14}$$

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 [66]. 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 rectified linear units (ReLU) as non-linearity, shown in Fig. 5. For normalization, weight normalization [67] is applied to the convolutional filters. In addition, a spatial dropout [52] is added after each dilated convolution for regularization, i.e., at each training step, a whole channel is zeroed out.

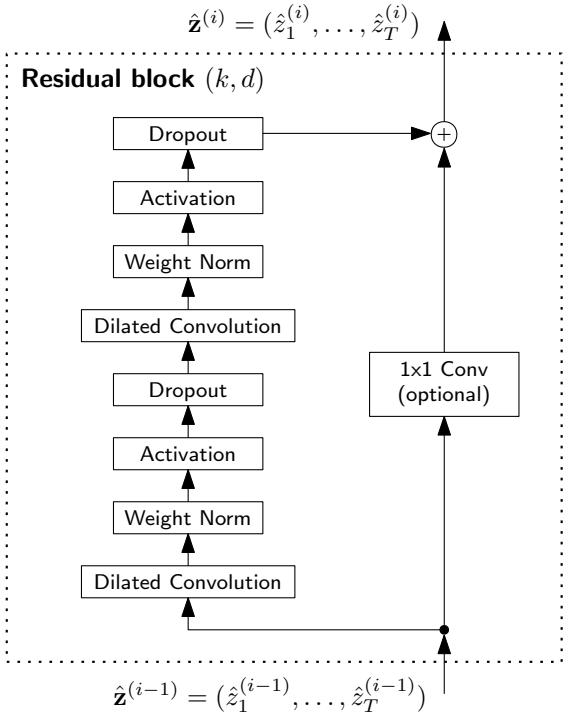


Figure 5: TCN residual block.

- 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] as a non-linearity
- for normalization a weight normalization [67] is applied to the convolutional filters
- additionally spatial dropout [52] is added after each dilated convolution for regularization (at each training step, a whole channel is zeroed out)
- The TCN architecture appears 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 emphasized the value of GPGPU for machine learning [70].

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 beat tracking system comprises three major stages, as shown in Fig. 6. In the first stage, the original audio signal is preprocessed. Generally, 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. The second stage represents the actual machine learning model, which is optimized based on the training data. The learned model takes the preprocessed audio as an input and outputs the beat activation which can be seen as a high-level feature of the musical piece. Finally, in the temporal decoding stage a dynamic Bayesian network (DBN) is used to model bars of various lengths and align the predicted beat positions to the global best solution.

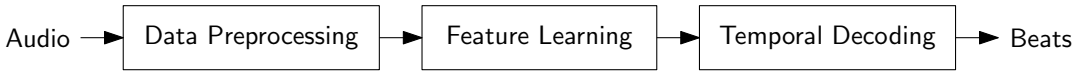


Figure 6: Overall structure of the beat tracking system.

To better understand the signal flow of the beat tracking system, the input signals to each stage are visualized in Fig. 7. For this purpose, the first verse of the track Yesterday by the Beatles is used as an example. The input to the data preprocessing stage is

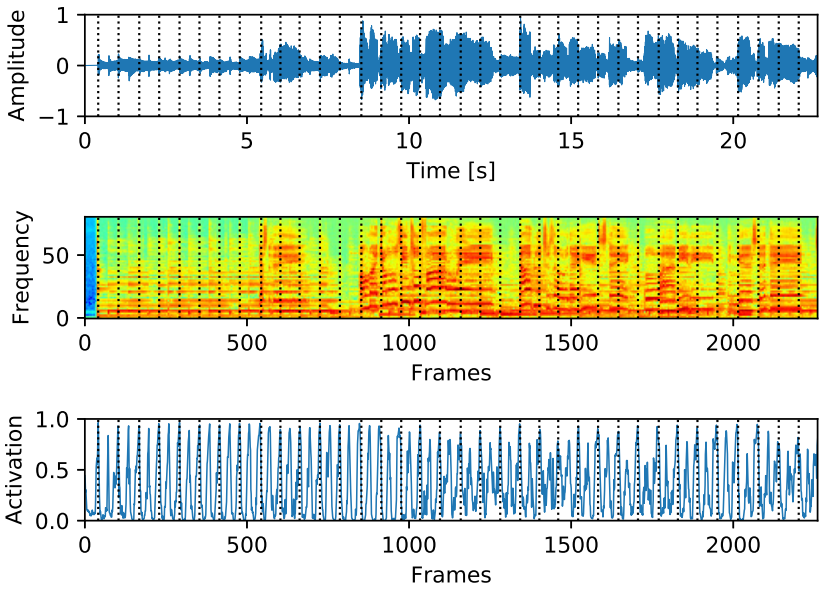


Figure 7: Input signals to the three main processing blocks of the beat tracking system for the first verse of the track Yesterday by the Beatles. On top the original audio signal is shown; in the middle the filtered log power spectrum; and bottom the beat activations. Dashed lines represent the annotated beats.

the discrete audio signal sampled at sampling rate f_s. In data preprocessing, the 1-dimensional audio file is transfered to a 2-dimensional filtered log power spectrum, which represents the amplitude of a particular frequency at a particular time. The output of the feature learning stage is the 1-dimensional beat activation, and represents the probability of a beat at each frame. In the last processing step, the temporal decoding stage take this high-level feature as an input and subsequently estimates the most probable beat sequence.

6.1 Dataset

The dataset that is used to train and evaluate the proposed beat tracking system is comprised of four common available beat tracking datasets. The different datasets are listed in Table 1 together with the number of files, and their total lengths. In total,

Table 1: Datasets used for training and evaluation.

Dataset	**Files**	**Length**
Ballroom [21, 71]	685	5 h 57 m
GTZAN [72, 73]	1000	8 h 20 m
Hainsworth [74]	222	3 h 19 m
SMC [75]	217	2 h 25 m
Total	2124	20 h 01 m

the compound dataset has a length of approximately 20 hours and contains 2124 files with ? beat annotations. The dataset comprises a wide range of different musical styles. However, the majority of the tracks represents western music. In the following, the properties of different subsets are described more detailedly.

Ballroom The Ballroom dataset [21, 71] contains 685 audio files[1] of ballroom dancing music. As musical genres the dataset covers Cha-Cha-Chá, 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 [72] dataset was originally proposed for music genre classification and later extended with beat annotations [73]. 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 [76].

Hainsworth The Hainsworth dataset [74] 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 [75] 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 [77]. 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

$$X(t,\omega) = \sum_{n=1}^{N} w(n)\, x(n + t\,h)\, e^{-2\pi j\omega/N} \tag{15}$$

where t refers to as the frame index, ω the frequency bin index, and the hop size $h = 441$, 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,

$$S(t,\omega) = \log\left(|X(t,\omega)|^2 \cdot F(t,\omega)^T + 1\right) \tag{16}$$

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 $S(t,\omega)$

$$\mathbf{x}_t = S(t,\omega), \quad \forall\, t = 1,\ldots,T \tag{17}$$

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 [78] 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.

6.3 Feature Learning

The neural network architecture consist of two main blocks, as shown in Fig. 8. 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 $a_{1:T}$.

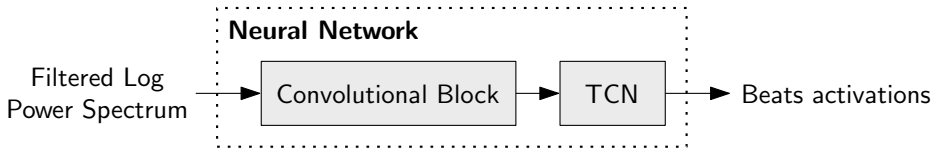


Figure 8: The neural network architecture consist of a convolutional block followed by a temporal convolutional network (TCN). The input corresponds to the filtered log power spectrum of the audio file, and the output is referred to as the beat activation, which represents the probability of a beat at each frame.

The convolutional block is a set of convolution and max pooling layers and reduce the dimensionality both in time and frequency. All convolutional layers contain 16 filters, with kernel sizes of 3×3 for the first two, and 1×8 for the last layer, as it is shown in Fig 9. The filters move with stride 1 and no zero padding is used. The intermediate max pooling layers apply pooling only in the frequency direction over 3 frequency bins. A dropout [79] rate of 0.1 is used with the exponential linear (ELU) [80] as activation function.

The output of the convolutional block is a 16-dimensional feature vector $\mathbf{z}$ and serves as the input to the TCN. The architecture of the TCN consists of 11 stacked residual blocks, as shown in Fig. 10. Each residual block contains 16 channels with filters of size 5, and the dilation $d = 2^i$ is geometrically spaced ranging from 2^0 up to 2^{10}. Thus, the resulting receptive field is approximately 81.5 s long. A spatial dropout with rate 0.1 and the ELU activation function is used. The output layer of the TCN is a fully-connected linear layer with two units, representing the two classes "beat" and "no beat". In order to normalize the output and to ensure computational stability, the log softmax function

$$\log \mathrm{softmax}(\hat{y}_i) = \log \left(\frac{\exp(\hat{y}_i)}{\sum_j \exp(\hat{y}_j)} \right) \tag{18}$$

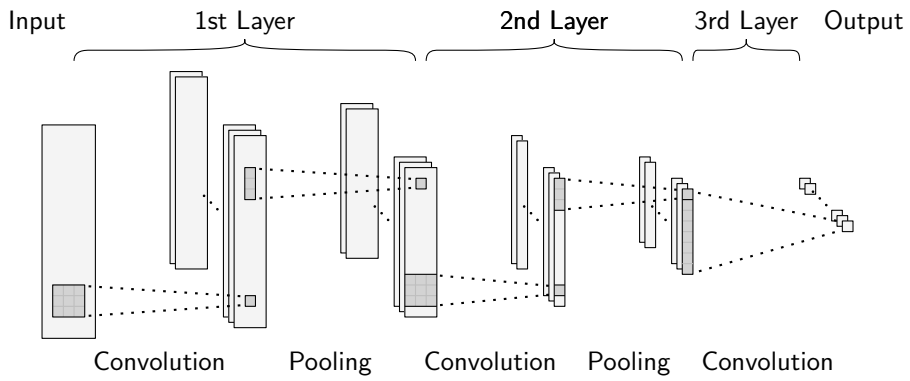


Figure 9: The convolutional block consists of 3 layers, all of which have 16 filters in total. The first two layers comprise filters of size 3×3 followed by pooling with a kernel of size 1×3. The last layer is a convolution with filter size 1×8. The resulting output is 16-dimensional vector $\mathbf{z}$.

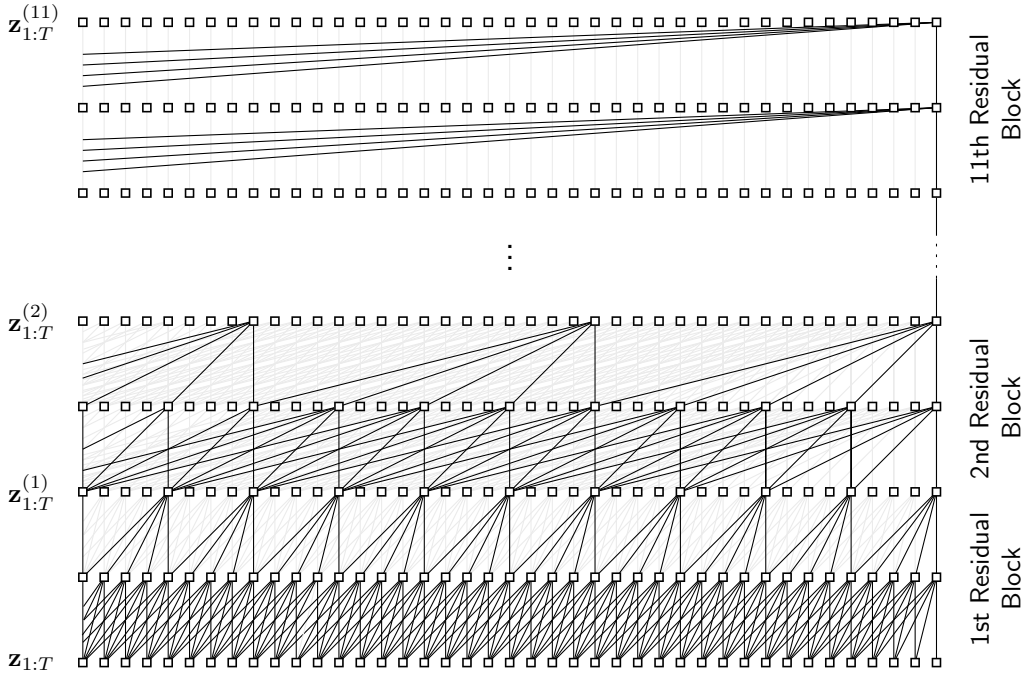


Figure 10: The architecture of the temporal convolutional network (TCN) consists of 11 stacked residual blocks. Each block comprises two layers of dilated convolution with filters of size $k = 5$. The dilation factor $d = 2^i$ geometrically widens with added stack level i. The black solid lines represent the dependencies for the last element $\mathbf{z}_T^{(11)}$ in the upper right corner of the figure.

is used which is added to the last layer of the neural network. Hence, 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. The network can thus be trained as a binary classifier with the cross entropy loss function, as defined in (4). The "beat" class is weighted with the factor $w_1 = 70$ to compensate the unbalancedness of the two classes. Finally, the beat activation a_t can be calculated by

$$a_t = \exp\left(\hat{y}_{1_t}\right), \quad \forall\, t = 1, \ldots, T \tag{19}$$

which represent the probability of a beat at frame t.

6.4 Temporal Decoding

Generally, the metrical structure of a musical piece builds upon a hierarchy of approximate regular pulses, as elaborated in Section 3. To exploit this sequential 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.

In the temporal decoding stage, following the feature learning, a dynamic Bayesian network (DBN) is employed which jointly infers the period and phase of the beat. Dynamic Bayesian networks are popular frameworks for meter tracking in music because they are able to incorporate prior knowledge about the dynamics of rhythmic parameters. More precisely, the post-processing stage constitutes a hidden Markov model (HMM), in which a sequence of hidden variables that represent the beat structure of an audio piece is inferred from the sequence of observed beat activations. The probabilistic state space consists of two hidden variables, namely the beat period, i.e. the reciprocal of the tempo, and the position inside the beat period.

The probabilistic model is based upon the bar pointer model of Whiteley et al. [81], which was originally proposed to jointly infer the tempo and downbeats of a musical piece. Where a downbeat refers to the first beat of a bar. In this model, mutual dependencies between rhythmic parameters are exploited, which increases the robustness of automatic rhythm detection systems. The introduced bar pointer is defined as being a hypothetical hidden object located in state space consisting of the period of a latent rhythmical pattern. The velocity of the bar pointer is defined to be proportional to tempo. This approach has the advantage that the bar pointer continues to move whether or not a note onset is observed. Hence, it 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 divided into discrete cells [18].

Given a sequence of beat activations $a_{1:T}$, the goal of the hidden Markov model is to identify the most probable hidden state sequence $\mathbf{s}^*_{1:T}$. For every time frame t, the hidden state is defined as $\mathbf{s}_t = (\phi_t, \tau_t)^T$ where $\phi_t \in \{1, 2, \ldots, \tau_t\}$ denotes the position inside a beat period, and $\tau_t \in \{\tau_{\min}, \ldots, \tau_{\max}\}$ refers to the length of the current beat period measured in frames. Due to the principle to use exactly one state per frame to indicate the position inside the beat period, the number of position states corresponds

exactly to length of the current beat period τ_t. Usually, the tempo Θ of a musical piece is measured in beats per minute (BPM), and therefore needs to be converted to the beat period measured in frames, by

$$\tau(\Theta) = \left\lfloor \frac{\Theta}{60} f_r \right\rfloor \tag{20}$$

where the frame rate f_r is defined as 100 frames per second. The limits of the beat period can be easily obtained as $\tau_{\min} = \tau(\Theta_{\max})$ and $\tau_{\max} = \tau(\Theta_{\min})$ respectively. Operating on a tempo range from 55 to 215 BPM results in a beat period ranging from $\tau_{\min} = 27$ to $\tau_{\max} = 107$ frames, and thus constructs a state space with 5427 states in total. As an illustration, a toy example with a significantly smaller state space is shown in Fig. 11.

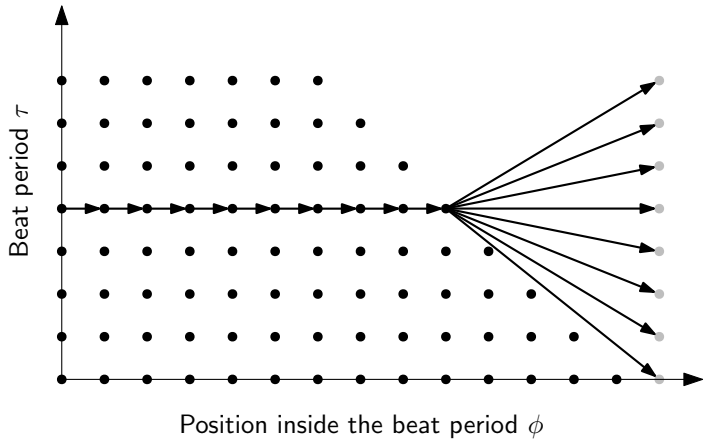


Figure 11: State space toy example with beat period range $\tau \in [7, 14]$. Each dot corresponds to a hidden state in the state space. The arrows indicate examples of possible state transitions starting at state $\mathbf{s}_1 = (\phi_1 = 1, \tau_1 = 10)^T$.

Given the aforementioned state space discretization, the most likely hidden state sequence $\mathbf{s}^*_{1:T} = (\phi^*_t, \tau^*_t)^T$ with $t = 1, \ldots, T$, given a sequence of beat activations $a_{1:T}$, is computed by

$$\mathbf{s}^*_{1:T} = \arg\max_{\mathbf{s}_{1:T}} P(\mathbf{s}_{1:T} \,|\, a_{1:T}) \tag{21}$$

with

$$P(\mathbf{s}_{1:T} \,|\, a_{1:T}) \propto P(\mathbf{s}_1) \prod_{t=2}^{T} P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})\, P(a_t \,|\, \mathbf{s}_t) \tag{22}$$

where $P(\mathbf{s}_1)$ is the initial state distribution, the term $P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})$ corresponds to the transition model, and $P(a_t \,|\, \mathbf{s}_t)$ refers to the observation model. The most likely hidden state sequence $\mathbf{s}^*_{1:T}$ can be solved using the well-known Viterbi algorithm [22]. Finally,

the set of beat instants $\mathcal{B}$ can be extracted from the sequence of the position within the beat period as

$$\mathcal{B} = \{t : \phi_t^* = 1\}. \tag{23}$$

In order to infer the hidden variables from an audio signal, the three entities, the initial distribution, the transition model, and the observation model, need to be specified. The transition 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 With the initial distribution $P(\mathbf{s}_1)$, any prior knowledge about tempo distributions can be incorporated into the model. For instance, if the music to be tracked from one genre, a specific tempo distribution can be used. To make it genre independent, 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 τ_t, this is

$$P(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) = P(\phi_t \,|\, \phi_{t-1}, \tau_{t-1})\, P(\tau_t \,|\, \tau_{t-1}) \tag{24}$$

where the first factor is

$$P(\phi_t \,|\, \phi_{t-1}, \tau_{t-1}) = \mathbf{1}_A \tag{25}$$

with the indicator function $\mathbf{1}_A$ that equals one if $\phi_t = (\phi_{t-1} + \tau_{t-1} - 1) \mod M + 1$. The modulo operator makes the bar position cyclic. If $\phi_t = \tau_t$, the second factor is defined by a Laplace distribution

$$P(\tau_t \,|\, \tau_{t-1}) = \exp\left(-\lambda \left| \frac{\tau_t}{\tau_{t-1}} - 1 \right| \right) \tag{26}$$

otherwise it is

$$P(\tau_t \,|\, \tau_{t-1}) = \begin{cases} 1, & \tau_t = \tau_{t-1} \\ 0, & \text{else.} \end{cases} \tag{27}$$

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 small value for p_ω would be quite appropriate, 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 [82]. The observation likelihood is defined as

$$P(a_t \,|\, \phi_t) = \begin{cases} a_t, & 1 \leq \phi_t \leq \frac{\lambda}{\Lambda} \\ \frac{1-a_t}{\lambda-1}, & \text{else.} \end{cases} \tag{28}$$

7 Evaluation

Robust evaluation is essential, not only to determine the individual successes and failures of the algorithm, but also to measure its relative performance among different algorithms. With the development of the Music Information Retrieval Evaluation eXchange (MIREX) [83], evaluation has become a fundamental aspect of research in music information retrieval. MIREX represents a community-based formal evaluation framework for the evaluation of algorithms and techniques related to music information retrieval.

In this section, existing beat tracking evaluation methods are reviewed, then subsequently applied to the proposed beat tracking system. The result is compared against the current state-of-the-art beat tracking algorithm, which is freely available within the Python library Madmom [84].

7.1 Evaluation Methods

The choice of evaluation method can have a significant impact on the relative performance of different beat tracking algorithms. An evaluation method should adequately contend with the inherent uncertainty or ambiguity while providing a measurement of performance which is both meaningful and easy to interpret [85].

Each evaluation method takes a set of annotated beat times $\mathcal{A} = \{a_1, \ldots, a_A\}$ and a corresponding set of predicted beat times $\mathcal{B} = \{b_1, \ldots, b_B\}$, where A refers to the number of annotations, and B to the number of detections, respectively.

F-measure In statistical analysis of binary classification, the F-measure, also called the F_1 score, is a generic measure to test accuracy. It considers both the precision p and the recall r of the test to compute a numerical score. The precision p is the number of true positives tp divided by the number of all positive results returned by the classifier, i.e, the number of true positives tp plus the number of false positives fp

$$p = \frac{tp}{tp + fp} \tag{29}$$

The recall r is referred to as the number of true positives tp divided by the number of all relevant samples, i.e, all samples that should have been identified as positive

$$r = \frac{tp}{tp + fn} \tag{30}$$

The F-measure is the harmonic mean of precision and recall

$$F_1 = \frac{2\,p\,r}{p + r} = \frac{2\,tp}{2\,tp + fp + fn} \tag{31}$$

where an F-measure reaches its best value at 1 (perfect precision and recall) and worst at 0. In beat tracking, a detected beat is regarded as true positive if it is in ± 70 ms range of an annotated beat. One can understand, the number of false positives as extra detections of the system, while the number of false negatives correspond to the missed detections.

Continuity measures In contrast to the F-measure, where the accuracy is determined from the overall beat tracking success, Hainsworth [86] and Klapuri et al. [87] developed an evaluation method, which measure the success of regions of continuously correctly tracked beats. Continuity is enforced by the implementation of tolerance windows of $\theta = \pm 17.5\,\%$ of the current inter-annotation-interval around each annotation. The closest detected beat to each annotation is only regarded as correct if it falls within this tolerance window and the previous detected beat is also within the tolerance window of the previous annotation. This condition addresses beat phase, but also requires consistency between inter-annotation-intervals.

Comparing each detected beat b_i to each annotation a_j, there is a number of correct beats in each continuously correct segment Y_m, where there are M continuous segments. The ratio of the longest continuously correct segment to the length of the annotations J defines the correct metrical level with continuity required

$$\mathrm{CML_c} = \frac{\max(Y_m)}{J}. \tag{32}$$

CMLc only reflects information about the longest segment of correctly detected beats. Therefore, it does not include the contribution of any other beats which may also be correct. For instance, if a single wrong detected beat occurs in the middle of the excerpt, this would lead to $\mathrm{CML_c} = 0.5$. To include the effect of beats in other segments Y_m, a less strict measure is defined as the total number of correct beats at the correct metrical level

$$\mathrm{CML_t} = \frac{\sum_{m=1}^{M} Y_m}{J}. \tag{33}$$

To account for ambiguity in metrical levels, Equations (32) and (33) are recalculated, such that the annotation sequence can be resampled to permit accurate tapping at double or half the correct metrical level and to allow for off-beat tapping. These conditions are referred to as allowed metrical level, which result in the two measures, allowed metrical levels continuity required ($\mathrm{AML_c}$), and allowed metrical levels continuity not required ($\mathrm{AML_t}$).

Information Gain Davies et al. [85] developed an evaluation method without reliance on tolerance windows, by measuring the timing error between beats and annotations. They propose to use the information gain D, which is a measure of how much information the beats provide about the annotations. To this end, an error histogram with K bins is formed from the resulting timing error sequence. The numerical score is then calculated as the Kullback-Leibler divergence with respect to a uniform distribution. The range for the information gain is 0 bits to approximately 5.3 bits, where the upper limit is determined as $\log_2(K)$. An information gain of 0 bits is obtained, when the beat error histogram is uniform, i.e. where the beat sequences are totally unrelated.

7.2 Results

The performance of the proposed beat tracking system is determined using the aforementioned evaluation methods. Each method is individually applied to the subsets of the training set. The achieved performance is compared against the state-of-the-art beat tracker by Böck et al. [17]. The overview of the performances of both beat tracking algorithms is shown in Tab. 2, where the best result is indicated as bold numbers.

Table 2: Overview of beat tracking performance.

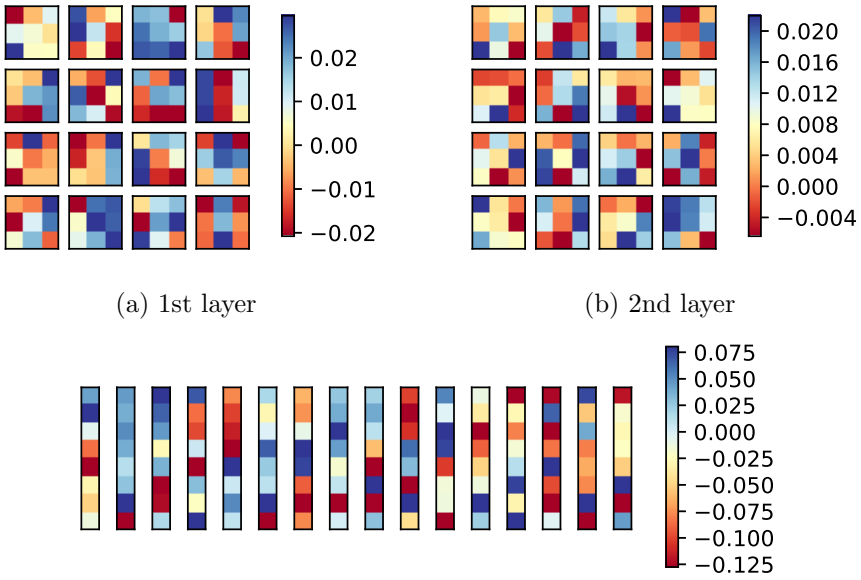
	F-Measure	CMLc	CMLt	AMLc	AMLt	D
Ballroom						
TCN	**0.848**	**0.826**	**0.850**	0.906	**0.946**	**4.761**
Madmom	0.839	0.797	0.815	**0.913**	0.933	4.626
SMC						
TCN	0.244	0.145	0.183	0.237	0.426	3.476
Madmom	**0.324**	**0.347**	**0.468**	**0.463**	**0.631**	**3.859**
Hainsworth						
TCN	0.536	0.615	0.695	0.711	0.845	4.661
Madmom	**0.560**	**0.740**	**0.830**	**0.818**	**0.919**	**4.796**
GTZAN						
TCN	**0.740**	0.700	0.726	0.866	0.914	4.543
Madmom	0.722	**0.755**	**0.775**	**0.902**	**0.930**	**4.610**

The inspection of Tab. 2 demonstrates that across all datasets and evaluation methods, the performance of the proposed beat tracking system is highly comparable to the existing state-of-the-art. For the Ballroom dataset, the performance is on a very high level and outperforms the approach in [17] in almost every evaluation method. Conversely, the SMC dataset reveals a significant drop in performance due to the large proportion of highly challenging musical excerpts. As a reminder, the approach in [17] uses multiple models which are specialized on certain datasets. In this case, exactly one model was trained only on the SMC dataset. Therefore the performance is essentially higher when compared against the proposed beat tracking system. This also confirms the assumption, that with the selection of training data the learned model can be specialized to certain musical styles.

In this case, exactly one model was trained only on the SMC dataset. Therefore the performance is essentially higher when compared against the proposed beat tracking system. This also confirms the assumption, that with the selection of training data the learned model can be specialized to certain musical styles.

7.3 Filter Activation Maps

In general, artificial neural networks are used because of good results. However, it is often difficult to understand what they actually do, or on which basis they generate their output. Inspecting the model's parameters sometimes gives a good intuition about the what exactly the network learned. For this purpose, the two-dimensional filters of the convolutional block of the final model are visualized in Fig. 12.



(a) 1st layer

(b) 2nd layer

(c) 3rd layer

Figure 12: Visualization of the 16 trained filters of the convolutional block. In (a) filters of the first layer, in (b) of the second layer, and in (c) of the third layer are shown.

and visualized to discover the types of features that the model will detect, and the activation maps output by convolutional layers can be inspected to understand exactly what features were detected for a given input image.

7.4 Network Training

The final model is trained with the ADAM optimizer [49] with a batch size $M = 20$ and a learn rate of $\mu = 1\,e^{-3}$. The whole system has only 30818 trainable parameters.

- Conclusion by Korzeniowski: 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 predictions 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 [88]

7.5 Labeling

- Enhance beat annotations the Sonic Visualiser [89] editor, 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 visualized using tools like Sonic Visualiser. 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 [90], where beats and downbeats were first obtained through the Ircambeat software and then errors were manually corrected by human annotators.
- 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 [85].
- Due to uncertainties in the annotation process, for many types of input signals (especially multi-instrument excerpts) it may not possible to determine onset locations with greater precision than 50 ms. [91]
- The ideal outcome of the annotation is an unambiguous representation of the start points of musical events

8 Conclusion

In this theses, a beat tracking system for musical audio was proposed, that uses convolutional neural networks to learn the beat activation as a high-level feature directly from the preprocessed audio. In a subsequent temporal decoding stage, a dynamic Bayesian network is used to model beat periods of various lengths and align the predicted beat positions to the global best solution.

Goal of supervised learning: predict not yet observed input.

Certain convolutional architectures can reach state-of-the-art accuracy in audio synthesis (e.g. Google WaveNet [64])

Common association between sequence modeling and recurrent networks: "*Given a new sequence modeling task or dataset, which architecture should one use?*" [24]

An efficient audio based beat tracking algorithm was presented, that is able to produce equivalent results to the current state-of-the-art.

Causal model and less parameters in contrast to [17].

System operates causally, thus an real-time implementation is possible.

What is most noteworthy about the proposed approach is that it can maintain competitive performances but with two distinct computational advantages over the state-of-the-art. The approach uses only 3xxxx parameters compared to

The no free lunch theorem for machine learning [50] states that, averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points. This implies that we must design our machine learning algorithms to perform well on a specific task.

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.

The domain of music is extremely complicated, and the machine learning model essentially involves simulating the entire universe.

Controlling the complexity of the model is not a simple matter of finding the model of the right size, with the right number of parameters. Instead, we might find that the best fitting model is a large mode that has been regularized appropriately.

Motivation: "*It is probable that no universal beat tracking model exist which does not utilize a switching model to recognize style and context prior to application.*" [92]

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 [74].

Theoretically, a single network large enough should be able to model all different music styles simultaneously but unfortunately this optional solution is hardly achievable. The main reason for this is the difficulty to choose an absolutely balanced training set with an evenly distributed set of beats over all the different dimensions relevant for detecting beats. These include rhythmic patterns, harmonic aspects and many other features

In practice, an overly complex model family does not necessarily include the target function or the true data.generating process, or even a close approximation of either.

The best way to make a machine learning model generalize better is to train it on

more data.

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 generally 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 [43].

Outlook:

Participate MIREX beat tracking contest 2019

Leave higher level analysis to the neural network

Instead of calculating the STFT as for input features 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.*" [43])

"*Feature extraction is an essential step towards effective and accurate beat/downbeat positions extraction.*" [93]

"*The system shows state-of-the-art beat and downbeat tracking performance on a wide range of different musical genres and styles. It does so by avoiding hand-crafted features such as harmonic changes, or rhythmic patterns, but rather leans the relevant features directly from audio.*" [94]

Take raw audio as input and learn Fourier transform with neural net.

Data augmentation (transformation or adding noise) ⇒ 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) [95].

Tried out Trellis Networks [96], but not better than TCN.

9 References

[1] F. Gouyon and S. Dixon, "A review of automatic rhythm description systems," *Computer music journal*, vol. 29, no. 1, pp. 34–54, 2005.

[2] C. Drake, M. R. Jones, and C. Baruch, "The development of rhythmic attending in auditory sequences: attunement, referent period, focal attending," *Cognition*, vol. 77, no. 3, pp. 251–288, 2000.

[3] E. Quinton, M. Sandler, and S. Dixon, "Estimation of the reliability of multiple rhythm features extraction from a single descriptor," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 256–260, IEEE, 2016.

[4] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals.," in *ISMIR*, vol. 5, pp. 304–311, Citeseer, 2005.

[5] A. Robertson and M. D. Plumbley, "B-keeper: A beat-tracker for live performance," in *NIME*, 2007.

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

[7] P. E. Allen and R. B. Dannenberg, "Tracking musical beats in real time.," in *ICMC*, 1990.

[8] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," in *Proceedings of the second ACM international conference on Multimedia*, pp. 365–372, ACM, 1994.

[9] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

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

[11] S. E. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, pp. 39–58, 08 2001.

[12] J. Laroche, "Efficient tempo and beat tracking in audio recordings," *Journal of the Audio Engineering Society*, vol. 51, no. 4, pp. 226–233, 2003.

[13] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2005.

[14] M. E. P. Davies and M. D. Plumbley, "Context-dependent beat tracking of musical audio," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1009–1020, March 2007.

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

[16] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11*, 2011.

[17] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," in *ISMIR*, 2014.

[18] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking.," in *ISMIR*, pp. 72–78, 2015.

[19] M. E. P. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," *European Association for Signal Processing*, 2019.

[20] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[21] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[22] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.," *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[23] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[24] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[25] O. E. Dictionary, "New york: Oxford university press," *Compact Edition)*, 1971.

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

[27] J. Lester, *The rhythms of tonal music*. Pendragon Press, 1986.

[28] J. London, "Rhythm," *The new Grove dictionary of music and musicians*, vol. 21, pp. 277–309, 2001.

[29] S. Handel, "Listening," *An introduction to the perception of auditory events, Cambridge, MA*, 1989.

[30] F. Lerdahl and R. S. Jackendoff, *A generative theory of tonal music.* MIT press, 1985.

[31] D.-J. Povel and P. Essens, "Perception of temporal patterns," *Music Perception: An Interdisciplinary Journal*, vol. 2, no. 4, pp. 411–440, 1985.

[32] R. Parncutt, "A perceptual model of pulse salience and metrical accent in musical rhythms," *Music Perception: An Interdisciplinary Journal*, vol. 11, no. 4, pp. 409–464, 1994.

[33] B. H. Repp, "On determining the basic tempo of an expressive music performance," *Psychology of Music*, vol. 22, no. 2, pp. 157–167, 1994.

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

[35] M. Yeston, "The stratification of musical rhythm," 1976.

[36] D. B. Huron, *Sweet anticipation: Music and the psychology of expectation.* MIT press, 2006.

[37] M. R. Jones and M. Boltz, "Dynamic attending and responses to time.," *Psychological review*, vol. 96, no. 3, p. 459, 1989.

[38] C. Drake and D. Bertrand, "The quest for universals in temporal processing in music," *PsychoL Sci*, vol. 13, pp. 71–4, 2001.

[39] C. Drake, L. Gros, and A. Penel, "How fast is that music? the relation between physical and perceived tempo," *Music, mind, and science*, pp. 190–203, 1999.

[40] C. Drake and M.-C. Botte, "Tempo sensitivity in auditory sequences: Evidence for a multiple-look model," *Perception & Psychophysics*, vol. 54, no. 3, pp. 277–286, 1993.

[41] C. Drake, A. Penel, and E. Bigand, "Why musicians tap slower than nonmusicians," *Rhythm perception and production*, pp. 245–248, 2000.

[42] E. F. Clarke, "Rhythm and timing in music," in *The psychology of music*, pp. 473–500, Elsevier, 1999.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[44] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[45] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*, pp. 5–13, Springer, 2012.

[46] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[47] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[48] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[50] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[51] A. N. Tikhonov, "On the stability of inverse problems," in *Dokl. Akad. Nauk SSSR*, vol. 39, pp. 195–198, 1943.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[53] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613, IEEE, 2013.

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

[55] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[56] A. Zell, *Simulation neuronaler netze*, vol. 1. Addison-Wesley Bonn, 1994.

[57] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[58] Y.-T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1141–1151, 1988.

[59] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[61] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[62] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[63] G. Weiss, Y. Goldberg, and E. Yahav, "On the practical computational power of finite precision rnns for language recognition," *arXiv preprint arXiv:1805.04908*, 2018.

[64] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[65] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[67] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

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

[69] K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004.

[70] D. Steinkraus, I. Buck, and P. Simard, "Using gpus for machine learning algorithms," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pp. 1115–1120, IEEE, 2005.

[71] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio.," in *ISMIR*, pp. 227–232, 2013.

[72] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[73] U. Marchand and G. Peeters, "Swing ratio estimation," pp. 423–428, 2015.

[74] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 15, p. 927847, 2004.

[75] 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*, vol. 20, no. 9, 2012.

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

[77] P. Grosche and M. Muller, "Extracting predominant local pulse information from music recordings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1688–1701, 2010.

[78] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 6979–6983, IEEE, 2014.

[79] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656, 2015.

[80] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[81] N. Whiteley, A. T. Cemgil, and S. J. Godsill, "Bayesian modelling of temporal structure in musical audio.," in *ISMIR*, pp. 29–34, Citeseer, 2006.

[82] 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*, vol. 20, pp. 290–301, Jan 2012.

[83] J. S. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.

[84] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new python audio and music signal processing library," *CoRR*, vol. abs/1605.07008, 2016.

[85] N. D. Matthew E. P. Davies and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," tech. rep., Centre for Digital Music, Queen Mary University of London, 2009.

[86] S. W. Hainsworth, *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, UK, September 2004.

[87] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 342–355, Jan 2006.

[88] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," *arXiv preprint arXiv:1612.05065*, 2016.

[89] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the ACM Multimedia 2010 International Conference*, (Firenze, Italy), pp. 1467–1468, October 2010.

[90] G. Peeters and H. Papadopoulos, "Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1754–1769, 2010.

[91] P. Leveau and L. 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.

[92] N. Collins, "Towards a style-specific basis for computational beat tracking," 2006.

[93] M. Khadkevich, T. Fillon, G. Richard, and M. Omologo, "A probabilistic approach to simultaneous extraction of beats and downbeats," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 445–448, IEEE, 2012.

[94] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks.," in *ISMIR*, pp. 255–261, 2016.

[95] P. K. Sebastian Böck, Matthew E.P. Davies, "Multi-task learning of tampo and beat: Learning one to improve the other," 2019.

[96] S. Bai, J. Z. Kolter, and V. Koltun, "Trellis networks for sequence modeling," *arXiv preprint arXiv:1810.06682*, 2018.