Tempo estimation via self-supervised learning

Giovana Vieira de Morais

Thesis presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
in partial fulfillment
of the requirements
for the degree of
Master of Science

Program: Computer Science
Advisor: Marcelo Gomes de Queiroz

São Paulo
August, 2023

Tempo estimation via self-supervised learning

Giovana Vieira de Morais

Dedico o texto à vó Lúcia, que confiou mais em mim do que eu mesma e sempre jurou para toda as suas vizinhas que eu era a mulher mais inteligente do Brasil. Saudades.

Acknowledgements

Irmão, você não percebeu que você é o único representante dos seus sonhos na face da terra? Se isso não fizer você correr, chapa, eu não sei o que faz.
— Emicida

I wish I could do this in English, but I can't. To my fellow English-speaker friends, I leave the "too long; didn't read" version of this acknowledgment: thank you so much!

Gostaria de iniciar agradecendo ao meu orientador, Marcelo Queiroz. Obrigada pela paciência, pela compreensão com as condições na quais eu trabalhei durante o mestrado, pelas discussões interessantíssimas e pelo esforço em manter o laboratório unido e com todos compartilhando conhecimento.

Aproveito o tópico "orientação" para agradecer à minha segunda orientadora Magdalena Fuentes por ter me aturado por 9 meses durante o programa do IEEE Mentoring Experiences for Underrepresented Young Researchers (ME-UYR), junto do Matthew (e do Félix) Davies. Eu sou muito grata de ter tido a oportunidade de trabalhar e aprender com pesquisadores tão talentosos e comprometidos. Espero um dia chegar perto disso!

Queria agradecer minha família pela paciência gigantesca, mesmo não fazendo ideia do que é que eu faço e estudo. Em mais de um momento eu pensei em desistir, mas minha mãe me mataria se eu fizesse isso, então não me restou nenhuma opção que não fosse terminar.

Finalmente, queria agradecer à Sofia, que esteve do meu lado do começo ao fim, me arrancando do lado do computador quando eu precisei respirar e me trazendo um café quando eu precisei correr com prazos malucos. Obrigada por crescer junto comigo e aceitar minhas ideias doidas.

Um agradecimento especial aos meus colegas de laboratório do COMPMUS, principalmente ao meu colega Carlos Castro: obrigada pelas conversas e sessões de música no

Discord. Agradeço também a Delia Fano e Elio Quinton por me mentorarem no programa do WiMIR, compartilhando suas experiências e me dando conselhos valiosos sobre indústria, academia e vida pessoal. Não posso deixar de agradecer à Katia por me salvar um milhão de vezes e me ajudar a navegar as burocracias da USP! Finalmente, aos meus amigos Rita, Zé, Chipa e Zamur fica o obrigada final por acompanharem essa doideira que foi o mestrado. Que logo menos possamos nos encontrar e bater um papo. Quem tem um amigo tem tudo e, por sorte, eu tenho vários!

Muito obrigada, galera!

Resumo

Giovana Vieira de Morais. **Estimação de andamento via self-supervised learning**. Dissertação (Mestrado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Métodos auto-supervisionados (self-supervised) aprendem representações de dados por meio da resolução de tarefas de pretexto (pretext tasks) que não necessitam de rótulos gerados por humanos, diminuindo a necessidade de dados anotados para o treinamento dos modelos. Esses métodos foram aplicados em problemas de visão computacional, processamento de linguagem natural, análise de som ambiente, e, recentemente, em recuperação de informação musical. Particularmente no contexto da música, existem poucos insights sobre a fragilidade desses modelos no que diz respeito à diferentes distribuições de dados e como elas podem ser mitigadas. Nesse trabalho, exploramos essas questões ao dissecar um modelo auto-supervisionado, que foi adaptado da estimação de pitch para a estimação de andamento, por meio de uma exploração rigorosa com dados sintéticos, cujo desempenho foi comparado ao uso de dados reais. Discutimos as escolhas de design a respeito do método e das representações dos dados de entrada. Finalmente, estudamos a relação entre a representação de entrada e a distribuição dos dados para a estimação de andamento.

Palavras-chave: estimação de andamento. self-supervised learning.

Abstract

Giovana Vieira de Morais. **Tempo estimation via self-supervised learning**. Thesis (Master's). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

Self-supervision methods learn representations by solving pretext tasks that do not require human-generated labels, alleviating the need for time-consuming annotations. These methods have been applied in computer vision, natural language processing, environmental sound analysis, and recently in music information retrieval, e.g. for pitch estimation (Gfeller *et al.*, 2020). Particularly in the context of music, there are few insights about the fragility of these models regarding different distributions of data, and how they could be mitigated. In this work, we explore these questions by analyzing a self-supervised model for pitch estimation that we adapted for tempo estimation via rigorous experimentation with synthetic data and contrasting its behavior with real data. We discuss the design choices regarding the method and the input representation. Finally, we study the relationship between the input representation and data distribution for self-supervised tempo estimation.

Keywords: tempo estimation. self-supervised learning.

List of Abbreviations

AI	Artificial Intelligence
BPM	Bar Pointer Model
BPM	Beats Per Minute
CNN	Convolutional Neural Network
CQT	Constant-Q Transform
CSD	Complex Spectral Difference
DBN	Dynamic Bayesian Network
DFT	Discrete Fourier Transform
DL	Deep Learning
DNN	Deep Neural Networks
GMM	Gaussian Mixture Models
GRU	Gated Recurrent Units
HF	Harmonic Features
ICASSP	International Conference on Acoustics, Speech, and Signal Processing
IEEE	Institute of Electrical and Electronics Engineers
IME	Instituto de Matemática e Estatística
ISMIR	International Society for Music Information Retrieval
KNN	K-Nearest Neighbors
MAF	Mel Auditory Feature
MFCC	Mel Frequency Cepstral Coefficients
MIR	Music Information Retrieval
MIREX	Music Information Retrieval EXchange
ML	Machine Learning
ODF	Onset Detection Function
PSF	Phase Slope Function
PGM	Probabilistic Graphic Model
RNN	Recurrent Neural Networks
SF	Spectral Flux

SSL	Self-Supervised Learning
SSM	Self-Similarity Matrix
STFT	Short-Time Fourier Transform
SVM	Support Vector Machines
TCN	Temporal Convolutional Network
USP	Universidade de São Paulo
ZeroNS	Zero-Note Samba

List of Symbols

ACC_1	Accuracy 1
ACC_2	Accuracy 2
OE_1	Octave Error 1
OE_2	Octave Error 2
Θ	Tempi range considered for the tempogram
$\mathcal{T}_\mathrm{A}$	Autocorrelation tempogram
$\mathcal{T}_\mathrm{F}$	Fourier tempogram
$\mathcal{T}_\mathrm{H}$	Hybrid tempogram

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Tempo is a fundamental dimension of music, representing the speed at which a listener would tap along to mark the underlying pulse. Despite its significance, its definition remains unclear, and when people are asked to annotate the tempo for a piece of music, they often disagree on the values they assign, as revealed in an experiment conducted by McKinney and Moelants (2006). Typically, these differences are multiples of the most salient tempo, leading to the terms ”tempo harmonics” or ”tempo octaves.”

Automatic tempo estimation has become a prominent research topic within the Music Information Retrieval (MIR) community, garnering significant attention over the past decade. However, it remains a challenging task due to the intrinsic ambiguity of tempo, resulting from the various possible interpretations of a given rhythm's metrical structure. In 2005, it was officially incorporated into the Music Information Retrieval Evaluation eXchange (MIREX) competition (Downie, 2008) as the ”Audio Tempo Extraction” task[1], later renamed ”Audio Tempo Estimation” [2]. Each year, new algorithms and improvements are proposed and evaluated using standard datasets and metrics.

Despite more than 25 years of research on tempo estimation, various dimensions, such as metrics, estimation algorithms, datasets, and possible biases, continue to be revisited and debated (Schreiber, Urbano, *et al.*, 2020). Outside of genres like pop and techno music, which inherently possess well-established and periodic tempos, tempo estimation is still considered an unsolved problem.

While early tempo estimation methods were largely based on signal processing, recent approaches exploit deep neural networks (DNNs), bringing new perspectives to the problem. This introduction meant a change of paradigm and improvement in the accuracy of tempo estimation models, but introduced a new problem: DNNs are very data-hungry (Böck and Davies, 2020).

[1] https://www.music-ir.org/mirex/wiki/Audio_Tempo_Extraction

[2] https://www.music-ir.org/mirex/wiki/2021:Audio_Tempo_Estimation

Within the MIR domain, the data bottleneck poses a challenge, as obtaining annotated data requires musical expertise. Additionally, available datasets are biased towards Western and "mainstream" music, typically found on popular media platforms like streaming services.

To address the lack of new data and the bias of the available datasets, researchers are developing and exploring techniques such as:

- Data augmentation to increase the size of existing datasets (Böck and Davies, 2020; C. Zhang *et al.*, 2021).
- Methods that can be adapted to new data with small samples, e.g. Few-Shot Learning (Wang, Salamon, *et al.*, 2020; Wang, Stoller, *et al.*, 2022).
- Unsupervised, semi-supervised or self-supervised alternatives, where data annotation is not needed (Jansen *et al.*, 2018).

In parallel with other research fields like computer vision and natural language processing, the data bottleneck has spurred the adoption of self-supervised learning. This set of methods derives labels algorithmically from the data, eliminating the need for human supervision. Self-supervised learning trains a model by solving a *pretext task*, which may not directly relate to the final application (*downstream task*) but leverages intrinsic information within the data (Balestriero *et al.*, 2023). When well-designed, solving the pretext task enables the model to learn meaningful internal representations that can be applied to the problem of interest (Goodfellow *et al.*, 2016).

This approach has been exploited in computer vision, environmental sound analysis, and very recently in MIR for pitch (Gfeller *et al.*, 2020), beat tracking (Desblancs *et al.*, 2022) and tempo estimation (Quinton, 2022).

1.2 Related Work

Previous work on self-supervised pitch estimation, SPICE (Gfeller *et al.*, 2020), introduced the concept of estimating relative pitch by inputting two shifted slices of a constant-Q transform (CQT) to the same convolutional autoencoder, and trained the encoder so that the difference in its outputs would be proportional to the introduced shift. This pretext task led to very good results while keeping pre-processing to a minimum: the computation of a CQT and the shift.

Similar to how a magnitude spectrogram or CQT represents time-frequency varying contents of a signal which are important for pitch analysis, the tempogram representation indicates for each time frame the local relevance of a tempo estimate for a given music recording, see Section 2.1.4. This makes it a compelling case to adapt an architecture such as SPICE to tempo estimation using a tempogram as input representation.

Desblancs *et al.* (2022) proposed a self-supervised method for beat tracking in which they trained two different Convolutional Neural Networks: one to process the percussive part and another to process the non-percussive part of a music signal. The networks should be able to find interesting data representations by learning how to synchronize the percussive and non-percussive parts of the audio.

Finally, the most recent study of self-supervision applied to tempo estimation was done in QUINTON (2022). In this work, the author creates a contrastive pretext task that aims to find a tempo representation by comparing two time-stretched versions of an audio sample.

On this basis, we explore the adaptation of SPICE to tempo estimation from a tempogram and study the different design choices as well as their effect on tempo estimation, such as the use of a logarithmic BPM sampled axis instead of the linear one, following GROSCHE, MÜLLER, *et al.* (2010).

The Self-supervised learning approaches will be discussed in-depth in Section 2.2.2.

1.3 Objectives

- **Understand if we can use a SPICE-inspired architecture to do tempo estimation:** In this work, we adapt the SPICE architecture and explore the performance of the model when using tempograms as input data.
- **Explore the effects of the dataset distribution in the model training and evaluation performance, with both synthetic and real data:** We create different datasets of synthetic data and analyze how they affect the model output.
- **Propose data augmentation strategies:** We want to understand if the insights we have found by training the model with synthetic data hold for real data. To do so, we want to use the synthetic data distributions to guide data augmentation within the real-world datasets and validate the results.

1.4 Text Outline

This document is structured as follows: Chapter 2 is an overview of onset detection, tempo representations (tempograms), tempo estimation, deep learning, deep learning methods for tempo estimation and self-supervised learning. In particular, the self-supervised learning subsection introduces SPICE, the method that we use as a reference for our own network framework implementation. Chapter 3 describes the proposed architecture and the experiments, also discussing the results achieved.

Chapter 2

Literature Review

2.1 Tempo Estimation

Musical beats are rhythmic units usually associated with our perception of the periodicity of note events. Tempo estimation is a task that aims to infer the tempo value of a given song, i.e. the number of beats occurring within a certain time period, typically measured in Beats Per Minute (BPM). This task is more challenging for songs that have local tempo changes (e.g. *accelerando*) (Schreiber, Zalkow, *et al.*, 2020) , syncopated rhythms (where strong events appear outside the beat structure, and asymmetric rhythms (where beats have varying durations) (Fouloulis *et al.*, 2012). It can be further divided into other two tasks, local tempo estimation (tempo as a function of musical time) and global tempo estimation (tempo as a global value).

Automatic tempo estimation is a task that is researched for over 25 years in MIR (Schreiber, Urbano, *et al.*, 2020) and the proposed methods have evolved from signal processing-based methods, i.e. classical methods, to deep learning based methods, which are currently the state of the art (Böck and Davies, 2020).

Over the past 25 years, automatic tempo estimation has been a significant area of research in Music Information Retrieval (MIR). The methods have evolved from classical signal processing-based approaches to the current state-of-the-art deep learning-based methods.

According to Oliveira *et al.* (2012), there are two main ways of analyzing tempo and beat information: *predictive* and *descriptive* methods. The predictive method, also known as online tracking, mimics human behavior by estimating beats and rhythmic information causally, meaning the tracker does not have access to the full music audio. On the other hand, the descriptive approach, or offline tracking, provides the tracker with access to the entire music input.

Although early methods like (Goto and Muraoka, 1995; Scheirer, 1998; D. P. W. Ellis, 2007) were online trackers, offline tracking has become more prevalent in research due to its empirical accuracy and the fact that it does not require predicting tempo and beats in a causal manner, as seen in the online approach. Online tempo estimation is outside of the scope of this work, so from now on "tempo estimation" will refer as "offline tempo estimation", unless defined otherwise.

The extraction of tempo and beat information follows a general scheme for signal processing approaches, as shown in Figure 2.1. First, we calculate onset features (see Section 2.1.1), which identifies timestamps characterized by a sudden energy increase in the audio signal. Then, one can use onsets' inner periodicity to infer tempo, by assuming that onsets timestamps are associated with note events, some of which coincide with beats.

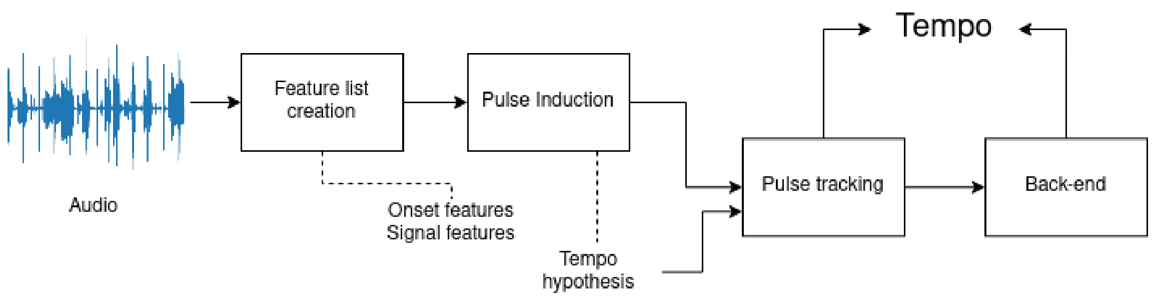


Figure 2.1: *Tempo estimation general workflow (Gouyon et al., 2006)*

As mentioned in Chapter 1, tempo can be perceived differently by different people (McKinney and Moelants, 2006). This leads to a big difficulty in the tempo estimation problem definition: are we estimating the perceptual tempo or are we estimating the annotated data? McKinney and Moelants (2006) shows that the tempo ambiguity is highly related to the music genre being analyzed.

cadê a tabela daqui? manter ou remover?

Quinton (2017) reinforces this argument by showing the disagreements between annotators for the GTZAN dataset, as shown in Table **??**

This discussion is not new: some works try to estimate the perceived tempo by estimating two values and showing their strength when compared to each other (Levy, 2011; Geoffroy Peeters and Flocon-Cholet, 2012), while others try to estimate the annotated tempo and then reduce the the octave error (Xiao *et al.*, 2008; Gkiokas, Katsouros, and G. Carayannis, 2012; Schreiber and Müller, 2017).

In this work, we will not try to estimate the perceived tempo, but it is interesting to understand how psychoacoustics and perception guide some of the design choices presented in the next pages.

2.1.1 Novelty Function

Classical tempo estimation approaches usually start from the onset detection. Formally, the onset definition depends on how the audio was annotated. Lerch (2012) describes the three different annotation methods proposed by Repp (1996):

Note Onset Time (NOT): considers the moment in which the instrument made a sound.

Acoustic Onset Time (AOT): considers the moment in which the signal or event is technically measurable.

Perceptual Onset Time (POT): consider the moment in which the signal is perceived by the listener.

Onset detection is especially challenging in polyphonic music, where simultaneous events can occur at the same time, therefore making it difficult to understand where each event starts. The method and the parameters that one chooses when trying to detect onsets also relies on some properties of the analyzed music. For example, Böck and Widmer (2013) proposes a method that tackles classical songs and their softer onsets, while Nunes *et al.* (2015) take into account specific frequency bands that have important information for Candombe beat and tempo information.

Knowing that the task is signal-dependent, datasets, such as the one provided by Bello *et al.* (2005) and the ones provided in MIREX, have distinctions between pitched non-percussive (e.g. bowed strings), pitched percussive (e.g. piano), non-pitched percussive (e.g. drums), and a complex mixture of onsets (e.g. pop songs). This is important to assess the quality and robustness of the proposed method in different scenarios.

More recent works on onset detection apply machine learning and deep learning techniques to it, such as Eyben *et al.* (2010), which uses a Recurrent Neural Network, and the current state-of-the-art method by Schlüter and Böck (2014) that uses Convolutional Neural Networks. Our implementation, as seen in Section 3.1, is based purely on signal processing techniques.

Multi-band Analysis

For tempo and beat detection, one common pre-processing technique used is the division of the signal in multiple frequency bands, that are analyzed independently, i.e. each subband has its novelty function.

For example, Nunes *et al.* (2015) create the onset detection function knowing that one specific instrument, the Piano, was responsible for carrying the tempo in the Candombe songs, therefore they adapted the function to take into account the frequency band that this instrument produced. This led to a great accuracy improvement for Candombe songs when compared to other beat-tracking algorithms that considered the full spectrum range.

Zapata and Gómez (2011) showed that analyzing the onset function in different frequencies subbands is beneficial and increases the accuracy of the methods. McFee and D. P. Ellis (2014) showed that not only the detection in subbands is important but the aggregation of the results also makes a difference: using the median instead of the average proved to generate a more robust beat tracking results.

Reduction

The reduction step is responsible for transforming the input audio signal x in a subsampled *novelty function*, also known in the literature as *detection function* and *onset function*, that reflects the onsets occurrences in the original signal.

According to Bello *et al.*, 2005 methods to achieve the detection function can be divided into two main categories: based on signal features and based on probability models. We will not dive into the second category of methods because it is out of the scope of this work.

The works of Bello *et al.* (2005) and S. Dixon (2006) present the most common classical approaches, that once achieved state-of-the-art for the task, also analyzing the pros and cons of the methods and discuss scenarios to apply them. Here, we describe the process to create a Spectral-Based Novelty function (also known as *Spectral Flux*). The Spectral Flux is a robust method that has been shown to work well in different scenarios without the need for much fine-tuning. It is also the base method for other novelty functions, such as the SuperFlux Böck and Widmer (2013).

Spectral-Based Novelty (Spectral Flux) The energy-based approach does not work properly when we have polyphonic audio because the energy values are overlaid, making it hard to detect every by looking only at the signal's energy.

The spectral-based method, or spectral flux, analyzes the audio spectrum and identifies the increase of energy of one or multiple bands. We first calculate the spectrogram X (Equation 2.1) of the audio signal through the Short-Term Fourier Transform (STFT).

$$X(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N) \tag{2.1}$$

where w is a window function of length N, and $m \in \mathbb{Z}$ and $k \in [0 : K]$.

We can enhance the spectral components by applying the logarithmic compression (Equation 2.2).

$$\mathcal{Y} = \Gamma_\gamma(|X|) = \log(1 + \gamma \cdot |X|) \tag{2.2}$$

where $\gamma \in \mathbb{R}_{>0}$ regulates the degree of compression.

Then, following the same approach as the energy-based novelty, we calculate the first-order derivative between the frames of X and discard the negative values by using the Half-Wave Rectification (Equation 2.3). The idea is that we should only account for new events, i.e. energy increases.

$$|r|_{\geq 0} = \frac{r + |r|}{2} = \begin{cases} r, & \text{if } r \geq 0, \\ 0, & \text{if } r < 0. \end{cases} \tag{2.3}$$

Finally, we aggregate the frequency bands and the result is the novelty function Δ_{Spectral}, shown in Equation 2.4. The frequency bands' most frequent aggregation is either the median or average. McFee and D. P. Ellis (2014) discusses the effects of different aggregations in beat tracking results.

$$\Delta_{\text{Spectral}}(n) = \sum_{k=0}^{K} |\mathscr{Y}(n+1,k) - \mathscr{Y}(n,k)|_{\geq 0} \tag{2.4}$$

There are still some postprocessing steps that one can apply to enhance the novelty function onsets. For example, Müller (2015) mentions that subtracting the local average can help to reduce small fluctuations.

$$\mu(n) = \frac{1}{2M+1} \sum_{m=-M}^{M} \Delta_{\text{Spectral}}(n+m), \tag{2.5}$$

$$\bar{\Delta}_{\text{Spectral}}(n) = |\Delta_{\text{Spectral}}(n) - \mu(n)|_{\geq 0} \tag{2.6}$$

Figure 2.2 shows how Δ_{Spectral} looks like with and without the multi-band analysis. Equation 2.4 shows the aggregation made by taking the mean of all frequencies coefficients, but if one chooses to do the multi-band analysis, then the aggregation would be over frequencies bands, also called *channels*.

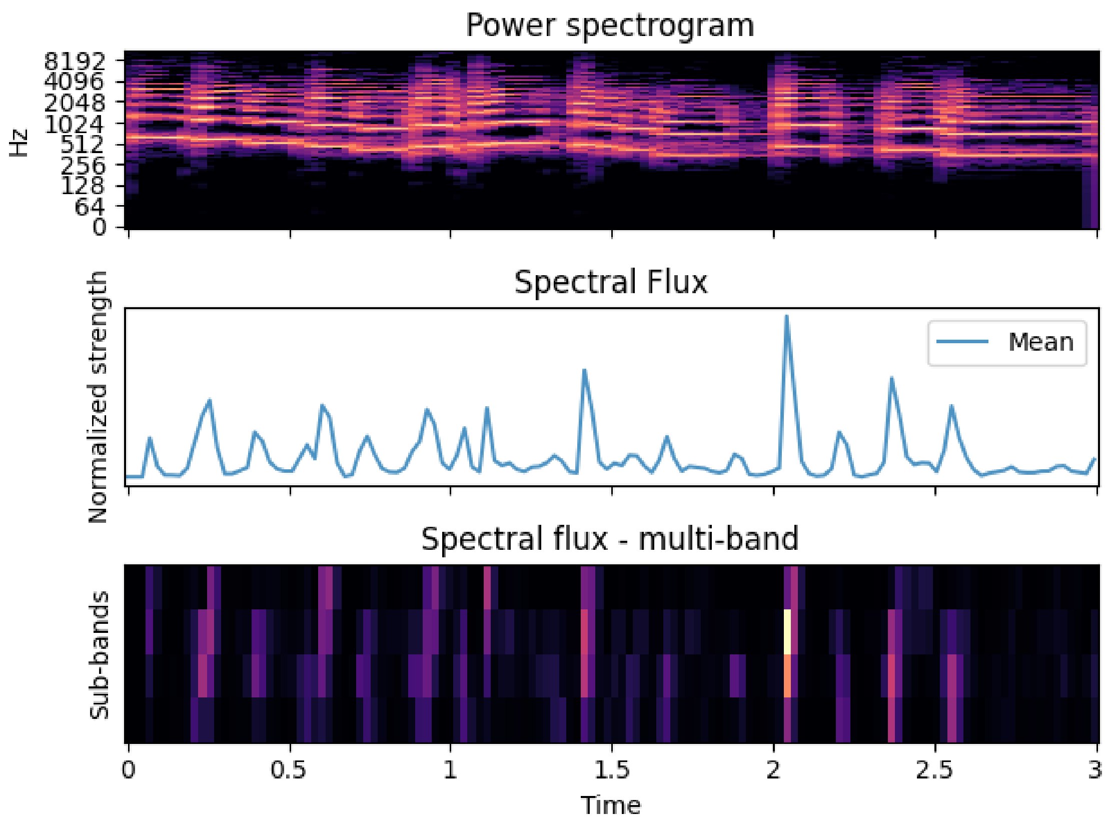


Figure 2.2: *Spectral flux with and without multi-band analysis. The multi-band spectral flux is made over 4 frequency bands.*

Other Approaches Müller (2015), Bello *et al.* (2005), and S. Dixon (2006) present other approaches to derive the novelty function from the signal, such as a phase-deviation approach and a complex domain approach (considering both the spectrum and the phase).

It is also possible to combine different novelty function methods in the same signal, as shown in Duxbury *et al.* (2002) work where they combine energy methods and spectral methods in different subbands, by assuming that soft onsets, i.e. non-percussive onsets, have more information in low frequency than high frequency.

Besides the spectral flux, one can choose to use other signal's properties to detect onsets. For example, Böck and Widmer (2013) proposed a method that incorporates the signal's phase, pitch, and a combination of both, reducing by 60% the false positives previously presented in datasets with classical songs, which are known for having much softer and hard-to-detect onsets.

Peak picking

Peak picking is the final step to choosing the onset times for all the onset candidates. It is a detection function that aims to find the local maxima of the novelty function derived from the audio signal. We will not dive much into this topic because this will not be used in this work, i.e. we are going to analyze the novelty function (onset envelope) directly.

To choose the peaks that correspond to the onsets, discarding any false positives, one needs to set a threshold, which can be fixed throughout the whole signal or adaptative, which is a more robust choice in general. An in-depth comparison of peak picking methods for onset detection methods based on signal processing can be found in Rosão *et al.* (2012) work.

2.1.2 Tempo Induction

Given a novelty function Δ, the tempo induction, also known as *pulse induction*, step is the one responsible for measuring the periodicities.

One of the most common approaches is the use of the autocorrelation function (ACF) to estimate the periodicities. Between the 23 systems evaluated in Zapata and Gómez (2011), 13 used ACF as the Pulse Induction algorithm, 3 used Inter-Onset-Interval (IOI) Clustering, one used Spectral Product, one used Bankcomb Filter, and 4 did not provide the Pulse Induction Method.

Simon Dixon (2001) and later on Oliveira *et al.* (2012) examines times between pairs of note onsets and uses a clustering algorithm to find significant clusters of IOIs. Each cluster represents a tempo hypothesis that is tested by a multi-agent beat tracking system. A similar idea of clustering tempo hypothesis is shown in Eronen and Klapuri (2010), where the periodicities of the novelty function is measured using an autocorrelation function, followed by the tempo estimation using k-Nearest Neighbor (KNN) regression.

According to Gouyon *et al.* (2006) and Zapata and Gómez (2011) the pulse induction is more efficient when combined with the subband division, i.e. when one analyzes the periodicities in each frequency subband and then aggregate the results.

Interestingly, in both comparisons (Gouyon *et al.*, 2006; Zapata and Gómez, 2011) the ”winner” method (Klapuri *et al.*, 2006) did not find onsets based on the approaches presented in Subsection 2.1.1, but uses the differentials of the loudness in 36 frequency bands which are the input for a bankcomb filter responsible for tempo induction. Zapata and Gómez (2011) assumes that the success of Klapuri’s approach lies in good feature extraction rather than in the tempo induction step.

Finally, other pre-deep-learning approaches rely on interesting techniques, such as harmonic and noise separation (Alonso *et al.*, 2007) and source separation (Gkiokas, Katsouros, George Carayannis, *et al.*, 2012), but that is out of the scope of this work.

2.1.3 Metrics and Evaluation

In 2004, there was a tempo induction contest at the International Conference on Music Information Retrieval[1] (ISMIR) with the goal to evaluate the state-of-the-art methods to extract the global tempo value of an except, measured in beats per minute. The evaluation system proposed in this conference is described in-depth by Gouyon *et al.* (2006) and revisited by Schreiber, Urbano, *et al.* (2020).

The main metrics proposed were Accuracy 1 (ACC_1) and Accuracy 2 (ACC_2). The two metrics remain the standard metrics for evaluating the tempo estimation system and are defined as follows:

- *Accuracy 1:* The percentage of tempo estimates within 4% (the precision window) of the ground-truth tempo.
- *Accuracy 2:* The percentage of tempo estimates withing 4% of either the ground-truth tempo, or half, double, three times, or one-third of the ground-truth tempo.

Schreiber, Urbano, *et al.* (2020) discuss that the value of the precision window is rather arbitrary and may be too high for music with stable tempi, but also too strict for music with less stable tempi. The precision window, also known as *tolerance window*, raises another problem related to the binary nature of these metrics. Even if ACC_2 is less strict than ACC_1 because it takes octave errors into account, both metrics do not give any additional information about the model performance. One just knows that the estimation is wrong, but doesn’t know *how much wrong* it is.

Implementation for all metrics mentioned in this section can be seen in Appendix B.1.

2.1.4 Tempograms

Tempograms are one type of audio representation that indicates the local relevance of a specific tempo for each audio time instance t. For example, if the tempogram $\mathscr{T}$ has a high value $\mathscr{T}(3, 160)$, that means that the signal has a dominant tempo $\tau = 160$ BPM at time $t = 3$. It is a representation used to analyze the periodicities of the novelty function and obtain a local tempo salience map.

[1] https://ismir.net/

One of the main challenges of the periodicity analysis relates to the multiple periodicities that exist in the hierarchical rhythmic structure of music, which are reflected in the tempogram as harmonics or sub-harmonics. Although this "harmonic" organization resembles harmonic-sound structures in a spectrogram representation, it can be more ambiguous, as the predominantly perceived tempo of a piece might have a low salience in the tempogram while its harmonics (or sub-harmonics) might have strong salience. This "harmonic structure" depends on the music and the method used to compute the tempogram (e.g autocorrelation or Fourier analysis).

Tempograms are not the only time-tempo representation available. There are also other representations known as rhythmograms (Quinton, 2017), predominant local pulses (Grosche and Muller, 2011), beat spectrograms (Foote and Uchihashi, 2001), and cyclic beat spectrum (Kurth and Gehrmann, 2006), but they are out of the scope of this work.

Tempograms can also be the first step to more complex mid-level representations, such as the tempo salience map (Thoshkahna *et al.*, 2015), rhythmic salience map (Quinton *et al.*, 2016). Tian *et al.* (2015) uses tempogram-based features, such as *Tempogram Principal Component Analysis (TPCA)*, *Tempogram Cepstral Coefficients (TCC)*, *Tempo Intensity (TI)* and *Tempo Intensity Ratio (TIR)* in order to perform music structural segmentation. The results showed that tempogram-based features have a strong capacity of describing music structure.

We define a discrete tempogram $\mathscr{T}$ as

$$\mathscr{T} : \mathbb{Z} \times \Theta \to \mathbb{R}_{>0}, \tag{2.7}$$

where $\Theta \subset \mathbb{R}_{>0}$ is a finite set o tempi specified in BPM, usually $\Theta = [30 : 600]$ BPM. According to Müller (2015, chapter 6) the motivation behind the boundaries relies on the assumption that only events that show a temporal separation between roughly 100 ms (600 BPM) and 2 sec (30 BPM) contribute to the perception of tempo. In works like Grosche, Müller, *et al.* (2010), the interval is even smaller, ranging from $[30 : 480]$ BPM, covering four tempo octaves. Quinton (2022) defines the BPM interval as $[0, 300]$ BPM.

For generating the tempogram calculation, we first need to compute the novelty function and then analyze the salience of the novelty to obtain a local tempo salience map. This general workflow is based on the premise that pulse positions usually go along with note onsets, therefore we can gather useful information from the novelty function.

The analysis and estimation of the periodicity can be done using a short-time Fourier transform (STFT), resulting in the Fourier Tempogram $\mathscr{T}_{\mathrm{F}}$ (Equation 2.9), or using the autocorrelation function, which results in the Autocorrelation Tempogram $\mathscr{T}_{\mathrm{A}}$ (Equation 2.12).

Implementations examples can be found in the FMP notebooks (Müller and Zalkow, 2019)[2].

[2] https://www.audiolabs-erlangen.de/resources/MIR/FMP/C6/C6.html

The figures below show tempograms of some examples in the GTZAN dataset. We compare and discuss the tempograms in the following pages. More tempogram examples from the dataset can be found in Appendix A.1

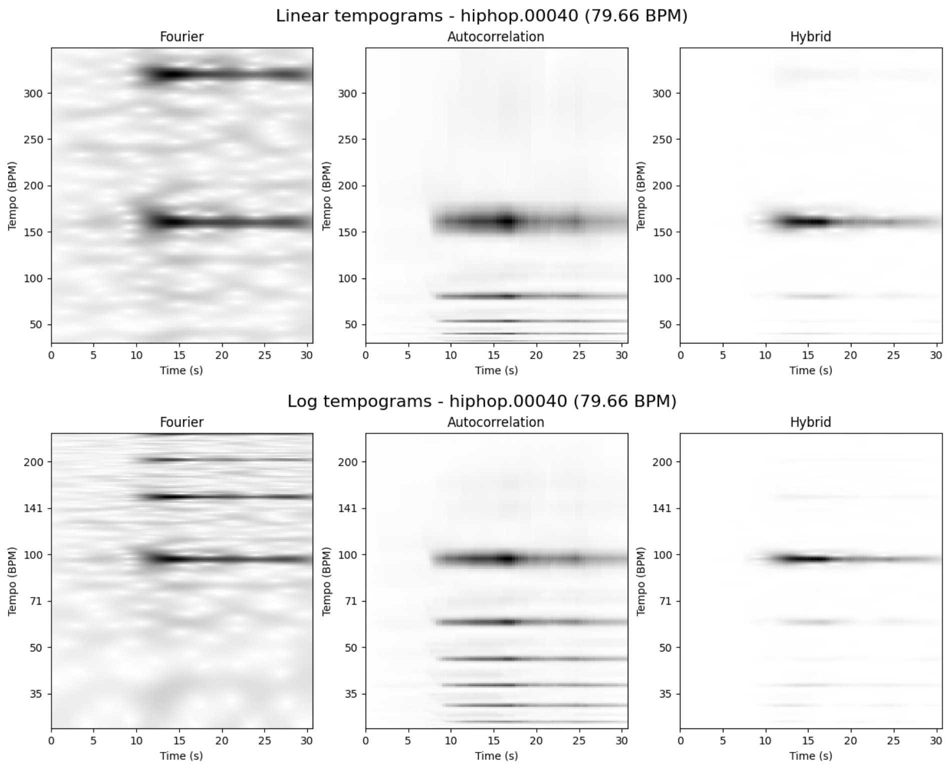


Figure 2.3: *Linear and log tempograms example for GTZAN hip-hop track*

In Figure 2.3 we see that the three variations of tempograms have a darker line around 160 BPM even if the annotated tempo of the track is around 80 BPM. At the first column, we have the Fourier tempogram, showing the harmonics in 160 BPM and 320 BPM, while the Autocorrelation variation shows sub-harmonics in 80 BPM, 60 BPM and 40 BPM.

Some music genres have a more stable tempi than others, for example it is common that genres such as hip-hop (Figure 2.3) and metal (Figure 2.6) have a more steady tempo than jazz (Figure 2.7).

Fourier Tempogram

The Fourier tempogram is the result of analyzing the novelty function via the discrete STFT (Equation 2.1) but using $H = 1$.

$$\mathscr{F}(n,\omega) := \sum_{m\in\mathbb{Z}} \Delta(m)\bar{w}(m-n)\exp(-2\pi i\omega m) \tag{2.8}$$

where Δ is the novelty function, $\omega \in \Theta$ and w is a window function of length N.

The coefficients $\mathscr{F}(n,\omega)$ are in Hertz (Hz), so we use the relationship $\tau = 60 \cdot \omega$ to convert the frequency results to BPM, leading to the final formulation

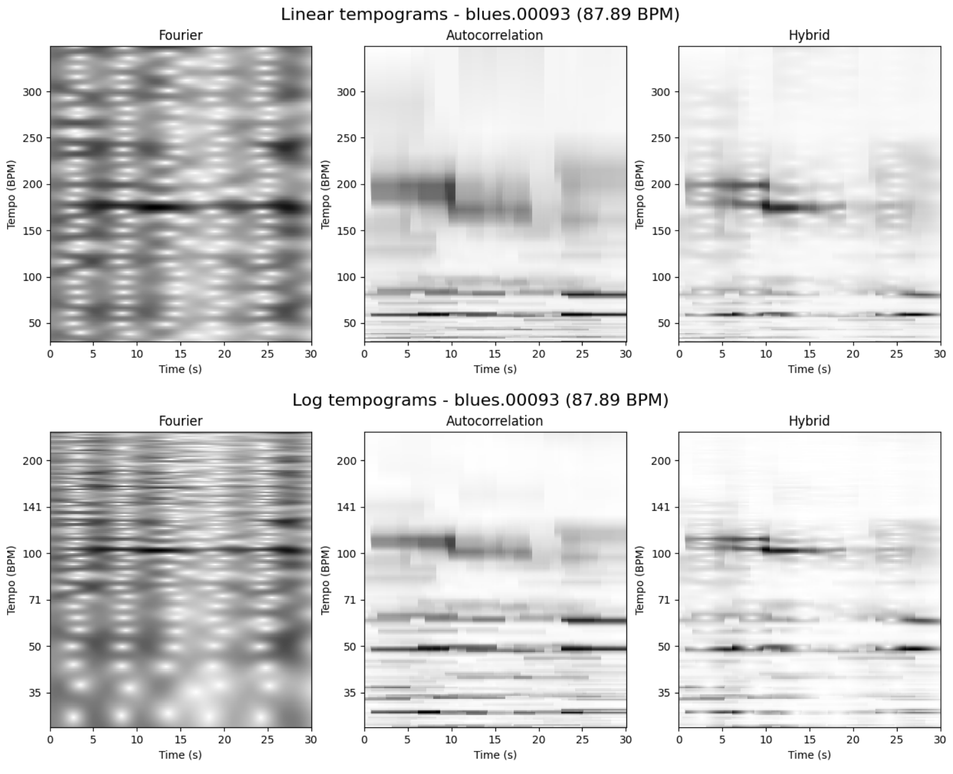


Figure 2.4: *Linear vs log tempograms example for GTZAN blues tracks*

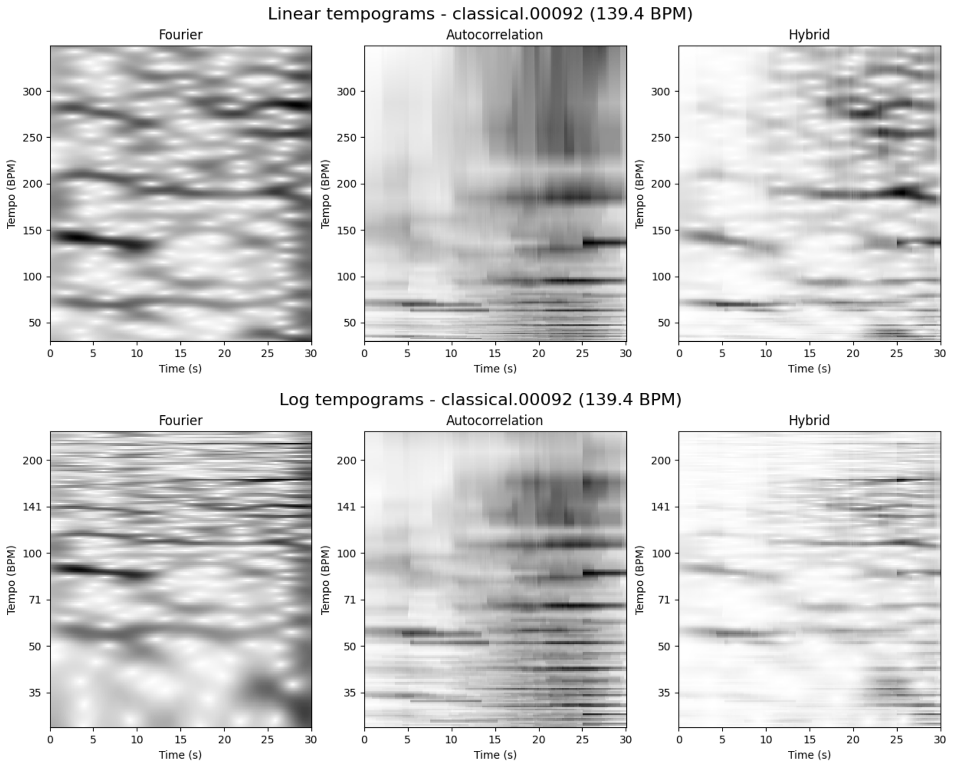


Figure 2.5: *Linear and log tempograms example for GTZAN classical track*

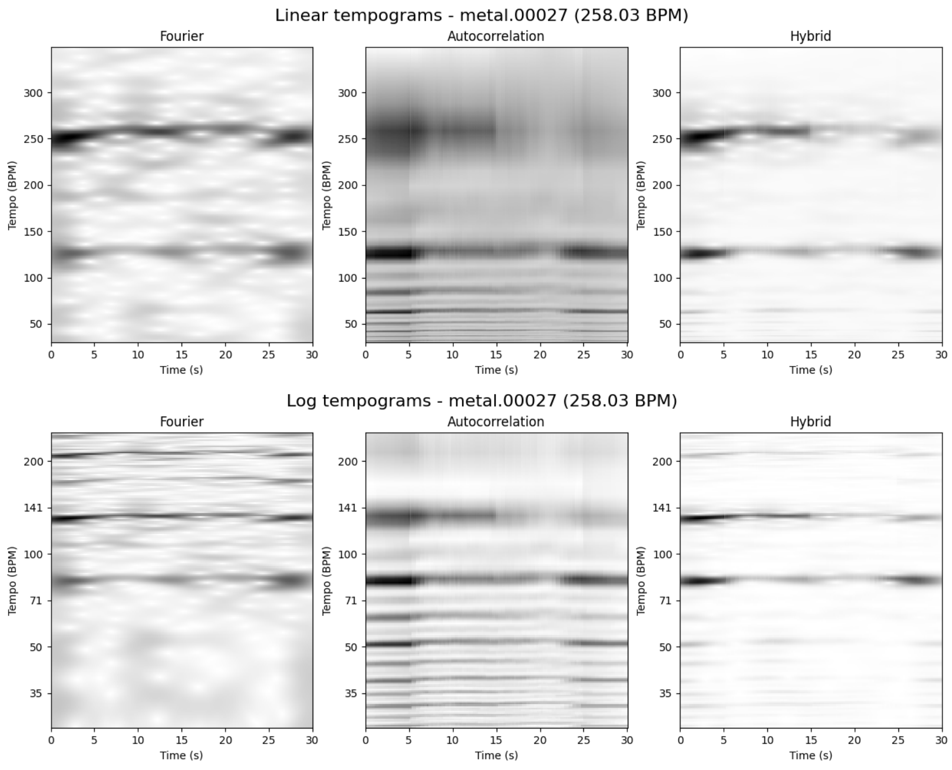


Figure 2.6: *Linear and log tempograms example for GTZAN metal track*

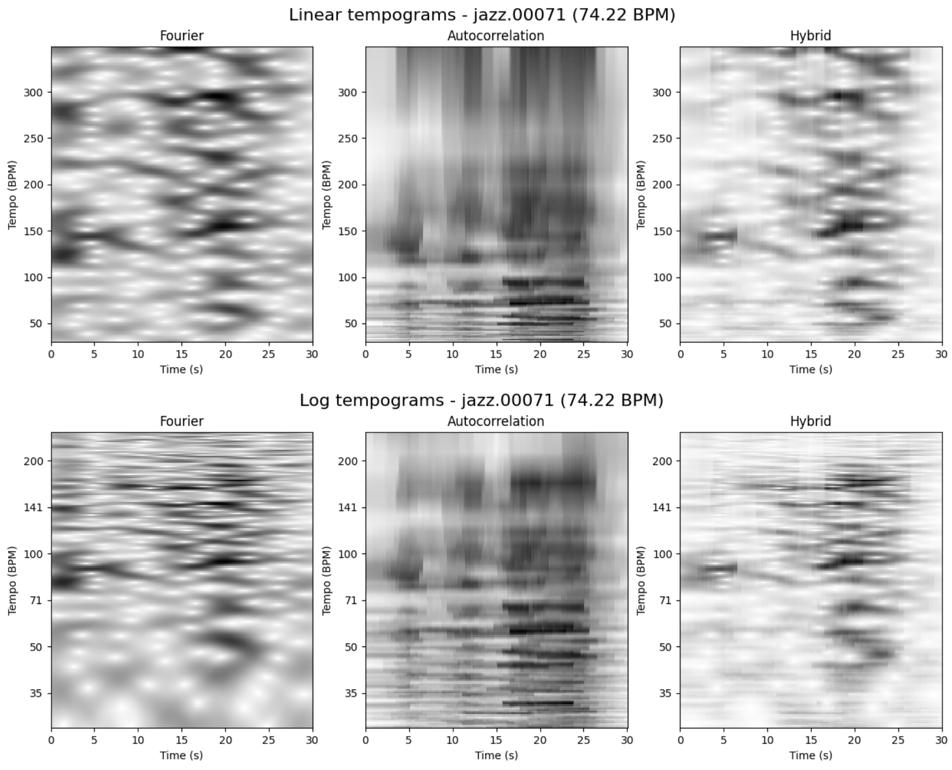


Figure 2.7: *Linear and log tempograms example for GTZAN jazz track*

$$\mathcal{T}_{\mathrm{F}}(n,\tau) := |\mathcal{F}(n,\tau/60)|. \tag{2.9}$$

The Fourier tempogram emphasizes tempo harmonics but suppresses tempo subharmonics.

Autocorrelation Tempogram

In the context of tempo estimation, the autocorrelation function is widely used to analyze and estimate local periodicities, so it is natural to derive a tempogram representation from this technique.

To calculate Autocorrelation tempogram $\mathcal{T}_{\mathrm{A}}$, we first need to calculate the windowed short-term autocorrelation $\mathcal{A}(n,\ell)$ of the novelty function Δ, as in Equation 2.10.

$$\mathcal{A}(n,\ell) = \frac{\sum_{m\in\mathbb{Z}} \Delta(m)w(m-n)\Delta(m-\ell)w(m-n-\ell)}{N+1-\ell} \tag{2.10}$$

The correspondence between lag and tempo is defined by 2.11.

$$\tau = \frac{60}{r\cdot\ell}\mathrm{BPM} \tag{2.11}$$

where r corresponds to the frame rate in seconds and $r\cdot\ell$ is the time lag of ℓ (in frames) that corresponds to $r\cdot\ell$.

A shift of $r\cdot\ell$ corresponds to a rate of $1/r\cdot\ell$ Hz, which corresponds to the relation shown in Equation 2.11.

We then define the autocorrelation tempogram $\mathcal{T}_{\mathrm{A}}$ by

$$\mathcal{T}_{\mathrm{A}}(n,\tau) = \mathcal{A}(n,\ell), \tag{2.12}$$

for each tempo $\tau = 60/(r\cdot\ell)$, $\ell \in [1, L]$.

There are a couple of things that we should highlight here:

- While ℓ is linear in the lag domain when we translate it to the tempo domain it is not. Therefore we need to interpolate the values to the tempo axis, which causes aliasing in higher frequencies (see Figures 2.4, 2.5, 2.6, 2.3, and 2.7).
- The lag ℓ represents an amount of periodicity, which is the inverse of frequency. Looking again at Equation 2.11 we can notice that harmonics that happen at $k\cdot\ell$ lags will correspond to $\frac{60}{r\cdot(k\cdot\ell)}$ BPM. When k gets bigger, the value of τ gets smaller. This is why we see subharmonics instead of harmonics for this type of tempograms.

Hybrid Tempogram

Even before the proposal of a hybrid tempogram itself, G. Peeters (2006) proposes the combination of temporal and spectral periodicities analysis in order to do pitch estimation.

As shown above, the octave uncertainty occurs in the inverse domain for the Fourier (harmonics) and Autocorrelation tempogram (sub-harmonics). The Hybrid Tempogram was proposed in Geoffroy Peeters (2006) and Geoffroy Peeters (2011) for several different tasks, such as pitch and tempo estimation, and rhythmic description.

Geoffroy Peeters (2011) shows different ways of creating Hybrid Tempograms. Such as the Product DFT and ACF (haDFTACF), the Product DFT/FM-ACF, Product TM-DFT/ACF, Product Hybrid Axis DFT/ACF. Each of them combines the Fourier tempogram and autocorrelation tempogram differently, for example, mapping frequencies from $\mathcal{T}_\mathrm{F}$ to lag-axis, mapping $\mathcal{T}_\mathrm{A}$ to the frequency axis or creating a hybrid axis and making the best use of the resolutions. We show a brief description of the combinations:

Product DFT and ACF (haDFTACF) The product DFT and ACF is obtained by multiplying the values of the two functions (DFT and ACF) after mapping one function to the domain of definition of the other: mapping the lags of the ACF to the frequencies of the DFT, or vice-versa.

Product DFT/FM-ACF FM-ACF stands for ”Frequency-Mapped ACF” because in this case we map every lag p to the frequency domain by using the fact that the lag p represents the amount of periodicity at the lag $l_p = p/sr$, where sr is the sampling rate. For this hybrid combination, we need to interpolate the resulting values to the linearly spaced frequencies f_k of the DFT. Now, both DFT and FM-ACF have the same frequencies f_k and the combination can be done by computing their product at each frequency.

Product TM-DFT/ACF Inverserly, we can map the DFT frequencies to the lag domain. The result is called ”Temporally Mapped DFT” (TM-DFT) and the combination of it with the ACF is done by computing the product at each lag l_p.

Hybrid Axis DFT/ACF (haDFTACF) This is the only combination that does not include a map between domains because this results in a loss of information and aliasing. The loss of information happens because the ACF resolution is smaller than the constant resolution of f_k, so it causes aliasing when interpolating the values to the frequencies. Instead of mapping the domains, this approach creates a new axis f_q made of the values of the lag axis mapped to the frequency domain $1/l_p$ for the lags $p < p_c$ and the values for the frequency axis f_k for $f_k > 1/l_{p_c}$. Both ACF and DFT are mapped to this new axis, interpolated at the new positions f_q, and then combined by computing the product at each frequency.

It is essential to point out that Product Hybrid Tempograms have aliasing embedded to it because of the frequencies and resolutions of the DFT and ACF calculations i.e. ACF has a better resolution on smaller frequencies while DFT has a better resolution on higher frequencies, thus when you translate the lag axis to a frequency domain, it is visible that the resolution gets bad.

The hybrid tempogram variation used in this work is the **Product DFT/FM-ACF**.

Cyclic Tempogram

There is a final way of calculating the tempogram which is the Cyclic Tempogram $\mathscr{T}_C$. The idea behind this approach is to have a representation that is more robust to harmonics, i.e. create tempo equivalences that differs by a power of two.

According to Grosche, Müller, *et al.* (2010) we say that two pitches having fundamental frequencies f_1 and f_2 are considered octave equivalent if $f_1 = 2^k f_2$ for some $k \in \mathbb{Z}$. Similarly, we say that two tempi τ_1 and τ_2 are *octave equivalent* if $\tau_1 = 2^k \tau_2$ for some $k \in \mathbb{Z}$. Therefore, one type of error that tempo estimation systems have to take into account is the so-called *octave error.*

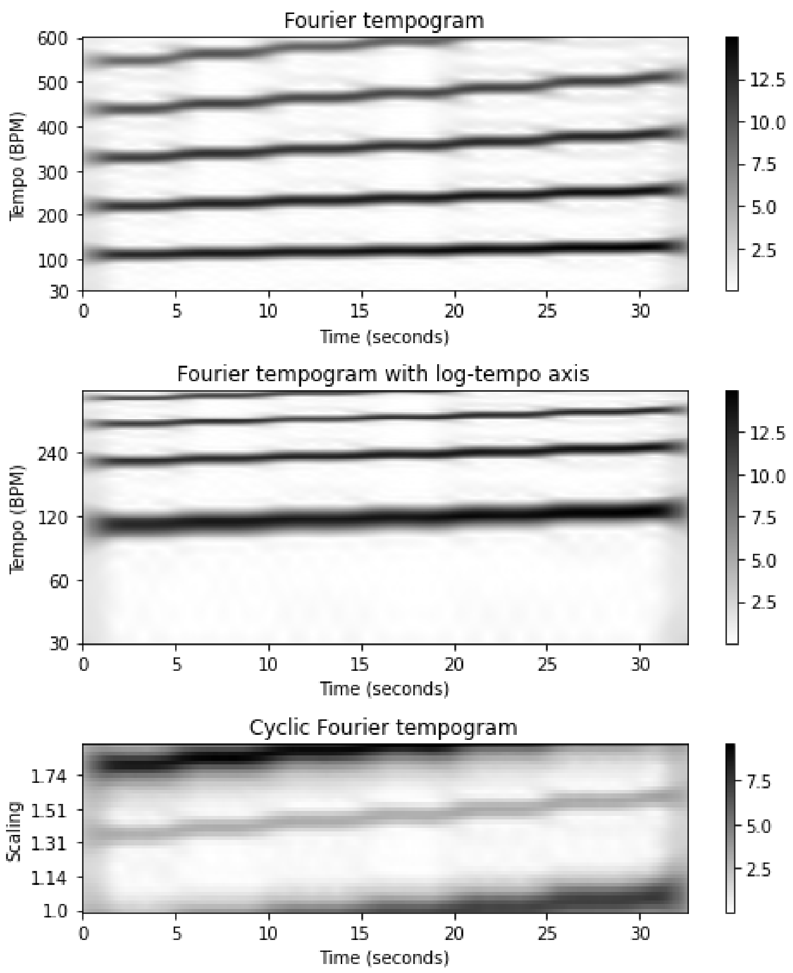


Figure 2.8: *Fourier tempogram, Fourier tempogram with log-axis and cyclic tempogram. Reproduced from Müller and Zalkow (2019).*

We will not explain the representation and its formulation in detail, but more context can be found in Grosche, Müller, *et al.* (2010). The authors also provide the "Tempogram Toolbox": a set of MATLAB implementations of tempo and pulse representations[3].

As shown in Thoshkahna *et al.* (2015), cyclic tempogram features can be used as the

[3] https://www.audiolabs-erlangen.de/resources/MIR/tempogramtoolbox

foundation to understand if a part of a song has or not a salient tempo, i.e. whether that part has a clear rhythm or a vague tempo.

Estimating tempo from the tempogram

There is a straightforward approach to estimating the global tempo of a piece of music given a tempogram $\mathscr{T}$: we can average the tempogram values over the time axis, resulting in a function $\mathscr{T}_{\text{Average}} : \Theta \rightarrow \mathbb{R}_{>0}$, defined by Equation 2.13, that has the salience values for each $\tau \in \Theta$.

$$\mathscr{T}_{\text{Average}}(\tau) = \frac{1}{N} \sum_{n \in [1:N]} \mathscr{T}(n, \tau) \tag{2.13}$$

To estimate the global tempo from $\mathscr{T}_{\text{Average}}$, one has to simply pick the maximum value for every τ.

$$\hat{\tau} = \max \mathscr{T}_{\text{Average}}(\tau) | \tau \in \Theta \tag{2.14}$$

This method is very simplistic and can be refined in a number of different ways, e.g. using the median instead of the average to have a more robust method against outliers.

2.2 Deep Learning and Audio

According to Choi *et al.* (2017) in 2010 there were only 2 deep learning articles at ISMIR conferences. This number increased to 6 in 2015 and 16 in 2016. Nowadays it is the most common approach to use, which followed a trend in other research areas such as Computer Vision and Natural Language Processing.

In conventional machine learning, one has to engineer features and then train a model that maps the labels to the input data. On the other hand, when using Deep Learning, every layer added is a set of features based on the input data, being able to learn more complicated relations between input and label.

Choi also explains that the success of deep learning in MIR can be explained by the fact that the research tasks can be subjective. For example, tempo and beat can be perceived differently by different people, therefore making it difficult to design features. Deep learning comes in handy when using a *data driven* end-to-end learning to achieve the desired goal.

There are clear tradeoffs between using or not deep learning approaches versus conventional machine learning approaches or even signal processing approaches. While deep learning allows better results and higher accuracy, these methods need a huge amount of data. There are several ways of approaching this specific problem, such as few-shot learning (Wang, Salamon, *et al.*, 2020; Wang, Stoller, *et al.*, 2022) (teaching the model to learn from a few examples), transfer learning (Takahashi and Barthet, 2022; Ou *et al.*, 2022) (fine-tuning a pre-trained model), data augmentation (Böck and Davies, 2020; C. Zhang *et al.*, 2021) and, more recently, self-supervision (see 2.2.2). On the other hand, signal processing methods are very fast, even if the accuracy is not the best, they are dependent of domain-specific knowledge to build the appropriate features for the appropriate task.

Purwins *et al.* (2019) offers an overview of deep learning for audio processing in general, comparing approaches for speech, music and environmental audio processing.

2.2.1 Deep Learning and Tempo Estimation

Researchers explored the use of deep learning in different steps of the tempo estimation pipeline (Figure 2.1) such as onset detection (Eyben *et al.*, 2010) or a mid-representation to analyze the periodicities and peak-picking (Widmer, 2013).

Böck, Krebs, *et al.* (2015) used a recurrent neural network to learn an intermediate beat-level representation of the signal and then processed it with a bank of resonating comb filters to detect the periodicities. The network used was a network that was the state-of-the-art for beat tracking at the time (Böck, 2011). This first work was able to outperform the state-of-the-art tempo estimation methods for most of the datasets used in the comparison for both ACC_1 and ACC_2 metrics.

Schreiber and Müller (2019) adapted a VGG-style network that was at first designed for key estimation to tempo estimation. A VGG network, or VGG model, is a deep convolutional neural network first proposed to image classification. The idea is to treat tempo estimation as a classification problem, so the network output an integer that is mapped to the tempo value, covering from 30 to 285 BPM.

Schreiber (2020) proposed a Convolutional Neural Network (CNN) that estimates the tempo value directly instead of a mid-level representation, such as the novelty function or the beat activation function, that needs to be further processed by other system components.

The work by Böck and Davies (2020), a follow-up of Böck, Davies, and Knees (2019), showed a great performance improvement for tempo estimation, especially looking at the ACC_1 metric. The key idea of the work is not to have a specialized tempo estimation model, but to have a multi-task model, responsible for tempo, beat and downbeat tracking. The core component is a deep neural network architecture based on dilated convolutions. The Temporal Convolutional Network (TCN) introduced by Davies and Böck (2019) was first proposed in Lea *et al.* (2016) in order to join in the same network architecture the capabilities of encoding spatiotemporal information locally (CNN) and also the capability of capturing high-level temporal relationships (RNN).

Unlike Schreiber, Zalkow, *et al.* (2020), the work by Böck and Davies (2020) does not output the tempo value directly, but instead a tempo activation function that has to be further processed to estimate the tempo value.

Sun *et al.* (2021) proposes a Convolutional Neural Network with an Attention Mechanism to estimate tempo. When compared to the specialized tempo model, it overcomes the performance of the competitors in the ACC_1 metric but falls behind Böck, Krebs, *et al.* (2015) in the ACC_2. In another experiment, the authors also try to reframe the problem as a multi-task problem and the performance has a similar performance to Böck and Davies (2020).

Changing the approach, Quinton (2022) proposed a self-supervised model to solve tempo estimation without the need for annotated data. His model is able to have a similar accuracy with unsupervised models but still did not reach the accuracy of the state-of-the-art systems.

Because tempo can be inferred from beat information, we also provide a non-exhaustive list of recent beat tracking methods that use deep learning in Table 2.1, inspired by Fuentes (2019, Chapter 3) and Matthew E. P. Davies (2021, Chapter 3). We omit downbeat tracking and online beat tracking methods because they are out of the scope of this work.

authors	approach		
	features	likelihood	post-proc
Böck (2011)	log STFT	BLSTMs	peak-pick
Böck, Krebs, *et al.* (2014)	mel log STFT	RNNs	DBN
Korzeniowski *et al.* (2014)	log STFT	BLSTMs	DBN
Böck, Krebs, *et al.* (2016)	mel log STFT	BLSTMs	DBN
Holzapfel and Grill (2016)	mel log STFT	CNN	DBN
Vogl *et al.* (2017)	log STFT	CRNNs	peak-pick
Cheng *et al.* (2018)	mel log STFT	RNNs	DBN
Fuentes, Maia, *et al.* (2019)	mel-STFT + ODF	RNNs	CRFs
Davies and Böck (2019)	log-STFT	TCN	DBN
Böck, Davies, and Knees (2019)	log-STFT	TCN	DBN
Böck and Davies (2020)	log-STFT	TCN	DBN
Steinmetz and Reiss (2021)	waveform	TCN	DBN
Pinto *et al.* (2021)	log-STFT	TCN	DBN
Chen and Su (2022)	mel log STFT	TCN	–

Table 2.1: *Overview of recent deep learning methods for beat tracking. The variants adopted at the different stages of the pipeline are indicated: input features (features), likelihood estimation (likelihood) and post-processing (post-proc).*

The acronyms used in the table:

- BLSTM: Bi-directional Long-Short Term Memory
- CNN: Convolutional Neural Network
- DBN: Dynamic Bayesian network
- LSTM: Long Short-Term Memory
- ODF: onset detection function
- TCN: Temporal Convolutional Network

According to Matthew E. P. Davies (2021), Probabilistic Graphic Models (PGMs), such as the Dynamic Bayesian Network, are the most used post-processing technique since 2010. The authors argue that this might be because these models offer a flexible way to incorporate musical knowledge and can be adapted to diverse music cultures.

There is some research on trying to estimate beats positions without the post-processing, such as Chen and Su (2022) and Steinmetz and Reiss (2021). The main reason is that deep learning is supposed to be an end-to-end method, i.e. the methods should discard as many human parametrizations as possible so the process is data-driven.

2.2.2 Self-Supervised Learning

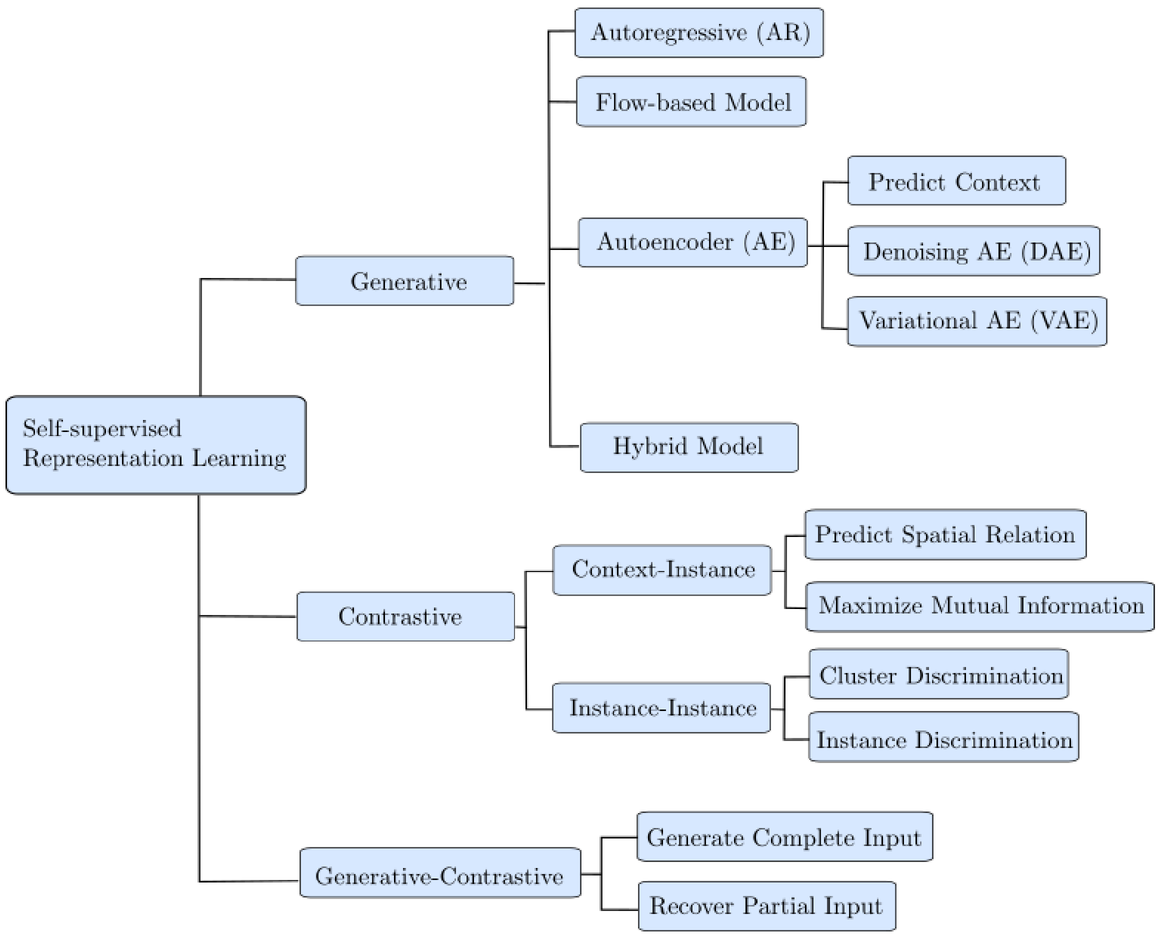


Figure 2.9: *Categorization of SSL methods (Balestriero et al., 2023)*

The use of deep learning increased a lot the accuracy of methods in different tasks, also introducing a new problem: data annotation. Deep learning approaches are data-hungry and annotated data is a problem in Music Information Retrieval. Dealing with music data involves copyright issues, it is time-consuming and expensive to annotate the data because someone with musical training is needed. Because of the reasons above, the models available right now are also biased toward the existing datasets.

Self-Supervised Learning (SSL) is a recent deep learning paradigm that aims to solve the lack of annotated data in some fields. The development of this the technique started with Computer Vision and more recently started to gain popularity in the audio-visual domain, in works such as Arandjelovic and Zisserman (2017).

The framework divides the deep learning problem into two stages: the *pretext task* and the *downstream task*. The pretext task is based on unlabeled inputs and aims to produce descriptive and intelligible representations (Goodfellow *et al.*, 2016), teaching the network some semantic information about the data. Balestriero *et al.* (2023) argues that the advantage of SSL when compared to supervised learning is that SSL aims to learn generic representations that may be useful across many different tasks.

According to Balestriero *et al.* (2023), pretext tasks can be grouped into three categories (detailed in Figure 2.9):

Generative: train an encoder to encode the input x into an explicit vector z and a decoder to reconstruct x from z;

Contrastive: train an encoder to encode input x into an explicit vector z to measure similarity;

Generative-Contrastive (Adversarial): train an encoder-decoder to generate fake samples and a discriminator to distinguish them from real samples.

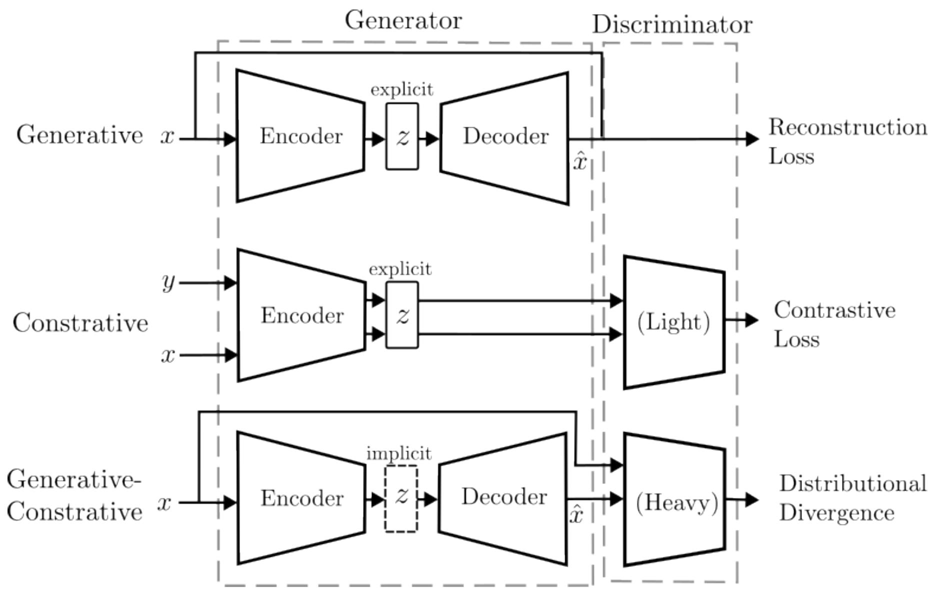


Figure 2.10: *Conceptual comparison between Generative, Contrastive and Generative-Contrastive methods (Balestriero et al., 2023).*

The creation of labels is usually done by hiding part of the data and trying to predict it by letting the rest visible, for example, one can remove part of a spectrogram and make the network predict the missing piece by showing its surroundings. Other examples of pretext tasks are: colorization (R. Zhang *et al.* (2016), image rotation (Pathak *et al.* (2017)), and audio and video correspondence (Arandjelovic and Zisserman (2017)). Once the network has learned the data representations, it can be used to solve the downstream task, with little or no re-training.

As briefly mentioned in Section 1.2, the applications of SSL in MIR tasks are new and come down to three works: Zero-Note Samba (Desblancs *et al.*, 2022) for beat tracking, SPICE (Gfeller *et al.*, 2020) for pitch estimation and more recently the work by Quinton (2022) for tempo estimation. We briefly review those works.

Zero-Note Samba

In Desblancs *et al.* (2022), the authors create two neural networks to process each part of the song: a percussive and a non-percussive. The pretext task, called syn-pred, tries to synchronize the samples. The input of the network is a variable-Q transform (VQT), i.e. a variant of the CQT with an improved time resolution in the low-frequency range. The motivation for this input is that important beat-tracking systems rely on information of frequencies below 100 Hz. In total, the model was trained with 98 hours of audio.

This contrastive learning approach (second model of Figure **fig:ssl_methods**) uses the non-percussive part as the anchor and the percussive part as the positive sample. The negative part of this scenario is a randomly lagged version of the percussive part. The full framework is described in Figure 2.11.

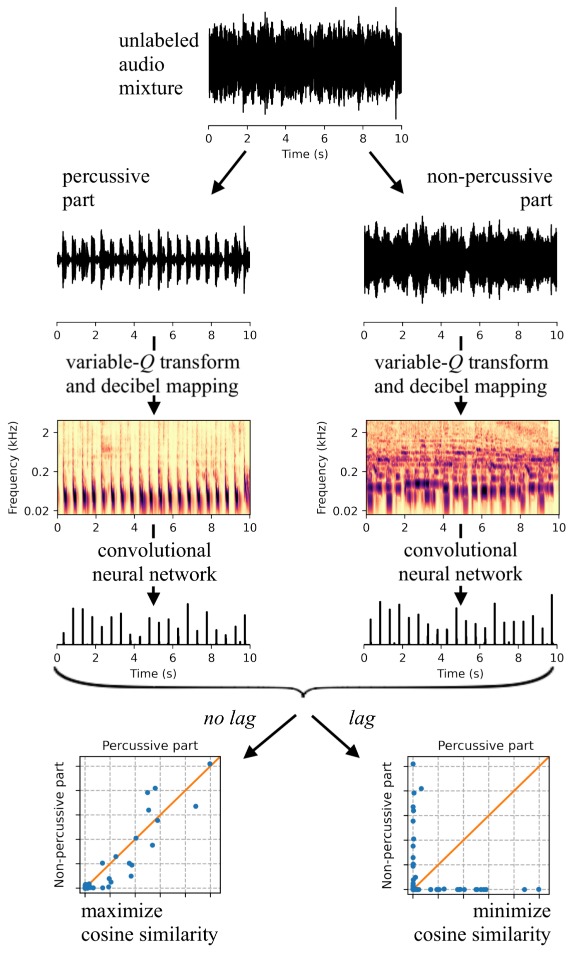

unlabeled audio mixture
Time (s)
percussive part
non-percussive part
variable-Q transform and decibel mapping
variable-Q transform and decibel mapping
Frequency (kHz)
convolutional neural network
convolutional neural network
Time (s)
no lag
lag
Percussive part
Non-percussive part
maximize cosine similarity
minimize cosine similarity

Figure 2.11: *Zero-Note Samba (Desblancs et al., 2022) workflow.*

Following the approaches of methods such as Böck, Davies, and Knees (2019), Davies and Böck (2019), and Böck and Davies (2020), the Zero-Note Samba (ZeroNS) model does not output the beat positions, but instead, output a beat activation function, which has to be further processed by a Dynamic Bayesian Network (DBN) to track the beat positions.

The model is not able to overcome the state-of-the-art for supervised beat tracking, but it surpasses unsupervised methods the trained model has an interesting capability of being fine-tuned with few data (only 1 - 24 tracks are required) and still reach competitive performance.

Although this work is highly promising, it relies on computationally expensive sound source separation using Spleeter (Hennequin *et al.*, 2020)[4] to obtain the percussive and non-percussive parts of the signal, which increases training time and can add noise and artifacts to the training data. The code for Zero-Note Samba is freely available on GitHub[5].

Equivariant Self-Supervised Tempo Estimation

Quinton (2022) also proposes a contrastive learning approach to tempo estimation. In this case, he uses two branches that will receive a time-stretched version of the signal.

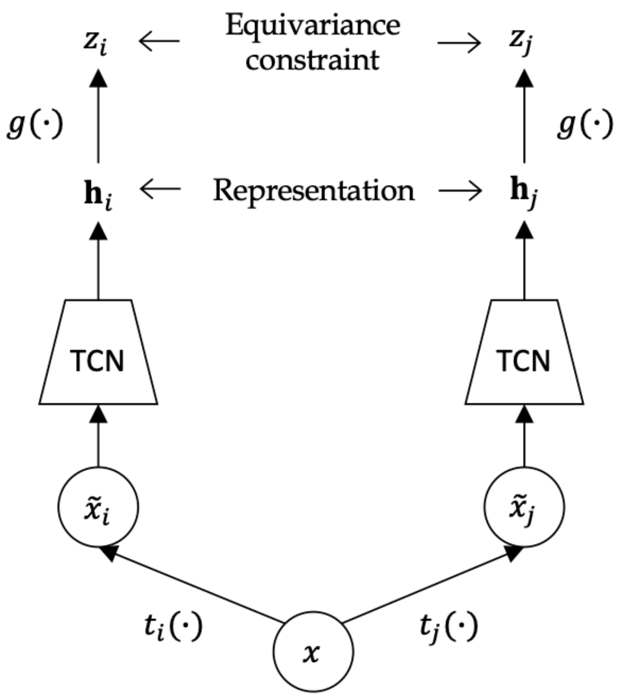


Figure 2.12: *Quinton (2022) workflow.*

[4] https://github.com/deezer/spleeter

[5] https://github.com/deezer/zeroNoteSamba

In the contrastive learning framework, two views are generated by applying random data augmentations so it can be compared to the *anchor*, or the positive value. The training objective then constrains the model to produce representations that are invariant to the augmentations applied to the training data and yet discriminative between different samples. What Quinton (2022) proposes to use the equivariant constraint to learn audio representations that capture tempo information.

The network is trained with 25k tracks and no tempo annotation. Once the network is trained, the projection head is discarded, the weights frozen and a linear classification head is attached.

The network used in this work is based on Böck and Davies (2020) work, which uses the Temporal Convolutional Network (TCN). This network started to be used because it showed robustness to find temporal patterns. So Quinton adapted the architecture proposed by Bock and Davies and removed the beat tracking branches from it, resulting in only in the tempo branch, which creates an embedding with temporal information.

The code is available on GitHub[6] and includes a pre-trained model.

SPICE

SPICE (Gfeller *et al.*, 2020) is a self-supervised pitch estimation approach. The core idea of the algorithm relies on the fact that a pitch shift maps to a simple translation when the audio signal is analyzed through the constant-Q, transform. Also, it is perceptually easier to find the relative pitch difference rather than the absolute difference.

A CQT is a representation first proposed by Brown (1991), to address the fact that the linear spacing between DFT frequencies yields components that do not map efficiently to musical frequencies. Therefore the CQT takes into account the logarithmic nature of perception. and spreads the center frequencies in.

The representation proposed in Brown (1991) is equivalent to a 1/24th-oct bank of filters. The idea is that frequency components correspond to the quarter-tone spacing of the equal-tempered scale.

The authors proposed a pretext task that learns how to encode the pitch difference between two CQT-shifted frames. Each network branch encodes one of the frames to a single scalar y, then reconstructs the input x from it.

The shift k plays an important role. Each slice can be randomly shifted by a integer number between $[0, 8]$, which translates to 0 to 4 semitones. The reason why this value translates to a 4 semitones shift when $k = 8$ is because the CQT, as mentioned before, does not have a linear spacing between the DFT frequencies, yet a logarithmic spacing.

Their method was able to estimate the relative pitch, and also to provide confidence in the estimation. SPICE was able to achieve similar accuracy to the fully-supervised methods but with the advantage of not using a labeled dataset.

[6] https://github.com/Quint-e/equivariant-self-supervision-tempo

SPICE source code and training data are not available, but the trained model can be downloaded and used via the Tensorflow Hub[7].

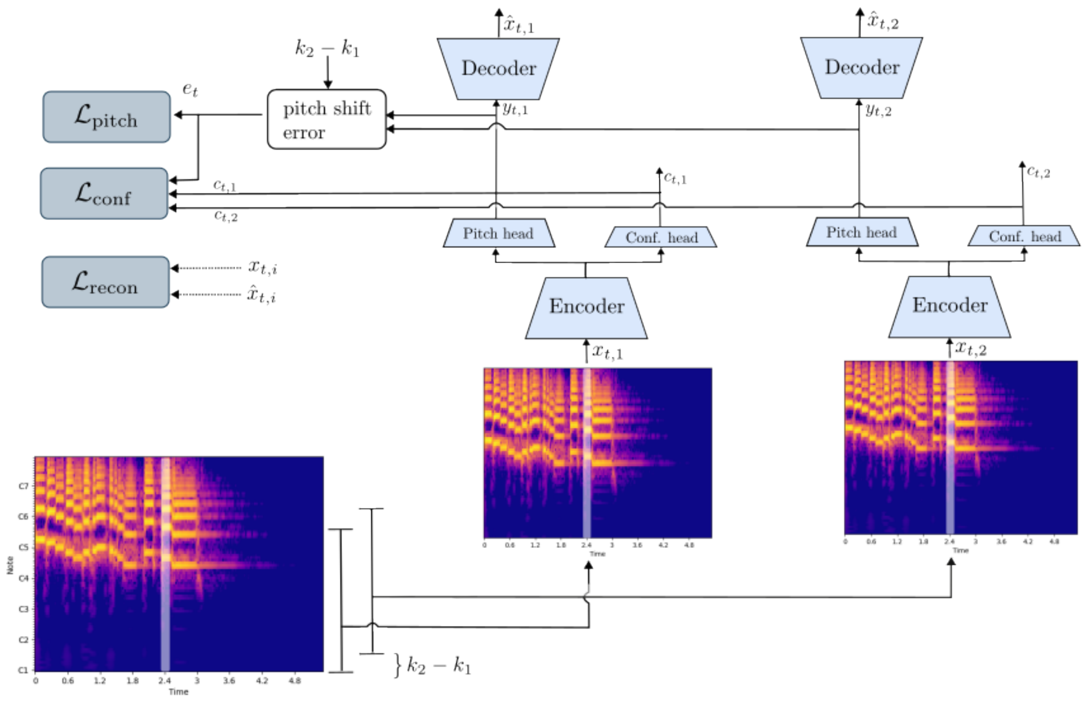


Figure 2.13: *Gfeller et al. (2020) architecture*

Unline ZeroNS and the Equivariant Tempo Estimation works, SPICE pretext task is not contrastive, i.e. it does not compare an anchor with positive and negative samples, but instead generative, meaning that the pretext task is based on the reconstruction of the input by using only the embedding outputted by the encoder.

SPICE code is not available, but the trained model is at TensorflowHub[8], along with a tutorial[9], a demonstration video of estimations of the model provided[10] and a short blog post[11].

[7] https://www.tensorflow.org/hub/tutorials/spice

[8] https://tfhub.dev/google/spice/2

[9] https://www.tensorflow.org/hub/tutorials/spice

[10] https://www.youtube.com/watch?v=pxIUtubtTws

[11] https://ai.googleblog.com/2019/11/spice-self-supervised-pitch-estimation.html

Chapter 3

Contributions

Between the contributions of the present work, we can list a poster presentation at the KHIPU - the Latin America Meeting in Artificial Intelligence[1], and a published paper and a poster presentation at the 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2023)[2] (Morais *et al.*, 2023).

3.1 Methodology

3.1.1 Framework

We first adapt the SPICE model to tempo estimation by following the information provided in Gfeller *et al.* (2020).

The framework described is all implemented in Python with the support of the libraries Keras (*Keras: The Python Deep Learning API* 2023)[3] and Tensorflow (Abadi *et al.*, 2016)[4] for machine learning and librosa (McFee, McVicar, *et al.*, 2023)[5] for signal manipulation. The implementation is open-source and hosted on GitHub[6].

Given a tempogram $\mathscr{T}$, our method extracts two random slices x_1 and x_2 which correspond to the same time instant, but are shifted vertically by k_1 and k_2 bins, respectively. These shifts translate to artificial changes in tempo in the different slices fed to the model, as the lines of the tempogram are vertically displaced. We sample the values k_1 and k_2 from the uniform distribution $\mathcal{U}(11, ..., 18)$, so the tempo range seen by the model ranges from 30 BPM to 310 BPM.

The model is then trained to estimate the relative tempo within each slice using the following loss function:

[1] https://khipu.ai/

[2] https://2023.ieeeicassp.org/

[3] https://keras.io/

[4] https://www.tensorflow.org/

[5] https://librosa.org

[6] https://github.com/giovana-morais/steme

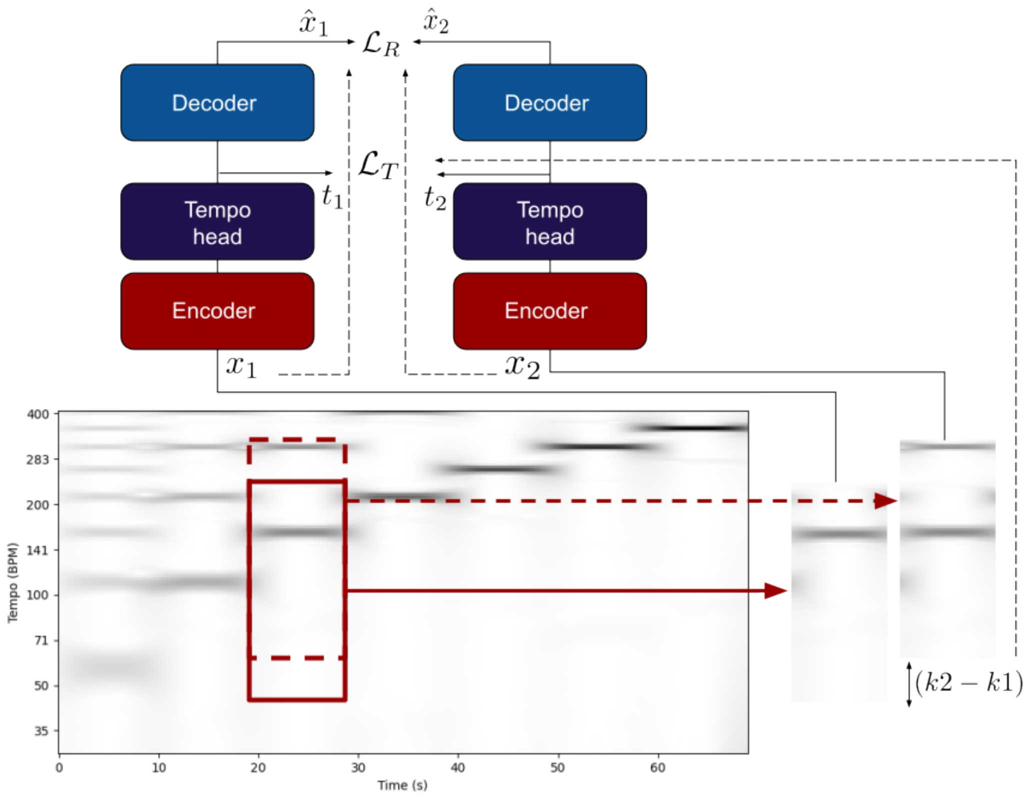


Figure 3.1: *Overview of the training framework.*

$$\mathcal{L}_T = |(t_1 - t_2) - \sigma(k_2 - k_1)|, \tag{3.1}$$

so the difference between estimated tempi in each branch has to be proportional to the artificially introduced difference $(k_2 - k_1)$. The constant σ ensures that the difference in the output model $t_1 - t_2$ corresponds to the right amount in bins within the training set, and is set to

$$\sigma = \frac{1}{Q \log_2\left(\frac{t_{\max}}{t_{\min}}\right)}, \tag{3.2}$$

as in Gfeller *et al.* (2020), where $t_{\max}$ and $t_{\min}$ are the maximum and minimum tempi present in the training set and Q is the number of bins per tempo octave. We set $Q = 40$ to ensure a reasonable tempo range and resolution, e.g. three tempo octaves 30-240 BPM correspond to 160 bins. Unlike Gfeller *et al.* (2020), we observed that $k_2 - k_1$ (instead of $k_1 - k_2$) preserves the monotonic correspondence between BPM and model output values, as higher shifts correspond to tempogram slices with lower salience lines, and thus lower tempi. Besides the loss in Equation 3.1, analogously to Gfeller *et al.* (2020) we use a reconstruction loss as follows:

$$\mathscr{L}_R = \frac{1}{T}\sum_T \|x_1 - \hat{x}_1\|_2^2 + \|x_2 - \hat{x}_2\|_2^2, \tag{3.3}$$

where x_1, x_2 are the input slices to each model branch and $\hat{x}_1, \hat{x}_2$ are the reconstructed slices at the output of the decoder in Fig. 3.1. The final combined loss is given by:

$$\mathscr{L} = \omega_T \mathscr{L}_T + \omega_R \mathscr{L}_R, \tag{3.4}$$

with $\omega_T = 10^4$ and $\omega_R = 1$.

We use a 6-layered convolutional encoder that receives a 128-dimensional vector corresponding to a slice of the tempogram and outputs one scalar representing tempo. We use filters of size 3 and stride equal to 2, and the number of channels is equal to $d \cdot [1, 2, 4, 8, 8, 8]$, where $d = 64$. The output of the last convolutional layer is flattened and fed into a tempo estimation head that consists of two fully connected layers with 48 and 1 units respectively. During training, the tempogram frames of the input audio are shuffled to ensure that, for a given batch, different tempos are seen. The scalar output of the encoder is then fed into a decoder which has the objective of reconstructing the tempogram slice from this tempo estimation. The decoder also has 6 layers as $d \cdot [8, 8, 8, 4, 2, 1]$.

There are a few, but important differences between the architecture proposed in this work and the architecture of SPICE, such as

- we do not have a confidence/voicing head,
- we do not make use of BatchNormalization or MaxPooling layers,
- our filter values d are the same for both encoder and decoder.

These changes are due to the results of informal testing that evaluated the capacity of the model of reconstructing tempogram frames.

3.1.2 Tempogram parameters

Once the model was implemented, we started to investigate the impact with the input itself. At the beginning, we decided to use linear tempograms, which were the most straightforward implementation and would require almost no effort for using it. When we started to analyze the input, we noticed a problem: the shift we were using was not enough for us to cover the whole tempo range we needed.

The main idea of the SPICE formulation is that we analyze and compare shifted versions of the same input. The shifts that are used in their method work because the CQT has a log axis instead of a linear axis.

Suppose that we calculate the tempograms with $\theta \in [30, 300]$ BPM and we want to use the same shifts as SPICE, i.e. $k \in [0, 8]$.

We defined an input slice as a 128-dimensional vector. Therefore, a 0-sample shift means that we have no shift at all, so we just take the 128 first positions of our tempogram

T. This means that the slice we input in our model covers from 30 BPM to 158 BPM (30 + 128 positions) (blue rectangle on Figure 3.2).

Then, if the biggest k value we can have is 8, then our 128-dimensional vector will cover from 38 BPM (30 + 8-position shift) to 166 BPM (38 + 128 positions) (red rectangle on Figure 3.2).

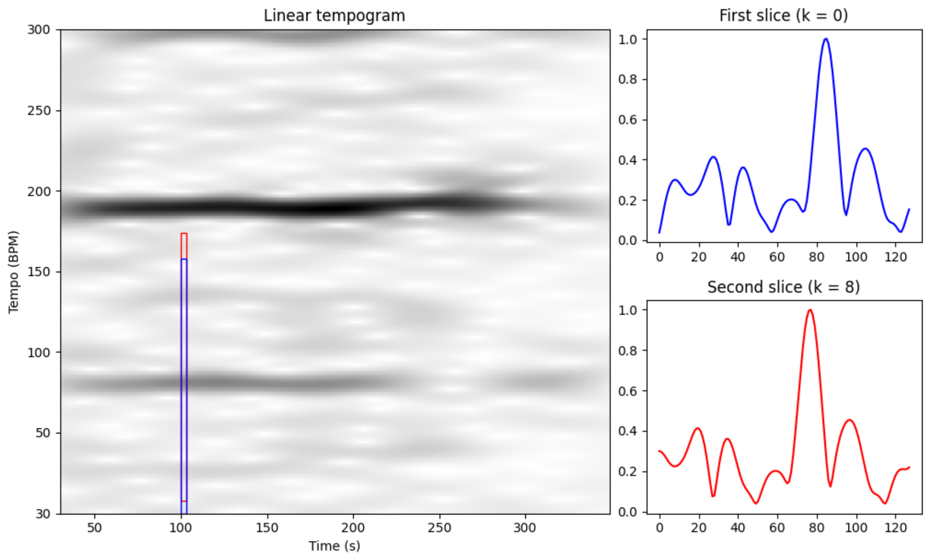


Figure 3.2: *Model input. The left image shows a linear tempogram for GTZAN track "metal.00070", which has an annotated tempo of 116 BPM. Each rectangle is a slice that will be inputted to the network. Here for example purposes we get the minimum possible shift ($k = 0$) as the blue rectangle and the maximum possible shift ($k = 1$) as the red rectangle.*

This means that if a song has an annotated tempo higher than 166 BPM, the model will not be able to learn from that song because no shift would be able to reach the region of the tempogram that actually has the tempo information we need.

One could argue that the solution would be to increase the shift, but Figure 3.3 shows what kind of problem this may introduce. If the shift is too big, one of the slices could have meaningful tempo information while the other might only have harmonic information.

To dig deeper into the discussion of the role of the harmonics and subharmonics in this scenario we would need further experiments, but the first informal results we had were not satisfactory, therefore we decided to choose a log-tempo axis instead of a linear-tempo axis. The reasoning is that the in the log axis the shifts can cover the a whole range of interests.

3.1.3 Datasets

To understand how the distribution of the data affects the self-supervised model performance, representations of Section 2.1.4 and the distribution of the data, we synthesize metronome excerpts ($N = 1000$) following four different distributions[7]. For three,

[7] The click track implementation is available on B.2

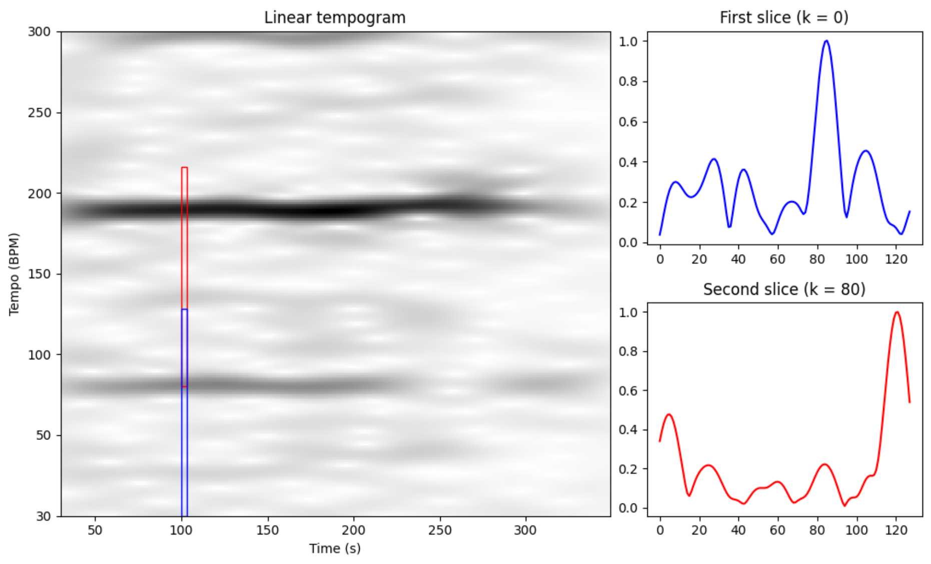


Figure 3.3: *Model input. The left image shows a linear tempogram for GTZAN track "metal.00070". Each rectangle is a slice that will be inputted into the network. In this scenario, the red slice would only have the harmonic information because the shift is too big, while the blue slice would miss the harmonic information because it only covers from 30 to 158 BPM.*

we model tempo as a *log-normal* random variable X such that $\log_2(X) \sim \mathcal{N}(\mu, \sigma^2)$, with $\mu \in \{70, 120, 170\}$ BPM and $\sigma = 0.25$. These three distributions are modeled after real-world datasets (e.g. GTZAN Tzanetakis and Cook (2002) and Marchand *et al.* (2015)) shifted to different tempo ranges. We also include a *log-uniform* distribution where $\log_2(X) \sim \mathcal{U}([30, 240])$, representing a well-balanced ideal tempo distribution, which is difficult to find in practice but serves as an upper-limit reference for our experiments.

The lognormal datasets were created by using SciPy lognormal function[8], with the parameters shown in Table 3.1.

	s	loc	scale	size	random_state
lognorm @ 70	0.25	30	50	1000	42
lognorm @ 120	0.25	70	50	1000	42
lognorm @ 170	0.25	120	50	1000	42

Table 3.1: *SciPy's function parameters*

Besides the synthetic datasets presented, we also tested our model with real data by using the GTZAN dataset (Tzanetakis and Cook, 2002; Marchand *et al.*, 2015). GTZAN is a dataset that was first created for genre classification, but it also received tempo annotations (Marchand *et al.*, 2015). It is composed of 1000 audio music excerpts of 30 seconds duration. It is divided into 10 genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock, with 100 tracks each. The tracks are named with a pat-

[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.lognorm.html

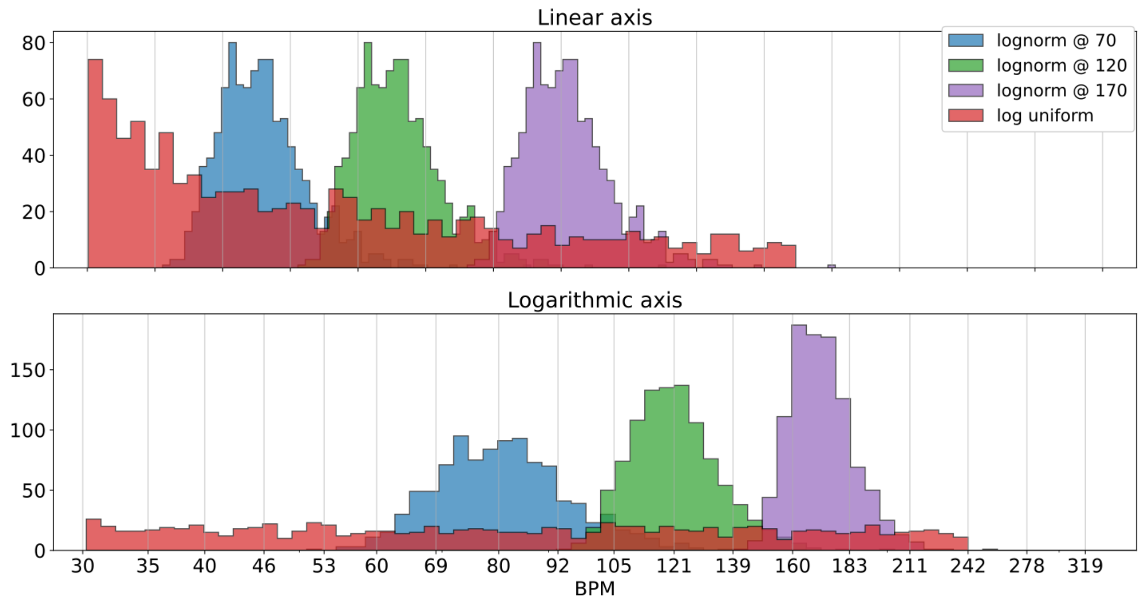


Figure 3.4: *Distributions of the generated synthetic datasets.*

tern "genre.number", e.g. "country.00001" and "metal.00100". Even though there are 1000 tracks available for GTZAN, we discard one that misses annotations (reggae.00086).

An analysis made by Sturm (2012) showed that at least 100 published papers used the GTZAN dataset. The rhythmic annotations of the GTZAN, made by Marchand *et al.* (2015) were made semi-automatically: first the beats and downbeats were automatically tracked by a system and then corrected by the human annotators when necessary. A comprehensive explanation of the dataset and explanation of disagreements of annotations is made by Quinton (2017).

In more practical terms, we use the mirdata library to load GTZAN tracks and tempo information. mirdata is an open-source Python library aimed to foster the reproducible use of MIR datasets available on GitHub[9]. We use the latest version, namely v.0.3.0 (Fuentes, Bittner, *et al.*, 2021). It is important to notice that GTZAN is not a public dataset anymore [10].

Figure 3.5 shows the datasets tempi distributions including GTZAN.

The first row shows the synthetic datasets distributions in a linear axis. The second row shows GTZAN dataset distribution is also in a linear axis. Finally, the third row shows the four distributions in the logarithmic axis.

3.1.4 Tempogram variations

As explained in 2.1.4, there are three variations of the tempogram: Fourier Tempogram ($\mathscr{T}_F$), Autocorrelation Tempogram ($\mathscr{T}_A$) and Hybrid Tempogram ($\mathscr{T}_H$). We want to compare the effects of harmonics, sub-harmonics and no harmonics in the learning process, so

[9] https://github.com/mir-dataset-loaders/mirdata

[10] https://github.com/mir-dataset-loaders/mirdata/issues/549

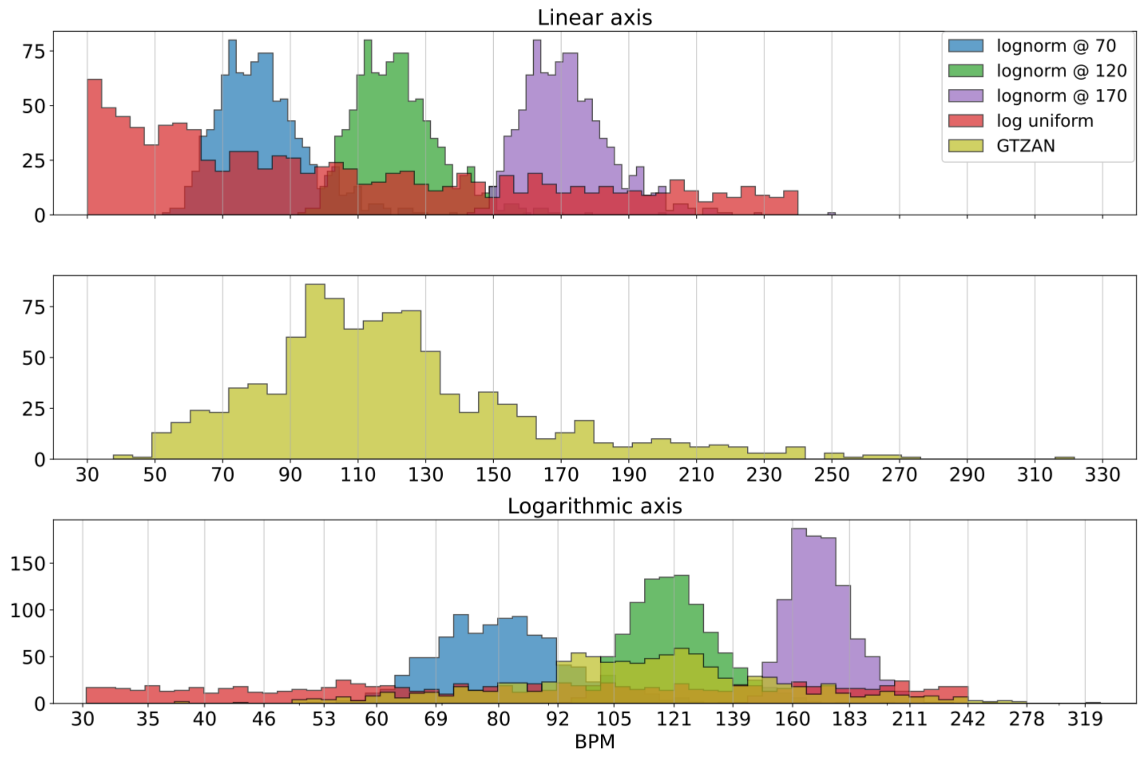


Figure 3.5: *Distributions of the used datasets*

for each track in the datasets (synthetic and real-world) we compute the three variations to use as input data.

The novelty function used to create all three variations is the spectral flux (Equation 2.4) with the following parameters: FFT window size of 2048 samples, Hann window function and hop length (H) of 512 samples. We use the STFT implementation provided by librosa[11]. We apply the logarithmic compression (Equation 2.2) with $\gamma = 100$ and pos-process the novelty function by subtracting the local average, considering a window of 10 samples, and normalizing it. The implementation for the spectral flux is provided in Appendix B.3.

Changes in frequency are perceived differently depending on the frequency range they happen, e.g. an increase of 40 Hz from 40 Hz to 80 Hz is perceived as an octave, but an increase of 40 Hz from 120 Hz to 160 Hz is perceived as a perfect fourth. Similarly, tempo changes are perceived relative to their speed (Grosche, Müller, *et al.*, 2010), so an increase of 30 beats per minute (BPM, BPM) from 60 BPM to 90 BPM is perceived as a bigger change than 130 BPM to 160 BPM. With this in mind, we convert the tempo axis of the tempogram representations from a linear to a logarithmic scale, by combining the linear bins into logarithmic ones centered at

$$t_k = t_0 2^{\frac{k}{Q}}, \tag{3.5}$$

[11] https://librosa.org/doc/latest/generated/librosa.stft.html

with $t_0 = 25$ BPM and $Q = 40$ bins per octave. This bears a resemblance to how a CQT (which has a log-frequency axis) can be derived from a (linear-frequency) spectrogram.

Tempograms implementations based on MÜLLER and ZALKOW (2021) are provided in Appendix B.4.

3.1.5 Model calibration

The output of the tempo head is a real-valued scalar $t \in [0, 1]$, which indicates the estimated tempo within a given slice. To perform inference with this model, we need to map this output to a BPM range using a calibration step (GFELLER *et al.*, 2020). Considering the logarithmic nature of the tempogram, given a linear range of BPM values, the model output will have a logarithmic distribution of values, with the values for higher tempi being squeezed into closer activation values of t. On this basis, we generate synthetic metronome clicks following a logarithmic tempo range and map the model to BPM with a linear model.

3.2 Results and discussions

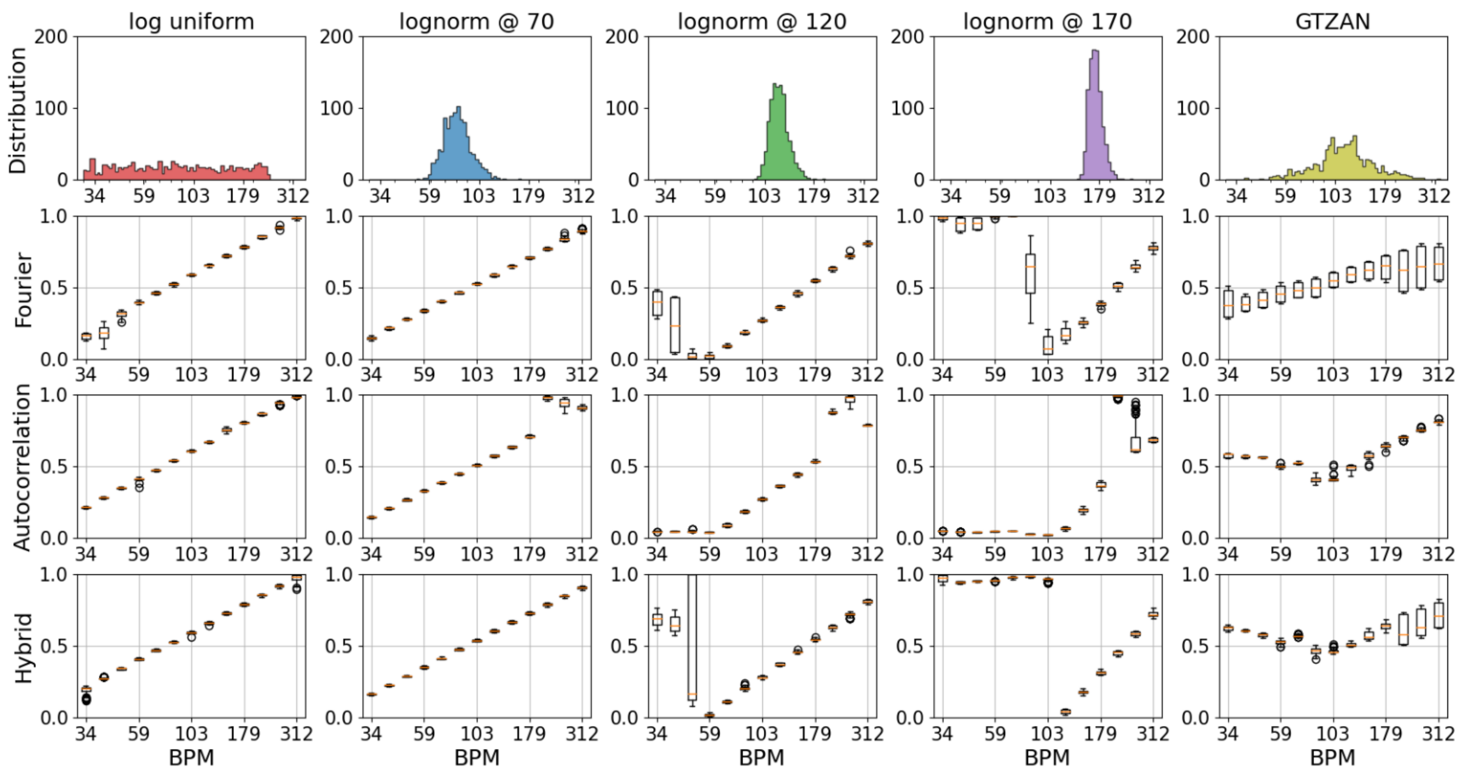


Figure 3.6: *Calibration results.*

Since calibration results correlate with model performance, i.e. a model with more linear and less disperse calibration curves leads to more consistent estimations GFELLER *et al.* (2020), we look at calibration results for each model, depicted in Figure 3.6. The first row reflects the distributions that we tested in training. The three lines below are the calibration results for Fourier, Autocorrelation, and Hybrid tempograms variations. These results are published in Morais *et al.* (2023).

3.2.1 Data distributions

The experiments with the different data distributions show that the model does not benefit from the wider range of a log-uniform distribution, and instead can extrapolate from a log-normal distribution centered in the lower tempi end (70 BPM). When we look at this result closely, an explanation may be found in Figure 3.5. Given a tempo distribution expressed in linear BPM values (Figure 3.5 top), the model will be fed a slightly different tempo distribution (Figure 3.5 bottom) because of the logarithmic effect of the input tempogram. This logarithmic representation compresses the tempo range increasingly in higher tempi, meaning that the model has access to a smaller range of log BPM values. In more extreme cases, such as the log-normal distribution centered at 170 BPM, the model learns to use the whole range of its output for the rather small range of tempo values it sees, and "saturates" for values out of range, i.e. it assigns the same value to all unseen tempi. This suggests that it would be more beneficial to annotate or augment data in this lower range of tempi to ensure a more stable model.

Finally, when we look at the results for the model trained with real data, we see that the same observations hold. The Fourier tempogram leads to a more stable model, which shows a larger variance than models trained with synthetic data due to the inherent variability and noisiness of real data which complicates training. The GTZAN distribution has similarities with the lower-BPM log-normal distributions, which shows in the smooth trend of Figure 3.6 with the Fourier tempogram representation. For the GTZAN data, the hybrid tempogram also does worse than the other two variations, which suggests that we might introduce artifacts and noise by combining the two tempograms without further processing of the signal, as done in Geoffroy Peeters (2006), even though this is enough for synthetic data given its simplicity and steadiness.

3.2.2 Tempogram variations

The Fourier tempogram shows more consistent calibration results across data distributions and leads to smoother curves for both synthetic and real data. This result aligns with the intuition that the Fourier tempogram is the most similar to the CQT representation in Gfeller *et al.* (2020) as it tends to have upper harmonics but not sub-harmonics. On the contrary, the autocorrelation tempogram results present the most variation, meaning that for similar tempo values, the model struggles to assign the same tempo output. Surprisingly, the hybrid tempogram does worse than the Fourier tempogram, although it is supposed to be a more consistent representation. This suggests that the multiplication of the autocorrelation and Fourier tempograms removes harmonic information that the model relies on to perform tempo estimation. This multiplication also may introduce noise when we are using real data. In order to reduce this noise, Geoffroy Peeters (2011) proposes several methods of creating hybrid tempograms, such as creating a hybrid-axis tempograms that makes the best use of both Fourier and ACF resolutions.

3.2.3 Data augmentation

The first augmentation proposed is an augmentation based on the audio, i.e. a time-stretching algorithm. We decided to use Rubberband[12], a C++ library that has a Python wrapper (pyrubberband[13]). We chose the finest configuration in order to preserve the audio quality.

The target distribution to the augmentation was the lognorm @ 70, because we've seen in previous results that even though this distribution is not uniform, the model is able to extrapolate to unseen tempi. Figure 3.7 shows the difference between distributions before and after augmentation and Figure 3.8 shows Fourier tempograms for the original audio and the augmented audio.

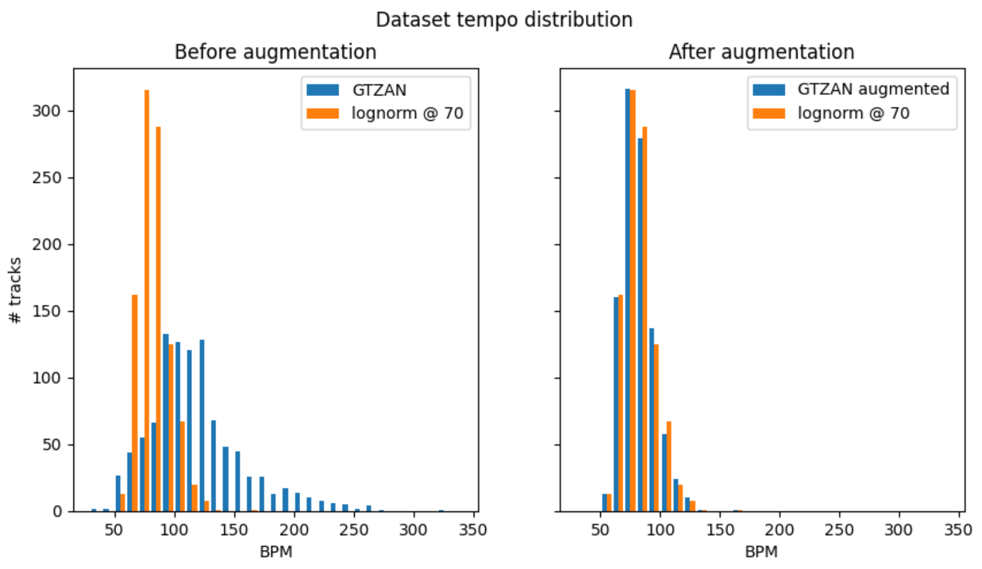


Figure 3.7: *Dataset distributions before and after data augmentation. The lognorm @ 70 distribution was the target distribution for the augmentation.*

In Figure 3.7 we can see that after augmentation the number of tracks was very similar. Still, because we are doing a time-stretching operation and slowing most of the tracks, they are bigger than the original. For example, if we slow a 30s track by half the tempo, the resulting augmented track has 60s. To have a fair comparison, we created two experiments: one considering the full dataset and the other cropping the augmented audios to 30 seconds to have a "fair" comparison.

Finally, we analyze all of the EarlyStopping parameter in the training. For this experiment, we want to understand if the distribution type affects the model convergence. The EarlyStopping parameter had a patience of 1 epoch, meaning that if the validation loss does not decrease in 1 epoch, the training is stopped and the best model is saved.

The grouped results of the experiments can be seen in Figures 3.9 and 3.10. For readability purposes, we show in those figures only the results for GTZAN. The full plot with

[12] https://breakfastquay.com/rubberband/

[13] https://pyrubberband.readthedocs.io/en/stable/index.html

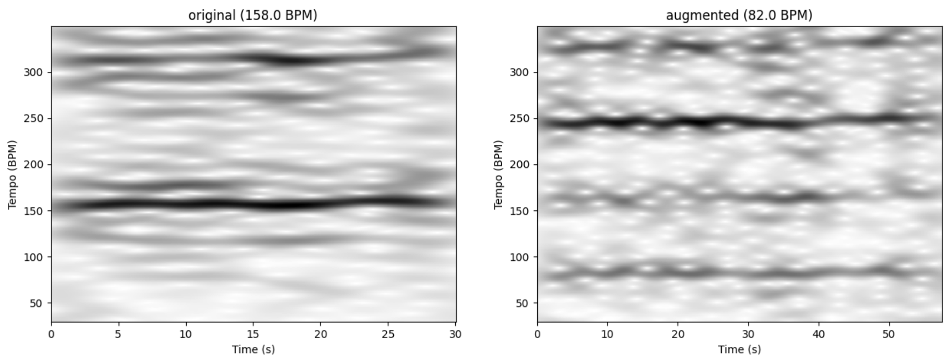


Figure 3.8: *Comparison between tempograms before and after augmentation via Rubberband. On the left, we have the tempogram for the track "blues.00002". In the right column, the tempogram after the rubberband's augmentation.*

all of the synthetic distributions can be found in Apendix A.2.

Figures 3.9 and 3.10 have unexpected results. Unlike Figure 3.6, the Fourier tempogram has a poor result for the GTZAN dataset, while the autocorrelation variation has a very good calibration performance and only a little variation. This is exactly the opposite of what we have reported in Morais *et al.*, 2023. One way of looking at and interpreting these results is to remember that the autocorrelation has sub-harmonics and, because of the log-axis of the tempogram, we have a high resolution in lower tempi than in higher tempi. So the information of the subharmonics, that can hint the tempo value, is in a tempogram region that the model can discriminate between very small tempi variations. In any case, this does not explain why the Fourier approach suddenly had this decline in performance when the training setup had not changed.

The way to investigate these discrepancies is to look at the training and validation data, and the model itself that is stored and will be available on GitHub in a separate repository for reproducibility purposes. Because the train/validation split is randomly selected at the data generation step, this could be something that affects the performance of the model.

3.3 Future work

3.3.1 Data augmentation

There are three ways of augmenting our dataset.

Augmenting by time-streching In this scenario, we change the dataset tracks tempi until we have a distribution that matches our needs. The downside of this approach is that the algorithm responsible for time-stretching can have artifacts that the model can use to cheat.

Augmenting by combining datasets We combine datasets and then remove the tracks that do not belong to the desired distribution. This might reduce the number of artifacts that the time stretch algorithm creates, but it is not certain that time-stretching

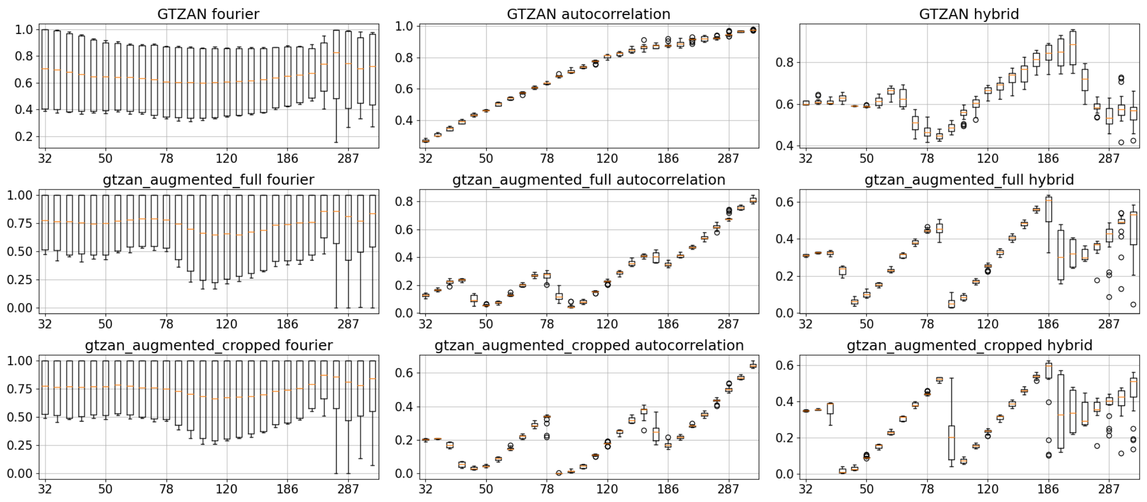


Figure 3.9: *Experiments with the EarlyStopping as True. Each row represents a training data distribution and each column represents a tempogram variation. The left column stands for Fourier tempograms, the middle column autocorrelation tempogram and the right column hybrid tempograms.*

could be 100% avoided. It heavily depends on the aimed distribution.

Augmenting on the tempogram domain In this scenario, we do the augmentation on the tempogram itself instead of the audio. This in theory allows us to avoid artifacts related to the time-stretching algorithm. The downside is that we cannot reconstruct the audio from the tempogram, so we might lose some information to inspect the samples qualitatively.

In this work, we explored option 1, but the effects of options 2 and 3 in the final result remain open.

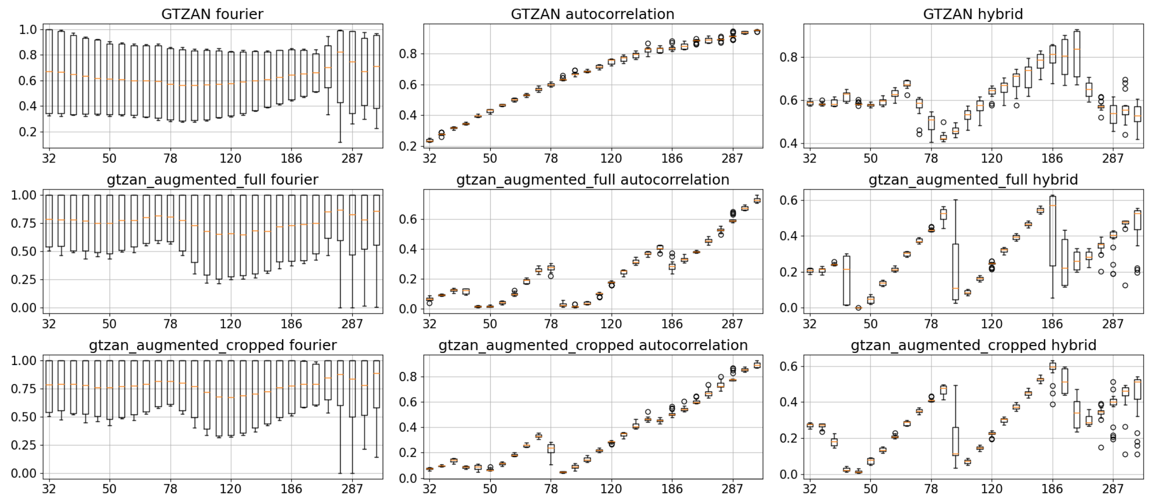


Figure 3.10: *Experiments with the EarlyStopping as False. Each row represents a training data distribution and each column represents a tempogram variation. The left column stands for Fourier tempograms, the middle column autocorrelation tempogram and the right column hybrid tempograms.*

Chapter 4

Conclusions

In this work, we explored the possibility to adapt a self-supervised model to tempo estimation. There are still some open questions that we need to answer to assess the performance of the model per se.

We explore the use of the log-axis tempogram as the main representation, experimenting with three variations: Fourier, autocorrelation and hybrid. The results showed that the hybrid tempograms have the worst results, which indicates that the network can use both harmonic and sub-harmonic information as useful signals to learn representations. Using the hybrid tempogram could work if a more robust approach to calculating it was used.

We also looked into the role of data distribution and its effects on model learning. Experiments showed that in our scenario the model benefits more from data concentrated in lower tempi than higher tempi.

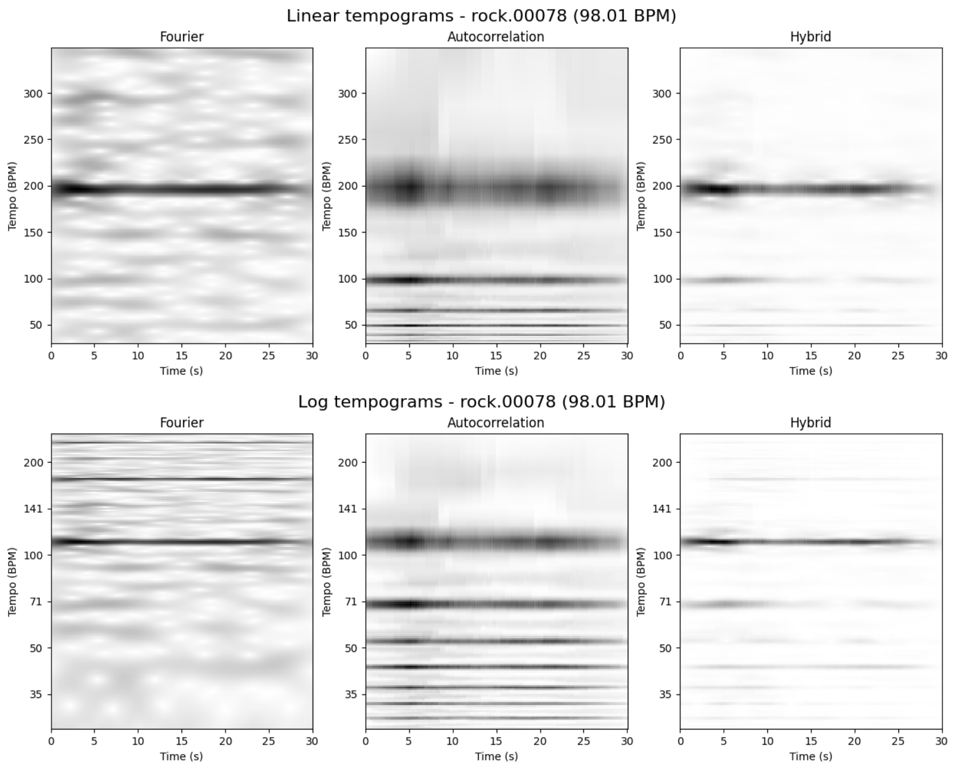
While we saw that it is possible to adapt the SPICE model to tempo estimation, a lot of questions and details remain unclear because we were comparing three different variations of the input representation with different synthetic and real-world data distributions. For future work, it would be better to focus on real-world datasets and focus on only the Fourier and Autocorrelation tempograms.

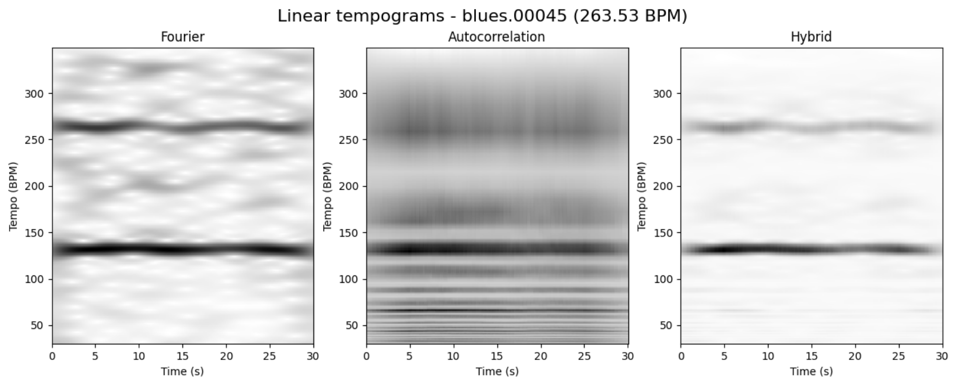
Another question that should be discussed is: does it make sense to have the tempogram as a representation or are we introducing too many parameters that are optimized before the model starts learning? For the tempogram, we have a set of parameters (from both the novelty function step and the pulse induction step) that are empirically defined. In this work, we did not explore the effects of those parameters in the final model, but it goes against the idea of a single-step model, such as the ones proposed by Schreiber (2018) and Steinmetz and Reiss (2021).

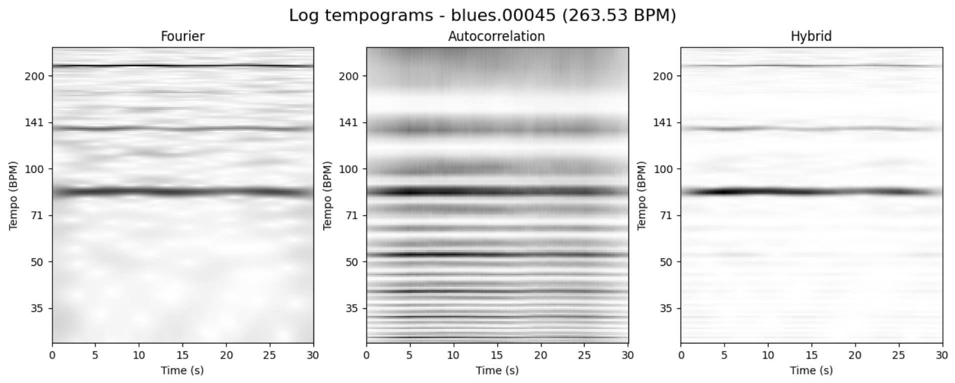
Appendix A

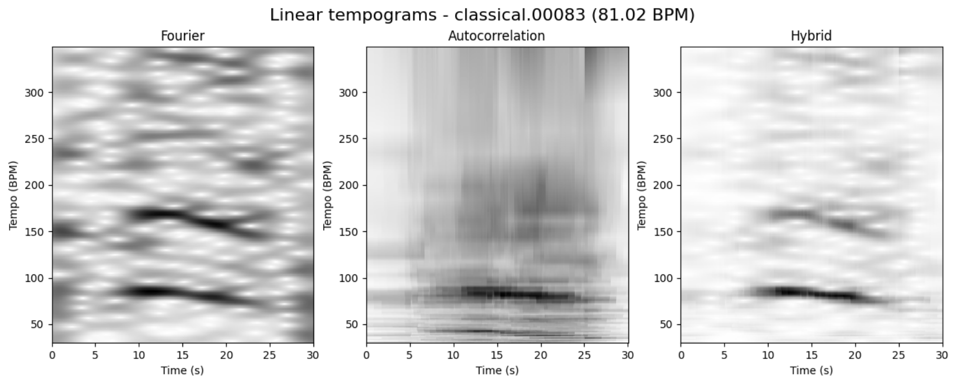
Additional Figures

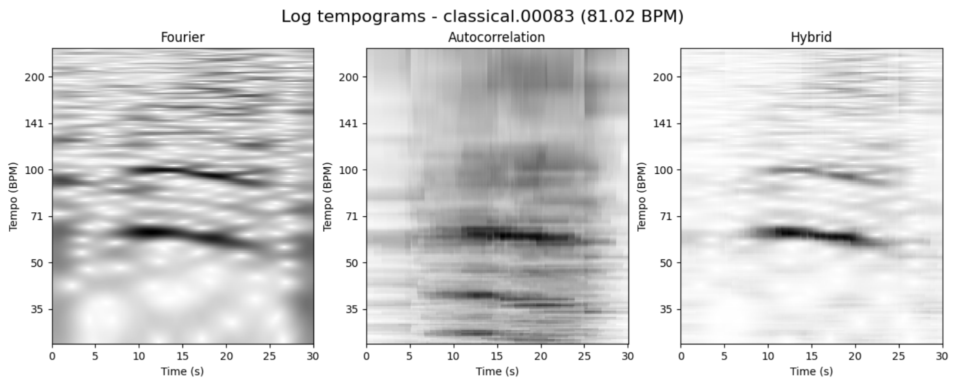
A.1 Linear and Logarithmic Tempograms

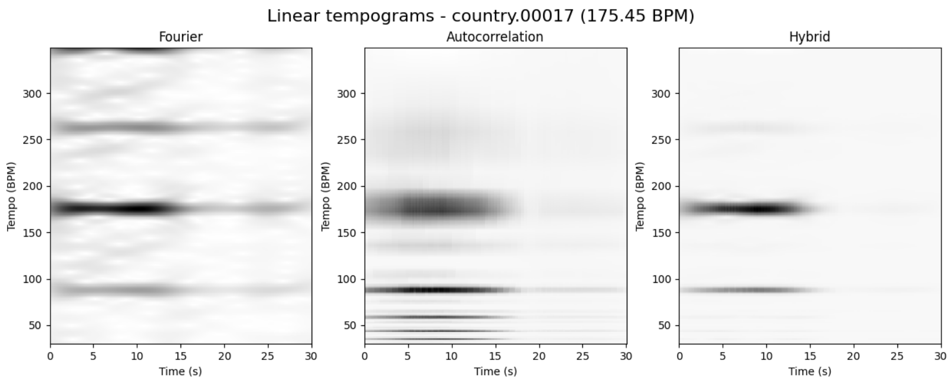


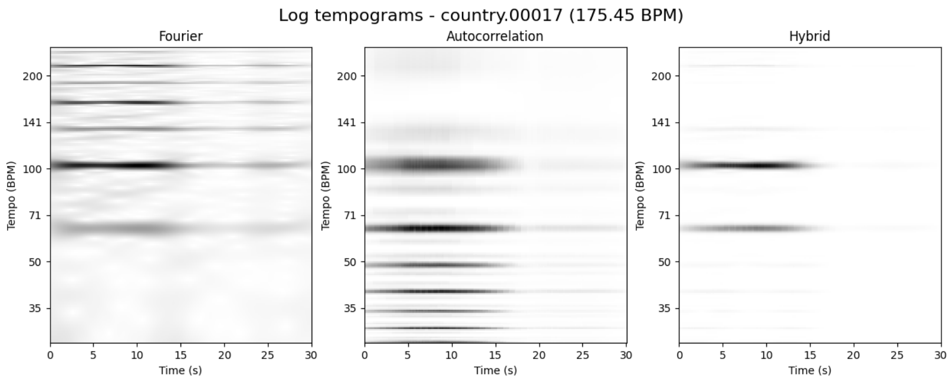

Linear tempograms - blues.00045 (263.53 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)

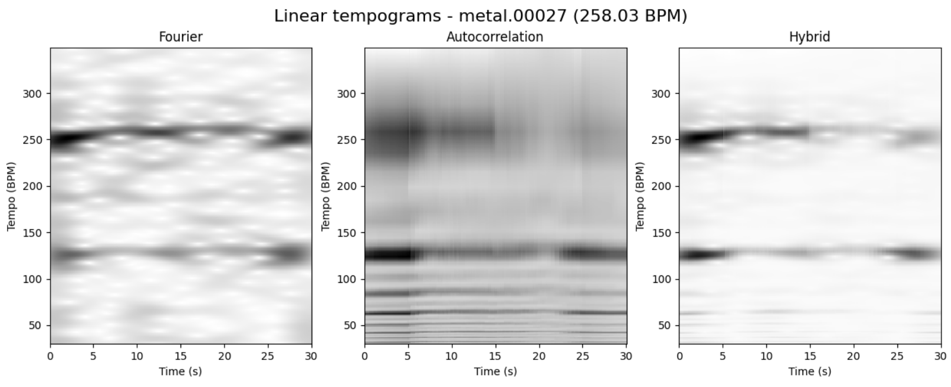

Log tempograms - blues.00045 (263.53 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)

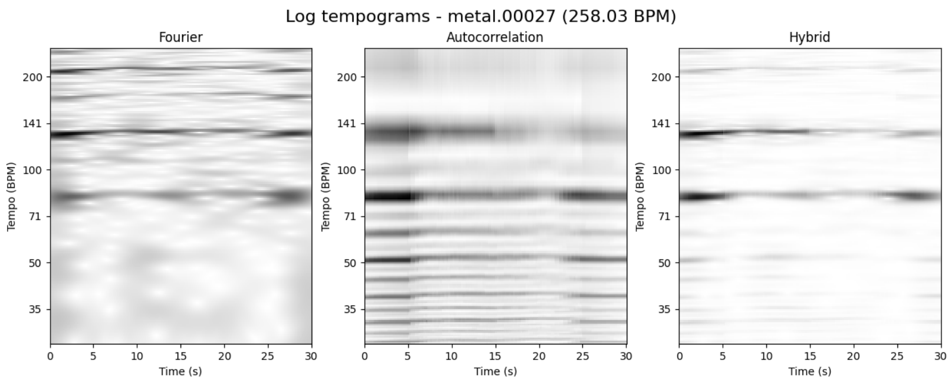

Linear tempograms - classical.00083 (81.02 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)


Log tempograms - classical.00083 (81.02 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)


Linear tempograms - country.00017 (175.45 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)


Log tempograms - country.00017 (175.45 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)


Linear tempograms - metal.00027 (258.03 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)


Log tempograms - metal.00027 (258.03 BPM)
Fourier
Autocorrelation
Hybrid
Tempo (BPM)
Time (s)

A.2 Calibration Results

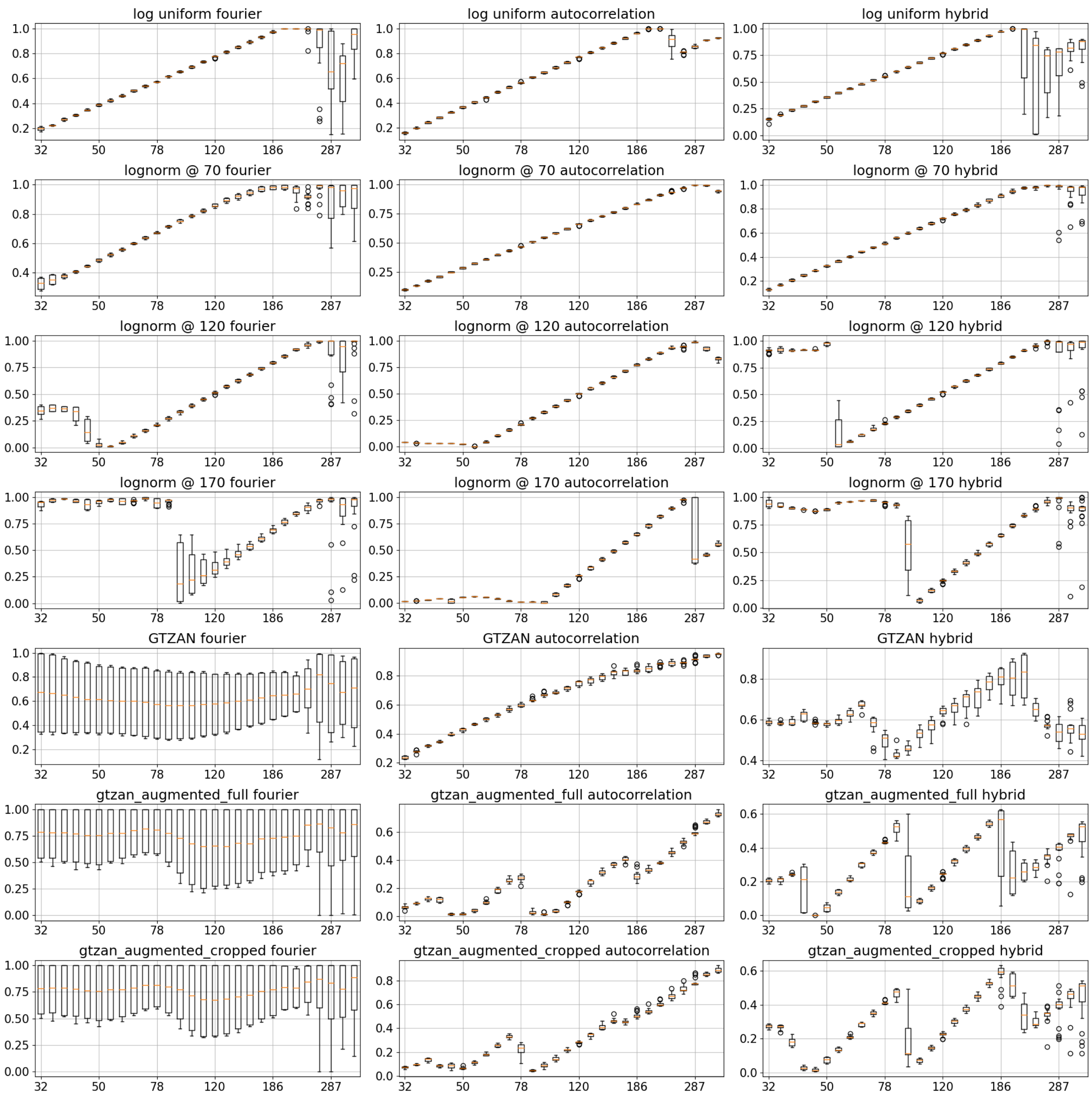


Figure A.1: *Calibration results. Models trained here had the EarlyStopping parameter as False, meaning that they were trained for 15 epochs.*

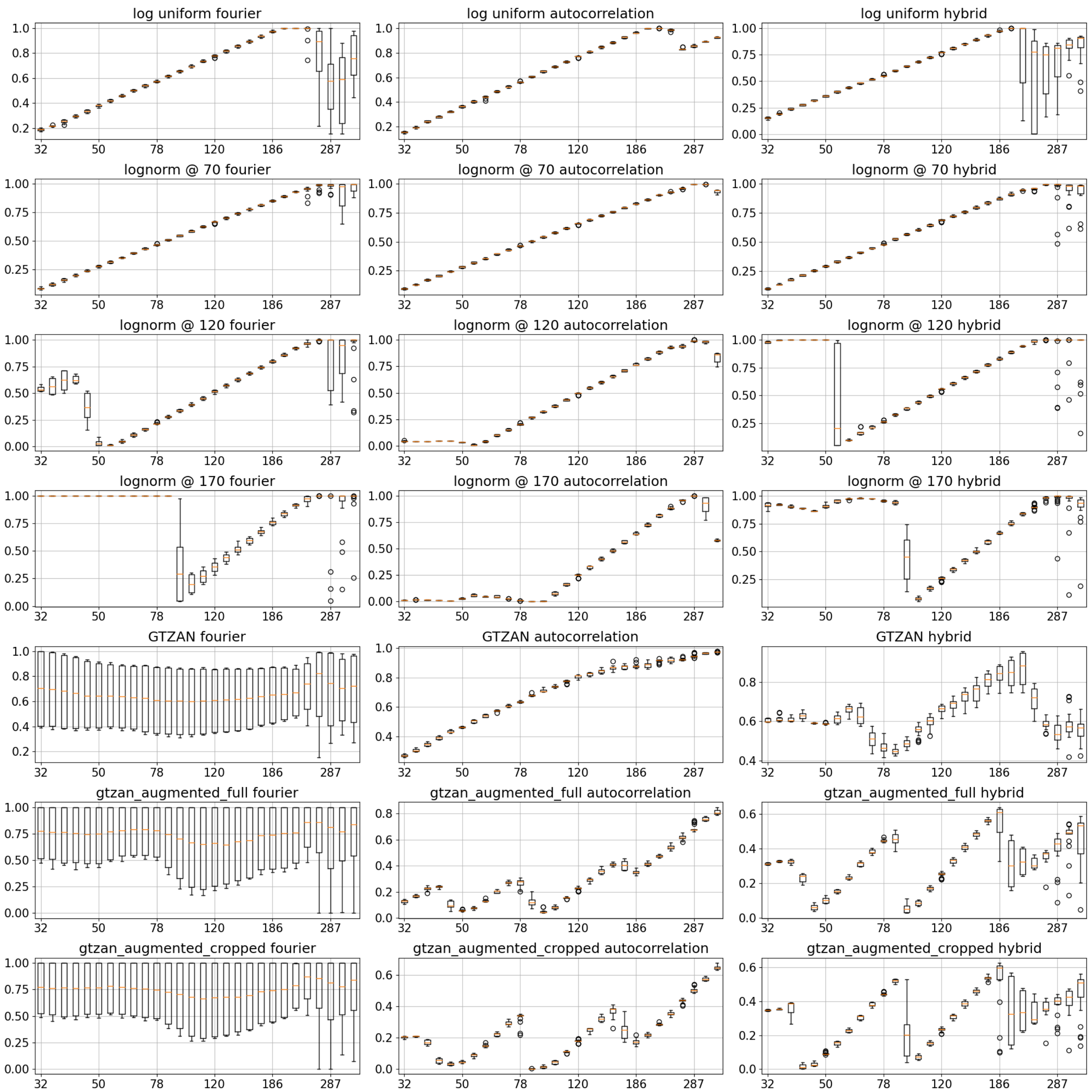


Figure A.2: *Calibration results. Models trained here had the EarlyStopping parameter as True, meaning that they were trained until the validation loss stopped reducing its value.*

Appendix B

Implementation

B.1 Metrics

```python
import numpy as np


def acc1(reference_tempo, estimated_tempo, tolerance=0.04, factor=1.0):
   return np.abs(reference_tempo * factor - estimated_tempo)\
      <= (reference_tempo * factor * tolerance)


def acc2(reference_tempo, estimated_tempo, tolerance=0.04):
   return (
      (acc1(reference_tempo, estimated_tempo, tolerance, 1.0))
      | (acc1(reference_tempo, estimated_tempo, tolerance, 2.0))
      | (acc1(reference_tempo, estimated_tempo, tolerance, 3.0))
      | (acc1(reference_tempo, estimated_tempo, tolerance, 1.0 / 2.0))
      | (acc1(reference_tempo, estimated_tempo, tolerance, 1.0 / 3.0))
   )


def oe1(reference_tempo, estimated_tempo, octave_factor=1.0):
   return np.log2((estimated_tempo * octave_factor) / reference_tempo)


def oe2(reference_tempo, estimated_tempo):
   factors = [1 / 3, 1 / 2, 1, 2, 3]
   oe = np.zeros_like(factors)

   for idx, factor in enumerate(factors):
      oe[idx] = oe1(reference_tempo, estimated_tempo, factor)

   return oe.min()
```

Program B.1: *ACC_1 and ACC_2 implementations*

B.2 Click tracks

```python
def click_track(bpm, sr=22050, duration=60):
    """
    Generates a 60 seconds click track with the desired BPM

    Parameters
    ----------
        bpm : int
            desired tempo
        sr : int, optional
            sampling rate
        duration : int
            duration in seconds
    """

    step = 60 / bpm

    times = np.arange(0, duration, step)

    return librosa.clicks(times=times, sr=sr)
```

Program B.2: *Function that generates metronome tracks*

B.3 Spectral Flux

```python
def spectral_flux(
        x,
        sr,
        n_fft=1024,
        hop_length=256,
        gamma=100.0,
        avg_window=10,
        norm=True):
    """
    Compute the spectral flux of a signal and apply logarithmic compression.

    Parameters
    ---------
        x : np.ndarray
            audio signal
        sr : int
            sampling rate
        n_fft : int, optional
            fft window size
        hop_length : int, optional
            step between fft windows
        gamma : float, optional
            logarithmic compression factor
        avg_window : int, optional
            window size (in samples) to compute local average
        norm : bool, optional
            boolean flag to normalize or not the novelty function
    Return
    ------
       novelty : np.ndarray
          the novelty function
       sr_novelty : float
          the sampling rate of the novelty function. defined as (sampling
          rate)/hop length
    """
    X = librosa.stft(
       x,
       n_fft=n_fft,
       hop_length=hop_length,
       win_length=n_fft,
       window="hann")
    sr_novelty = sr / hop_length

    Y = np.log(1 + gamma * np.abs(X))

    Y_diff = np.diff(Y)
    Y_diff[Y_diff < 0] = 0

    novelty = np.sum(Y_diff, axis=0)
    novelty = np.concatenate((novelty, np.array([0.0])))

    # subtract local avg
    if avg_window > 0:
       L = len(novelty)
       local_avg = np.zeros(L)
       for m in range(L):
          init = max(m - avg_window, 0)
          end = min(m + avg_window + 1, L)
          local_avg[m] = np.sum(novelty[init:end]) * \
             (1 / (1 + 2 * avg_window))
       novelty = novelty - local_avg
       novelty[novelty < 0] = 0.0

    if norm:
       max_value = max(novelty)
       if max_value > 0:
          novelty /= max_value

    return novelty, sr_novelty
```

Program B.3: *Spectral flux calculation*

B.4 Tempograms

```
def tempogram(x, sr, window_size_seconds, t_type, theta):
    """
       x : np.ndarray
          signal
       sr : float64
          sampling rate
       window_size : int, optional
          size in seconds of the tempogram window. default is 5s.
```

```
        type : string, optional
            tempogram type. accepted values are "fourier", "autocorrelation",
            "hybrid"
        theta : np.arange, optional
            tempi interval (BPM). default is (30,300,1), i.e from 30 to 300, 1
            at a time.
    """

    if not isinstance(theta, np.ndarray):
        raise ValueError(
            f"theta type incorrect. it should be np.ndarray, but is {type(theta)
                }")

    novelty, sr_novelty = spectral_flux(x, sr, n_fft=2048, hop_length=512)

    window_size_frames = int(window_size_seconds * sr_novelty)
    hop_size = 1

    if t_type == "fourier":
        T, t, bpm = fourier_tempogram(
            novelty,
            sr_novelty,
            window_size=window_size_frames,
            hop_size=hop_size,
            theta=theta
        )
    elif t_type == "autocorrelation":
        T, t, bpm, _, _ = autocorrelation_tempogram(
            novelty, sr_novelty, window_size=window_size_frames, hop_size=
                hop_size, theta=theta)
    elif t_type == "hybrid":
        ft, t, bpm = fourier_tempogram(
            novelty, sr_novelty, window_size=window_size_frames, hop_size=
                hop_size, theta=theta)
        at, ta, freqsa, _, _ = autocorrelation_tempogram(
            novelty, sr_novelty, window_size=window_size_frames, hop_size=
                hop_size, theta=theta)

        T = ft * at
    else:
        raise ValueError("tempogram_type incorrect. accepted values are \
            ['fourier', 'autocorrelation', 'hybrid']")

    return T, t, bpm
```

Program B.4: *Tempogram implementation*

B.4.1 Fourier Tempogram

```
def fourier_tempogram(novelty, sr_novelty, window_size, hop_size, theta):
    """
    Compute Fourier tempogram

    Parameters
    ----------
```

```
      novelty : np.ndarray
         novelty function
      sr_novelty : np.float
         sampling rate of the novelty function
      window_size : int
         window size in frames. 1000 corresponds to 10s in a signal sampled
         at 100 Hz
      hop_size : int
         hop size
      theta : np.ndarray
         range of BPM to cover
   """
   window = np.hanning(window_size)
   pad_size = int(window_size // 2)

   L = novelty.shape[0] + 2 * pad_size

   novelty_pad = np.concatenate(
      (np.zeros(pad_size), novelty, np.zeros(pad_size)))
   t_pad = np.arange(L)

   M = np.int64(np.floor(L - window_size) / hop_size + 1)
   K = len(theta)
   X = np.zeros((K, M), dtype=np.complex_)

   for k in range(K):
      omega = (theta[k] / 60) / sr_novelty

      exponential = np.exp(-2 * np.pi * 1j * omega * t_pad)
      x_exp = novelty_pad * exponential

      for n in range(M):
         t_0 = n * hop_size
         t_1 = t_0 + window_size
         X[k, n] = np.sum(window * x_exp[t_0:t_1])

   times = np.arange(M) * hop_size / sr_novelty
   tempi = theta

   return np.abs(X), times, tempi
```

Program B.5: *Fourier tempogram*

B.4.2 Autocorrelation Tempogram

```
def autocorrelation_tempogram(
      novelty, sr_novelty, window_size, hop_size, theta):
   """
   Compute autocorrelation-based tempogram

   Parameters
   ----------
      novelty : np.ndarray
         input novelty function
      sr_novelty : float64
```

```
            sampling rate
        window_size : int
            window length in frames
        hop_size : int
            hop size
        theta : np.ndarray
            array with BPM values we want to interpolate the autocorrelation

    Return
    ------
        tempogram : np.ndarray
            autocorrelation tempogram
        times : np.ndarray
            time axis (seconds)
        bpms : np.ndarray
            tempo axis (BPM)
        A_cut : np.ndarray
            time-lag representation A_cut (cut according to theta)
        lags_cut : np.ndarray
            Lag axis lags_cut
    """
    tempo_min = theta[0]
    tempo_max = theta[-1]
    lag_min = int(np.ceil(sr_novelty * 60 / tempo_max))
    lag_max = int(np.ceil(sr_novelty * 60 / tempo_min))

    A, times, lags = local_autocorrelation(novelty, sr_novelty,
                                           window_size, hop_size)
    # getting the min/max lag interval to use in the interpolation
    A_cut = A[lag_min:lag_max + 1, :]

    # "cut" the frequencies out of the max/min
    lags_cut = lags[lag_min:lag_max + 1]

    # translate to BPM
    bpms_cut = 60 / lags_cut
    bpms = theta

    # interpolate
    axis_interpolation = interp1d(
        bpms_cut,
        A_cut,
        kind='linear',
        axis=0,
        fill_value='extrapolate')

    tempogram = axis_interpolation(bpms)
    return tempogram, times, bpms, A_cut, lags_cut
```

Program B.6: *Autocorrelation tempogram*

References

[Abadi *et al.* 2016] Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.* Mar. 16, 2016. doi: 10.48550/arXiv.1603.04467. arXiv: 1603.04467 [cs]. url: http://arxiv.org/abs/1603.04467 (visited on 02/06/2023) (cit. on p. 29).

[Alonso *et al.* 2007] Miguel Alonso, Gaël Richard, and Bertrand David. "Accurate tempo estimation based on harmonic + noise decomposition". In: *EURASIP Journal on Advances in Signal Processing* 2007.1 (Jan. 2007), p. 161. issn: 1110-8657. doi: 10.1155/2007/82795. (Visited on 07/26/2023) (cit. on p. 11).

[Arandjelovic and Zisserman 2017] Relja Arandjelovic and Andrew Zisserman. "Look, Listen and Learn". In: Proceedings of the IEEE International Conference on Computer Vision. 2017, pp. 609–617. url: https://openaccess.thecvf.com/content_iccv_2017/html/Arandjelovic_Look_Listen_and_ICCV_2017_paper.html (visited on 02/06/2023) (cit. on p. 23).

[Balestriero *et al.* 2023] Randall Balestriero *et al. A Cookbook of Self-Supervised Learning*. Apr. 2023. doi: 10.48550/arXiv.2304.12210. arXiv: 2304.12210 [cs]. (Visited on 05/27/2023) (cit. on pp. ix, 2, 22, 23).

[Bello *et al.* 2005] J.P. Bello *et al.* "A tutorial on onset detection in music signals". In: *IEEE Transactions on Speech and Audio Processing* 13.5 (Sept. 2005), pp. 1035–1047. issn: 1063-6676. doi: 10.1109/TSA.2005.851998. url: http://ieeexplore.ieee.org/document/1495485/ (visited on 02/04/2023) (cit. on pp. 7, 8, 10).

[Böck 2011] Sebastian Böck. "Enhanced Beat Tracking With Context-Aware Neural Networks". In: (2011) (cit. on pp. 20, 21).

[Böck and Davies 2020] Sebastian Böck and Matthew E. P. Davies. "DECONSTRUCT, ANALYSE, RECONSTRUCT: HOW TO IMPROVE TEMPO, BEAT, AND DOWNBEAT ESTIMATION". In: (2020) (cit. on pp. 1, 2, 5, 19–21, 25, 26).

[Böck, Davies, and Knees 2019] Sebastian Böck, Matthew E. P. Davies, and Peter Knees. "Multi-Task Learning of Tempo and Beat: Learning One to Improve the Other". In: *Proceedings of the 20th International Society for Music Information Retrieval Conference* (Delft, The Netherlands). Delft, The Netherlands: ISMIR, Nov. 2019, pp. 486–493. doi: 10.5281/zenodo.3527850. url: https://doi.org/10.5281/zenodo.3527850 (cit. on pp. 20, 21, 25).

[Böck, Krebs, *et al.* 2014] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "A Multi-Model Approach to Beat Tracking Considering Heterogeneous Music Styles". In: (2014) (cit. on p. 21).

[Böck, Krebs, *et al.* 2015] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Accurate tempo estimation based on recurrent neural networks and resonating comb filters". In: *International Society for Music Information Retrieval Conference*. 2015 (cit. on p. 20).

[Böck, Krebs, *et al.* 2016] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Joint Beat and Downbeat Tracking with Recurrent Neural Networks". In: (2016) (cit. on p. 21).

[Böck and Widmer 2013] Sebastian Böck and Gerhard Widmer. "Maximum Filter Vibrato Suppression for Onset Detection". In: (2013) (cit. on pp. 7, 8, 10).

[Brown 1991] Judith C. Brown. "Calculation of a constant q spectral transform". en. In: *The Journal of the Acoustical Society of America* 89.1 (Jan. 1991), pp. 425–434. issn: 0001-4966. doi: 10.1121/1.400476 (cit. on p. 26).

[Chen and Su 2022] Tsung-Ping Chen and Li Su. "TOWARD POSTPROCESSING-FREE NEURAL NETWORKS FOR JOINT BEAT AND DOWNBEAT ESTIMATION". In: (2022) (cit. on pp. 21, 22).

[Cheng *et al.* 2018] Tian Cheng, Satoru Fukayama, and Masataka Goto. "Convolving Gaussian Kernels for RNN-Based Beat Tracking". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. Sept. 2018, pp. 1905–1909. doi: 10.23919/EUSIPCO.2018.8553310 (cit. on p. 21).

[Choi *et al.* 2017] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. "A tutorial on deep learning for music information retrieval". In: *CoRR* abs/1709.04396 (2017). arXiv: 1709.04396. url: http://arxiv.org/abs/1709.04396 (cit. on p. 19).

[Davies and Böck 2019] Matthew E. P. Davies and Sebastian Böck. "Temporal convolutional networks for musical audio beat tracking". In: *2019 27th European Signal Processing Conference (EUSIPCO)*. 2019 27th European Signal Processing Conference (EUSIPCO). Sept. 2019, pp. 1–5. doi: 10.23919/EUSIPCO.2019.8902578 (cit. on pp. 20, 21, 25).

[Desblancs *et al.* 2022] Dorian Desblancs, Romain Hennequin, and Vincent Lostanlen. “Zero-Note Samba: Self-Supervised Beat Tracking”. In: (2022) (cit. on pp. ix, 2, 24).

[S. Dixon 2006] S. Dixon. “ONSET DETECTION REVISITED”. In: 2006. url: https://www.semanticscholar.org/paper/ONSET-DETECTION-REVISITED-Dixon/ee97a58bc2813b56bb0d2319d99756bd731802e3 (visited on 02/06/2023) (cit. on pp. 8, 10).

[Simon Dixon 2001] Simon Dixon. “Automatic Extraction of Tempo and Beat From Expressive Performances”. In: *Journal of New Music Research* 30.1 (Mar. 1, 2001), pp. 39–58. issn: 0929-8215. doi: 10.1076/jnmr.30.1.39.7119. url: https://www.tandfonline.com/doi/abs/10.1076/jnmr.30.1.39.7119 (visited on 02/06/2023) (cit. on p. 10).

[Downie 2008] J. Stephen Downie. “The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research”. In: *Acoustical Science and Technology* 29.4 (2008), pp. 247–255. doi: 10.1250/ast.29.247 (cit. on p. 1).

[Duxbury *et al.* 2002] Christopher Duxbury, Mark Sandler, and Mike Davies. “A Hybrid Approach to Musical Note Onset Detection”. In: (2002) (cit. on p. 10).

[D. P. W. Ellis 2007] Daniel P. W. Ellis. “Beat Tracking by Dynamic Programming”. In: *Journal of New Music Research* 36.1 (Mar. 1, 2007), pp. 51–60. issn: 0929-8215. doi: 10.1080/09298210701653344. url: https://doi.org/10.1080/09298210701653344 (visited on 02/06/2023) (cit. on p. 5).

[Eronen and Klapuri 2010] A. J. Eronen and A. P. Klapuri. “Music tempo estimation with k-nn regression”. In: *Trans. Audio, Speech and Lang. Proc.* 18.1 (Jan. 2010), pp. 50–57. issn: 1558-7916 (cit. on p. 10).

[Eyben *et al.* 2010] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. “Universal onset detection with bidirectional long short-term memory neural networks”. In: *International Society for Music Information Retrieval Conference.* 2010 (cit. on pp. 7, 20).

[Foote and Uchihashi 2001] Jonathan Foote and Shingo Uchihashi. “The Beat Spectrum: A New Approach To Rhythm Analysis”. In: *Tokyo.* Vol. 22. Aug. 2001. doi: 10.1109/ICME.2001.1237863 (cit. on p. 12).

[Fouloulis *et al.* 2012] Thanos Fouloulis, Aggelos Pikrakis, and Emilios Cambouropoulos. “Asymmetric beat/tactus: Investigating the performance of beat-tracking systems on traditional asymmetric rhythms”. In: (2012) (cit. on p. 5).

[Fuentes 2019] Magdalena Fuentes. “Multi-Scale Computational Rhythm Analysis : A Framework for Sections, Downbeats, Beats, and Microtiming”. PhD thesis. Nov. 2019 (cit. on p. 21).

[Fuentes, Bittner, *et al.* 2021] Magdalena Fuentes, Rachel Bittner, *et al. Mirdata v.0.3.0*. Version 0.3.0. Jan. 2021. doi: 10.5281/zenodo.4355859. url: https://doi.org/10.5281/zenodo.4355859 (cit. on p. 34).

[Fuentes, Maia, *et al.* 2019] Magdalena Fuentes, Lucas S Maia, *et al.* “TRACKING BEATS AND MICROTIMING IN AFRO-LATIN AMERICAN MUSIC USING CONDITIONAL RANDOM FIELDS AND DEEP LEARNING”. In: (2019) (cit. on p. 21).

[Gfeller *et al.* 2020] Beat Gfeller *et al.* “SPICE: Self-supervised Pitch Estimation”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), pp. 1118–1128. issn: 2329-9290, 2329-9304. doi: 10.1109/TASLP.2020.2982285. arXiv: 1910.11664 [cs, eess]. url: http://arxiv.org/abs/1910.11664 (visited on 10/05/2022) (cit. on pp. v, ix, 2, 24, 26, 27, 29, 30, 36, 37).

[Gkiokas, Katsouros, and G. Carayannis 2012] Aggelos Gkiokas, Vassilis Katsouros, and G. Carayannis. “Reducing Tempo Octave Errors by Periodicity Vector Coding and SVM Learning”. In: *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*. Oct. 2012, pp. 301–306 (cit. on p. 6).

[Gkiokas, Katsouros, George Carayannis, *et al.* 2012] Aggelos Gkiokas, Vassilis Katsouros, George Carayannis, and Themos Stajylakis. “Music tempo estimation and beat tracking by applying source separation and metrical relations”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Mar. 2012), pp. 421–424. doi: 10.1109/ICASSP.2012.6287906. (Visited on 07/26/2023) (cit. on p. 11).

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. url: https://www.deeplearningbook.org/ (cit. on pp. 2, 23).

[Goto and Muraoka 1995] Masataka Goto and Yoichi Muraoka. “A Real-time Beat Tracking System for Audio Signals”. In: (1995) (cit. on p. 5).

[Gouyon *et al.* 2006] F. Gouyon *et al.* “An experimental comparison of audio tempo induction algorithms”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.5 (Sept. 2006), pp. 1832–1844. issn: 1558-7916, 1558-7924. doi: 10.1109/TSA.2005.858509. url: https://ieeexplore.ieee.org/document/1678001/ (visited on 02/06/2023) (cit. on pp. ix, 6, 10, 11).

[Grosche and Muller 2011] Peter Grosche and Meinard Muller. “Extracting Predominant Local Pulse Information From Music Recordings”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.6 (Aug. 2011), pp. 1688–1701. issn: 1558-7924. doi: 10.1109/TASL.2010.2096216 (cit. on p. 12).

[Grosche, Müller, *et al.* 2010] Peter Grosche, Meinard Müller, and Frank Kurth. "Cyclic tempogram—A mid-level tempo representation for musicsignals". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. Mar. 2010, pp. 5522–5525. doi: 10.1109/ICASSP.2010.5495219 (cit. on pp. 3, 12, 18, 35).

[Hennequin *et al.* 2020] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020). Deezer Research, p. 2154. doi: 10.21105/joss.02154. url: https://doi.org/10.21105/joss.02154 (cit. on p. 25).

[Holzapfel and Grill 2016] Andre Holzapfel and Thomas Grill. "BAYESIAN METER TRACKING ON LEARNED SIGNAL REPRESENTATIONS". In: *New York City* (2016) (cit. on p. 21).

[Jansen *et al.* 2018] Aren Jansen *et al.* "Unsupervised Learning of Semantic Audio Representations". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Apr. 2018, pp. 126–130. doi: 10.1109/ICASSP.2018.8461684 (cit. on p. 2).

[*Keras: The Python Deep Learning API* 2023] *Keras: The Python Deep Learning API*. url: https://keras.io/ (visited on 02/17/2023) (cit. on p. 29).

[Klapuri *et al.* 2006] A. P. Klapuri, A. J. Eronen, and J. T. Astola. "Analysis of the meter of acoustic musical signals". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (Dec. 2006), pp. 342–355. issn: 1558-7916. doi: 10.1109/TSA.2005.854090 (cit. on p. 11).

[Korzeniowski *et al.* 2014] Filip Korzeniowski, Sebastian Böck, and Gerhard Widmer. "Probabilistic Extraction of Beat Positions from a Beat Activation Function". In: (2014) (cit. on p. 21).

[Kurth and Gehrmann 2006] Frank Kurth and Thorsten Gehrmann. "The Cyclic Beat Spectrum: Tempo-Related Audio Features for Time-Scale Invariant Audio Identification". In: (2006) (cit. on p. 12).

[Lea *et al.* 2016] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D. Hager. *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*. Aug. 2016. doi: 10.48550/arXiv.1608.08242. arXiv: 1608.08242 [cs]. (Visited on 07/27/2023) (cit. on p. 20).

[Lerch 2012] Alexander Lerch. *An Introduction to Audio Content Analysis*. Guide books. 2012. doi: 10.5555/2392638. url: https://dl.acm.org/doi/abs/10.5555/2392638 (visited on 02/06/2023) (cit. on p. 6).

[Levy 2011] Mark Levy. “IMPROVING PERCEPTUAL TEMPO ESTIMATION WITH CROWD-SOURCED ANNOTATIONS”. In: *Oral Session* (2011) (cit. on p. 6).

[Marchand *et al.* 2015] Ugo Marchand, Quentin Fresnel, and Geoffroy Peeters. *GTZAN-Rhythm: Extending the GTZAN Test-Set with Beat, Downbeat and Swing Annotations*. Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conference, 2015. Oct. 2015. URL: https://hal.science/hal-01252607 (cit. on pp. 33, 34).

[Matthew E. P. Davies 2021] Magdalena Fuentes Matthew E. P. Davies Sebastian B ock. *Tempo, Beat and Downbeat Estimation*. 2021. URL: https://tempobeatdownbeat.github.io/tutorial/intro.html (cit. on pp. 21, 22).

[McFee and D. P. Ellis 2014] Brian McFee and Daniel P.W. Ellis. “Better beat tracking through robust onset aggregation”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). May 2014, pp. 2154–2158. DOI: 10.1109/ICASSP.2014.6853980 (cit. on pp. 7, 8).

[McFee, McVicar, *et al.* 2023] Brian McFee, Matt McVicar, *et al. Librosa/librosa: 0.10.0rc1*. Zenodo, Feb. 17, 2023. DOI: 10.5281/zenodo.7650421. URL: https://zenodo.org/record/7650421/export/hx (visited on 02/17/2023) (cit. on p. 29).

[McKinney and Moelants 2006] Martin F. McKinney and Dirk Moelants. “Ambiguity in Tempo Perception: What Draws Listeners to Different Metrical Levels?” In: *Music Perception* 24.2 (Dec. 1, 2006), pp. 155–166. ISSN: 0730-7829. DOI: 10.1525/mp.2006.24.2.155. URL: https://online.ucpress.edu/mp/article/24/2/155/62298/Ambiguity-in-Tempo-Perception-What-Draws-Listeners (visited on 02/06/2023) (cit. on pp. 1, 6).

[Morais *et al.* 2023] Giovana Morais, Matthew E. P. Davies, Marcelo Queiroz, and Magdalena Fuentes. *Tempo vs. Pitch: understanding self-supervised tempo estimation*. 2023. arXiv: 2304.06868 `[cs.SD]` (cit. on pp. 29, 37, 39).

[Müller 2015] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. 1st. Springer Publishing Company, Incorporated, 2015. Chap. 6. ISBN: 3319219448 (cit. on pp. 9, 10, 12).

[Müller and Zalkow 2019] Meinard Müller and Frank Zalkow. “FMP NOTEBOOKS: EDUCATIONAL MATERIAL FOR TEACHING AND LEARNING FUNDAMENTALS OF MUSIC PROCESSING”. In: Sept. 26, 2019 (cit. on pp. ix, 12, 18).

[Müller and Zalkow 2021] Meinard Müller and Frank Zalkow. “Libfmp: A Python Package for Fundamentals of Music Processing”. In: *Journal of Open Source Software* 6.63 (July 20, 2021), p. 3326. ISSN: 2475-9066. DOI: 10.21105/joss.03326. URL: https://joss.theoj.org/papers/10.21105/joss.03326 (visited on 02/08/2023) (cit. on p. 36).

[Nunes *et al.* 2015] Leonardo O. Nunes, Martín Rocamora, Luis Jure, and L. Biscainho. “Beat and Downbeat Tracking Based on Rhythmic Patterns Applied to the Uruguayan Candombe Drumming”. In: International Society for Music Information Retrieval Conference. 2015. URL: https://www.semanticscholar.org/paper/Beat-and-Downbeat-Tracking-Based-on-Rhythmic-to-the-Nunes-Rocamora/99228cc4537bdcbf1646f84dda74ac68dd914c2b (visited on 02/05/2023) (cit. on p. 7).

[Oliveira *et al.* 2012] J. L. Oliveira, M. E. P. Davies, F. Gouyon, and L. P. Reis. “Beat Tracking for Multiple Applications: A Multi-Agent System Architecture With State Recovery”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.10 (2012), pp. 2696–2706. ISSN: 1558-7916. URL: https://www.academia.edu/2877208/Beat_tracking_for_multiple_applications_A_multi_agent_system_architecture_with_state_recovery (visited on 02/08/2023) (cit. on pp. 5, 10).

[Ou *et al.* 2022] Longshen Ou, Xiangming Gu, and Ye Wang. *Transfer Learning of Wav2vec 2.0 for Automatic Lyric Transcription.* Oct. 2022. DOI: 10.48550/arXiv.2207.09747. arXiv: 2207.09747 [cs, eess]. (Visited on 08/02/2023) (cit. on p. 19).

[Pathak *et al.* 2017] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. *Learning Features by Watching Objects Move.* 2017. arXiv: 1612.06370 [cs.CV] (cit. on p. 23).

[G. Peeters 2006] G. Peeters. “Music Pitch Representation by Periodicity Measures Based on Combined Temporal and Spectral Representations”. In: *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings.* Vol. 5. Toulouse, France: IEEE, 2006, pp. V-53-V–56. ISBN: 978-1-4244-0469-8. DOI: 10.1109/ICASSP.2006.1661210. (Visited on 07/26/2023) (cit. on p. 16).

[Geoffroy Peeters 2006] Geoffroy Peeters. “Template-Based Estimation of Time-Varying Tempo”. In: *EURASIP Journal on Advances in Signal Processing* 2007.1 (Dec. 2006), p. 067215. ISSN: 1687-6180. DOI: 10.1155/2007/67215. URL: https://asp-eurasipjournals.springeropen.com/articles/10.1155/2007/67215 (visited on 02/06/2023) (cit. on pp. 17, 37).

[Geoffroy Peeters 2011] Geoffroy Peeters. “Spectral and Temporal Periodicity Representations of Rhythm for the Automatic Classification of Music Audio Signal”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.5 (July 2011), pp. 1242–1252. ISSN: 1558-7924. DOI: 10.1109/TASL.2010.2089452 (cit. on pp. 17, 37).

[Geoffroy Peeters and Flocon-Cholet 2012] Geoffroy Peeters and Joachim Flocon-Cholet. “Perceptual tempo estimation using GMM-regression”. In: *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies.* Nara Japan: ACM, Nov. 2012, pp. 45–50. ISBN: 978-1-4503-1591-3. DOI: 10.1145/2390848.2390861. (Visited on 07/26/2023) (cit. on p. 6).

[Pinto *et al.* 2021] António Pinto, Sebastian Böck, Jaime Cardoso, and Matthew Davies. "User-Driven Fine-Tuning for Beat Tracking". In: *Electronics* 10.13 (June 2021), p. 1518. issn: 2079-9292. doi: 10.3390/electronics10131518. (Visited on 07/26/2023) (cit. on p. 21).

[Purwins *et al.* 2019] Hendrik Purwins *et al.* "Deep Learning for Audio Signal Processing". In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (May 2019), pp. 206–219. issn: 1932-4553, 1941-0484. doi: 10.1109/JSTSP.2019.2908700. (Visited on 07/26/2023) (cit. on p. 20).

[Quinton 2017] Elio Quinton. "Towards the Automatic Analysis of Metric Modulations". PhD thesis. Queen Mary University of London, 2017 (cit. on pp. 6, 12, 34).

[Quinton 2022] Elio Quinton. *Equivariant Self-Supervision for Musical Tempo Estimation.* Sept. 3, 2022. arXiv: 2209.01478 [cs, eess]. url: http://arxiv.org/abs/2209.01478 (visited on 10/05/2022) (cit. on pp. ix, 2, 3, 12, 21, 24–26).

[Quinton *et al.* 2016] Elio Quinton, Mark Sandler, and Simon Dixon. "Estimation of the reliability of multiple rhythm features extraction from a single descriptor". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Mar. 2016, pp. 256–260. doi: 10.1109/ICASSP.2016.7471676 (cit. on p. 12).

[Repp 1996] Bruno H. Repp. "Patterns of note onset asynchronies in expressive piano performance". In: *The Journal of the Acoustical Society of America* 100.6 (Dec. 1996), pp. 3917–3932. issn: 0001-4966. doi: 10.1121/1.417245 (cit. on p. 6).

[Rosão *et al.* 2012] C. Rosão, Ricardo Ribeiro, and David Martins de Matos. "Influence of Peak Selection Methods on Onset Detection". In: International Society for Music Information Retrieval Conference. 2012. url: https://www.semanticscholar.org/paper/Influence-of-Peak-Selection-Methods-on-Onset-Ros%C3%A3o-Ribeiro/7bfc35ab2febcd31bec0bae071e00487d47d3508 (visited on 02/06/2023) (cit. on p. 10).

[Scheirer 1998] Eric D. Scheirer. "Tempo and beat analysis of acoustic musical signals". en. In: *The Journal of the Acoustical Society of America* 103.1 (Jan. 1998), pp. 588–601. issn: 0001-4966. doi: 10.1121/1.421129 (cit. on p. 5).

[Schlüter and Böck 2014] Jan Schlüter and Sebastian Böck. "Improved musical onset detection with Convolutional Neural Networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). May 2014, pp. 6979–6983. doi: 10.1109/ICASSP.2014.6854953 (cit. on p. 7).

[Schreiber 2018] Hendrik Schreiber. "A SINGLE-STEP APPROACH TO MUSICAL TEMPO ESTIMATION USING A CONVOLUTIONAL NEURAL NETWORK". In: (2018) (cit. on p. 43).

[Schreiber 2020] Hendrik Schreiber. “Data-Driven Approaches for Tempo and Key Estimation of Music Recordings”. doctoralthesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020, p. 188 (cit. on p. 20).

[Schreiber and Müller 2017] Hendrik Schreiber and Meinard Müller. “A Post-Processing Procedure for Improving Music Tempo Estimates Using Supervised Learning”. In: *International Society for Music Information Retrieval Conference*. 2017. (Visited on 07/26/2023) (cit. on p. 6).

[Schreiber and Müller 2019] Hendrik Schreiber and Meinard Müller. “Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters”. In: *ArXiv* (Mar. 2019). (Visited on 07/27/2023) (cit. on p. 20).

[Schreiber, Urbano, *et al.* 2020] Hendrik Schreiber, Julián Urbano, and Meinard Müller. “Music Tempo Estimation: Are We Done Yet?” In: *Transactions of the International Society for Music Information Retrieval* 3.1 (1 Aug. 24, 2020), pp. 111–125. issn: 2514-3298. doi: 10.5334/tismir.43. url: http://transactions.ismir.net/articles/10.5334/tismir.43/ (visited on 02/05/2023) (cit. on pp. 1, 5, 11).

[Schreiber, Zalkow, *et al.* 2020] Hendrik Schreiber, Frank Zalkow, and Meinard Müller. “MODELING AND ESTIMATING LOCAL TEMPO: A CASE STUDY ON CHOPIN’S MAZURKAS”. In: (2020) (cit. on pp. 5, 20).

[Steinmetz and Reiss 2021] Christian J. Steinmetz and Joshua D. Reiss. *WaveBeat: End-to-end Beat and Downbeat Tracking in the Time Domain.* https://arxiv.org/abs/2110.01436v1. Oct. 2021. (Visited on 07/29/2023) (cit. on pp. 21, 22, 43).

[Sturm 2012] Bob L. Sturm. “An analysis of the GTZAN music genre dataset”. In: *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies.* MIRUM ’12. New York, NY, USA: Association for Computing Machinery, Nov. 2, 2012, pp. 7–12. isbn: 978-1-4503-1591-3. doi: 10.1145/2390848.2390851. url: https://doi.org/10.1145/2390848.2390851 (visited on 02/06/2023) (cit. on p. 34).

[Sun *et al.* 2021] Xiaoheng Sun, Qiqi He, Yongwei Gao, and Wei Li. “Musical Tempo Estimation Using a Multi-scale Network”. In: *ArXiv* (Sept. 2021). (Visited on 07/26/2023) (cit. on p. 20).

[Takahashi and Barthet 2022] Takuya Takahashi and Mathieu Barthet. “EMOTION-DRIVEN HARMONISATION AND TEMPO ARRANGEMENT OF MELODIES USING TRANSFER LEARNING”. In: (2022) (cit. on p. 19).

[Thoshkahna *et al.* 2015] Balaji Thoshkahna, Meinard Muller, Venkatesh Kulkarni, and Nanzhu Jiang. “Novel audio features for capturing tempo salience in music recordings”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* South Brisbane, QLD, Australia: IEEE, Apr. 2015, pp. 181–185. isbn: 978-1-4673-6997-8. doi: 10.1109/ICASSP.2015.7177956. (Visited on 07/26/2023) (cit. on pp. 12, 18).

[Tian *et al.* 2015] Mi Tian, György Fazekas, Dawn A. A. Black, and Mark Sandler. “On the use of the tempogram to describe audio content and its application to Music structural segmentation”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2015, pp. 419–423. doi: 10.1109/ICASSP.2015.7178003 (cit. on p. 12).

[Tzanetakis and Cook 2002] G. Tzanetakis and P. Cook. “Musical genre classification of audio signals”. In: *IEEE Transactions on Speech and Audio Processing* 10.5 (2002), pp. 293–302. doi: 10.1109/TSA.2002.800560 (cit. on p. 33).

[Vogl *et al.* 2017] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. “Drum Transcription via Joint Beat And Drum Modeling using Convolutional Recurrent Neural Networks”. In: Oct. 2017 (cit. on p. 21).

[Wang, Salamon, *et al.* 2020] Yu Wang, Justin Salamon, Mark Cartwright, Nicholas J. Bryan, and Juan Pablo Bello. *Few-Shot Drum Transcription in Polyphonic Music*. Aug. 2020. doi: 10.48550/arXiv.2008.02791. arXiv: 2008.02791 [cs, eess]. (Visited on 08/02/2023) (cit. on pp. 2, 19).

[Wang, Stoller, *et al.* 2022] Yu Wang, Daniel Stoller, Rachel M. Bittner, and Juan Pablo Bello. “Few-Shot Musical Source Separation”. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2022, pp. 121–125. doi: 10.1109/ICASSP43922.2022.9747536 (cit. on pp. 2, 19).

[Widmer 2013] Gerhard Widmer. “Enhanced peak picking for onset detection with recurrent neural networks”. In: 2013 (cit. on p. 20).

[Xiao *et al.* 2008] Linxing Xiao, Aibo Tian, Wen Li, and Jie Zhou. “USING A STATISTIC MODEL TO CAPTURE THE ASSOCIATION BETWEEN TIMBRE AND PERCEIVED TEMPO”. In: *Rhythm and Meter* (2008) (cit. on p. 6).

[Zapata and Gómez 2011] Jose R Zapata and Emilia Gómez. “COMPARATIVE EVALUATION AND COMBINATION OF AUDIO TEMPO ESTIMATION APPROACHES”. In: (2011) (cit. on pp. 7, 10, 11).

[C. Zhang *et al.* 2021] Chen Zhang *et al. PDAugment: Data Augmentation by Pitch and Duration Adjustments for Automatic Lyrics Transcription*. Sept. 2021. arXiv: 2109.07940 [cs, eess]. (Visited on 08/02/2023) (cit. on pp. 2, 19).

[R. Zhang *et al.* 2016] Richard Zhang, Phillip Isola, and Alexei A. Efros. *Colorful Image Colorization*. 2016. arXiv: 1603.08511 [cs.CV] (cit. on p. 23).