POLITECNICO DI MILANO
Master in Computer Science Engineering
Dipartimento di Elettronica e Informazione

Beat Tracking using Recurrent Neural Network: a Transfer Learning Approach

Master graduation thesis by
Davide Fiocchi

Supervisor:
Prof. Augusto Sarti
Co-supervisor:
Dr. Massimiliano Zanoni
Assistant supervisor:
Dr. Michele Buccoli

Academic Year 2016-2017

Alla mia famiglia ma soprattutto
ai miei nonni Elena, Flora e Franco.

Abstract

In the last decades we witnessed to crucial changes in the context of music fruition. With the increasing popularity of streaming services, that today are the main music content providers, there has been an exponential growth of the amount of music available. To provide suggestions to their users, content providers need to classify their vast catalogs, thus, meaningful information has to be extracted out of a musical piece. This has laid the foundation of the Music Information Retrieval (MIR), a research field which goal is to find approaches to automatically retrieve information from musical excerpts. One of the aspects that characterize most a music piece is the rhythm, which has the beat as its basic element. While for human beings the beat perception is an almost instinctive capability, its machine-based equivalent is a non-trivial task. One important research field of MIR is Beat Tracking which aims to automatically extract the beat out of a musical piece. Several methods have been proposed and one of the most promising involves Deep Neural Networks due to their ability to emulate the human mind. The effectiveness of deep learning networks is limited by the amount and the variety of data used for the training. For this reason, deep-learning models can be applied in scenarios where a huge amount of annotated data is available. In MIR this is the case of popular genres due to the wider availability of annotated datasets. Instead, to find sufficient data is an hard task for non widespread genres like folk music. A recent approach for overcoming the need of large datasets is transfer learning. It is based on miming the ability of human brain to address novel problems by applying knowledge acquired to solve similar problems in different contexts.

In this work, we propose an approach to apply transfer learning for beat tracking. We use a deep RNN as the starting network trained on popular music, and we transfer it to track beats of folk

music. Moreover, we test if the resultant models are able to deal with highly variable music. In order to evaluate the effectiveness of our approach, we collect a dataset of Greek folk music, and we manually annotate the pieces.

Sommario

Negli ultimi decenni abbiamo assistito ad una rivoluzione del consumo di musica. Con l'avvento dei servizi di streaming musicale, gli utenti sono diventati testimoni di una crescita esponenziale dell'offerta musicale a loro disposizione. Questa organizzazione mirata dei contenuti multimediali è resa possibile dall'estrazione di informazioni significative dai brani musicali. La necessità di un'estrazione automatizzata delle informazioni dall'audio ha gettato le basi dell'area di ricerca del Music Information Retrieval (MIR). In particolare questo studio si focalizza sugli aspetti legati alla struttura ritmica dei brani, che vede il beat come suo elemento base. Mentre la percezione del beat è una capacità istintiva per l'uomo, automatizzarla non è un compito banale. Infatti, l'automatizzazione del Beat Tracking è uno degli ambiti di studio più importanti nel contesto MIR. Tra i metodi proposti, uno dei più promettenti coinvolge le reti neurali. L'efficacia di questi modelli è limitata dalla quantità di dati utilizzati per l'apprendimento. Dunque, le reti di deep learning possono essere allenate e applicate in scenari in cui sono disponibili enormi quantità di dati. Nel caso del MIR, mentre per generi musicali diffusi sono disponibili molti dati, per generi di nicchia come la musica folk la disponibilità di dati utilizzabili diminuisce sensibilmente, per cui risulta complesso allenare e utilizzare modelli di deep learning. Un approccio recente per superare questa problematica è chiamato transfer learning. Si basa sul mimare la capacità del cervello umano di affrontare nuovi problemi applicando le conoscenze acquisite per risolvere problemi simili in contesti diversi.

In questo lavoro, proponiamo un approccio al beat tracking basato sul transfer learning. In particolare un RNN come rete di partenza addestrata alla musica popolare, e ne trasferiamo la conoscenza acquisita in questa fase preliminare per tracciare la musica folk. Inoltre, testiamo se i modelli risultanti sono in grado di gestire musica

molto variabile. Per valutare l'efficacia del nostro approccio, abbiamo raccolto e annotato manualmente un set di dati di musica folk greca.

Ringraziamenti

Il primo pensiero lo rivolgo alla passione per la musica che mi ha spinto (e mi spinge) a compiere delle scelte che mi rendono felice. Sono contento che questo mi guidi in percorsi interessanti, con assoluta naturalezza.

Vorrei ringraziare Max e Michele, i quali hanno seguito e guidato i nostri inesperti passi verso il raggiungimento della meta. Il loro contributo è stato fondamentale, grazie. Un ringraziamento anche al Prof. Augusto Sarti e a tutte le persone del laboratorio con le quali sono stati condivisi i mesi della tesi.

Un grosso ringraziamento va a tutti i nuovi amici del team di Como con cui sono stati condivise gioie e dolori. Vorrei ringraziare le amiche della Banda della Mañana con le quali ci siamo spalleggiati nei momenti di disagio all'estero.

L'esperienza comasca mi ha allontanato fisicamente da ciò che ho sempre chiamato "casa" ma non sentimentalmente. Voglio ringraziare tutti i miei amici di sempre per il loro supporto costante.

Ovviamente tutto ciò non sarebbe stato possibile senza l'aiuto e l'appoggio della mia famiglia che è con me sempre, grazie Ma, Pa, Luca e nonni tutti.

Infine, il ringraziamento più grande va a chi, nella quotidianità, è sempre presente, mi sopporta e supporta e colma il vuoto al mio fianco. Grazie topa.

Contents

List of Figures

Chapter 1

Introduction

In the last decades, we have witnessed a significant revolution in the context of music fruition. The raise of content providers such as Spotify, Apple Music, Tidal, SoundCloud, has encouraged a shift of the musical habits of the customers from the purchase of physical media to the music consumption based, above all, on streaming services. As a result, a crucial augment of the amount of available music has been observed. Vast catalogs needs to be classified in order to provide meaningful suggestion to their users. In fact, in this new context, it is possible to provide music catalogs based on the user's tastes by acquiring an information on user's preferences and choices. This can lead to a targeted recommendation of new music for the user, or to a targeted browsing methods which can be tuned on the user's preferences. The classification of the musical catalogs is done by extracting musically semantical descriptors of the audio (such as genre, mood, etc.). It is not feasible nor cheap to manually describe each piece in those catalogs since the amount of data is growing exponentially. Moreover, a human-based description is not systematic nor replicable, with the consequence of possible ambiguous characterization. All these opened questions led to the birth of the research field of *Music Information Retrieval* (MIR). MIR's main goal is to find ways to automatically extract useful information out of a musical piece by applying techniques coming from signal processing, machine learning combined with musical knowledge, psychoacoustic theory, etc.. The information is encoded into *features* which are computed to provide a description of the musical excerpt. These features can be divided in three levels of abstraction, intended as how much the provided description is close to the human perception. Low-level features are extracted directly from the audio signal and are not perceptually motivated but are used to encode the characteristics of the audio signal. Mid-level features are computed from the low-level ones, combined with musical knowledge that lead to a more complete and human-comprehensible set of informations such as harmony and rhythm. Lastly, the high-level features represent the highest level of abstraction since they contribute to build a perceptive and emotional description of a piece. In this study we will concentrate on rhythm and in particular on the detection of one of its fundamental characteristics: the beat.

At a perceptual level, the beat is what make us, humans, to tap our feet or snap our fingers while listening to music. The beat is, first of all, a perceptual phenomenon and the capability of the listeners to perceive it is due to our instinctive ability to discern periodicities in a series of events of a musical excerpt (such as a chord, a note in the melody, a percussive events, etc.). The perception of an event is related to acoustic characteristics of the signal such as the transient. The transient can be understood as a short-time interval in which a significant increase followed by a decay occurs in the signal's amplitude. In fact, the sequence of musical events in a song is perceived as the sequence of transients (or, more specifically, peaks of transients) detectable in the audio signal. Furthermore, this perception of the events in a musical piece is not only due to the local position of the transient, but it is given also and especially by the overall structure of the piece. The rhythmical structure provides an organized and periodical disposition of the musical events. Thus, based on this structure, the listeners are able not only to perceive the local beat occurrence, but also to to foresee the next occurrences thanks to their periodical disposition. As a result, the perception of the rhythm of a piece depends on a hierarchical arrangement of its characteristics which starting from its unit, the beat, grows to higher structural levels.

Beat tracking is one of the most important research field of MIR that aims to automatically extract the beat sequence out of a musical piece. This is vitally important in order to analyze the rhythm structure of an audio signal, which is obviously useful for classifying songs. Furthermore, being the beat a mid-level feature, it is also used for higher level of audio analysis (such as segmentation or pattern discovery). Moreover, the automatic extraction of the beat sequence is the core of tools in music production such as automatic time correction of recorded audio or self-adapting digital audio effects [12].

Several methods have been developed in literature to address automatic beat tracking issues. Most of the them employ probabilistic approaches to mimic the human perception of rhythmical structure. Among the approaches, *deep learning* seems one of the most promising.

In Music Information Retrieval (MIR), deep learning has been applied to several tasks, including automatic music classification [8], music structural analysis [6] and beat tracking. With regard to the latter, some architectures have been proposed that employ deep neural network [20] specifically designed to deal with sequential data such as Recurrent Neural Networks [4, 5]. One of the most crucial issues related to the use of deep learning is the need of a huge amount of data to properly and effectively train the models. Another crucial subject for deep neural networks to be effective is related to the variety of the training data: networks trained with collections that present a high inner variability are more likely to succeed in processing new data [34]. Nevertheless, acquire large datasets with an high variability is an hard task and is not always feasible.

A recent approach to overcome the issue of the absence of sufficient data is based on miming the ability of human brain to address novel problems by applying methods and knowledge acquired to solve similar problems in different contexts [47]. In deep learning, one of the possible approaches is to exploit networks knowledge acquired to accomplish a *source* task to achieve a *target* task, different from the initial one. In other words, the parameters resulting from a previous network's training phase, focused on a primary task, are used to define a network capable of perform a new task. Thanks to this process, the network can benefit from the large available datasets to build a hierarchical representation of the input data, and then transfer the learned knowledge for another task. This procedure is known in literature as *Transfer Learning.* Transfer Learning been effectively used for different tasks, and in [10] the authors present an investigation of the approach in MIR.

In this work, we study the effect of applying transfer learning to beat tracking of traditional folk music. In fact, for this genre, the lack of large and diverse datasets is particularly burdensome and that is why using transfer learning comes in handy. We started from the deep recurrent bi-directional networks with Long Short-Term Memory cells proposed in [5], as the starting network. We then employ the learned parameters on a network designed for folk music beat tracking. In order to evaluate the effectiveness of our approach, we collect a dataset of Greek folk music, which was smaller but with an higher variety of data with respect the source dataset. In the new dataset the beat is manually annotated for each piece.

1.1 Thesis outline

Chapter 2 illustrates the theoretical background needed in order to understand the proposed method. Chapter 3 provides an overview on state-of-the-art techniques regarding beat tracking and transfer learning. Chapter 4 gives an in-depth description of the system in all its components. The experimental setup and the obtained results are shown in Chapter 5. Lastly, in Chapter 6 we present conclusions and proposed future work.

Chapter 2

Theoretical Background

This chapter dives into the theoretical background on which this study lays its foundation.

Section 2.1 provides a music theory background that will be used in the rest of our study. Sections 2.2, 2.3 and 2.4 describes the tools that are needed to understand how our beat tracking method works. In particular, Section 2.2 supplies a definition of the signal processing methodologies used in our method. Section 2.3 provides a description of neural networks, with a particular focus on the type used in this study. In Section 2.4 we provide a mathematical description of the probabilistic framework used.

2.1 Musical background

Music is, in general, an event-based phenomenon characterized by changes occurring at specific time positions. Musical events can be roughly divided in two non-disjointed categories: percussive and harmonic events. The former is described as an abrupt change in the signal's amplitude. The latter is characterized by a variation in the frequency components of the signal but signal's energy might not change that much.

Each event is characterized by four time-related properties: attack, decay, transient and onset [1]. A graphical representation of these properties is depicted in Figure 2.1. The attack is the time interval in which the signal's amplitude increases. Immediately after the attack, the decay is a time interval in which the event's contribution tends to a more stable progression, characterized by

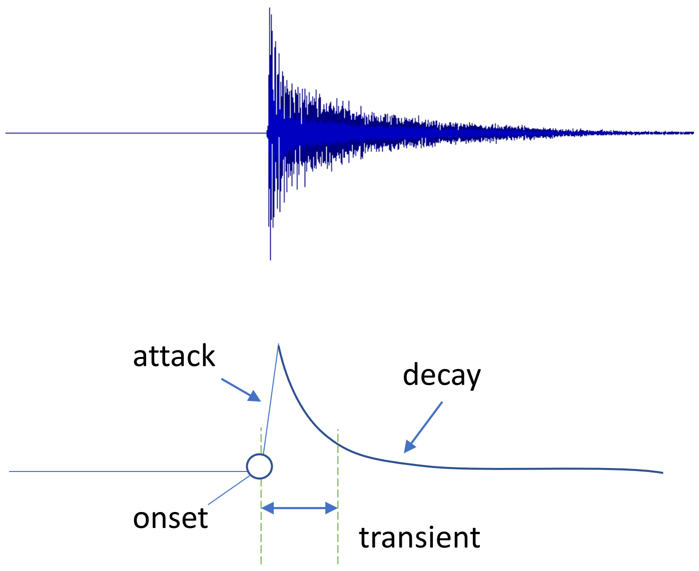


Figure 2.1: Single event with its time properties.

a slowly variation towards a “rest position”. The transient is the initial time of the event in which the signal evolves quickly and in non-predictable way. It incorporates the whole attack and part of the decay phase. Lastly, the onset is the exact time at which the event starts, i.e. the beginning of the transient.

As aforementioned, the beat is perceived through the series of events, based on their periodical disposition in time. Depending on past event’s positioning, humans naturally infers the rhythmical structure of the piece and are able to foresee where the next beats’ locations are. In music notation, beats are grouped into bars or measures, which are segments that contain a constant number of almost equally-spaced beats. The number of beats within a bar depends on the meter of the measure that is a formal way to define its “rhythmic shape”. The first beat of the bar is called downbeat.

2.2 Signal processing

Given a raw audio signal, some processing needs to be done in order to extract meaningful information. The *Short-Time Fourier Transform* renders the original signal into its evolution of time-frequency components and it is the core block of most systems that deal with audio signals. We are going to describe in Chapter 4 how signal processing techniques are employed to transform the audio signal into its representation used as input to a deep learning model. Since the latter mimics the capabilities of the human mind, it is better to give as input a representation as close as possible to human perception. Therefore, the *filter-bank* is employed to convert the linear frequency resolution of the STFT into a logarithmic one.

2.2.1 Short-Time Fourier Transform

The *Short-Time Fourier Transform* (STFT) is a transform used to determine the sinusoidal frequency and phase component of local sections of a signal $s(t)$ as it changes over time t.

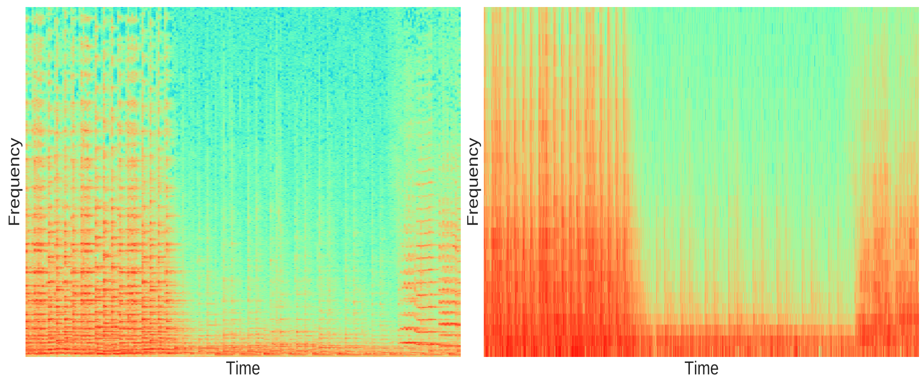
In general, the signal is sliced into frames by the multiplication of a sliding window $\mathrm{w}(t)$ where $\mathrm{w}(t) \neq 0$ for $\frac{L_\mathrm{w}}{2} \leq t \leq \frac{L_\mathrm{w}}{2} - 1$, and L_w is the window length. To obtain the STFT coefficients, each sample of the windowed signal is multiplied by a complex sinusoid. Formally, the STFT is defined as

$$\begin{gathered} STFT_i(k) = \sum_{t=-N_{FT}/2}^{N_{FT}/2-1} \mathrm{w}(t) s(t + Hi) e^{-j\omega_k t} \\ \text{with } k = 0, \ldots, N_{FT} - 1 \ , \end{gathered} \tag{2.1}$$

where $\omega_k = 2\pi k / N_{FT}$ is the radiant frequency assigned to bin k, i indicates the frame index, H is the hop-size, N_{FT} is the size of the transform.

In other words, every H time samples, a Discrete Fourier Transform (DFT) of length N_{FT} is applied to portion of the signal of length L_w. In order to be able to compute the DFT $L_\mathrm{w} \leq N_{FT}$ since the number of samples of the audio signal needs to be smaller than the transform dimension. At the same time, the hop-size must be such that $H \leq L_\mathrm{w}$ in order to be able to analyze the whole audio signal. To efficiently compute the DFT, $N_{FT} = 2^n$ since the algorithm used to calculate the DFT, called Fast Fourier Transform, has maximized performances if the size of the transform is a power of two.

The frequency resolution of the DFT is linked to the transform dimension, in fact, each frequency bin of the DFT represent a range of continuous frequencies equal to $\Delta\psi = \psi_s / N_{FT}$, where ψ_s is the



(a) STFT with N_{FT}=4096. *(b) STFT with N_{FT}=512.*

Figure 2.2: STFT comparison of fifteen seconds of song. Please note that if we increase L_{w}, we are able to see the frequency progress through time (horizontal lines). On the contrary, with a small L_{w}, we can clearly see the vertical lines denoting the onsets. In both cases $L_{\mathrm{w}} = N_{FT}$.

sampling frequency of the original signal. Moreover, the bigger L_{w}, the more samples of the signal are taken, leading to a better analysis of the frequency component of the signal. Figure 2.2a shows this concept graphically. This has to be taken into account if we aim to detect an harmonic event (i.e. whenever there is a change in the frequency component of a signal). On the other hand, a lower value of L_{w} led to higher time-resolution (see Figure 2.2b). This phenomena lead the procedure to be more effective in capturing percussive events in the signal. In fact, percussive events are characterized by abrupt change in the signal's amplitude, which, if falling inside a window, will be distributed to all its length. Thus, in case two percussive events fall within the same window, it is not possible to discriminate them. This phenomena is called *time-frequency compromise*. To summarize, a big window is preferred to have an higher frequency resolution a narrow window is preferred to have a higher time resolution. To have a graphical visualization of the problem see Figure 2.2.

2.2.2 Filter-bank

To better match the human earing system and to reduce the number of STFT coefficients, it is possible to filter each frame of the STFT with a *filter-bank*. It consists of a series of overlapped filters that acts on an STFT frame. Each filter analyzes a portion of

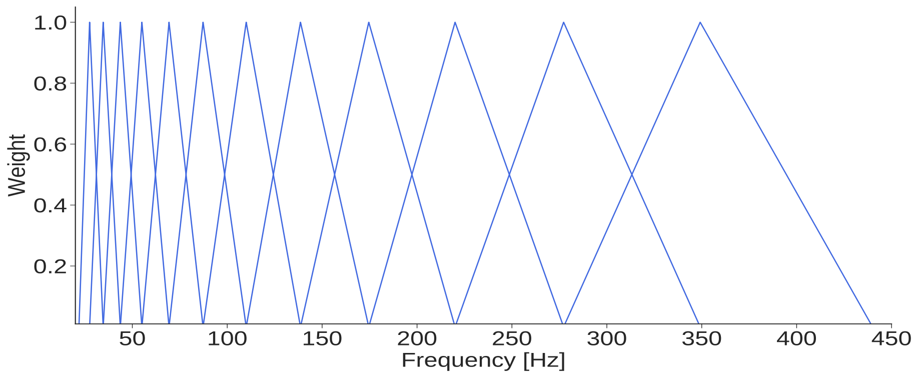


Figure 2.3: Triangular filter-bank logarithmically spaced on the frequency axis.

a frame by extracting a single coefficient out of it. Since the frequency perception of human earing system is logarithmic, filters are logarithmically spaced in frequency domain. The filter-bank acts on every frame of the STFT producing a new frequency representation much closer to human perception.

In practice, each filter applies the dot product of part of the frame's coefficients multiplied with a window coefficient. Taking into account that the STFT has a linear frequency resolution whereas the filter-bank is logarithmically spaced, the resultant windows will have different widths (in terms of number of coefficients): windows that deal with lower frequencies will be narrower compared to the ones at higher frequencies. A graphical visualization of a triangular filter-bank is provided in Figure 2.3.

2.3 Neural Network

Neural networks are powerful non-linear models that map the input features $\mathbf{x}$ to the output $\mathbf{y}$. In a formal way, $\mathbf{y} = f(\mathbf{x}; \Theta)$, where f is the mapping function and Θ are its parameters. The mapping is learned during the *training* phase in which the network parametrization Θ change accordingly. Neural networks are loosely inspired by how biological brain works: simple processing units (the neurons) are connected to each other. Weighting the connections by modifying Θ during the training phase, the network is actually able to learn an optimal representation of the input. The model capabilities to describe a particular representation strongly depends on its *topology*.

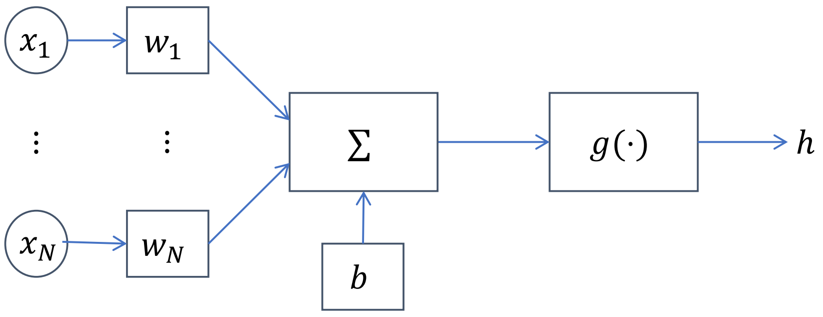


Figure 2.4: A graphical representation of an artificial neuron.

In the following sections we introduce and describe the network topologies used in this study. For a more in-depth discussion, we refer the reader to [26].

2.3.1 Topology

A neural architecture is defined by its elementary processing units and by the way these are interconnected. In the following subsections we review the network structure starting from its basic element (the artificial neuron) and fundamental topology, up to more complex deep architectures and advanced type of units.

Artificial neuron

The basic processing unit is the artificial neuron. The output h is calculated as a non-linear function g (called *activation function*) of the weighted sum of the inputs $\mathbf{x}$. Formally,

$$h = g(\mathbf{x}^T\mathbf{w} + b) = g(\mathbf{x}; \boldsymbol{\theta}) \ , \tag{2.2}$$

where $\mathbf{w}$ is the set of weights and b is the bias. To ease the notation, we incorporate $\mathbf{w}$ and b into a single variable $\boldsymbol{\theta}$. Figure 2.4 gives a graphical representation of the artificial neuron.

The activation function is a static, non-linear function that plays an important role in the learning phase. As will be explained in Section 2.3.2, some properties of the activation function (such as the differentiability) are crucial for the learning algorithm to work properly. Several activation functions are present in the literature. Some of the most commonly adopted activation functions are: sigmoid, hyperbolic tangent, softmax. In Section 2.3.1 we will provide a formal description of the activation functions we use in this study.

It could be demonstrated that with a single neuron is possible to solve linearly separable problems (such as bitwise AND or bitwise OR). In fact, its computational power is very limited but the power of neural networks lies in the connections among neurons.

Multiple Layer Perceptron

A connection between two neurons exists if the output of the first is one of the inputs of the second. It could be demonstrated that, by connecting more neurons, the overall function from input to output, is able to solve non-linearly separable problems.

One of the most common structure is the *Multiple Layer Perceptron* (MLP) also known as *Feed-Forward Neural Network* (FFNN). In this kind of topology, neurons are partitioned into layers, and adjacent layers are connected in an unidirectional way, without feedbacks. Basically, each neuron receives an input from all the neurons of the previous layer (or the network's input in case it belongs to the first layer) and its output is used to feed next layer's neurons (or the network's output in case it belongs to the last layer). A graphical representation of the MLP can be found in Figure 2.5. Formally, given a network of L_l layers, the output of the l-th layer with $0 < l < L_l$ is:

$$\mathbf{h}^{(l)} = g^{(l)}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) = g^{(l)}(\mathbf{h}^{(l-1)}; \boldsymbol{\theta}^{(l)}) \ , \tag{2.3}$$

where $\mathbf{h}^{(l)}$ is the layer's output, $g^{(l)}$ is the activation function of the layer's neurons, $\mathbf{W}^{(l)}$ is the weight's matrix and $\mathbf{b}^{(l)}$ is the bias vector. Similarly,

$$\mathbf{h}^{(0)} = g^{(0)}(\mathbf{x}; \boldsymbol{\theta}^{(0)}) \tag{2.4}$$

and

$$\mathbf{y} = g^{(L_l-1)}(\mathbf{h}^{(L_l-2)}; \boldsymbol{\theta}^{(L_l-1)}) \ , \tag{2.5}$$

represent the first layer's output and network's output, respectively.

The layers with $0 < l < L_l - 1$ are called *hidden layers* since their contribution is not visible outside the model, whereas $l = 0$ and $l = L_l - 1$ are called the input and output layer, respectively. In general a network is denoted as *deep* if $L_l > 3$, i.e. has more than one hidden layer.

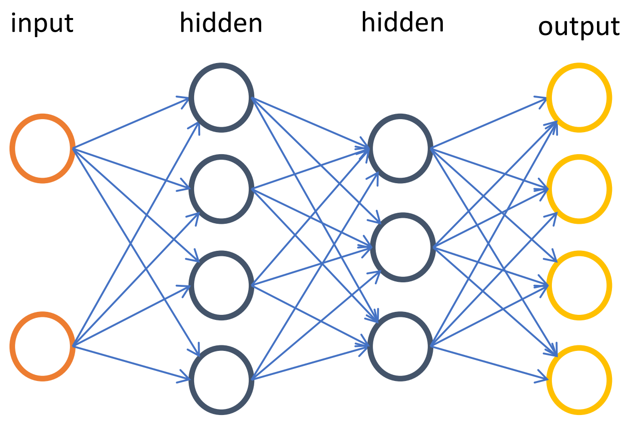


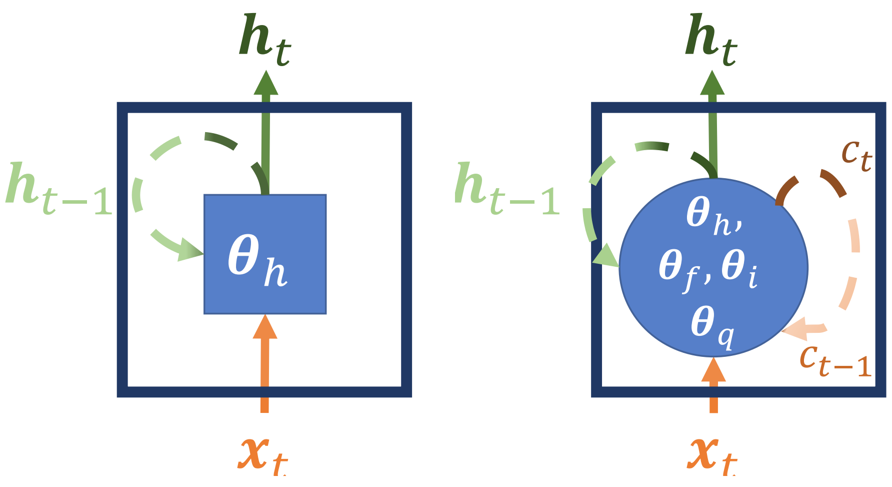
Figure 2.5: Multiple Layer Perceptron with two hidden layers.

Recurrent Neural Network

RNNs are a family of neural networks specialized in modeling sequential data, i.e., the temporal evolution of a signal. Due to this ability, RNNs are used in MIR to model the time-varying evolution of musical properties. In this Section we provide an overview of the basic Recurrent Neural Networks (*Vanilla* RNNs) and the Long Short-Term Memory cells we used in our study.

Vanilla Recurrent Neural Network Given a sequence $\mathbf{x}_t \in \mathbb{R}^{N_\mathbf{x}}$ representing a temporal sequence of data over a time index t, a RNN aims at modeling its time evolution and generate hidden units $\mathbf{h}_t = \sigma\left(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\theta}^{(l)}\right) \in \mathbb{R}^{N_l}$ as a function σ of the current sample $\mathbf{x}_t$ and of the hidden units of the previous samples with parameters $\boldsymbol{\theta}^{(l)}$. Doing this, the function recursively access to past samples and hidden units: $\mathbf{h}_t = \sigma_l\left(\sigma_l\left(\mathbf{h}_{t-2}, \mathbf{x}_{t-1}\right), \mathbf{x}_t\right)$, etc., allowing the RNN to model the evolution of the sequence (we use σ_l to indicate σ using $\boldsymbol{\theta}^{(l)}$ parameters). See Figure 2.6a for a simple graphical representation of a RNN unit.

RNNs can be stacked together in a deep architecture (shown in Figure 2.7), allowing the model to exploit a hierarchical representation of information and effectively bridge the gap between low-level input and high-level desired output. More formally, given a network with L_l layers, the units $\mathbf{h}_t^{(l)} \in \mathbb{R}^{N_l}$ from the generic layer l are



(a) Vanilla RNN's unit. (b) LSTM unit.

Figure 2.6: A graphical comparison of Recurrent units.

computed as

$$\mathbf{h}_t^{(l)} = \sigma_l \left(\mathbf{h}_{t-1}^{(l)}, \mathbf{h}_t^{(l-1)} \right), \text{ with } \mathbf{h}_t^{(0)} = \mathbf{x}_t \ . \tag{2.6}$$

Each layer l learns a different parametrization $\boldsymbol{\theta}^{(l)}$. The predicted output $\hat{\mathbf{y}}_t$ can be computed from the top (and most semantic) hidden layer's output $\mathbf{h}_t^{(l)}$ or from the stacking of two or more hidden layers, depending on the task.

In some cases it might be useful to exploit not only the past hidden values but also future ones. In [51] Bidirectional RNNs (BRNNs) are proposed. In a BRNN, each layer of the RNN is composed of two sub-layers, one that learns $\mathbf{h}_t$ given $\mathbf{h}_{t-1}$, and one that learns $\mathbf{h}_t$ given $\mathbf{h}_{t+1}$. On the one side, BRNNs are more effective than RNN to predict a desired output, since they are able to exploit more information and, in particular, to "look into the future"; on the other side, BRNNs are not causal and their application is limited to offline tasks.

There is, however, an important issue that prevents RNNs from learning an effective multi-layer representation of an input sequence, namely the *vanishing gradient* problem [5], i.e., the gradient of the loss function decreases dramatically, making the weights' update to stop. The LSTM cells [32] address this issue by defining an internal memory and using *gates* to control how much information from input and previous output is stored in the memory, or when to forget past

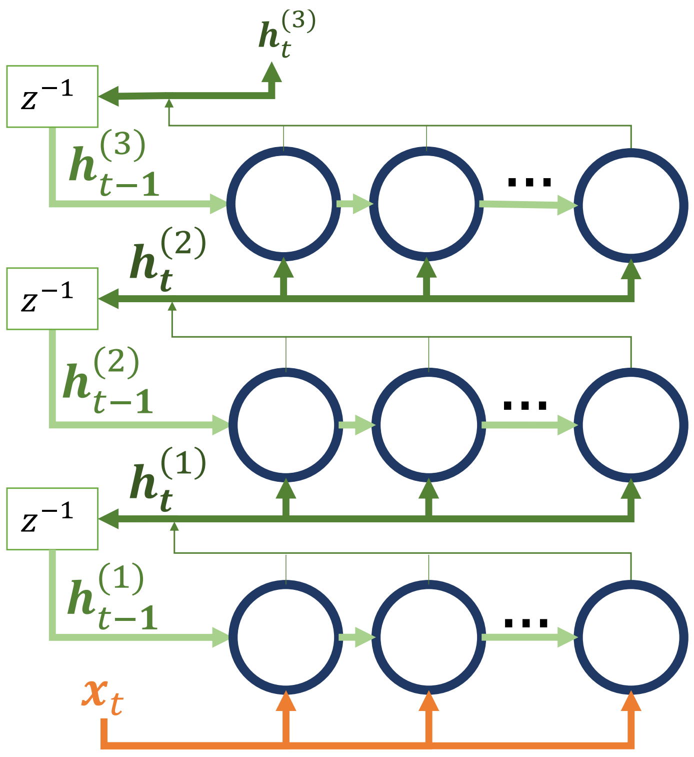


Figure 2.7: A representation of 3-layered RNN with varying number of hidden units.

samples, and to weigh the output of the network.

Long Short-Term Memory units With respect to vanilla RNN units, LSTM units include a cell state c_t that acts as a memory for the unit, and three *gates* f, i and q, named *forget*, *input* and *output* gates respectively, defined as scalars ranging between 0 and 1 to weigh the contribution from $\mathbf{x}$, $\mathbf{h}_{t-1}$ and c_{t-1} from the previous samples.

More formally, given a one-layer RNN , at a certain instant t, the forget gate f_t is computed as $f_t = \sigma_f(\mathbf{x}_t, \mathbf{h}_{t-1})$, using the function σ with specific parameters $\boldsymbol{\theta}_f$. The gates i_t and q_t are computed accordingly over the same input and parametrized by $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_q$ respectively.

Each gate is composed of artificial neuron whose output range is between 0 and 1, so the activation function g for each neuron has to be selected accordingly. In [32] the *sigmoid* and the *hyperbolic tangent* activation functions are used and their formal respective

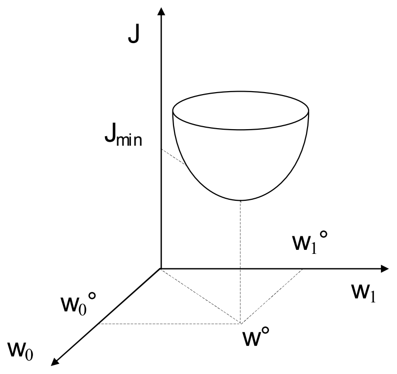


Figure 2.8: A loss surface as a function of two parameters[1].

definition is

$$\varsigma(x) = \frac{e^x}{e^x+1} \qquad \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}\ . \tag{2.7}$$

In LSTM, a vector $\tilde{\mathbf{h}}_t = [\mathbf{x}_t, \mathbf{h}_{t-1}]$, is defined and is used as input to all the LSTM gates. The state cell is first updated as $c_t = f_t c_{t-1} + i_t \tilde{\mathbf{h}}_t$. Basically, the state is updated by weighing the contribute of the previous state (first addend) and of the input (second addend). Lastly, the hidden units are computed by weighing the contribution of the cell state as $\mathbf{h}_t = \tanh(c_t) q_t$.

LSTM is a variation of the modeling and learning of the input sequence, while keeping the same external architecture. For this reason, LSTM-based layers can be stacked to compose a deep network. In this study we use Bi-Directional LSTM-based RNNs (BLSTM). The training of the network with respect to a desired input y_t is analogous to the classical BRNNs, except that the space of parameters has increased, since now the network also needs to learn $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_q$ (Figure 2.6b), increasing the requirements in terms of training data.

2.3.2 Training

During the training phase, the model iteratively adjust its parameters according to the mapping function f we aim to obtain. In particular, at each iteration, the model process a set of data and predicts

[1]Image taken from Model Identification and Adaptive Systems lecture notes. Chapter 5: Adaptive Filtering

an output. The training algorithm acts on the model parametrization Θ in order to minimize the "difference" between the predicted output $\hat{\mathbf{y}}$ and the desired one $\mathbf{y}$. The relation between the output difference and the model's parametrization is described by the *loss function*, also known as *cost function* (Figure 2.8). The training process aims to detect the global minima of the loss function by iteratively searching for the point in which its derivative with respect to the parameters (called gradient) is zero

$$\frac{\partial \mathcal{L}(\Theta^*)}{\partial \Theta} = 0 \; . \tag{2.8}$$

Iterative methods make local estimates of the gradient and move incrementally downward the loss surface. One of the most used iterative algorithms is the *Stochastic Gradient Descent* (SGD). For each iteration i, the SGD calculates the instantaneous error out of a single training element to estimate the loss function and to update the parameters accordingly

$$\Theta_{i+1} \longleftarrow \Theta_i - \eta \frac{\partial \mathcal{L}(\Theta i)}{\partial \Theta} \; , \tag{2.9}$$

where η is a constant called *learning rate*. The learning rate sets how much the parameters at the next iteration have to change according to the actual estimate of the loss function. To have a better estimate of the cost function is possible to calculate the gradient using a subset of the training data instead of a single value. The algorithm that uses this approach is called *Mini-batch Gradient Descent*: the parameter update is the same of the SGD (Eq. 2.9) but the gradient estimate is computed over a batch of training data instead of a single element.

The parameters' update depends on the output error but, considering the case of a deep neural network, this is not directly applicable since the input and the desired output of the neurons in the hidden layers are not known. A procedure called *Back Propagation* is usually applied: the outputs of the previous layers are used as inputs to the neurons of the hidden layer and the desired hidden output is obtained by propagating backwards the output error along the network. Recalling Eq. 2.2 and Eq. 2.3, the output of each neuron is a function of functions since $\mathbf{h}^{(l)} = g^{(l)}(g^{(l-1)}(\mathbf{h}^{(l-2)}; \boldsymbol{\theta}^{(l-1)}); \boldsymbol{\theta}^{(l)})$, then, since the gradient is basically a derivative, Back Propagation is a way of computing gradients of expressions through recursive application of the chain rule (derivative of composite functions). Because the chain rule computes the derivative of a composite function as the product of the derivatives of its component functions, is vitally important that $g^{(l)}$ is differentiable. Otherwise, would be impossible to compute the gradient and, as a consequence, the model will not be able to learn.

A machine learning model is useful if it is able to generalize, i.e. to deal with new data with respect to the one used to train it. Two separate parts of the same dataset are used in the learning process: the training set and the validation set. To verify the model's ability to generalize, the cost function is calculated on both training and validation set. If the loss function was calculated solely on training set, the learning process would find a minimum in the loss function that might not correspond to the optimal one. Instead, using both training and validation set, even if the training loss decreases, the model starts losing the ability to generalize when the validation loss starts increasing. When both training and validation costs are diminishing, the model is actually learning useful informations and this said that the model is in *underfitting*, still has to learn something. In case validation and training loss are going in opposite directions, is said that the model is in *overfitting* and is learning the training data by heart and losing its ability to generalize on new data. See Figure 2.9 for a graphical representation.

2.4 Bayesian probabilistic framework

The network's output is not descriptive enough to properly model the beat sequence because it is necessary to include some musical knowledge in its computation. This is done by introducing probabilistic methodologies such as Dynamic Bayesian Networks, which combine prior information (in our case, musical knowledge) with the ability to deal with time series of data. In this section we start introducing Bayesian Network's general features, followed by a more specific explanation of its dynamic variant. To conclude, we introduce the Viterbi Algorithm, which extracts the most probable sequence of beat based on the probability distribution specified in a DBN. See Chapter 4 for the implementation details.

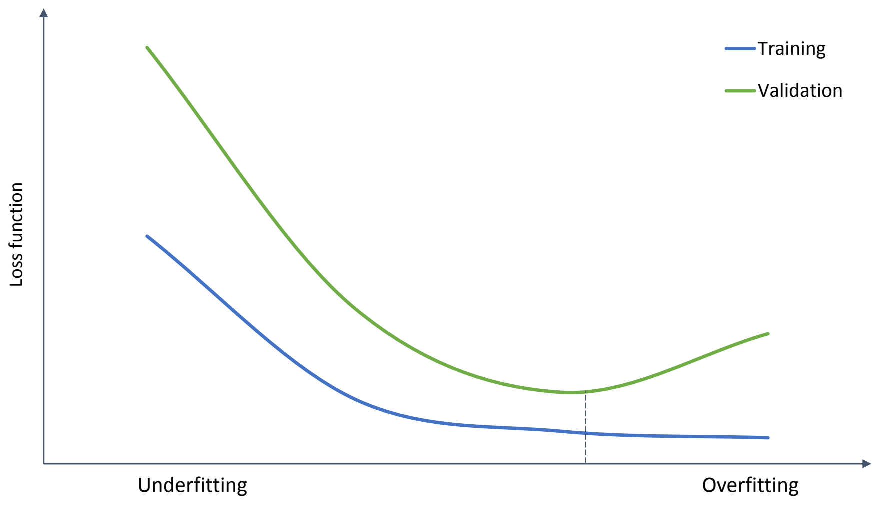
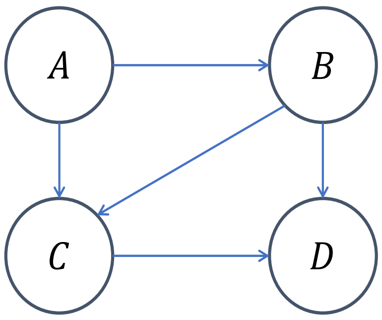


Figure 2.9: Difference between training and validation loss. The dashed line indicates the optimal stopping point.



$$\begin{tikzcd}
A \arrow[r] \arrow[d] & B \arrow[d] \arrow[dl] \\
C \arrow[r] & D
\end{tikzcd}$$

Figure 2.10: A Bayesian Network with four nodes and five directed arches.

2.4.1 Bayesian Network

A Bayesian Network (BN) is a graphical model for representing conditional dependencies between a set of discrete random variables $(R_1, \ldots, R_N)$. The graphical model used is the *Directed Acyclic Graph* and is defined as a set of node and arches $G = (Nodes, Arches)$, where each arch is directed (i.e. implies a non-symmetric relationship) and is impossible to follow a path from $Node_a$ that arrives back at $Node_a$. A directed connection between two nodes represent a parental relation between them: if there is a directed arch from $Node_a$ to $Node_b$, $Node_a$ is said to be a parent of $Node_b$. A random variable R is a variable whose possible values are outcomes of a random phenomenon. In case the random variable is discrete, the number of possible outcomes are limited to the set $\{v_1, \ldots, v_n\}$. Each random variable is represented by a node in the graph and is associated with a *Conditional Probability Distribution* (CPD) that specifies the distribution over the values of the random variable associated with its parents. The absence of arcs implies conditional independence between the two considered variables. The state of a Bayesian Network represent the simultaneous combination of values of its random variables, i.e. $S = (R_1 = v_1, \ldots, R_N = v_n)$.

Given the above definitions, a Bayesian Network fully specify the factorization of the joint probability of all its random variables: in fact, is known from basic probability theory that is possible to define the joint probability of some random variables as the multiplication of their conditional probability. In a Bayesian Network, the joint probability is

$$P(Node_1, \ldots, Node_N) = \prod_{i=1}^{N} P(Node_i | Parents(Node_i)) \ . \quad (2.10)$$

As an example, given the DAG in Figure 2.10, its joint probability is $P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|B, C)$. Thus, given a Bayesian Network is possible to calculate the probability of being in a determined state as the joint probability that all the random variables have a particular combination of outcomes. The probability of being in a state S is

$$P(S) = P(Node_1 = v_1, \ldots, Node_N = v_n) \ . \quad (2.11)$$

Is possible to distinguish two different type of random variables: observed and hidden. The hidden ones are not directly visible but are inferred based on the observed ones. Some evidence (i.e. an outcome of an observed variable) propagates in the network updating marginal probabilities of all the variables in the network to incorporate this evidence.

2.4.2 Dynamic Bayesian Network

A Dynamic Bayesian Network (DBN) is a static Bayesian Network that is modeled over an arrangement of time series. In a DBN, each time slice is conditionally dependent from the previous one. Thus,

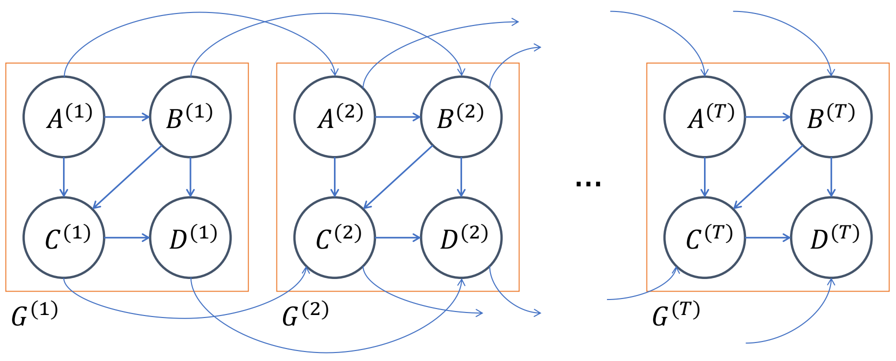


Figure 2.11: A representation of a DBN.

we can picture a DBN as a static Bayesian Network G duplicated for a time series $t = (1, \ldots, T)$ in which the state probability of $G^{(t)}$ conditionally depends on the state of $G^{(t-1)}$.

There are two sets of aches: intra-slice and inter-slice. Intra-slice aches represent the influence between the random variables in the original Bayesian Network G. The second set of aches represent the conditional influence between random variables in two adjacent time slices. Basically, the value of a random variable $R_i^{(t)}$ can be calculated from the internal regressors $R_{Parent(R_i)}^{(t)}$ and its immediate prior value $R_i^{(t-1)}$.

A graphical representation of the DBN is depicted in Figure 2.11.

2.4.3 Viterbi algorithm

The Viterbi algorithm is a formal technique that finds the single best state sequence $S_{1:T}^* = \{S_1^*, \ldots, S_T^*\}$ that maximize the probability of the state sequence given the observations $P(S_{1:T}|O)$, where $O = \{o_1, \ldots, o_T\}$ is the sequence of observations.

Starting at the beginning of the sequence, it computes score (probability) for each state given the first observation and the initial state probabilities. The algorithm keeps track of the highest score and the previous best state which, at the beginning, is none. For the successive time slices, the best score and the best previous state are recursively computed from the previous time slice's scores and also transition's and observation's probabilities. Note that the previous best state is computed taking into account just the transition probabilities but not the observation ones. At the end of the time sequence, $S^*_{1:T}$ is computed by following the path traced by the best previous state sequence backwards, starting from the state that has the highest probability at the final time slice.

Chapter 3

State of the Art

In this chapter we provide an overview of the methods for beat tracking proposed in the literature and we describe two relevant applications of transfer learning.

Section 3.1 gives a brief description of a general beat tracking scheme followed by a review of state-of-the-art methods, divided by category. Section 3.2 gives a picture of some methods that are used to automatically analyze rhythm patterns in folk music. Finally, section 3.3 focuses on transfer learning and its applications related to this study.

3.1 Beat tracking

The beat is perceived through the series of events that are part of a musical piece. Those events are temporally arranged in a particular way that let humans be able to predict where the next beat will be. Generally, a beat tracking system consists of three main components [28] each one devoted to mimic a particular task that humans naturally do (i.e. detect the events, analyze the inner periodicities and predict the beats' positions). A graphical representation of the general beat tracking scheme is depicted in Figure 3.1.

The first stage receives an audio signal as input and aims to describe the position in time of the musical events through the use of some audio descriptors. We are going to call this the *feature extraction* phase. Based on the type of event we aim to detect, different tools are employed to achieve this goal [1]. As an example, the signal amplitude envelope can be used to detect a percussive events

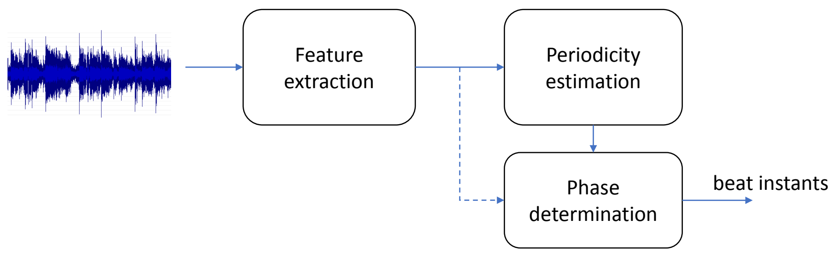


Figure 3.1: General beat tracking scheme.

since they are characterized by an abrupt change in the signal's energy. However, the same tool becomes useless in case of a chord change since the energy might not vary significantly but frequency components do.

The beginning of a musical event is called *onset* and the function that describes the proximity of an onset is called Onset Detection Function (ODF). The ODF is a single-valued function, with a sample rate usually lower than the musical input signal, that is characterized by high values close to onset's locations. Different ODFs are proposed in literature, some meant for specific kind of event (such as *Beat Emphasis Function*[16]) and some more general-purpose (such as *Spectral Flux Log Filtered*[2]). In a beat tracking system, the feature extraction phase is often employed to extract the ODF from the audio signal. In few cases, depending on how the computation is done in the next stages of the beat tracking system, it might be required a detailed frame analysis so other kind of multi-dimensional features are computed (such as first-order difference between frames) instead of the ODF.

Once the events have been detected, humans naturally try to find a pattern in that series. This is the goal of the second phase of the beat tracking scheme, which analyzes and *estimate the feature's periodicities*. The output of this second stage is an initial estimate of the beat period, which is the distance between two beats that is, generally, time-varying. This is usually accomplished using methods that are able to model patterns in a series of data such as autocorrelation [22], comb filter-bank [39], Hidden Markov Models [17], Recurrent Neural Network [5].

The last stage exploits the information coming from the periodicity estimation stage (often combined with feature extraction stage's information) to compute the most probable sequence of beat locations $\mathcal{B}$. In literature, the beat location is defined as "phase" [5] thus, final stage is called *phase detection*. Its task is to predict the beat locations with two constraints: the position has to be close to a musical event's position whilst maintaining a distance between two successive beats as coherent as possible to the information about the periodicity. Popular methods to achieve the goal of the last stage use tools that are able to to solve problems of joint optimization or use a probabilistic approaches combined with musical knowledge. In literature Dynamic programming [22], multiple agents [27], Kalman filtering [7], Dynamic Bayesian Network [41] approaches are used and are briefly defined in the next subsections.

In some methods (such as [40]) the second and the third stages are incorporated if beat periodicities and phases are jointly estimated.

The following subsections review state-of-the-art method for beat tracking. The subsections are divided into different category based on how the phases and periodicities are extracted.

3.1.1 Dynamic Programming approaches

One of the possible approaches to address the beat tracking task involves Dynamic Programming (DP). DP is a method of solving complex tasks by subdividing them into collections of simpler problems of the same type. As an example, finding the optimal beat sequence out of a musical piece can be seen as finding the optimal beat sequence of a portion of the same song.

In [22] Ellis uses DP to extract the beat location by searching the beat sequence that jointly matches the high ODF values and the tempo estimate (an estimate of the distance between beats). A greedy approach that evaluates all the possible sequences is computationally prohibitive, and DP comes in handy by considering a local "runtime" score that evaluates the sequence up to a certain time. Once the whole sequence has been considered, the beat instants are computed as the local maxima of the runtime score.

Some adaptation of the previous method is presented by Stark [53], which makes the algorithm able to work on-line by predicting the next beat location without the need to wait until the whole audio signal has been processed. Wu et al. extended the approach to time-varying situations [55].

Di Giorgi, in [25] further elaborates the concept previously introduced by searching only for potential candidates and by applying a rule that forces the exploration of different sequence hypothesis.

3.1.2 Multiple agents approaches

An agent is a component that is programmed to perform a specific task. Considering the beat tracking task, multiple-agents architectures are used in order to examine more beat sequence hypothesis in parallel. Moreover, depending on the type of architecture, if one agent lose track of the beat, the whole system is capable of detecting the correct sequence as long as other agents maintain the right hypothesis.

In [27] Goto et al. propose a model in which from several frequency bands, the onsets sequences are extracted and assigned to two paired agents which task is to estimate the beats given the onset sequence. Each agent is paired with another one and they cooperate by forcing each other to explore a different strategy. Moreover, each agent self-evaluates itself using musical knowledge, according to the input signal. If its hypothesis becomes highly reliable, the agent adapts its parameters to maintain the current hypothesis. At the end, an agent manager selects the beat sequence of the most promising agent to be the system's output.

A different approach is proposed by Dixon [18] in which there is no agent manager. The onsets are analyzed and the time interval in between is clustered considering different tempo hypothesis corresponding to various metrical level. These hypothesis and the onsets are passed to a group of agents that analyzes different possibilities regarding the phase of beats. Each agent evaluates itself considering how its beat estimates are evenly spaced and how many predicted beats correspond to onset's location. At the end, the best agent is chosen.

3.1.3 Kalman filtering approaches

The filtering problem consists in estimate a variable based on its related observations (such as a direct measurement affected by noise). The Kalman filter addresses this problem by defining a state space model (comparable to the one described by a Bayesian Network) and applying Bayesian prediction to estimate the joint probability distribution over the random variables. The more observations are available, the more reliable will be the estimate. The Kalman filter is known and widely used in robotic motion planning control, time series analysis, vehicles navigation [58, 31].

In order to use the original formulation of the Kalman filter, the problem has to be stated as a linear dynamical system [38]. Cemgil et al. in [7] define the tempo tracking process as a linear dynamical system and a Kalman filtering process is applied to retrieve an estimate of the hidden state (which consists of beat locations and period). At every time frame, Kalman predicts where the next beat should be and adjust its estimation based on the new observations. At the end of the process, all the beat estimates are updated based on the whole sequence of evidence (Kalman smoothing).

Shiu et al. in [52] improved this idea by applying probabilistic data association to make Kalman filtering approach able to deal with tempo fluctuations and expressive timing deviations.

3.1.4 Ensemble approaches

In the ensamble approaches more than one beat tracking algorithm is used to determine the final system's output.

A measure of Mutual Agreement (MA) is used in to determine the correlation between two sequences of beats estimates. Holzapfel et al. in [34] compares the output of five different beat tracking algorithms by measuring the MA of their sequence estimates. Each algorithm's output is compared with all the others' by means of MA. The one that agrees most with the others (with highest mean MA) is selected as the most representative among the committee and its output is selected as the system's output.

Since the execution of five different algorithms may be impractical due to computational issues, Zapata et al. in [57] used nine different ODF as input to the same beat tracking algorithm [17] and measured the MA to choose the best candidate.

3.1.5 Inference approaches

In the inference approaches, beats positions are deducted as the sequence that best explains the observed random variables of the system. Most of the times, beats are computed by applying the Viterbi algorithm on Bayesian probabilistic frameworks able to model the probability of series of data (such as Dynamic Bayesian Network [3] or Hidden Markov Models [39]).

Whiteley et al. in [54] propose a model called *bar pointer model* that use the movement of an imaginary pointer inside a bar to extract the beat positions. The pointer's motion is described as a function of two variables: its position and its velocity. Those are encoded into hidden variables of a Bayesian probabilistic framework and the pointer motion is described as the most probable sequence of position and velocity. In [3] Böck et al. uses the same approach to describe the pointer movement inside a beat period instead of a bar. In the original bar pointer model, the possible tempi and possible discrete positions are equally spaced. To better match the human perception, in [41] Krebs et al. propose a variation of the bar pointer model in which the possible tempi are logarithmically spaced and number of discrete positions depends on the actual tempo.

In [39] Klapuri et al. jointly estimates the beat's period and measure's length using a Hidden Markov Model (HMM). The HMM combines the estimation of period and phase coming from a comb filter resonator stage to infer the final output.

In [17], Degara et al. encode the elapsed time since the previous beat event into a hidden state of an HMM. Then, given the ODF and the beat period estimation, through the use of a decoding algorithm, the model finds the most likely sequence of hidden states, thus the sequence of beats. This approach also provides a reliability model that allows to determine the correctness of the beat estimation by evaluating the period estimation stage's output.

Peeters in [48] proposes two strategies for the beat tracking. The first involves a modified version of the Pitch Synchronous Over-Lap Add (PSOLA) method: local maxima in ODF with a distance close to the estimated period are found, then a least-squares optimization is used to find the beat as close as possible to the local maxima and with a distance close to the period estimation. The second procedure involves an inverse Viterbi-based beat tracking method that exploits Linear Discriminant Analysis [24] to define the observation probabilities.

3.1.6 Deep learning approaches

Recently deep learning is also used to improve the performance of a beat tracking system. Since music is a human perception, deep learning is widely used in the MIR community because of its ability to emulate the human thinking. In fact, it has proven to improve performances on task such as music structural analysis [6], music classification [9]. In these cases, differently from the methods explained above, no ODF is computed but a multi-dimensional feature is extracted from the musical excerpt. Since the deep learning models try to mimic the human brain, there is no need to extract hand-crafted features (such as the ODF) from the audio signal, since they might loose relevant informations. In these cases, the multi-dimensional feature can be interpreted as an emulation of the human earing system which, combined with the emulation of the human brain, leads to better results.

In [5], Böck et al. use a Recurrent Neural Network (RNN) to extract some information about the periodicities of the input signal. Spectral features are extracted from the audio excerpt and are used to feed the network which outputs a probability of having a beat. In order to extract the beat positions, a method based on autocorrelation and peak picking is used. In fact, as said in Section 2.1, the beats are perceived through a periodical disposition of musical events (analyzed by the autocorrelation stage), and the most salient periodicity is chosen (via the peak picking stage).

To deal with heterogeneous music style, more networks can be trained, each one on a particular style. In [3] the output of all the networks is analyzed and compared with the output of a reference one to choose which performs better. The output of the chosen one is selected as the input of a DBN that extracts the beat position. This can be related with the ensemble approaches discussed in Section 3.1.4.

In [20] Durand et al. address the problem of downbeat tracking (i.e. detect the first beat of a bar) using a Deep Neural Network (DNN). Six features are extracted and analyzed by six independent DNN which output the probability of having a downbeat for a given audio frame. The mono-dimensional probability is then analyzed by DBN which extracts the downbeat positions. In [42] Krebs et al. use a similar approach to detect the downbeat jointly with the other beats. Two kind of features are extracted, one to detect percussive events and the other to detect harmonic events. Their periodicities are analyzed using two separate RNN which outputs are examined by a DBN similar to the one proposed in [20] that jointly extracts beats' and downbeats' positions.

Deep learning approaches are used also for the onset detection task. Eyben et al. in [23] use the same method of [5] to find the sequence of onsets in a musical piece. In [50] another kind of network called Convolutional Neural Network (CNN) is trained to effectively predict the probability of having an onset in an audio frame. Being a probability, the output of the network spans from 0 to 1, and the onset detection is performed applying a threshold to the probability value. For every possible onset location, if the related probability value is greater with respect to the threshold, the onset is detected.

3.2 Rhythm analysis of Folk music

Most of the previous algorithms have been designed for Western popular music. Lately, in Music Information Retrieval (MIR), the focus has expanded also to folk music. Folk music, also known as traditional music or world music, is a musical genre typical of a particular geographic area. Differently from commercial popular music, usually is transmitted orally with unknown composers. For a non-native listener, folk music might contain unusual tempi, melodic and harmonic progression. So, the automatic rhythm analysis becomes an harder task with respect to popular music which is characterized by regular measures and steadier tempo. In this section we are going to present a couple of studies that highlight this complexity and an algorithm that solves the beat tracking issue on this particular music context.

Cornelis in [13] compares the results of automatic beat tracking algorithms with respect to human annotations in the context of Central-African music. The study shows that, in that particular environment, the set of algorithms behaves similarly with respect to the human tappers, but there is an ambiguity in the perception of tempo and meter in both cases. In fact, the tempo is often perceived doubled or halved (also known as octave error) and the correct meter is not always grasped.

Holzapfel et al. in [33] use symbolic data to study the distribution of onsets in Turkish Makam music with respect to their metrical structure. They compare the onset distribution with Western music and show that the correlation between onset distribution and meter is lower in case of Makam with respect to Western music.

In [35] Holzapfel et al. use a Gaussian Mixture Model (GMM) and a DBN to track the downbeats on non-Western datasets: one composed of Turkish songs, the other of Carnatic music and the last one of Cretan music. The DBN is derived from [40] but is adapted to be able to track the downbeat in those genres.

3.3 Transfer learning

Transfer learning or *knowledge transfer* comes in handy in cases when the collection of new training data requires too much effort. Thus, portion of the previously acquired knowledge can be transferred to a new model to ease the training of the latter.

In [10] Choi et al. use the feature vector extracted from a CNN pre-trained on music tagging task as input to a classifier to perform six new tasks ranging from genre classification to emotion prediction through vocal/non-vocal classification. The feature vector is extracted from different combinations of layers of the CNN. The idea is that, depending on the target task, the features extracted from some layers might be useful and some others not. As an example, for the target task of genre classification, the discriminant between different genres in the target dataset is loosely related to rhythmic patterns and tempo; in fact, the first layers are shown to be more useful, since features from higher levels of the CNN are mostly tempo invariant.

In the context of Natural Language Processing, Yang et al. in [56] apply transfer learning for sequence tagging task using hierarchical RNN. A specific style of transfer learning is applied called *parameter transfer*, so the knowledge is transferred between two models by the sharing of some parameters. In this particular type of transfer learning, the model weights consists of two parts: one shared and one task-specific. The shared component is optimized by the training on source and target tasks while the task specific one is optimized just one the training on the task which is referring to. Yang et al. show that the more parameters are shared, the better the improvement will be.

Chapter 4

Method

In this chapter we provide an in-depth description of our method for automatic beat tracking. Our method is based on the one proposed in [4]; Section 4.1 explains the details of our implementation. Section 4.2 dives into the details of transfer learning used here.

4.1 Beat tracking system

The scheme of our beat tracking system is depicted in Figure 4.1. Given an audio signal, we extract a perceptually-meaningful time-frequency representation (Section 4.1.1). This representation is the input of the deep network, precisely a RNN, whose implementation is described in Section 4.1.2. The network outputs a likelihood that a beat occurs at a certain instant of the audio signal. Since in music the sequence of beats follows some rules given by the music theory, we add a DBN at the end of the deep network in order to transform likelihoods to a sequence of beat instants. The DBN is explained in Section 4.1.3.

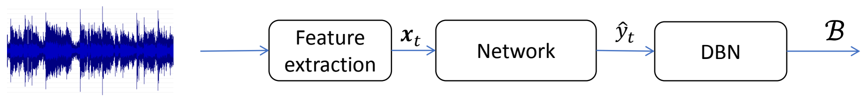


Figure 4.1: Our general scheme for beat-tracking

4.1.1 Pre-processing

Given a generic audio signal s, we extract a sequence of feature vectors $\mathbf{x}_t$ with $t = 1, ..., T$, to feed the RNN. We use a similar approach to the one proposed in [42] for the pre-processing stage. A graphical description of the process is given in Figure 4.2.

To reduce complexity, stereo signal are converted to monoaural by averaging the channels. We then compute the STFT of the signal with three different window sizes L_{w_1} L_{w_2} and L_{w_3}, to exploit the trade-off between frequency and time resolution. For each STFT, the transform size N_{FT} matches the window length. To combine the information into a single feature vector, the frame rate has to be equal for all the STFT, thus, a common hopsize H is used. After the STFT the signal passes through the same processing chain, independently from the window size. We discard the phase component of the STFT by taking the magnitude of the spectrum. Then, we filter the spectrograms using a bank of triangular filters with a constant frequency resolution per octave, to reduce the number of components of the STFT frame and extract a frequency representation closer to human perception. Additionally, to better match human perception, we compute the logarithm of the spectrogram. To enhance the evolution of the signal, the half-rectified difference of the spectrogram is computed as

$$D(t,k) = h(\xi(t,k) - \xi(t-1,k)) \ , \tag{4.1}$$

where h is the half-rectified function defined as $h(x) = \frac{x+|x|}{2}$ and $\xi(t,k)$ indicates the frame coefficient k at time t.

We stack together the three versions of the spectrogram and their first-order half-rectified difference to compose the final sequence $\mathbf{x}_t \in \mathbb{R}^{N_{\mathbf{x}}}$. We then use it as the input of the network.

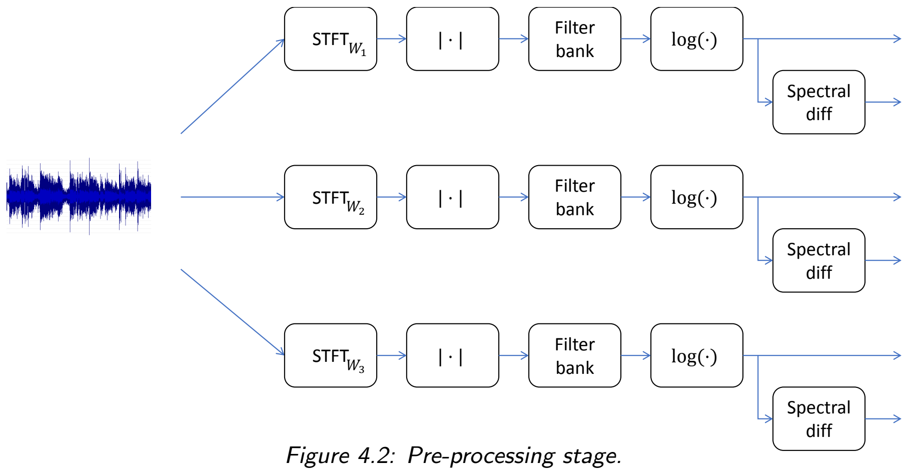


Figure 4.2: Pre-processing stage.

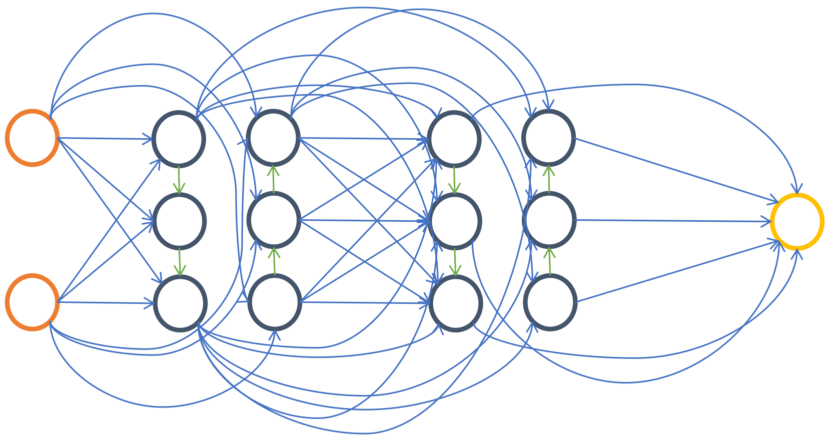


Figure 4.3: Bidirectional RNN with two hidden layers. Please note that the series of connection is non-exhaustive.

4.1.2 Network

The network input **x** represents how the musical piece evolves both in terms of rhythmic events and in terms of harmonic changes. The network input can be interpreted as an emulation of the human perception and the network itself simulates the first stages of the human thought by analyzing the periodicities of the input. To do so, we use a three-layer BLSTM-based RNN network followed by a fully-connected (FC) layer to compute a *beat activation* $\hat{y}_t$ as the probability that a beat occurs at time t. A graphical representation of the network is presented in Figure 4.3.

The BLSTM part of the network is a standard Bidirectional-RNN with LSTM units. Each LSTM unit of the last layer of the BLSTM part is connected to the single output neuron of the network. The latter outputs a scalar ranging from 0 to 1, since the activation function of the last neuron is a sigmoid as defined in 2.3.1. Figure 4.5 show the output for each layer, and is possible to see the hierarchical representation learned by the network. In fact, by comparing Figure 4.4 and Figure 4.5, we can see that the first layer present high activations at onset locations. On the other hand, the last two layers analyzes their input at higher metrical level, emphasizing the activation function values near the beat locations.

The network is trained using labeled annotation $y_t = \{0, 1\}$ that indicates whether ($y_t = 1$) or not ($y_t = 0$) a beat occurs at time

t for a given excerpt. Basically, for each frame presented to the network $\mathbf{x}_t$, the desired output y_t is given to the training procedure to calculate the output error. As previously said, the sequence $\hat{y}_t \in [0, 1]$ represent the probability of having a beat and has continuous co-domain. The output $\hat{y}_t$ is passed to the DBN that combines it with musical knowledge in order to extract the beat sequence $\mathcal{B} = \{b_1, \ldots, b_{N_\mathcal{B}}\}$.

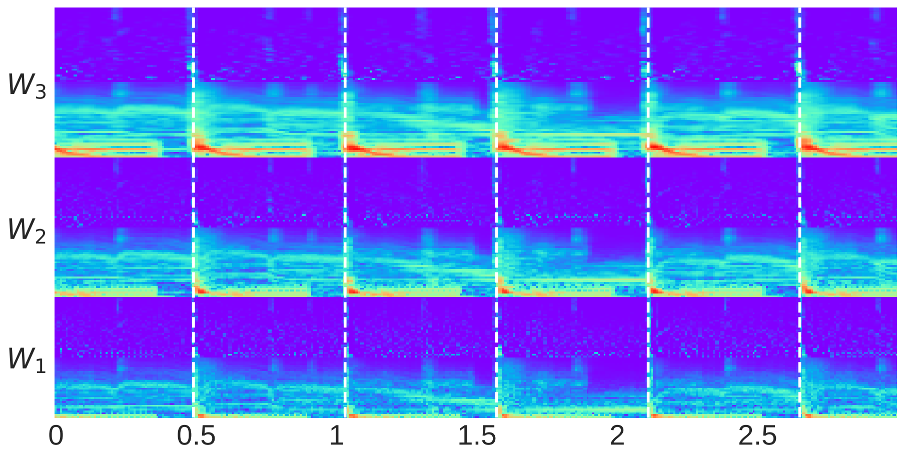
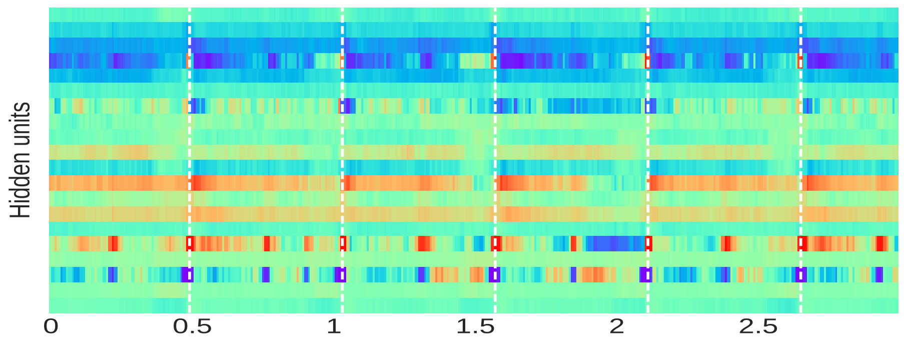
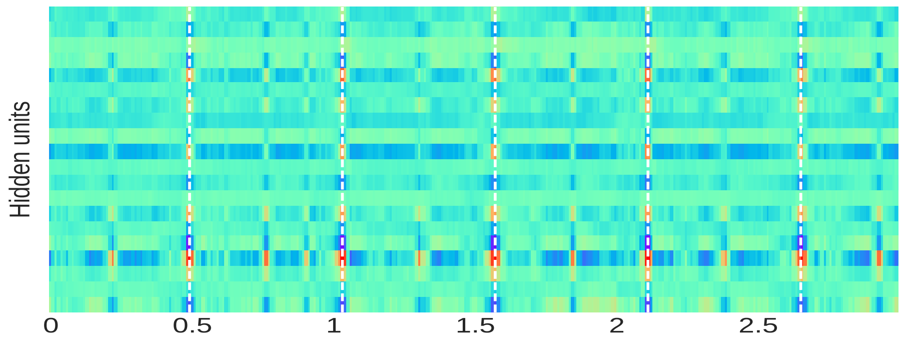


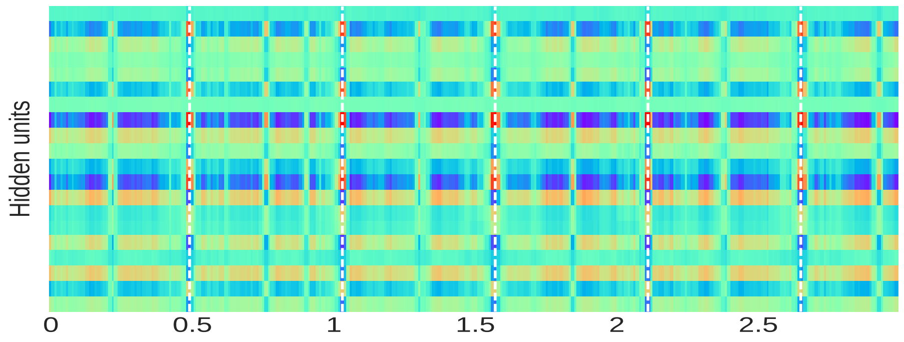
Figure 4.4: Network input.



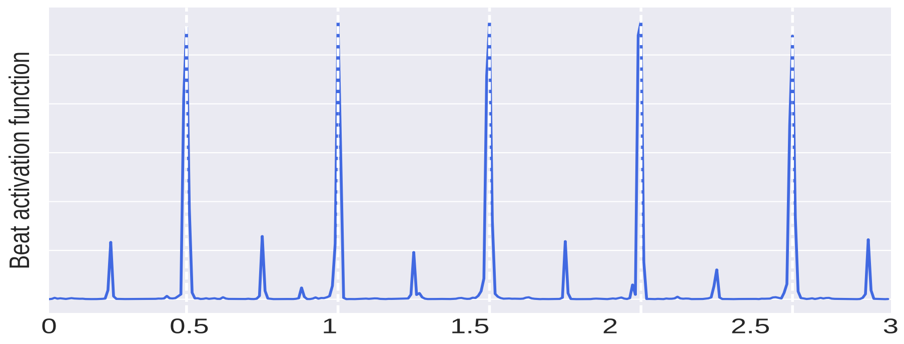
(a) Output of the first layer



(b) Output of the second layer



(c) Output of the third layer



(d) Output of the last layer

Figure 4.5: Output for all the layers of the network for three seconds of song. The dashed vertical white line denote a beat position

4.1.3 Post-processing

We use the DBN proposed in [3] (with perceptive and efficiency improvements described in [41]) to track and model the beat sequences. This model is a variation of the bar pointer model explained in Section 3.1.5. Basically, the DBN proposed in [3] describes the movement of an hypothetical pointer inside a beat period, which is the time between two beat instants. The pointer is only allowed to move forward in time and whenever moves to the next period, it means that a beat occurs.

In the DBN, states S_t are identified by two variables: a measure of the tempo ρ_t and the position of the pointer within a beat period ϕ_t. States can be imagined as in a grid spanned by these two components. At each time instant t, the *transition model* estimates the probability of a transition between states $P(S_t|S_{t-1})$. Given a sequence of states $S_{1:T}$, the probability sequence of transitions among states is:

$$P(S_{1:T}|\hat{y}_{1:T}) \propto P(S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(\hat{y}_t|S_t), \tag{4.2}$$

where: $P(S_1)$ is the *initial state distribution*, usually initialized as a uniform (equiprobable) distribution, and $P(\hat{y}_t|S_t)$ is the *observation model*, which employs the beat activation as the observable variable to guide the transitions. The hidden variables are discretized as

$$\phi_t \in \{1, \ldots, M\} \tag{4.3}$$

$$\rho_t \in \{\rho_{min}, \ldots, \rho_{max}\} \tag{4.4}$$

where M is the number of discrete positions per beat period and ρ_{min}, ρ_{max} are the slowest and the highest tempo expressed in discrete beat period positions, respectively.

In order to be able to calculate the probability of a sequence of states (see Eq. 4.2) we have to define the the transition and observation models.

The transition model is defined as

$$P(S_t|S_{t-1}) = P(\phi_t|\phi_{t-1}, \rho_{t-1})P(\rho_t|\rho_{t-1}) \tag{4.5}$$

where $P(\phi_t|\phi_{t-1}, \rho_{t-1})$ denotes the position dependency from the previous position and the previous tempo, and $P(\rho_t|\rho_{t-1})$ refers to the dependency of the actual tempo from the previous one. The first

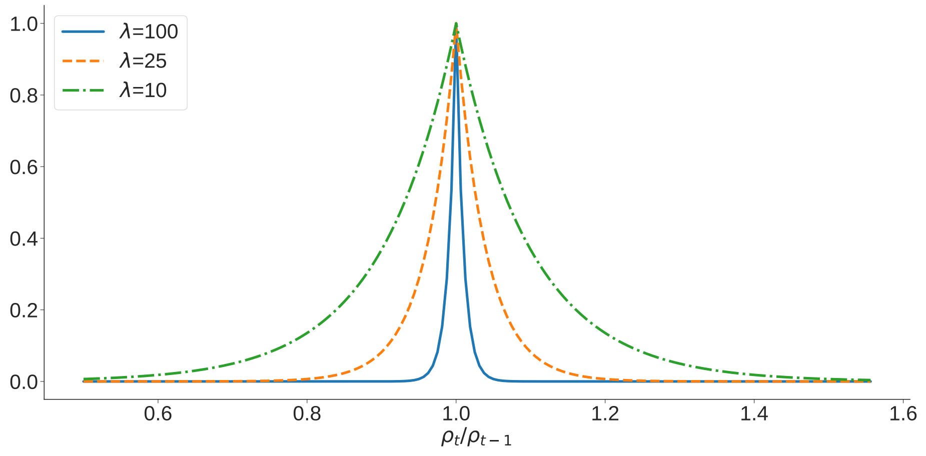


Figure 4.6: Tempo change probability as a function of λ

term is $P(\phi_t|\phi_{t-1}, \rho_{t-1}) = \mathbb{1}_\phi$ where $\mathbb{1}_\phi$ is the indicator function, that has value 1 if $\phi_t = (\phi_{t-1} + \rho_{t-1} - 1) \in \mathbb{Z}_{M+1}$ and 0 otherwise. The second term indicates the transition tempo model and is expressed as an exponential distribution $P(\rho_t|\rho_{t-1}) = \exp(-\lambda|\frac{\rho_t}{\rho_{t-1}} - 1|)$, where $\lambda \in \mathbb{Z}_{\geq 0}$ indicates the steepness of the distribution. Basically, λ controls the probability of tempo transition, the lower λ the higher the probability of tempo transition (see Figure 4.6).

Since the network outputs $\hat{y} \in [0, 1]$ which represents the probability of having a beat, this can be used directly as state-conditional observation distribution. The observation model is defined as a function of the position inside a beat period as

$$P(\hat{y}_t|S_t) = \begin{cases} \hat{y} & 1 \leq \phi_t \leq \frac{M}{c} \\ \frac{1-\hat{y}}{c-1} & \text{otherwise} \end{cases} \tag{4.6}$$

where $c \in [\frac{M}{M-1}, M]$ is a constant arbitrarily chosen that determines the portion of beat interval which is considered as beat and non-beat locations.

M defines the discretization of the beat period (i.e. the time resolution). In the original bar pointer model the discretization is independent from the tempo, which means that the time resolution of a piece played at a low tempo will have a lower time resolution with respect to the one played at higher tempi. To overcome this issue, we use a tempo-dependent discretization, i.e.

$$M(bpm) = \text{round}(\frac{6 \cdot 60 \cdot \Delta}{bpm}) \ . \tag{4.7}$$

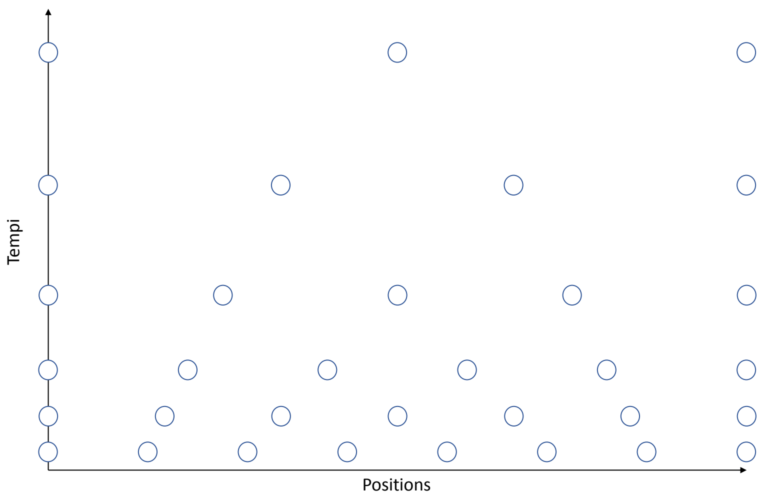


Figure 4.7: Representation of the state space

In the above equation, Δ is the frame rate measured in Hz and, since bpm is the tempo measured in beats per minute, the numerator is multiplied by 60 to match the unit measure of the frame rate. The higher M, the higher the precision in describing the pointer's movements and, as a consequence, the time resolution. Thus, to increase the discretization, the numerator is multiplied again by a constant $= 6$.

The tempo sensitivity of human brain depends on the speed of what they are listening to. As an example, a variation of 10 BPM on a song at 190 BPM is quite imperceptible, whereas the same variation on a song at 60 BPM is easy to notice. The ability to notice tempo difference is proportional to the actual tempo and is equal to the Just Noticeable Difference (JND) that corresponds to $2-5\%$ of the inter beat interval [19]. To overcome this problem, given the bpm_{min} and bpm_{max}, the tempo discretization is logarithmically distributed in N_{tempi} across the tempi range $[bpm_{min}, bpm_{max}]$ to better match the JND of the human auditory system. Finally, using Eq. 4.7 we can find the number of discrete positions for every tempo.

The tempo and position discretization describe the state space of the Bayesian Network, i.e. all the possible values of its random variables. A graphical representation of the state space is depicted in Figure 4.7.

Using the Viterbi algorithm, the most likely sequence of transitions among states $S^*_{1:T}$ is computed as:

$$S^*_{1:T} = \{S^*_1, \dots, S^*_T\} = \arg\max_{S_{1:T}} P(S_{1:T}|\hat{y}_{1:T}). \tag{4.8}$$

The hidden variables in the state completely define the position of the hypothetical pointer inside a beat period through the whole audio signal. Each state defines the values of the hidden variables, so, by looking at the transition of states occurring through $S^*_{1:T}$, we obtain the sequence of pointer's positions. The sequence of beat position is finally determined as $\mathcal{B} = \{t : \phi_t < \phi_{t-1}\}$, i.e. whenever the pointer moves to the next beat period.

4.2 Transfer learning

Transfer learning aims to exploit the knowledge acquired on a source task to perform a target task. This is consistent with what happens in real life, where pre-existing knowledge is used to process and understand new informations. Transfer learning is defined starting from two key concepts: domain and task.

A domain $\mathcal{D} = \{\mathcal{Z}, P(Z)\}$ is constituted by a feature space $\mathcal{Z}$ and a probability distribution $P(Z)$ in which $Z = \{z_1, \dots, z_n\} \in \mathcal{Z}$. Given the domain, a task is defined as $\tau = \{\mathcal{Q}, f(\cdot)\}$ where $\mathcal{Q}$ is a label space and $f(\cdot)$ is a function that maps a domain sample z into a co-domain sample q. The predictive function $f(\cdot)$ has to be learned with a training process using labeled pairs $\{z_i, q_i\}$ in order to minimize the prediction error on a given input z.

In the transfer learning case we use a model previously trained for a source task $\tau_\mathcal{S}$ on a source domain $\mathcal{D}_\mathcal{S}$, to perform a new target task $\tau_\mathcal{T}$ on a target domain $\mathcal{D}_\mathcal{T}$. In other words, the knowledge acquired by the model is exploited to learn the predictive target function $f_\mathcal{T}(\cdot)$ associated with the $\tau_\mathcal{T}$.

As described in [47], there are different types of transfer learning, based on source and target domains and tasks. Our case fall into the category of *inductive transfer learning* in which $\tau_\mathcal{S} \neq \tau_\mathcal{T}$ and the labeled data in the target domain induce a predictive function to be used in the target domain. This can be done in four different ways:

- *Instance Transfer*: re-weights some labeled data in the source domain to be used in the target domain [14].

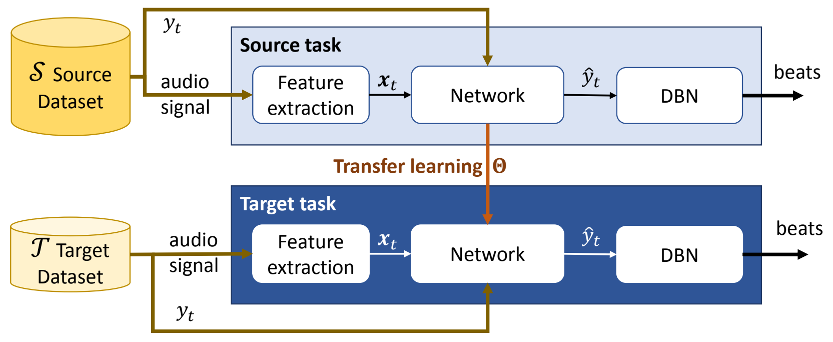


Figure 4.8: General scheme of the algorithm

- *Feature Representation Transfer*: aims to find a different feature description that reduces the difference between source and target domain [37].
- *Parameter Transfer*: assumes the existence of shared parameters between the source and target domain models that helps in determining the target model parameters [56].
- *Relational Knowledge Transfer*: builds a mapping between the source domain and target domain [45].

We have chosen to use a *parameter* transfer learning since $\mathcal{D}_\mathcal{S} \equiv \mathcal{D}_\mathcal{T}$. Parameter transfer assumes that model coefficients are composed of two parts: one shared among tasks and one task-specific. Therefore, a model coefficient is built as

$$w_\mathcal{S} = w_0 + \delta(w_\mathcal{S}) \text{ and } \quad w_\mathcal{T} = w_0 + \delta(w_\mathcal{T}). \tag{4.9}$$

where $w_\mathcal{S}$ is the model coefficient for the source task, $w_\mathcal{T}$ is the one for the target task. So, w_0 is the shared component whereas δ indicates the task-specific part.

Our implementation of parameter transfer is graphically described in Figure 4.8. Basically, there are two different stages of training. In the first one, we train our model for the source task over a large database $\mathcal{S}$, learning the parameters $\Theta_\mathcal{S}$. In the second training phase, we use a smaller database $\mathcal{T}$ for the target task, for training the last layer of the RNN and the output layer; while the lower two layers do not update the weights. This makes sense since, as shown in Figure 4.5, the first layers of RNN learn a basic representation of the music rhythm, and can be effectively used for both tasks, whereas the higher layers are more specialized. In practice, with reference to Eq. 4.9, in the second training phase the network learns the parameters $\Theta_{\mathcal{T}}$ starting from $\Theta_{\mathcal{S}}$. For the first two layers the set of weights $\boldsymbol{\theta}^{(l)}\ \forall l = 0, 1$ does not change during the training phase on $\mathcal{T}$, then $\boldsymbol{\theta}_{\mathcal{T}}^{(l)} \equiv \boldsymbol{\theta}_{\mathcal{S}}^{(l)}\ \forall l = 0, 1$ and can be considered shared by both tasks. Considering the last two layers, $\boldsymbol{\theta}_{\mathcal{T}}^{(l)} = \boldsymbol{\theta}_{0}^{(l)} + \delta(\boldsymbol{\theta}_{\mathcal{T}}^{(l)})\ \forall l = 2, 3$ where $\boldsymbol{\theta}_{0}^{(l)} \equiv \boldsymbol{\theta}_{\mathcal{S}}^{(l)}\ \forall l = 2, 3$. In fact, provided that $\mathcal{S}$ and $\mathcal{T}$ are different, the second training on $\mathcal{T}$ means a change in the loss function and, as a consequence, an update of the last two layers equal to $\delta(\boldsymbol{\theta}_{\mathcal{T}}^{(l)})\ \forall l = 2, 3$.

Chapter 5

Experimental setup

This chapter describes the experimental setup and the results obtained.

Section 5.1 describes the datasets we have used for the evaluation of the methods. Section 5.2 characterizes the system by explaining the training procedure and defining the system's parametrization. Section 5.3 presents the results of the proposed method.

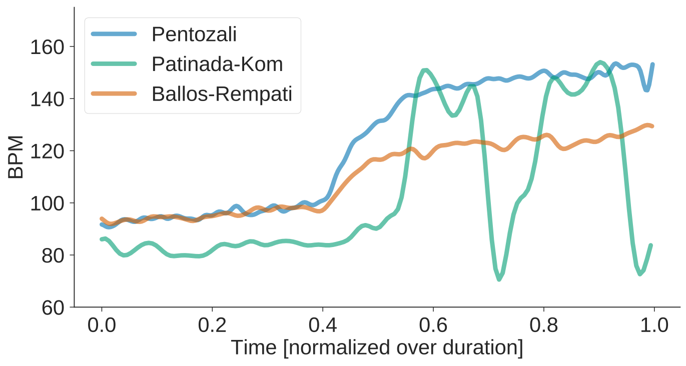
5.1 Data collection

Two kinds of datasets have been used. The first is the source dataset $\mathcal{S}$ which is a big dataset, composed of an high number of songs generally stable in terms of tempo fluctuations. To test the approach of transfer learning the target dataset $\mathcal{T}$ is used. This is characterized by an higher variability of data (with respect to $\mathcal{S}$) but is too small to be used for a single training. To find large datasets for folk music to properly train the network is an hard task. For this in our study the target dataset $\mathcal{T}$ are composed of folk music from specific regions to exploit the benefits of the use of transfer learning.

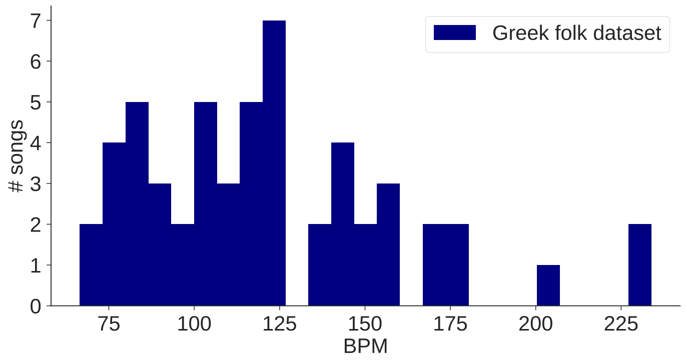
The source dataset $\mathcal{S}$ used in this study is the one described in [5], here named *Böck dataset*. It is composed of 120 songs from the Ballroom set [29], some training and bonus excerpts from the MIREX 2006 beat tracking contest[1] and six files from the set used in [1]. This dataset is used for training the source network. The result of the training is the set of parameters Θ.

With regard to the target dataset $\mathcal{T}$, two dataset have been used.

[1]`http://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking`



(a) BPM evolution of three song in Greek folk dataset

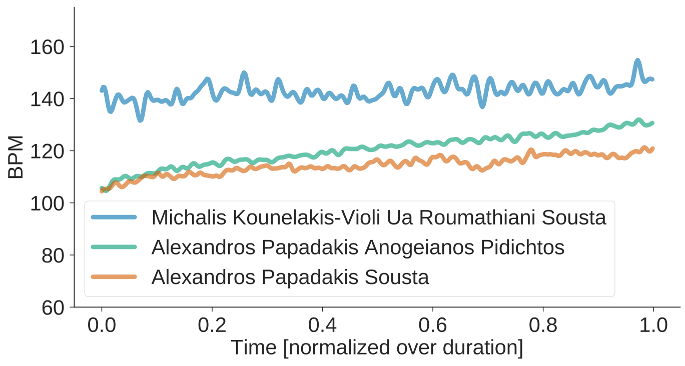


(b) Histogram of BPM in Greek folk dataset

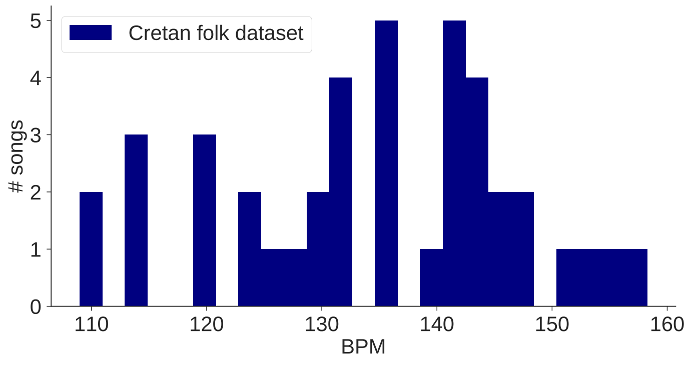
Figure 5.1: Greek folk dataset statistics

The first, named the *Greek folk dataset*, is composed of 56 music pieces typically used in traditional Greek folk dance. We manually annotate the beat for each song in the dataset, and we have asked to traditional Greek music experts to validate our annotations. This dataset contains a wide variety of binary, ternary and odd meters, which may also change throughout the same piece. Moreover, as usually happens in folk music, musicians rarely use the metronome during recordings and tempo fluctuation are extremely common. In Figure 5.1a we show some examples of the BPM evolution over time for three excerpts from the dataset. It demonstrates the variability of the first dataset chosen as $\mathcal{T}$: the tempo fluctuation is clearly visible in all the songs. Figure 5.1b shows the distribution of song-level average BPM in the dataset. The histogram provides useful information about the tempi distribution in the dataset, in particular about the minimum and maximum tempo.

The second dataset used as a target is the *Cretan dataset* presented



(a) BPM evolution of three song in Cretan dataset



(b) Histogram of BPM in Cretan dataset

Figure 5.2: Cretan dataset statistics

by Holzapfel in [35]. It is composed of 42 songs typical of Cretan leaping dance, each belonging to a different style. All these dances are in binary meter, which rarely change during each song. By looking at Figure 5.2a we can see that tempo variations are still common in this dataset but are less prominent with respect to the Greek folk dataset. Figure 5.2b shows that the tempi range of the Cretan dataset is smaller with respect to the Greek folk dataset.

$\mathcal{S}$ lacks of non-steady examples. Thus, a model trained on $\mathcal{S}$ will have a hard time tracking the beats in $\mathcal{T}$ due to the higher degree of tempo fluctuation with respect to $\mathcal{S}$. We highlight the difference of tempo fluctuations between the three datasets by computing the *Median Absolute Deviation* (MAD) over inter-beat intervals (IBI) $\Delta_{\mathcal{B}} = \{b_2 - b_1, b_3 - b_2, \dots, b_{N_{\mathcal{B}}} - b_{N_{\mathcal{B}}-1}\}$, where b_i is the generic beat instant and $N_{\mathcal{B}}$ is the total number of beats. MAD is a robust measure of the variability of a univariate distribution and it is computed

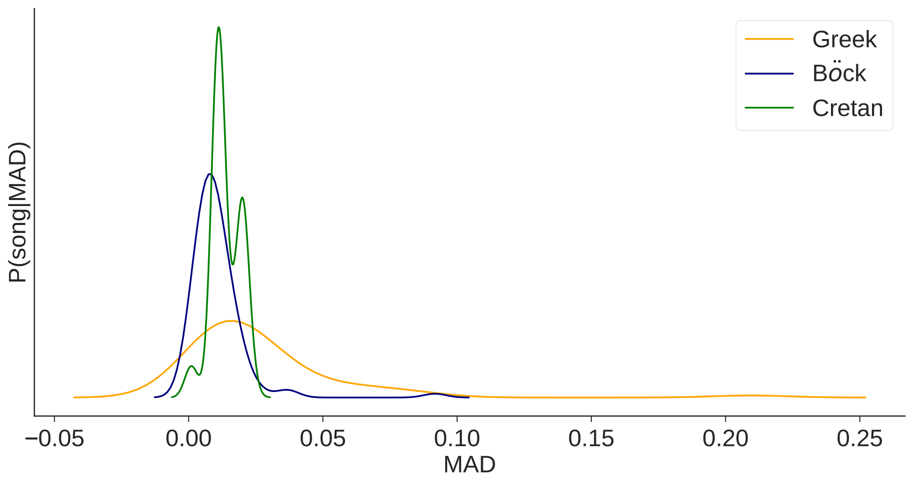


Figure 5.3: Comparison of the distributions of MAD in Böck, Greek folk and Cretan datasets

as:

$$MAD = \text{median}\left(|\Delta_{\mathcal{B}} - \text{median}(\Delta_{\mathcal{B}})|\right) \ . \tag{5.1}$$

The higher the MAD, the higher the variability of the inter-beat intervals and, hence, the degree of tempo fluctuations. In Figure 5.3 we show a comparison between the normalized distributions of the annotations-based MADs in *Böck*, *Cretan* and *Greek folk* datasets. While the former exhibits a high and narrow peak over low values of MAD, the last two are more skewed toward high values. This means that is more probable to find a song with high degree of tempo fluctuation in the Greek folk or Cretan datasets rather than in the Böck dataset. In fact, based on the MAD description given above, the Greek folk and the Cretan datasets present an higher degree of tempo fluctuations with respect to the Böck dataset, so the beat tracking task is inherently harder in the former datasets with respect to the latter.

5.2 Experimental Setup

Given a generic monauralized PCM audio signal, we use a filterbank with six bands per octave to filter the three STFTs with common hopsize of 10 ms and three different windows sized $L_{w_1} = 1024$, $L_{w_2} = 2048$ and $L_{w_3} = 4096$ samples, obtaining 39, 45 and 49 components per frame, respectively. Considering also the first order difference of the spectrograms, we obtain $N_{\mathbf{x}} = 266$ input units.

We follow the RNN setup described in [5]. The model weights are randomly initialized following a Gaussian distribution with mean 0 and standard deviation 0.1, i.e. $\sim \mathcal{N}(0, 0.1)$. As training algorithm, Mini-batch Gradient Descent with Back Propagation are used. Each song in the training set is divided into 10-seconds batches and the gradient is updated at each batch. The training phase is stopped if no improvement in the validation loss is achieved for 20 epochs, where each epoch is one forward pass on all the training samples and the validation set is a disjoint 15% of the training set. The deep neural network has been implemented using Keras [11].

To define the state space of the random variables of the DBN we need to tune its parameters accordingly to the statistics (such as the minimum and the maximum tempi) of the datasets used. In particular, taking into account the BPM range in the dataset (see Figure 5.1b for an example) and the variability of the BPM in the dataset $\mathcal{T}$, we set set $bpm_{min} = 55$ and $bpm_{max} = 240$. Focusing on the variability of the BPM, we set $\lambda = 25$ in order to increase the probability of tempo transitions and, as proposed in [3], $c = 16$, its default value.

In this work, we investigate whether a source network trained for beat tracking of a general-purpose dataset is able to *transfer* its knowledge to a target network for beat tracking on a specific dataset. For this reason, we compare the performance of three models: 1) the **source network** trained over the $\mathcal{S}$ dataset; 2) the target network **TL Freeze**, obtained using transfer learning only on the higher two levels (one BLSTM and one fully-connected), as described in Sec. 4.2, and trained on $\mathcal{T}$ dataset; 3) a network trained only with $\mathcal{T}$ dataset, named **No TL**.

We trained all models but **No TL** with all the pieces in $\mathcal{S}$; then, we randomly picked 60% of $\mathcal{T}$ to train all models but the **source network**, and we left the remaining 40% for the evaluation.

5.3 Evaluation

5.3.1 Beat tracking measures

The MIR community agrees that, given the complexity of the problem to address, it does not exist a unique metric able to capture all the aspects needed for the evaluation of beat-tracking methods. Therefore, we evaluated the performances of the three aforementioned models using a set of well-known metrics for beat tracking [15]: F-measure, Cemgil, P-score, Continuity-based scores and Information gain. Each of those metrics evaluates a different aspect of the beat tracking task, by comparing the estimated set of beats $\mathcal{B} = \{b_1, \ldots, b_{N_\mathcal{B}}\}$ with the ground truth $\mathcal{GT} = \{\gamma_1, \ldots, \gamma_{N_{\mathcal{GT}}}\}$. We denote as b_i the estimated i-th beat and γ_j as the j-th beat from the ground truth and $\Delta_i = b_i - b_{i-1}$, $\Delta_j = \gamma_j - \gamma_{j-1}$ as the i-th inter-beat interval (IBI) and the j-th inter-annotation interval (IAI) respectively.

We provide here a brief description of each beat tracking measures. An in-depth discussion about beat tracking measures is provided by Davies et al. in [15].

F-measure measures the proportion between hits, false positives and false negatives. An estimated beat is considered hit if falls within a tolerance window of $\pm 0.07s$ around an annotated beat. False positives are extra beat estimates and false negatives are missing estimation. The F-measure is calculated based on value of precision and recall. Precision indicates the portion of correctly tracked beats among the number of estimates $\mathcal{B}$. Recall indicates the portion of correct beats found, out of the total number of annotations $\mathcal{GT}$. The F-measure is calculated as

$$\text{F-measure} = \frac{2pr}{p + r} \ ,$$

where p and r stands for the precision and recall values, respectively. While the F-measure considers a correct beat estimation if falling within a tolerance window (i.e. a binary discriminant), **Cemgil** metrics weights the distance between each annotated beat and its closest estimate using a Gaussian window. The Cemgil value is computed as

$$\text{Cemgil} = \frac{\sum_j \max_i W(b_i - \gamma_j)}{(N_\mathcal{B} + N_{\mathcal{GT}})/2} \ ,$$

where W is the Gaussian window with a standard deviation $\sigma =$ 0.04s. A graphical visualization of the window is depicted in Figure 5.4.

Differently from the previous measures, **P-score** does not compare the number of hits and misses, instead it sums, over a small window,

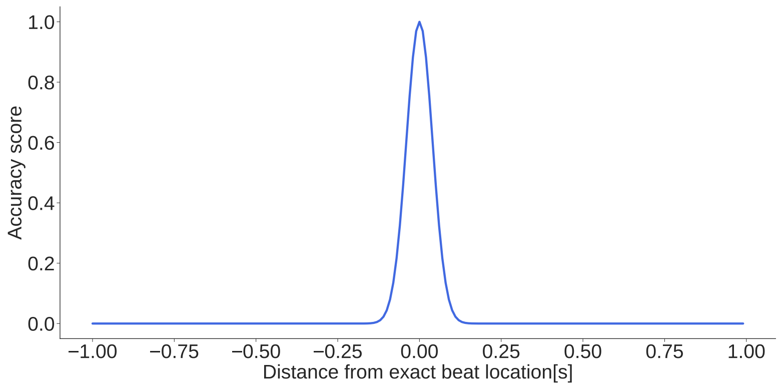


Figure 5.4: Gaussian window as a function of temporal distance between two beats

the correlation between two impulse trains representing the annotated and the estimated beats, respectively. Each impulse train has a sample rate of 100Hz and present a 1 in case there is a beat, 0 otherwise. In fact, both the annotations and the estimations are quantized to 0.01s in order to match the sample rate of the impulse train. The impulse trains are formally defined as

$$I_{\mathcal{B}}(n) = \begin{cases} 1 & n = b_i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad I_{\mathcal{GT}}(n) = \begin{cases} 1 & n = \gamma_j \\ 0 & \text{otherwise} \end{cases} .$$

The resolution of the impulse trains is so low that there is no need to go lower. In fact, it is implicitly assumed that the IBI and IAI are always bigger than the quantization, i.e. $\Delta_i > 0.01 \ \forall i$ and $\Delta_j > 0.01 \ \forall j$. An IAI less than quantization would correspond to a song at 6000 BPM, an imperceptible tempo. The cross correlation is calculated over a small window $w = 0.2\text{median}(\Delta_j)$ and is denoted as $\star_w$. Finally, the P-score is computed as

$$\text{P-score} = \frac{\sum_w I_{\mathcal{B}} \star_w I_{\mathcal{GT}}}{\max(N_{\mathcal{B}}, N_{\mathcal{GT}})} .$$

The above metrics do not consider the ability of the beat tracker algorithm to correctly track the beats in a continuous way. **Continuity-based** measures are based on the length of regions of *correctly* tracked beats. The continuity is assessed for a beat if i) it falls within a tolerance window around an annotation, ii) the previous beat falls within the previous tolerance window and iii) the actual

IBI is within a range defined by the actual IAI. Comparing each estimated beat b_i to each annotation γ_j under conditions (i)-(iii), we can find the number of beats within continuously correct segments Υ_m with $m = 0, \ldots, M$, where M is the number of correct regions. Continuity-based is a family of metrics, each of those considering a different point of view. The first measure is the Correct Metrical Level with continuity required (CML$_c$) computed as

$$\mathrm{CML}_c = \frac{\max_m(\Upsilon_m)}{N_{\mathcal{GT}}} .$$

CML$_c$ considers just the region of maximum length, so a single non-continuously tracked beat in the middle of a piece would lead to CML$_c$ = 0.5. A more loose measure is obtained by considering all the correctly tracked regions instead of just the bigger. Thus, the Correct Metrical Level considering the total number of correct regions is defined as:

$$\mathrm{CML}_t = \frac{\sum_{m=0}^{M} \Upsilon_m}{N_{\mathcal{GT}}} .$$

Similarly, it is possible to re-sample the annotation sequence $\mathcal{GT}$ in order to *allow* different metrical levels (such as double or half the correct metrical level). We can then define two different measures, namely Allowed Metrical Level with continuity required (AML$_c$) and the Allowed Metrical Level considering the total number of regions (AML$_t$).

The core idea behind the last measure, namely **Information gain**, is to measure "how much the beats are better than random" by comparing the histogram of beat error sequence with an histogram of an uniform distribution. The beat error sequence $\varepsilon_{\mathcal{B}|\mathcal{GT}}$ is calculated by measuring the temporal difference from each beat falling within half-IAI of an annotation and the annotation itself (see Figure 5.5). The divergence between the histogram of the calculated beat error sequence and the uniform histogram is computed as

$$D_{\mathcal{B}|\mathcal{GT}} = \log_2(N_{bin}) - H(p(\varepsilon_{\mathcal{B}|\mathcal{GT}}))$$

where N_{bin} is the number of bin of the histogram, $p(\varepsilon_{\mathcal{B}|\mathcal{GT}})$ is the probability mass function defined by the histogram of the timing error sequence $\varepsilon_{\mathcal{B}|\mathcal{GT}}$ and H is the entropy of the histogram, i.e. a measure of level of information. In fact, $\log_2(N_{bin})$ is the measure of the entropy of the uniform distribution. If the estimated

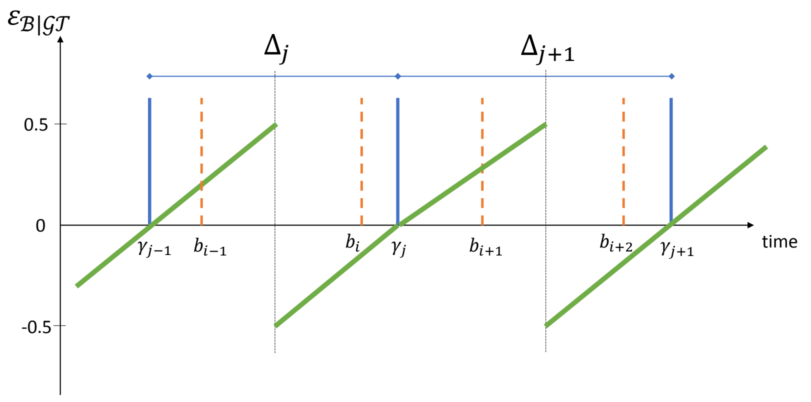


Figure 5.5: The beat error sequence definition of the estimated beats given the annotations. The timing error is calculated from each beat falling within the first half of the next IAI and the last half of the previous IAI with respect to the annotation.

beats were completely random, $H(p(\varepsilon_{\mathcal{B}|\mathcal{GT}})) \approx \log_2(N_{bin})$ since the histogram of the beat error sequence would seem close to uniform. Instead, if the histogram of the beat error sequence was deterministic $H(p(\varepsilon_{\mathcal{B}|\mathcal{GT}})) = 0$ and the divergence would be maximum. In fact, the worst beat tracker would be the one that randomly estimates the beat sequence. In order to determine the Information gain is necessary to calculate $\varepsilon_{\mathcal{GT}|\mathcal{B}}$ by measuring the temporal difference from each annotated beat falling within half-IBI of an estimate and the estimate itself. As before, it is trivial to compute $D_{\mathcal{GT}|\mathcal{B}}$. Finally, the Information gain is calculated as

$$\text{Information gain} = \min(D_{\mathcal{B}|\mathcal{GT}}, D_{\mathcal{GT}|\mathcal{B}}) \ .$$

5.3.2 Evaluation

We present here the evaluation of our method. Each metric is computed using the well-known *mir-eval* tool [49], a common standard adopted by the MIR community.

In Table 5.1 we show the evaluation on the Cretan dataset. Recalling the setup explained in Section 5.2, we compare the performance of three models. The **source network** is our base line and, as a result, its performance are quite low with respect to the other two models. Starting from the previous model, on **TL Freeze** transfer learning is applied and the Cretan dataset is taken as $\mathcal{T}$: it outperforms the model trained just on the $\mathcal{T}$ (**No TL**) of about 7% on

Table 5.1: Evaluation result on Cretan dataset

	TL Freeze	**Source Network**	**No TL**
F-measure	0.615	0.473	0.557
P-score	0.616	0.475	0.556
Cemgil	0.571	0.435	0.516
CMLc	0.583	0.432	0.488
CMLt	0.614	0.472	0.524
AMLc	0.911	0.868	0.826
AMLt	0.956	0.928	0.886
Info gain	0.518	0.493	0.478

average for all the measures. Moreover, it increases the performance of the **source network** of 10%, on average. This corresponds on what we are expecting: the Cretan dataset itself does not contain enough songs for a proper training, thus resulting in sub-optimal performance of **No TL**. However, **source network** obtains lower results with respect to **No TL**. In fact, probably due to the lack of training songs with an high degree of tempo fluctuations, **source network** is not able to deal with the higher variability of Cretan dataset.

Table 5.2: Evaluation result on Greek dataset

	TL Freeze	**Source Network**	**No TL**
F-measure	0.641	0.572	0.585
P-score	0.645	0.586	0.621
Cemgil	0.574	0.511	0.517
CMLc	0.496	0.421	0.278
CMLt	0.510	0.435	0.446
AMLc	0.752	0.707	0.476
AMLt	0.781	0.744	0.686
Info gain	0.556	0.529	0.383

Table 5.2 presents the result of the evaluation on the Greek folk dataset. In this case, we used the Greek folk dataset as $\mathcal{T}$ and the evaluation has been carried out with the same setup as before. By looking at the results, also in this case the **source network** is not capable to achieve an high quality performance due to its complete unawareness with respect to songs with an high degree of tempo fluctuation. On the other hand, the **No TL** model performances

are affected by the limited amount of data in the Greek dataset, that makes a proper training quite difficult and worsen the generalization capabilities of the model. Let the reader notice that in **No TL**, continuity-based CMLc and AMLc metrics obtained a very low value. We assume that, due to the limited training dataset and the high degree of tempo fluctuations in it, **No TL** is not able to correctly track the beats for wide temporal regions in the musical piece. This did not happened in the Cretan case since, as shown in Figure 5.3, the Cretan dataset has a lower degree of tempo fluctuations with respect to the Greek case. Using the proposed method based on transfer learning, as expected, we obtained a significant improvement. In the **TL Freeze** case, indeed, for all the metrics we obtained an average improvement of about 6%.

Table 5.3: Evaluation result on SMC dataset

	TL Freeze	**Source Network**	**No TL**
F.measure	0.388	0.375	0.369
P-score	0.507	0.488	0.506
Cemgil	0.309	0.298	0.299
CMLc	0.268	0.221	0.187
CMLt	0.287	0.234	0.253
AMLc	0.438	0.367	0.263
AMLt	0.472	0.400	0.379
Info gain	0.416	0.369	0.183

In order to prove the effectiveness of our method we also performed preliminary tests on the SMC dataset, which is well known to be extremely challenging for beat tracking algorithms [34]. This test is useful to prove the positive effect of transfer learning in increasing the performance on a dataset with higher degree of tempo fluctuations. In fact, it is possible to compare the variability of the Böck dataset and the SMC dataset by study their MAD distributions (see Figure 5.6). In this tests we kept the three models (TL Freeze, source network, no TL) of the Greek folk dataset unchanged, using the SMC only as the test set. In Table 5.3 we report the result on SMC dataset.

Also here is possible to notice the positive effect of using transfer learning, while the improvement is not as relevant as in the previous scenarios. Despite to perform beat tracking on the Greek dataset is an hard task, the SMC seems to be more challenging, as expected,

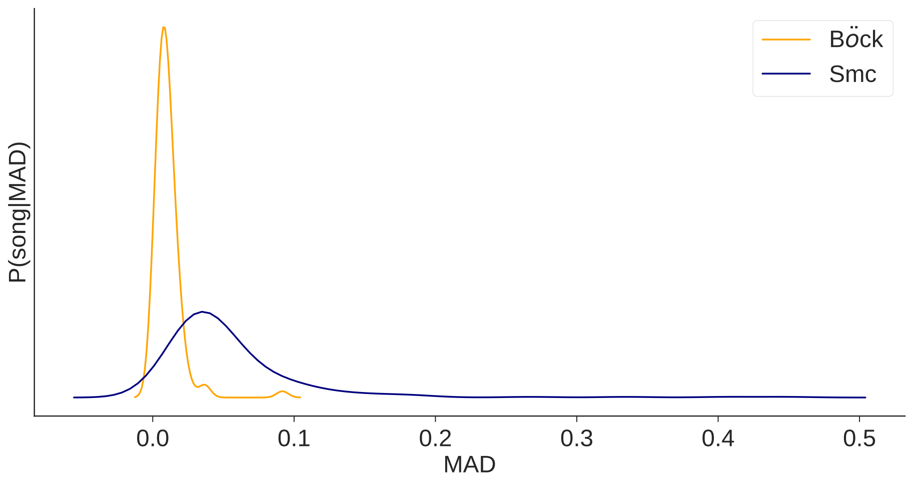


Figure 5.6: Comparison of the distributions of MAD in Böck and SMC datasets

and the result are much lower with respect to the previous cases.

Chapter 6

Conclusions

In this work we presented novel method for music beat tracking based on transfer learning.

As it has been observed, transfer learning has proved to be effective in tasks where large amount of annotated data, needed to train effective models, is not available and, in these cases, indeed, it is possible to use huge (and usually available) general-purpose datasets to learn a basic representation of the task and then transfer the acquired knowledge to achieve the desired task (with less data). We applied transfer learning for beat tracking task presenting two use cases, both regarding traditional folk music. For folk music to find annotated datasets of sufficient size to accomplish a proper training of deep learning models is not an easy task. Nevertheless, we have observed that, by applying transfer learning, it is possible to train a general purpose algorithm able to follow the higher rhythmic variety and complexity of folk music. We have tested our approach on Cretan Leaping dances and Greek folk music. With regard to the latter, we have manually annotated the beat for each song and the dataset will be publicly available. Moreover, we have proven that by applying transfer learning is possible to improve the original model's performance on a highly variable dataset. In fact, we have tested our models on the SMC dataset, known for its high variability and complexity, proving that the model on which transfer learning was applied performs better with respect to the original model trained on the source task.

To test the variability of the data, we have introduced a new measure to quantify the degree of tempo fluctuation for a song. Thanks to this metric, we have been able to demonstrate the higher variability of the target datasets with respect to the general-purpose one.

We evaluate the models using standard measures adopted by the MIR community and the performance achieved by the networks shows that the transfer learning is effective in re-tuning a RNN's parameters to track rhythmically-challenging music pieces. This can potentially be applied to other target task with properties similar to the folk music case (hard to find the data and high variability) provided that the knowledge acquired on the source task is useful for the target task.

6.1 Future work

Downbeat tracking is an harder task with respect to beat tracking, especially in case of frequent meter changes within a song. In the future we will use transfer learning to this task, possibly expanding the newly created Greek folk dataset with the annotation of the downbeat. Obviously, it is possible to continue the investigation on beat tracking and transfer learning by exploring a wider diversity across music genres. Moreover, together with the music pieces, we also collected a dataset of motion-capture movements of the Greek folk dataset. Future work includes an investigation of automatic approaches for tracking beats from dance motions with a focus on the synchronism between motion beats and the music beats.

Bibliography

[1] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, Sept 2005.

[2] S. Böck, F. Krebs, and M. Schedl. Evaluating the online capabilities of onset detection methods. 2012.

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

[4] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. 2016.

[5] 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)*, 2011.

[6] M. Buccoli, M. Zanoni, A. Sarti, S. Tubaro, and D. Andreoletti. Unsupervised feature learning for music structural analysis. In *2016 24th European Signal Processing Conference (EUSIPCO)*, Aug 2016.

[7] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 29(4):259–273, 2000.

[8] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017*, March 2017.

[9] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. In *IEEE Inter-*

national Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. IEEE, 2017.

[10] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Transfer learning for music classification and regression tasks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*. International Society of Music Information Retrieval, 2017.

[11] F. Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[12] T. Collins, S. Böck, F. Krebs, and G. Widmer. Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[13] O. Cornelis, J. Six, A. Holzapfel, and M. Leman. Evaluation and recommendation of pulse and tempo annotation in ethnic music. *Journal of New Music Research*, 42(2):131–149, 2013.

[14] W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, ICML '07, New York, NY, USA, 2007. ACM.

[15] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. 2009.

[16] M. E. P. Davies, M. D. Plumbley, and D. Eck. Towards a musical beat emphasis function. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*. IEEE, 2009.

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

[18] S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

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

[20] S. Durand, J. P. Bello, B. David, and G. Richard. Downbeat tracking with multiple features and deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015*. IEEE, 2015.

[21] C. Duxbury, J. P. Bello, M. Davies, M. Sandler, et al. Complex domain onset detection for musical signals. 2003.

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

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

[24] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[25] B. D. Giorgi, M. Zanoni, S. Böck, and A. Sarti. Multipath beat tracking. *Journal of the Audio Engineering Society*, 64(7/8):493–502, 2016.

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] M. Goto and Y. Muraoka. Beat tracking based on multiple-agent architecture-a real-time beat tracking system for audio signals. 1996.

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

[29] 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*, 14(5):1832–1844, 2006.

[30] S. W. Hainsworth, M. D. Macleod, et al. Onset detection in musical audio signals. 2003.

[31] A. C. Harvey. *Applications of the Kalman filter in econometrics*, volume 1 of *Econometric Society Monographs*. Cambridge University Press, 1987.

[32] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(9):1735–1780, November 1997.

[33] A. Holzapfel and B. Bozkurt. Metrical strength and contradiction in turkish makam music. In *Proceedings of the 2nd CompMusic Workshop*. Universitat Pompeu Fabra, 2012.

[34] A. Holzapfel, M. Davies, J. Zapata, J. Lobato Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. 20:2539–2548, 11 2012.

[35] A. Holzapfel, K. Florian, and S. Ajay. Tracking the "odd": meter inference in a culturally diverse music corpus. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 425–430, Taipei, Taiwan, 27/10/2014 2014.

[36] A. Holzapfel and Y. Stylianou. Beat tracking using group delay based onset detection. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, 2008.

[37] D. Hongker. Chicken chicken chicken: Chicken chicken. In *Proceedings of at the AAAS*, 2007.

[38] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[39] 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*, 14(1):342–355, 2006.

[40] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. 2013.

[41] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. An efficient state-space model for joint tempo and meter tracking. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[42] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat synchronous features with recurrent neural networks. 2016.

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

[44] P. Masri. *Computer Modelling of Sound for Transformation and Synthesis of Musical Signals*. University of Bristol, 1996.

[45] L. Mihalkova, T. N. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, Vancouver, BC, July 2007.

[46] J. L. Oliveira, F. Gouyon, L. G. Martins, and L. P. Reis. Ibt: A real-time tempo and beat tracking system. 2010.

[47] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[48] G. Peeters. Beat-tracking using a probabilistic framework and linear discriminant analysis. 2009.

[49] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common mir metrics. *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[50] J. Schlüter and S. Böck. Improved musical onset detection with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*. IEEE, 2014.

[51] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997.

[52] Y. Shiu, N. Cho, P. Chang, and C. J. Kuo. Robust on-line beat tracking with kalman filtering and probabilistic data association (kf-pda). *IEEE Transactions on Consumer Electronics*, 54(3), 2008.

[53] A. M. Stark, M. E. P. Davies, and M. D. Plumbley. Real-time beat-synchronous analysis of musical audio. 2009.

[54] N. Whiteley, A. T. Cemgil, and S. Godsill. Bayesian modelling of temporal structure in musical audio. Citeseer, 2006.

[55] F. F. Wu, T. Lee, J. R. Jang, K. K. Chang, C. Lu, and W. Wang. A two-fold dynamic programming approach to beat tracking for audio music with time-varying tempo. 2011.

[56] Z. Yang, R. Salakhutdinov, and W. W. Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*, 2017.

[57] J. R. Zapata, M. E. P. Davies, and E. Gómez. Multi-feature beat tracking. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(4):816–825, 2014.

[58] P. Zarchan and H. Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. Number v. 190 in Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated, 2000.