# Analyses of Convolutional Neural Networks for Automatic tagging of music tracks

**Master Thesis**

Aravind Sankaran

**Supervisors**

Prof. Paolo Bientinesi

Prof. Marco Alunno

**Examiners**

Prof. Paolo Bientinesi

Prof. Bastian Leibe

# Abstract

Describing music can be quite tricky and talking about music may simply require as much vocabulary as any technical subject. Musicians and composers usually discuss their work with jargon describing certain aesthetics of the song. As the amount of music recordings are constantly growing, finding a song that matches these aesthetic description is challenging. This work is an attempt to take a step towards developing algorithms that could tag music like an artist. The currently available state-of-art algorithms are trained and tested on datasets with tags that are socially biased. Moreover these datasets contain just short clips of songs, but an artist describes a song as a whole. Therefore, in this thesis, a repertoire of 900 songs with carefully labelled data describing the whole track is used and the aim is to find the model settings that would best approximate the audio features for such a dataset. The Mel-Frequency-Cepstral-Coefficients (MFCC) features are compared with features extracted by pre-trained Convolution Neural Networks (CNN) over mel-log power spectrogram. These features are extracted every 29.1s and approximated over time to a fixed size representation. The temporal approximation by sequence to one Long Term Short Memory (LSTM) Recurrent Neural Network is compared with approximation by Bag of Frames (BoF) approach and the weighted area under receiver operating characteristic curve (AUC) is reported. The experiments show that MFCC features summarized by LSTM outperforms pre-trained convolutions counterpart. It is also seen that LSTM perform better than Bag of Frames features for temporal approximation.

## Acknowledgments

I would like to thank *Prof. Paolo Bientinesi* (High performance and automatic comupting group, AICES, RWTH Aachen) for doing one of the most expensive work - *Listen to almost thousand songs and tag them.* I also thank *Prof. Marco Alunno* (Professor of Composition and Theory at University EAFIT, Columbia) for his valuable association and advices.

# Contents

# Chapter 1

# Introduction

Computers have been used to automate discovery and management of music in so many different ways. Automating the task of attaching a semantic meaning to a song is popularly known as *music auto-tagging*. Automatic tagging algorithms have been used to build recommendation systems that allow listeners to discover songs that match their taste. It also enables online music stores to filter their target audience. But semantic description of a song is not straightforward and there is this gap between music audio and listener's description, both linguistically and emotionally, which we term as *audio-semantic gap*. In this thesis, we test the state-of-art approaches of music auto-tagging on a dataset that is least affected by *audio-semantic* noise and find out the efficient model settings. In section 1.1, the need for this dedicated research is explained by describing some shortcomings of the currently available solution approaches. In section 1.2, a top to bottom overview of the contents of this research is presented.

## 1.1 Motivation

Captioning music from the point of view of an artist is an interesting application. Training an algorithm to do so takes music auto-tagging a step towards human intelligence. Although great technical progress have been made to enable efficient retrieval and organization of audio content, analysing music and communicating it's artistic properties is still challenging. Current music recommendation systems fall short in providing recommendations based of aesthetics of music because of the reasons described below,

### 1.1.1 Cold start problem with collaborative filtering methods

When the usage data is available, one can use collaborative filtering to recommend the tracks on a community-based trending lists (say, a community of experts). That is, if a listener liked songs A and B and you liked A, you might also like B. Such algorithms have proven to be extremely efficient and out-perform those algorithms that works by extracting acoustic cues from audio signal for the task of finding similar songs [14]. However, in absence of such usage data, one resorts to content-based methods, where just the audio signal is used for generating recommendations. Thus

collaborative filtering methods suffer from what is called a *cold start problem*, making it less efficient for new and unpopular songs.

### 1.1.2 Problems with content based methods

Using information from audio content to overcome the cold-start problem resulted in *content-based* recommendation methods. In such algorithms, a classifier is trained on some training data to learn acoustic cues. But a recommendation system can also be built without requiring such acoustic labels by combining *content based* and *collaborative* techniques[8] and training it from a collaborative view point. However, this is not sufficient if a recommendation system has to designed for artists and composers who search for songs based on some properties of music itself. In such cases, the current *content-based* methods fall short because of following training assumption,

#### Psychoacoustic assumptions

Music descriptions are often affected by social factors. However, it is possible to measure what percentage of subjects classify a music to certain mood (say, happy, dull etc.) and present the popular description. The currently available large datasets are built on this assumption [11][7]. Algorithms trained on such datasets may converge to learning social descriptions rather than actual descriptions termed by experts for the properties of music (also termed as *aesthetics* in popular journals of music psychology[20]). This psychoacoustic assumption stands as a barrier to discover songs based on aesthetics (which has applications in music therapy [6]).

#### Temporal summarization of audio content

The current state of art music tagging algorithms[19][15] are established by training on datasets that contain just short music excerpts. In run-time, these algorithms classify each short section separately and merge tags across different sections. But an artist might describe the music as a whole and not in sections. Hence there is a need to test algorithms that extract features approximating greater length of the song.

### 1.1.3 Need for adaptive glossary

Current recommendation systems including the ones that use collaborative filtering, restrict the user with the choice of tags. Moreover, it is not guaranteed that all users will perceive all the tags in the same way. A recent study in idiographic music psychology have indicated that different people use different aesthetic criteria to make judgement about music[20]. Hence it would be interesting to study if these models [18][19] trained on large datasets can be exploited for training on personal repertoire, which are usually small.

## 1.2 Overview

In an attempt to get rid of the *audio-semantic* noise, we use a dataset tagged by an expert and assumptions about song length are not made (Let us call them *aesthetic tags*). But such datasets are usually small because gathering them is expensive. Convolution neural networks (CNN) have

recently gained popularity for content-based multi-label classification task achieving state-of-art performance[18][19] on established datasets[11][7]. But these models were trained on large amount of training data containing short excerpts of music and it is not clear if section-wise merging of descriptions can approximate actual description of the whole song. So the aim of this thesis is to find out

- If the celebrated CNN models trained on large data can be exploited to show similar performance gains when *fine-tuned* on a small dataset (Generally termed as *transfer learning*).
- Models that can best approximate signals of arbitrary length to a fixed size representation (needed for classification).

The classification is done on a lower dimensional approximation of the audio signal known as *feature*. The general pipeline for music feature extraction is *signal representation*, systematic *dimensionality-reduction* followed by *temporal approximation*.

### 1.2.1 Signal Representation

Music is distinguished by the presence of many relationships that can be treated mathematically, including rhythm and harmony, by analysing the frequency content. In Chapter 2 (Sec. 2.1.3), representation of digital audio in time-frequency format is explained. Motivated by the way ear-brain handles the frequency information, myriad of features extracted from spectrogram representations were evolved. Each one of them will fit into one of these broad categories:

- **Mel based spectrogram :** Exploiting the fact that our ear cannot distinguish adjacent frequencies (say we cannot differentiate 300 KHz and 310 KHz), the information pertaining to frequencies are binned according to a what to popularly known as *mel-scale*. The features (MFCCs, etc). obtained from this representation are useful for any general purpose application like speech recognition, genre classification etc.
- **Chromagram :** Frequencies are binned by taking advantage of the periodic perception of *pitch* (perceived frequency). Features (Tonnetz, etc) following from this representation find applications in music mixing, chord recognition etc.
- **Tempogram :** For applications like tempo estimation and onset detection, the representation should encode the *change* of frequencies over time. This resulted in a set of features (Novelty curve, beat centroid etc) calculated from *differential* spectrograms.

In this thesis, we only study *mel-based spectrogram* representations. Chroma and tempo features can be obtained after binning to mel-scale (not necessarily), and in this sense mel-spectrogram can be viewed as a super set. But one should look at this categorization in terms of mathematical operation. Features resulting from *mel-spectrogram* means *explicit* operations involving periodic binning and temporal differentiation are not involved. I use the word *explicit* because it will be shown later that tempo and chroma information can be *implicitly* modelled by *feature learning* with CNNs (see Chapter 2, 2.2.2).

### 1.2.2 Dimensionality reduction

Features are usually obtained as a result of some dimensionality reduction operation on the spectrogram. In Chapter 2 (Sec. 2.2.1), dimensionality reduction through *basis transformation* is introduced in terms of convolution operation and the extraction of Mel-Frequency-Cepstral-Coefficients (MFCCs) is explained. Then in section 2.2.2, generalizing the MFCC computation into a learning problem by replacing the basis functions with learnable convolutional filters is discussed and eventually explaining the success of convolution neural networks. In Chapter 3, some of the successful algorithms reported for music-auto-tagging are discussed. Transfer-learning using CNN model in [18] which reports close to state-of-art performance is compared with MFCC features for our target task in Chapter 4 (Sec. 4.2) .

### 1.2.3 Temporal approximation

We are looking for an algorithm that will approximate features for songs of arbitrary length without losing much of the rhythmic information. In the current literature, this is handled using *Bag of Frame* approach which is explained in Chapter 2, section 2.3.1. But as the reported performance are for 29.1s song clips, their optimality for songs of greater length is questioned. So we test *Recurrent neural network (Sequence to One LSTM)* which is more motivating to use for rhythmic information extraction (see 2.3.2). In Chapter 4 (Sec. 4.2), the performance of approximation by *Bag of Frames* and *Recurrent neural network* is reported.

## 1.3 Outline of the report

In chapter 2, the fundamentals required for understanding remaining contents are explained and formalism of notations are introduced. In chapter 3, a detailed overview of previous research, their shortcomings for the current problem along with justification for proposed models are discussed. In chapter 4, details of the dataset and the experiment results of proposed models are discussed. In chapter 5, the inference from these experiments in understanding the development of algorithms for tagging music with aesthetic tags is explained and future directions are discussed.

# Chapter 2

# Formalisms

## 2.1 Representation of music signal

To retrieve information from a music signal, it is important to understand it's discriminants. In section 2.1.1, the abstractions that would help in analysing the organization of music-audio content are explained. The general aim of Music Information Retrieval (MIR) is to extract information about it's discriminants from the observed data. This observed signal is traditionally represented in the time domain. The time domain is a record of what happened to a parameter of the system versus time. Standard formats use amplitude versus time. The observed signal is then discretised by sampling and stored in digital format (see 2.1.2). This signal in the time domain is then changed to frequency domain. This is simply because our ear-brain combination is an excellent frequency domain analyser. Our brain splits the audio spectrum into many narrow bands and determines the power present in each band.[pp1] Conversion from time domain to frequency domain is done using the foundations from *Fourier theorem* (see 2.1.3). Currently used music signal representations for general MIR tasks are explained in section 2.1.4.

### 2.1.1 Discriminants of music signal - Harmonics and overtones

It was shown over one hundred years ago by Baron Jean Baptiste Fourier that any waveform that exists in the real world can be generated by series of sinusoids which are a function of frequencies. Hence any *stationary signal* $\mathbf{m}(t)$ (i.e signal at instantaneous time $t$ ) can be represented as a linear combination of function ($f$) of *fundamental frequency* $\omega_0$(lowest frequency), and other frequencies ($\omega_i \vee i \in \{1, 2..N\}$) which are multiples of fundamental. The formalism of this function $f$ will be elaborated in section 2.1.3. The following abstract representation is adapted for further explanations in this section,

$$\mathbf{m}(t) = f(\omega_0[t], c_i[t], k_i[t]) \qquad i \in \{1, 2, ..., N\}$$

Where $c_i$ are the coefficients in linear combination and $k_i$ are the multiples of the fundamental. It is important to note that all the variables of function $f$ changes over time except for *stationary signals* and hence the index specification $[t]$. Therefore, whenever the assumption of *stationarity* is

made, the signal is represented as

$$\mathbf{m}(t) = f(\omega_0, c_i, k_i) \qquad i \in \{1, 2, ..., N\}$$

For now, let us assume that signal is stationary. This can be imagined as a stroke of a musical note. When a note is played on an instrument, listeners hear the played tone as the fundamental, as well as a combination of its harmonics sounding at the same time (Hammond, 2011). Harmonics are tones that have frequencies that are integer multiples of the fundamental frequency. The fundamental and its harmonics naturally sound good together. So a *stationary* signal is harmonic if $k_i \in \mathbb{Z}$, where $\mathbb{Z}$ is a set of *integers*. The fundamental usually dominates the harmonics (i.e strength of higher harmonics are usually less). These additional frequencies with multiples $k_i$ determines the *timber*. It is this *timber* that differentiates the same note played by different instruments.

TODO:

The presence of multiple, simultaneous notes in polyphonic music renders accurate pitch tracking very difficult. However, there are many other applications, including chord recognition and music matching, that do not require explicit detection of pitches, and for these tasks several representations of the pitch and harmonic information commonly appear. Usually, there are more instruments being played simultaneously and sometimes accompanied by voices. In such cases, we hear the fundamental and overtones (chord). The overtones are any frequency above the fundamental frequency. The overtones may or may not be harmonics. So overtones are those frequencies which are not just restricted to integer multiples of fundamental. The fundamental and overtones together are called partials

EQ, ki != integers

m1(t) = f(440 ,880)

Where 440 = fundamental, 880 = first harmonic m2(t) = f(330 , 660)

Where 330 = fundamental, 660 = first harmonic

Thus the recorded signal has components of all these frequencies

m(t) = f(330,440,660,880)

Where 330 = fundamental, 440 = second partial, 660 = third partial, 880 = fourth partial

Most certainly, the signal evolves over time and hence the components of frequencies and its amplitudes will vary for each time t. The heat map representation of amplitudes, with frequency along y, time along x is called spectrogram.

Thus to discriminate a signal, we not only need the evolution of frequencies, but also information about harmonics. For instance, to discriminate the instruments from the recorded signal m(t), the classifier should infer the frequencies in the each harmonics m1(t) and m2(t). To discriminate the temporal pattern (Rhythm), we need the evolution of m1 and m2. To identify other aesthetics (warm, city) , the interaction between m1(t) and m2(t) should be inferred. To discriminate voices and other non-harmonic aspects (tempo, beat), the envelop curve of the spectrum will also be needed.

### 2.1.2 Sampling of continuous-time signal

The digital formats contain the discrete version of the signal obtained by sampling continuous-time signal. For functions that vary with time, let s(t) be a continuous function (or "signal") to be

sampled, and let sampling be performed by measuring the value of the continuous function every T seconds, which is called the sampling interval or the sampling period.[1][pp2]. The sampling frequency or sampling rate, $fs$, is the average number of samples obtained in one second (samples per second),

$$fs = \frac{1}{T}.$$

The optimum sampling rate is given by Nyquist-Shannon sampling theorem which says, the sampling frequency (fs) should be at least twice the highest frequency contained in the signal [pp2] Given the human hearing range lies between 20Hz - 20KHz [pp3], most of the digital audio formats use a standard sampling frequency of 44.4Khz. The signal is further down sampled depending on the kind of feature information needed for classification.

### 2.1.3 Time-Frequency transformations

The signal represented in the time domain is a set of ordered $n$-tuples of real numbers $(a_1, a_2, ..., a_N) \in \mathbb{R}^N$ in the vector space $V$ , specifically *Euclidean n-space*. That is to say, a discrete-time signal can be represented as a linear combination of Cartesian basis vectors.

$$\mathbf{a}(t) = (a_1, a_2, ..., a_N) = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + ... + a_N\mathbf{e}_N = \sum_{i=1}^{N} a_i\mathbf{e}_i \tag{2.1}$$

where:

**a** is a discrete-time signal

$\mathbf{e}_1 ... \mathbf{e}_N$ are Cartesian basis vectors (Unit vectors).

Mapping from time-domain to frequency-domain is looked up on as basis transformation. We need to find a set of basis vectors $\boldsymbol{\phi}_\omega$, whose coefficients $c_\omega$ then represents the components in frequency domain.

$$\mathbf{a}(t) = \sum_{\omega=0}^{M-1} c_\omega \boldsymbol{\phi}_\omega(t) \tag{2.2}$$

for some integer $0 < M < \infty$. Then $\mathbf{c}(\phi) = (c_0, c_1, ..., c_{M-1}) \in \mathbb{C}^M$ represents the components in frequency domain. Thus our aim is to compute $\mathbf{c}(\phi)$ by defining basis vectors $\boldsymbol{\phi}_\omega$ which are functions of frequency. Computing the Fourier coefficients for periodic and aperiodic signals are discussed below.

#### Periodic Signals

If $\mathbf{a}(t)$ is periodic in **T**, then we can apply the definition of **Exponential Fourier Series** expansion and define $\boldsymbol{\phi}$ in equation (2.2) as (See Appendix ??),

$$\boldsymbol{\phi}_k(t) = \frac{1}{\sqrt{T}} e^{ik\omega t} \tag{2.3}$$

Whose basis functions $\boldsymbol{\phi}$ now form *complete orthonormal* set [3]. That is,

$$< \boldsymbol{\phi}_i, \boldsymbol{\phi}_j >= \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{2.4}$$

The fourier series finds a set of discrete coefficients of **harmonically related frequencies** ($k\omega$) . To retrieve $c_k$, multiply $\boldsymbol{\phi}_k$ on both sides of equation (2.2) and apply the conditions of orthonormality in equation (2.4). Thus

$$c_k =< \mathbf{a}(t), \boldsymbol{\phi}_k(t) > \tag{2.5}$$

Although periodicity assumptions are not made for general music signals, it becomes relevant to deduce rhythmic patterns.

#### Aperiodic Signals

It is difficult to assume periodicity for a generalized signal. We need to estimate the coefficients $\mathbf{c}$ for continuous frequency variable $\boldsymbol{\omega}$ instead of discrete harmonics $\mathbf{k}\boldsymbol{\omega}$. The Fourier series can not be applied directly and hence Fourier Transform was developed. Here we aim to find out quantity of each sinusoids is the signal $\mathbf{a}(t)$. This can be done by dividing $\mathbf{a}(t)$ by $e^{i\omega t}$ over the time domain. We use the complex exponential in place of sinusoids because we know (see Appendix ??)

$$sin(\omega t + \Phi) \propto e^{i\omega t} \tag{2.6}$$

Where $\Phi$ is the phase difference. Thus, the coefficients in the frequency domain are

$$c_\omega = \sum_{t=0}^{N-1} a(t) e^{-i\omega t} \tag{2.7}$$

This is the N-point **Discrete Fourier Transform**. For the proof of existence of such coefficients, please refer to chapter ?? in [3]. From here, $\boldsymbol{\phi}_\omega(t)$ in equation (2.2) can be defined as

$$\boldsymbol{\phi}_\omega(t) = e^{i\omega t} \tag{2.8}$$

Thus, we can compute $\mathbf{a}(t)$ as a linear combination of complex exponentials. This is also known as **Inverse Fourier Transform**.

$$\mathbf{a}(t) = \sum_{\omega=0}^{M-1} c_\omega e^{i\omega t} \tag{2.9}$$

Hence, with Fourier Transform, we can go back and forth between time and frequency domain. It is important to note that these basis vectors need **not** be *orthogonal*.

**Fast Fourier Transform**(FFT) is an efficient implementation of Discrete Fourier Transform(DFT) which exploits the symmetry of *sines* and *cosines*. While DFT requires $O(N^2)$ operations, FFT requires only $O(NlogN)$ [3].

### 2.1.4 STFT, Mel-Spectrogram, Chromogram

It is useful to perform FFT locally over short segments. This is simply because FFT becomes very expensive for larger $N$.

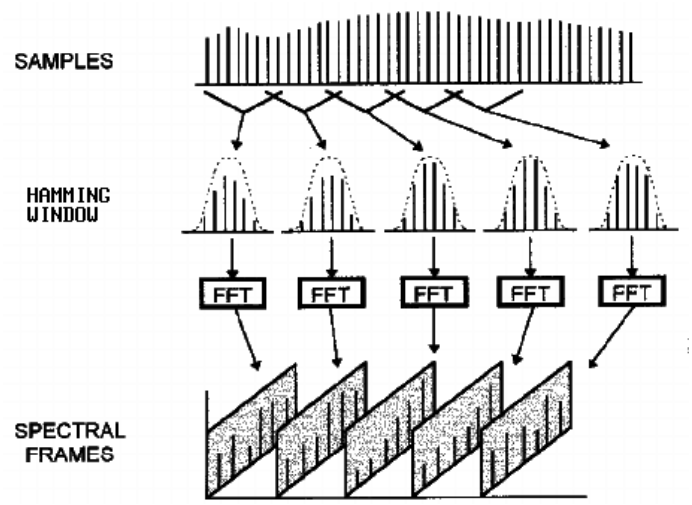
$$KNlog(N) < (KN)log(KN)$$

The full length signal is divided into short segments, and FFT is computed separately for each segment. This is known as **Short Time Fourier Transform (STFT)**. Usually the dimension of

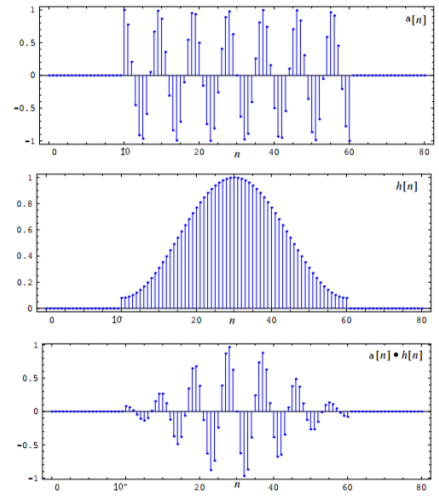
the frequency components are reduced by using bins. Every frequency component is assigned to it's nearest bin. This however causes **spectral leakage** when we divide the signal into rectangular windows. That is, components at the end of the segment can leak to the adjacent segment. This is avoided by modifying the original signal by applying some window function. The most common window function is the **Hamming Window** defined as,

$$h[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \tag{2.10}$$

Where $n \in 0, 1, ..., N-1$. The signal approaches zero near $n = 0$ and $n = N-1$, but reaches peak near $n = N/2$ [23]. To overcome the information loss at the ends of the window, signal is divided into segments that are partly *overlapping* with eachother. Figure (2.1 (a)) shows the extraction of spectral frames of a spectrogram.




(a) Windowing is applied on overlapping segments followed by FFT




(b) Application of Hamming Window on a segment of input signal

Figure 2.1: (a) Shows STFT Pipeline. (b) Shows the application of Window function

The discrete STFT (*slow*) for $p^{th}$ frame of signal $\mathbf{a}$ is obtained as,

$$\mathbf{C}(p, \omega) = \sum_{n=p.s}^{p.s+F} \mathbf{a}(n)\mathbf{h}(n - p.s)e^{-i\omega(n-p.s)} \tag{2.11}$$

Where:

$P$: is the number of spectral frames; $p \in [0, 1..P-1]$

$M$: is the dimension of discrete frequency space ; $\omega \in \mathbb{R}^M$

$F$: is the frame length

$s$: is stride (or) hop-length for the next segment

$\mathbf{a} \in \mathbb{R}^N$ ; $n \in [0, 1...N-1]$

$\mathbf{h} \in \mathbb{R}^F$

$\omega \in \mathbb{R}^M$

$\mathbf{C}$: is Fourier Coefficient Matrix ; $\mathbf{C} : \mathbb{R}^{F.P} \rightarrow \mathbb{C}^{M \times P}$

Equation (2.11) can be seen as a **convolution** over the signal $\mathbf{a}$ with $\mathbf{W}$ which has finite support over the set $\{0, 1.., F\}$ (more details in section ??)

$$\boxed{\mathbf{C}(p, \omega) = \mathbf{a}(n) \star \mathbf{W}_\omega(n - \tau)} \tag{2.12}$$

Where:

$\tau = p.s$

$\mathbf{W}_\omega(n - \tau) = \mathbf{h}(n - \tau)e^{-i\omega(n-\tau)}$

It is important to note that the coefficients $c_\omega$ may be complex valued. They are functions of the amplitude of corresponding sinusoidal component (see Appendix ??). But, to obtain useful metrics, we need to extract some physical quantity from the coefficients. This is where **Parseval's theorem** is used, which relates and time and frequency domain components in DFT as follows [3] :

$$\|\mathbf{c}\|^2 \propto \|\mathbf{a}\|^2 \tag{2.13}$$

If $\mathbf{a}$ represents amplitude in the time-domain, then as a consequence of Hook's law on energy equation (see Appendix ??), we know that

$$Energy \propto amplitude^2 \tag{2.14}$$

Relating equation 2.11 and 2.12, it can be inferred that **square** of the Fourier coefficients is proportional to the energy distributed in the corresponding frequencies. This is called the **Power Spectrum (E)**. It is often motivating to use this representation because *loudness* is proportional to *energy*.

$$\mathbf{E} = \mathbf{C} \odot \mathbf{C} \tag{2.15}$$

As mentioned earlier, the frequencies in the considered range are grouped into bins. It is useful to do so, not only to reduce dimension but also due to the aliasing effect of human auditory system. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on how frequencies are grouped, two different class of spectrograms are discussed.

### Mel Spectrogram

The *mel-scale* was developed to express measured frequency in terms of psychological metrics (i.e perceived pitch). The mel-scale was developed by experimenting with the human ears interpretation of a pitch. The experiment showed that the pitch is linearly perceived in the frequency range 0-1000

Hz. Above 1000 Hz, the scale becomes logarithmic. There are several formulae to convert Hertz to mel. A popularly used formula is noted in [1]

$$\omega_m = 2595 log_{10}\left(1 + \frac{\omega}{700}\right) \tag{2.16}$$

Where $\omega$ is the frequency in Hertz. In a mel spectrogram, the frequencies and converted to mels and then grouped into mel-spaced bins. This is done by multiplying the spectrum with some **filter bank ($\mathbf{M}_{\omega_m}$)**. For details about computation of mel-filter banks, refer [10]. Each filter bank is centered at a specific frequency. Hence, to compute R mel bins, we need R mel-filter banks.

$$\mathbf{Mel}(p, \omega_m) = \sum_{\omega=0}^{M} \mathbf{Y}(p, \omega)\mathbf{M}_{\omega_m}(\omega) \tag{2.17}$$

Where:

$\mathbf{Y} = f(\mathbf{C})$

$\omega_m$ = mel frequency

When the function $f$ is an defined by equation (2.15), we get **mel power spectrogram**

We can re-write equation (2.17) as,

$$\mathbf{Mel}(p, \omega_m) = \sum_{k=p.M}^{p.M+K} \mathbf{U}(k)\mathbf{M}_{\omega_m}(k - p.M) \tag{2.18}$$

Where:

$P$: is the number of spectral frames; $p \in [0, 1..P-1]$

$M$: is the dimension of discrete frequency space ; $\omega = k - p.M \in \mathbb{R}^M$

$K = M.P$ and $k \in [0, 1...K]$

$\mathbf{U}(k) = \mathbf{Y}(i, j)$ ; $i = floor(\frac{k}{M})$ ; $j = k - floor(\frac{Mk}{M-1})$

$\mathbf{Y} \in \mathbb{R}^{M \times P}$

$\mathbf{U} \in \mathbb{R}^{M.P}$

$\omega_m \in \mathbb{R}^R$

**Mel**: is Mel Spectrum Matrix ; $\mathbf{Mel} : \mathbb{R}^{M.P} \rightarrow \mathbb{R}^{R \times P}$

Hence, we can represent mel-spectrogram as **M-strided convolution** over *flattened* $\mathbf{Y}$ with mel filters $\mathbf{M}_{\omega_m}$ (i.e, the frequency axis of $\mathbf{C}$ is contracted with each mel-filte)r,

$$\boxed{\mathbf{Mel}(p, \omega_m) = \mathbf{U}(k) \star \mathbf{M}_{\omega_m}(k - p.M)} \tag{2.19}$$

### Chromagram

This representation takes advantage of the periodic perception of pitch. Two pitches are perceived similar in "color" if they differ by one or several octaves apart. Chromagram groups such periodic perceptions into same coefficient (chroma). All pitches that belong to the same chroma are said to be from same pitch class. [TODO: How is this computed?]

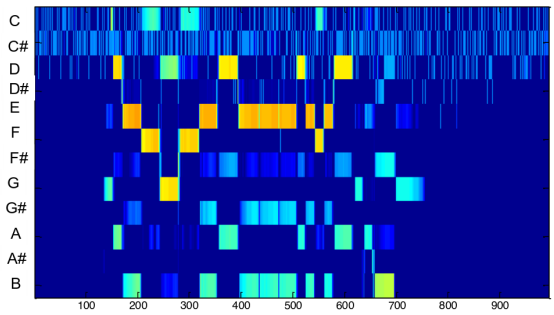



Figure 2.2: Chromagram of Western Pitch Scale

## 2.2 Dimensionality Reduction

The objective of dimensionality reduction is to retain only the desirable characteristics of the representations presented in section 2.1. This is done because the representation ($\mathbf{R}$) can be large for longer audio tracks (because number of frames $P$ depends on length of the audio). Reduction over a large frame at once can lead to loss of temporal information (i.e, change of variables across frames). Hence, dimensionality reduction is done hierarchically, sometimes by stacking combination of these techniques. We generalize the operations on input signal $\mathbf{a}$ as follows,

$$\mathbf{R} = Rep(\mathbf{a}) \qquad Rep : \mathbb{R}^N \rightarrow \mathbb{R}^{R \times P}$$

$$\mathbf{X} = f(\mathbf{R}) \qquad f : \mathbb{R}^{R \times P} \rightarrow \mathbb{R}^{S \times Q}$$

$$\mathbf{Y} = D(\mathbf{X}) \qquad D : \mathbb{R}^{S \times Q} \rightarrow \mathbb{R}^{T \times W}$$

The representation operations defined in the previous section can be a part of the function $Rep$. Since dimension reductions can be stacked, $f$ represents the previous reductions applied. $Q$ and $W$ are number of frames as a result of hierarchical windowing operation shown in fig (2.3). For the first reduction however, $f$ does not exist and hence represented by conditional arrow (dotted). The output of reduction $\mathbf{Y} \in \mathbb{R}^{T \times W}$ will then be the reduced representation ($T < S$ or $W < Q$). Depending on how the function $D$ is defined, we will classify the techniques into broad categories.

- Reduction by basis transformation
- Reduction by neural networks
- Reduction by clustering

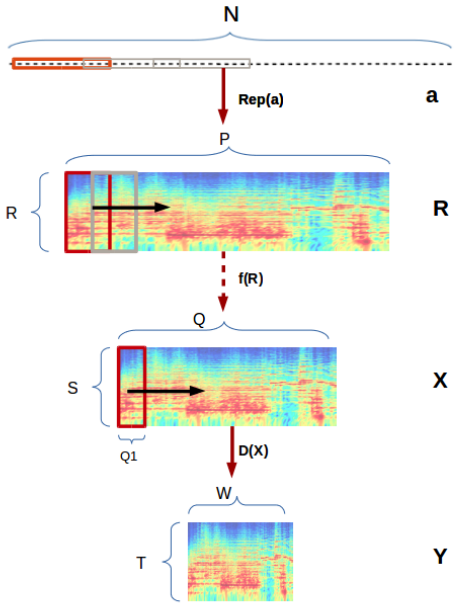



Figure 2.3: Dimensionality Reduction Pipeline

### 2.2.1 Basis Transformations - PCA, MFCC

The basis vectors are functions of the properties we want to encode. In equation (2.2), we used basis transformation to represent the signal in frequency domain. Now we want to use the same concept, but for dimensionality reduction. In general terms, this is done by *changing to a reduced basis*. That is, we need to find a change of coordinates matrix that will map the input to a basis system with lesser coordinates.

The input frame $\mathbf{X}_w$ is first operated with some window function $\mathbf{w}$,

$$\mathbf{z}_w = \{\mathbf{X}_w\mathbf{w} \quad | \quad \mathbf{X}_w \in \mathbb{R}^{S\times J}, \mathbf{w} \in \mathbb{R}^J, \mathbf{z}_w \in \mathbb{R}^S\}$$

Now we have to compute $\mathbf{y}_w \in \mathbb{R}^T$ which has least representation error with $\mathbf{z}_w \in \mathbb{R}^S$ such that $T < S$. Let us say, $\mathbf{z}_w$ can be written as a linear combination of some basis $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_T] \in \mathbb{R}^{S\times T}$ with coordinates $\mathbf{y}_w = [y_1, y_2, ..., y_T]$. Then $\mathbf{y}_w$ can be calculated by solving,

$$\mathbf{z}_w = \sum_{i=1}^{T} y_i\mathbf{v}_i = \mathbf{V}\mathbf{y}_w$$

$$\mathbf{y}_w = \mathbf{V}^{-1}\mathbf{z}_w$$

Thus, the operator $D$ in equation (??) is,

$$\mathbf{Y} = D(\mathbf{X}, \mathbf{V})$$

Therefore, dimension reduction through basis transformation are those class of techniques where $D$ is a function of some invertible change of coordinates matrix $\mathbf{V}$. Now, depending on how $\mathbf{V}$ is defined, some methods are discussed.

**Principal Component Analysis (PCA)**

The frequencies in the adjacent bins can be highly correlated and therefore contain redundant information. Principal component Analysis is a procedure to transform large number of correlated variables into smaller number of uncorrelated variables. This means, $\mathbf{y}_w$ should be approximated only with basis vectors that account for large variance. Hence, in PCA based techniques, dimension reduction is done through basis vectors of covariance matrix ($\mathbf{\Sigma}$). The coordinates of the resulting basis system are known as *principal components*. The steps of this abstraction are enumerated,

(a) The rows of $\mathbf{X}$ are centred by their mean and covariance is computed as,

$$\mathbf{\Sigma} = \mathbf{E}[(\mathbf{X} - \mathbf{E}[\mathbf{X}])(\mathbf{X} - \mathbf{E}[\mathbf{X}])^T] = \frac{1}{Q}\hat{\mathbf{X}}\hat{\mathbf{X}}^T \quad \in \mathbb{S}^{S\times S}$$

(b) The eigen values and eigen vectors of $\mathbf{\Sigma}$ are computed. At this point, we use the Orthogonal Eigenvector Decomposition Theorem and infer that eigen vectors of symmetric matrix ($\mathbf{\Sigma}$) form an orthogonal basis in $\mathbb{R}^S$.

$$\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \qquad \mathbf{V} \in \mathbb{O}^{S\times S}, \quad \mathbf{\Lambda} \in \mathbb{D}^{S\times S}$$

(c) The eigen values represent the magnitude of variance for each frequency. Hence, eigenvectors corresponding to large eigen values gives the coordinates corresponding to greater variance. So eigen vectors corresponding to top $T$ eigen values are retained, while ignoring coordinates of lower variance. The resulting change of coordinates matrix is then $\hat{\mathbf{V}} \in \mathbb{O}^{S\times T}$

(d) Since $\hat{\mathbf{V}}$ is orthogonal, $\hat{\mathbf{V}}^{-1} = \hat{\mathbf{V}}^T$, and we can compute $\mathbf{y}_w = \hat{\mathbf{V}}^T\mathbf{z}_w$

**Algorithm 1** $\mathbf{Y} = \text{PCA}(\mathbf{X})$

**Input :** $\mathbf{X} \in \mathbb{R}^{S\times Q}$
**Output :** $\mathbf{Y} \in \mathbb{R}^{T\times W}$

$W = \frac{Q}{Q_s}$
**for** $i \in \{0, 1, .., W\}$ **do**
  $\mathbf{X}_s = \mathbf{X}[i.Q_s : (i+1).Q_s]$ ▷ $\mathbf{X}_s \in \mathbb{R}^{S\times Q_s}$
  $\mathbf{\Sigma} = \frac{1}{Q_s}\mathbf{X}_s\mathbf{X}_s^T$ ▷ $\mathbf{\Sigma} \in \mathbb{S}^{S\times S}$
  $\mathbf{V}, \boldsymbol{\lambda} = EIG(\mathbf{\Sigma}, T)$ ▷ $\boldsymbol{\lambda} \in \mathbb{R}^T$ , $\mathbf{V} \in \mathbb{O}^{S\times T}$
  $\mathbf{Y}[i] \leftarrow \mathbf{V}^T\mathbf{X}_s$
**end for**

### 2.2.2 Feature learning with stacked convolutions

**Feature engineering Vs Feature Learning**

**Algorithm 2** $\mathbf{Y} = \text{PCA WHITENING}(\mathbf{X})$

**Input :** $\mathbf{X} \in \mathbb{R}^{S \times Q}$
**Output :** $\mathbf{Y} \in \mathbb{R}^{T \times Q}$ ▷ $W = Q$
1: $\mathbf{\Sigma} = \frac{1}{Q}\mathbf{X}\mathbf{X}^T$ ▷ $\mathbf{\Sigma} \in \mathbb{S}^{S \times S}$
2: $\mathbf{V}, \mathbf{\Lambda} = EIG(\mathbf{\Sigma}, T)$ ▷ $\mathbf{\Lambda} \in \mathbb{D}^{T \times T}$ , $\mathbf{V} \in \mathbb{O}^{S \times T}$
3: $\hat{\mathbf{X}} = ZERO\_MEAN(\mathbf{X})$
4: $\mathbf{Y} \leftarrow \mathbf{\Lambda}^{-1}\mathbf{V}^T\hat{\mathbf{X}}$

**Algorithm 3** $\mathbf{Y} = \text{MFCC}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $\mathbf{Y} \in \mathbb{R}^{S \times P}$ ▷ $T = S$, $W = Q = T$
1: $\mathbf{C} = STFT(\mathbf{a})$ ▷ $\mathbf{C} \in \mathbb{C}^{M \times P}$
2: $\mathbf{Y}_r = MODULUS(\mathbf{C})$ ▷ $\mathbf{Y}_r \in \mathbb{R}^{M \times P}$
3: $\mathbf{R} = MEL(\mathbf{Y}_r)$ ▷ $\mathbf{R} \in \mathbb{R}^{R \times P}$, $\mathbf{X} = \mathbf{R}$
4: $\mathbf{R} \leftarrow ln(\mathbf{R})$
5: $\mathbf{V} \leftarrow COSINE\_BASIS(R, S)$ ▷ $\mathbf{V} \in \mathbb{R}^{R \times S}$
6: $\mathbf{Y} \leftarrow \mathbf{V}^T\mathbf{R}$

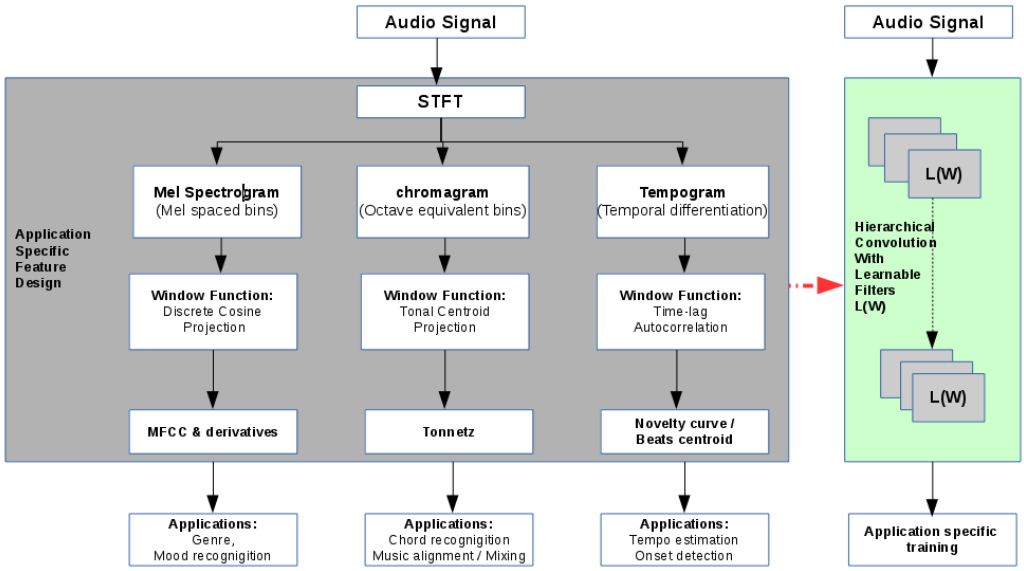



Figure 2.4: Motivation for deep architectures

## 2.3 Temporal pooling

### 2.3.1 Clustering

### 2.3.2 Recurrent Neural Networks

## 2.4 Training

# Chapter 3

# Literature Survey and Model Selection

Using content-based music information for solving several music information retrieval tasks is not new, but a decade long research efforts have been put. Hunting for the right model for our task and to justify it to be superior to the rest requires thorough understanding of evolution of such techniques. In section 3.1, the dynamics of the literature that has lead to the use of deep learning techniques for MIR tasks have been discussed. In section 3.2, the inferences from state of art techniques have been used to short list models for the experiments.

## 3.1 Literature Review

A number of surveys[13][4] amply document what is a decades-long research effort at the intersection of music, machine learning and signal processing. In a broader sense, all techniques have a two-stage architecture: first, features are extracted from music audio signals to transform them into a more meaningful representation. These features are then used as input to a classifier, which is trained to perform the task at hand. This dedicated analysis for music features emerged due to the fact that music signals possess specific acoustic and structural characteristics that distinguish them from spoken language or other non musical signals.

In a general sense, the goal of classification tasks in music informatics is to attach a semantic meaning to the content. The underlying issue is ultimately one of *organization* and *variance* of the features.

***Feature organization :*** The better organized a feature is to answer some question, the simpler it is to assign or infer semantic meaning. Thus a feature representation should explicitly reflect a desired semantic organization. That is, the information about the discriminants (refered in 2.1.1 ) should not be lost.
***Feature variance :*** A feature representation is said to be *noisy* when variance in the data is misleading or uninformative, and *robust* when it predictably encodes these invariant attributes.

Complicated classifying methods are necessary only to compensate for any noise and hence a *robust* feature representation is important.

In subsection 3.1.1, some of the early works indicating the need for better *feature organization* are discussed. In the remaining subsections, adoption of feature learning techniques for multi-label classification task are elaborated. The general motivation of all the works from section 3.1.2 - 3.1.4, was to use feature learning to obtain *robust* and *organized* features. (In section 2.2.2, how feature learning can increase *robustness* was explained) All the models (except [21]) were experimented on Magna Tag a Tune dataset(MTT)[7] with about 29K clips which are 29.1s long.

### 3.1.1 From classifier to feature emphasis

Looking back to our history before 2010, there is a clear trend in MIR of applying increasingly more powerful machine learning algorithms to the same feature representations to solve a given task. There are also ample surveys with evidence suggesting that appropriate feature representations significantly reduce the need for complex semantic interpretation methods[2]. Particularly in [5], ten different classifiers were compared on same set of features for genre classification task. It was seen that ceiling performance of 80% was achieved on GTZAN dataset, thereby suggesting the need for robust feature representation for further improvements. In [9], significant improvements were reported even for simplest classifiers by using appropriate filtering of features for chord recognition task.

### 3.1.2 From hand-crafting to feature learning

Great amount of technical research have been done to develop robust features. A number of features relevant for different MIR tasks were formulated (hand-crafted) and tested. MFCC features (ref. 2.2.1), originally developed for speech recognition task often proved efficient for genre classification and tagging. At the same time, some machine learning algorithms were also used to adopt feature learning on spectrogram frames. Several subsequent works rely on a Bag of Frames approach - where a collection of features are computed for each frame and then statistically aggregated (ref. 2.3.1). Some early feature learning approaches that proved more efficient than MFCCs are discussed in this section.

**Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. 2011 [12]:**

The pipe line of their algorithm is shown below. The formalism of the notations used are consistent with explanations in chapter 2. The PCA whitened mel-power spectrogram (ref. 2.2.1) is compared with MFCC features on Magna tag a tune dataset. It was shown that the former achieve a performance of **AUC 0.87** out performing MFCCs which was 0.77.

Signal (**a**) is sampled at 22.1 KHz. Then STFT with window length 1024 and stride 512 is computed with FFT algorithm (ref. 2.1.4). This is followed by conversion to mel power-spectrogram with 128 bins, followed by PCA Whitening which selects the top 120 variant frequencies. Another transformation is done by stacking a single layer perceptron ($L(\mathbf{W})$ means the weight matrix $\mathbf{W}$

is learned by training a neural network). The temporal pooling is done by summarizing every 2.3s frame with suitable functions (see [12] for details). The matrix $\mathbf{W}_1$ learns the optimal features for pooling. The resulting feature is then classified by two layer perceptron with 1000 hidden units with sigmoid ($\sigma$) activations.

**Algorithm 4** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^L$

$\mathbf{C} = STFT(\mathbf{a})$ ▷ $\mathbf{C} \in \mathbb{C}^{M \times P}$
$\mathbf{Y}_r = \mathbf{C} \odot \mathbf{C}$ ▷ $\mathbf{Y}_r \in \mathbb{R}^{M \times P}$
$\mathbf{R} = MEL(\mathbf{Y}_r)$ ▷ $\mathbf{R} \in \mathbb{R}^{128 \times P}$
$\mathbf{X}_1 = PCA\_WHITEN(\mathbf{R})$ ▷ $\mathbf{X}_1 \in \mathbb{R}^{120 \times P}$
$\mathbf{X}_2 = L(\mathbf{W}_1)\mathbf{X}_1$ ▷ $\mathbf{W}_1 \in \mathbb{R}^{S \times 120}, \mathbf{X}_2 \in \mathbb{R}^{S \times P}$
$\mathbf{y} = POOL(\mathbf{X}_2)$ ▷ $\mathbf{y} \in \mathbb{R}^{S.W}$
$Pred = \sigma(L(\mathbf{W}_3)\sigma(L(\mathbf{W}_2)\mathbf{y}))$ ▷ $\mathbf{W}_2 \in \mathbb{R}^{1000 \times S.W}, \mathbf{W}_3 \in \mathbb{R}^{L \times 1000}$

It is important to note that this algorithm is does not work on audio of arbitrary length because of their design of temporal pooling (because fixed sized features are needed for classification).

**Multiscale Approaches To Music Audio Feature Learning. 2012[15]:**

The result reported by this model is the current state-of-art on MTT dataset (**AUC 0.898**). Here, features are extracted from mel-power spectrogram by convolving with window functions (*gaussian pyramids*). This is done for $W$ window functions of different sizes (time length). The resulting features are then concatenated. PCA whitened frames in the mel-spectrogram are subjected to unsupervised learning with K-Means to get the Bag of Frames features (see 2.3.1). The efficient performance attributed to use of window functions of different time length suggests the existence of overlapping rhythms. (Recall the discussion about rhythmic traces in 2.1.1)

**Algorithm 5** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^L$

$\mathbf{C} = STFT(\mathbf{a})$ ▷ $\mathbf{C} \in \mathbb{C}^{M \times P}$
$\mathbf{Y}_r = \mathbf{C} \odot \mathbf{C}$ ▷ $\mathbf{Y}_r \in \mathbb{R}^{M \times P}$
$\mathbf{R} = MEL(\mathbf{Y}_r)$ ▷ $\mathbf{R} \in \mathbb{R}^{R \times P}$
**for** $i \in \{1, .., W\}$ **do**
    $\mathbf{X}_1 \leftarrow GAUSSIAN\_PYRAMID(\mathbf{R}, i)$ ▷ $\mathbf{X}_1 \in \mathbb{R}^{R \times Q1_i}$
    $\mathbf{X}_2 \leftarrow PCA\_WHITEN(\mathbf{X}_1)$ ▷ $\mathbf{X}_2 \in \mathbb{R}^{S1 \times Q1_i}$
    $\mathbf{X}_3 \leftarrow BAG\_OF\_FRAMES(\mathbf{X}_2, S2)$ ▷ $\mathbf{X}_3 \in \mathbb{R}^{S2 \times Q2_i}$
    $\mathbf{Y}[i] \leftarrow MAX\_POOL(\mathbf{X}_3)$ ▷ $\mathbf{Y}[i] \in \mathbb{R}^{S2}, \mathbf{Y} \in \mathbb{R}^{S2 \times W}$
**end for**
$\mathbf{y} = FLATTEN(\mathbf{Y})$ ▷ $\mathbf{y} \in \mathbb{R}^{S2.W}$
$Pred = \sigma(L(\mathbf{W}_3)ReLU(L(\mathbf{W}_2)\mathbf{y}))$ ▷ $\mathbf{W}_2 \in \mathbb{R}^{1000 \times S2.W}, \mathbf{W}_3 \in \mathbb{R}^{L \times 1000}$

The take-away is that modelling relation between features at rhythmic intervals does help.

### 3.1.3 Transfer Learning by supervised pre-training

Stacked feature learning techniques typically require large amounts of training data to work well. But sometimes, features learned on large datasets can be used for other datasets, either as a *black-box* extractor (feature extractor is not further trained on target dataset) or as a *fine-tuned* feature extractor (feature extractor is further trained after initializing weights). To do this, it is essential to have a source task that requires a very rich feature representation, so as to ensure that the information content of this representation is likely to be useful for other tasks

**Transfer learning by supervised pre-training for audio-based music classification. 2014[17]:**

In this research, features trained on MSD dataset ( 1000K clips) are used as *black-box* feature extractor while training on MTT dataset and the resulting classification performance still achieved **AUC 0.88** outperforming baseline MFCC. The workflow for source and target are shown below,

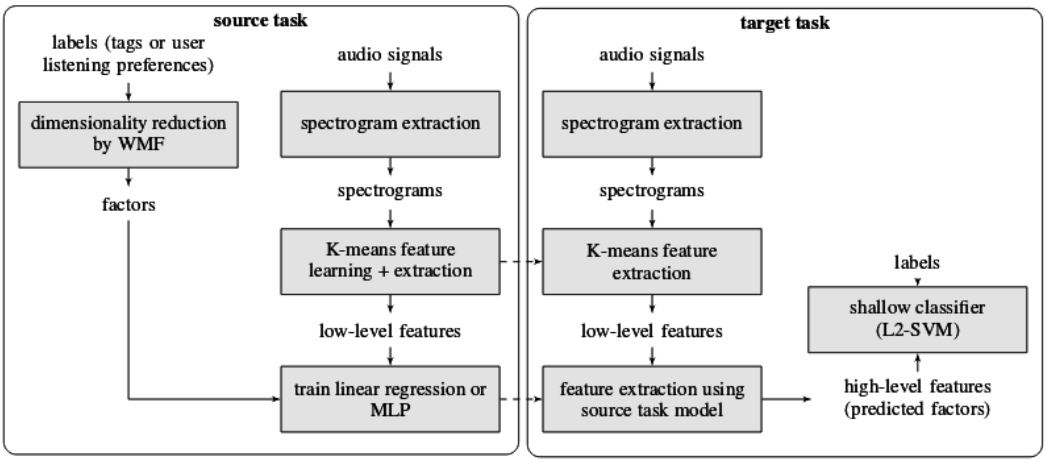



Figure 3.1: Schematic overview of the workflow of transfer learning[17]

**Source task**: The low-level features from audio spectrograms are learned through unsupervised learning by spherical K-Means. A multi layer perceptron is then stacked to obtain final prediction. So the output from the penultimate layer of MLP are treated as transferable features. To tackle problems created by redundant and sparse labels, dimensionality reduction is done in the label space using PCA. The model is then trained to predict the reduced label representation.

**Target task** Next, the trained models are used to extract features from other datasets, which are then passed to train shallow classifiers for different but related target tasks. This workflow is visualized in figure 3.1. Dashed arrows indicate transfer of the learned feature extractors from the source task to the target task.

### 3.1.4 Convolutional Neural Networks

It can be seen that deep signal processing structures can be realized by stacking multiple shallow architectures (ref. 2.2.2). As feature learning was proving to be more efficient than hand crafted features, stacking learnable layers over one another became a hot area of research. The idea was to replace the application specific dimension reductions with hierarchy of learnable convolution filters.

**End-to-end learning for music audio. 2014[16]:**

As shown in chapter 2, all operations including FFT can be defined in terms of convolutions. In this research they investigate whether it is possible to apply feature learning directly to raw audio signal. The signal was convolved with 3 layers of 1D convolutions followed by two fully connected layers. Thus, the feature and the classifier was trained in a single pipeline and this is called *end to end learning*. They compared the *end to end learning* approach with convolutions from mel-spectrogram on MTT dataset (i.e, retaining STFT). Their algorithm is described below. Function $f$ is an element-wise logarithmic compression. It was found that, discarding STFT hurt the performance. CNN from mel-spectrogram achieved **AUC 0.8815**, but on including convolutions on audio signal AUC dropped to 0.8487.

**Algorithm 6** CNN(raw audio) [**0.84**]

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^L$

1: $\mathbf{C}_1 = f(\mathbf{a} \star \mathbf{w}^{(256)}_{(256)})$
2: $\mathbf{C}_2 = MaxPool(ReLU(\mathbf{C}_1 \star \mathbf{w}^{(1)}_{(32)}))$
3: $\mathbf{C}_3 = MaxPool(ReLU(\mathbf{C}_2 \star \mathbf{w}^{(1)}_{(32)}))$
4: $\mathbf{y} = FLATTEN(\mathbf{C}_3)$
5: $Pred = \sigma(L(\mathbf{W}_2)ReLU(L(\mathbf{W}_1)\mathbf{y}))$

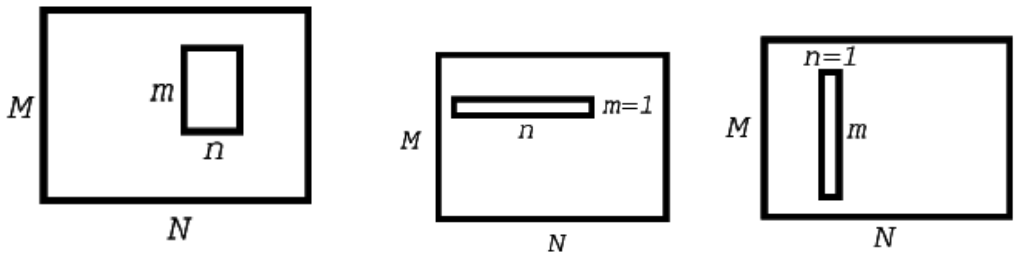
**Algorithm 7** CNN(Mel-Spectrogram) [**0.88**]

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^L$

1: $\mathbf{R} = f(MEL(||STFT(\mathbf{a})||^2))$
2: $\mathbf{C}_1 = MaxPool(ReLU(\mathbf{R} \star \mathbf{w}^{(1)}_{(32)}))$
3: $\mathbf{C}_2 = MaxPool(ReLU(\mathbf{C}_1 \star \mathbf{w}^{(1)}_{(32)}))$
4: $\mathbf{y} = FLATTEN(\mathbf{C}_2)$
5: $Pred = \sigma(L(\mathbf{W}_2)ReLU(L(\mathbf{W}_1)\mathbf{y}))$

**Experimenting with musically motivated convolutional neural networks. 2016[21]:**

In the previous section, only 1D convolution with filter sizes directly motivated by hand-crafted methods were tested for comparison. But usually, the convolution operation allows flexibility in choosing the filter sizes. In this research, the authors discuss how convolution filters with different shapes can fit specific musical concepts.




(a) Rectangular filter (b) Time filter (c) Frequency filter

Figure 3.2: Different Filter sizes

*Time filters* can learn temporal cues (Onset, BPM and other rhythmic patterns), while *frequency filters* can differentiate timbre and note. *Rectangular filters* can learn short time sub-bands (Bass, kick, drums)[21]. However, because of hierarchical nature of deep networks, any filter should be theoretically capable of picking up the relevant cues. It was shown in their experiments that *rectangular filters* or combination of time and frequency filters performed better than using just time / frequency filter. These experiments were however done for a genre classification task.

**Automatic tagging using deep convolutional neural networks. 2016[18]:**

Different CNN architectures were tested and the proposed model achieves close to state of art performance on MTT dataset (**0.894 AUC**). The audio samples were down-sampled to 12 KHz and convolutions were started from mel spectrogram (96 bins). They also compared MFCCs, convolutions over STFT and convolutions over Mel-log power spectrogram and report that the latter performs significantly better.

| **Model** | **AUC** |
|---|---|
| STFT → CNN | 0.846 |
| STFT → MEL → CNN | **0.894** |
| STFT → MEL → MFCC | 0.862 |

Also, to exploit the advantage of PCA Whitening proven in [15][12], Batch Normalization of frequency components is done. That is, data is centred to the batch mean and divided by batch variance. In Batch normalization however, the basis is not switched but the data is *learned* to be scaled and shifted.

**Algorithm 8** $\hat{\mathbf{X}} = \text{BATCHNORM}(\mathbf{X})$

**Input :** $\mathbf{X} \in \mathbb{R}^{B \times S \times Q}$, ▷ $B$ is batch size
**Output :** $\hat{\mathbf{X}} \in \mathbb{R}^{B \times S \times Q}$
**Parameters to learn :** $\gamma$ (Scale), $\beta$ (Shift)

$\boldsymbol{\mu}, \boldsymbol{\sigma}^2 = FREQUENCY\_MEAN\_VARIANCE(\mathbf{X})$ ▷ $\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \in \mathbb{R}^S$
**for** $i \in \{1, .., B\}$ **do**
  **for** $j \in \{1, .., Q\}$ **do**
    $\mathbf{X}[i, :, j] \leftarrow \frac{\mathbf{X}[i,:,j] - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 - \epsilon}}$
  **end for**
**end for**
$\hat{\mathbf{X}} = \gamma \mathbf{X} + \beta$

The five layer proposed CNN architecture is shown below. The filters $\mathbf{W}$ in each layer are the weights that will be learned. *Spatial_Bn* is similar to the normalization algorithm mentioned above, except that the normalization is done along the 1st axis of tensors $\mathbf{C}$. $MaxPool_{i,j}$ is a dimensionality reduction done by pooling $(i, j)$ elements along $S$ and $Q$ directions respectively. *Elu* is an element-wise non-linearity operation described in section 2.4

**Algorithm 9** $\mathbf{y} = CHOI\_CNN(\mathbf{R})$

**Input :** $\mathbf{R} \in \mathbb{R}^{1 \times 96 \times 1366}$
**Output :** $\mathbf{y} \in \mathbb{R}^{1024}$

$\mathbf{R}_n = BatchNorm(\mathbf{R})$
$\mathbf{C}_1 = \mathbf{R}_n \star \mathbf{W1}^{(1,1)}_{(32)}$ ▷ $\mathbf{W1} \in \mathbb{R}^{32 \times 3 \times 3}, \mathbf{C}_1 \in \mathbb{R}^{32 \times S1 \times Q1}$
$\mathbf{C}_1 \leftarrow MaxPool_{(2,4)}(Elu(Spatial\_Bn(\mathbf{C}_1)))$ ▷ $\mathbf{C}_1 \in \mathbb{R}^{32 \times T1 \times W1}$
$\mathbf{C}_2 = \mathbf{C}_1 \star \mathbf{W2}^{(1,1)}_{(128)}$ ▷ $\mathbf{W2} \in \mathbb{R}^{128 \times 3 \times 3}, \mathbf{C}_2 \in \mathbb{R}^{128 \times S2 \times Q2}$
$\mathbf{C}_2 \leftarrow MaxPool_{(2,4)}(Elu(Spatial\_Bn(\mathbf{C}_2)))$ ▷ $\mathbf{C}_2 \in \mathbb{R}^{128 \times T2 \times W2}$
$\mathbf{C}_3 = \mathbf{C}_2 \star \mathbf{W3}^{(1,1)}_{(128)}$ ▷ $\mathbf{W3} \in \mathbb{R}^{128 \times 3 \times 3}, \mathbf{C}_3 \in \mathbb{R}^{128 \times S3 \times Q3}$
$\mathbf{C}_3 \leftarrow MaxPool_{(2,4)}(Elu(Spatial\_Bn(\mathbf{C}_3)))$ ▷ $\mathbf{C}_3 \in \mathbb{R}^{128 \times T3 \times W3}$
$\mathbf{C}_4 = \mathbf{C}_3 \star \mathbf{W4}^{(1,1)}_{(192)}$ ▷ $\mathbf{W4} \in \mathbb{R}^{192 \times 3 \times 3}, \mathbf{C}_4 \in \mathbb{R}^{192 \times S4 \times Q4}$
$\mathbf{C}_4 \leftarrow MaxPool_{(2,4)}(Elu(Spatial\_Bn(\mathbf{C}_4)))$ ▷ $\mathbf{C}_4 \in \mathbb{R}^{192 \times T4 \times W4}$
$\mathbf{C}_5 = \mathbf{C}_4 \star \mathbf{W5}^{(1,1)}_{(256)}$ ▷ $\mathbf{W5} \in \mathbb{R}^{256 \times 3 \times 3}, \mathbf{C}_5 \in \mathbb{R}^{256 \times S5 \times Q5}$
$\mathbf{C}_5 \leftarrow Elu(Spatial\_Bn(\mathbf{C}_5))$
$\mathbf{y} = Flatten(\mathbf{C}_5)$ ▷ $\mathbf{y} \in \mathbb{R}^{1024}$

The features from convolutions then pass through a fully connected layer of size equalling number of tags. The authors have then trained this model on MSD dataset and made the weights publicly available.

## 3.2 Model Selection

In section 3.1, it was stated that when the feature is well *organized* and encodes the *variance* in the data, it becomes easier to attach a semantic meaning. Feature learning can increase robustness, but to learn an organized representation is not guaranteed. That is to say, the extracted feature should encode the information about it's discriminants related to the task. Our brain differentiates sounds with energy changes, and this is approximated by MFCCs (ref 2.2.1) through proportionate energy variance from a mel-spaced spectrogram (ref 2.1.4). But in section 2.1.1, it was argued that the difference between music and speech is that, a music signal is composed of several superimposed *rhythmic traces*. It was not clear if the classifier could decompose the rhythms from engineered features and hence there was this movement from feature engineering to feature learning. The results from the work [15], where fully unsupervised technique is adopted for feature learning, shows that hand-crafted features does lose some information necessary for classification. But even the learned features were extracted from mel-spaced frequency spectrogram that does not exploit the harmonic encodings. That is, we still do not know if the learning algorithms extract the rhythms, thereby questioning the optimality of *feature organization* (i.e, would the features learned for one task be optimal for auxiliary but related task?). However in general, it could be seen that feature learning performs better than MFCCs for multi-label classification task.

Music tagging problem is further complicated by the complexity of semantic assignments that reflect user preference. To get the right discriminants, the training method should be properly defined in the first place. In section 1.1.2, the problems with content based methods that stem from training assumptions were pointed out. One of which was the social-factor assumption resulting from training on large datasets. But we want train on a specialized dataset which is small. This leaves us with the question, if the models trained by supervised learning on larger datasets[18][16] can be used for smaller datasets with different label-context. Secondly, the assumptions resulting by training on short excerpts of music rather than whole song cause vagueness. This is because, the currently available large datasets only contain short clips and the current algorithms generalize the tags for the whole song by merging tags from short sections of the song. Therefore, methods that hold better feature organization for songs of arbitrary length are also explored.

### 3.2.1 Transfer learning Vs MFCC

To check if models trained on large datasets can be exploited for smaller datasets, *transfer learning* from the model which achieves state of art performance with CNN[18] is compared with MFCC features. (It makes perfect sense to compare the state of art unsupervised feature learning algorithm[15] as well, but in this thesis I stick to analysing CNNs). This will also show if features learned (stacked convolutions - ref. 2.2.2) through supervised training on large dataset attain better *feature organization* than MFCCs. It is important to note that, better *feature organization* simply does not mean that classifier identifies the *rhythmic traces*.

### 3.2.2 Bag Of Frames vs RNN

To summarize tags for songs of arbitrary length, most of the current algorithms classify short sections of the spectrogram separately and finally merge tags across different sections[16]. It is also

possible to stack a *decision tree* over section-wise tags and improve performance, but that would not tell anything about the optimality of *feature organization*. Hence for temporal pooling, only methods that directly work on content information are considered. Algorithm in [15] is designed to handle songs of arbitrary length and it was seen that Bag of Frames features trained using K-Means algorithm (see 2.3.1) proved efficient while testing on 29.1s excerpts from MTT dataset. However, it is not clear if these features are optimal choice for identifying rhythms. It is also not known if the efficiency of K-Means will be retrained when tested on songs longer than 30s. Hence, this algorithm is compared with temporal approximation using Recurrent Neural Network (see 2.3.2). Supervised training with RNN might force the classifier to look for rhythmic content.

# Chapter 4

# Experiments and Results

The aim of this thesis is to find the optimal algorithm for content-based multi-label classification of music tracks, that can be solved with minimal training data. We are looking for a classifier that can discriminate aesthetics in music, but the public datasets are socially biased and contains only short excerpts. Hence the multi-label classifiers have to be tested on our unbiased target dataset which has full clips. In Chapter 3, the state of art models were reviewed and in section 3.2, the short-comings of these algorithms in addressing the problems discussed in Chapter 1 (see 1.1.2) were pointed out. In this chapter, the experiments that will lead to finding the best algorithm using the components short-listed in 3.2 will be described.

## 4.1 Dataset and Evaluation

More specific to our task than representing audio is finding a proper dataset of labelled pairs. Recalling that our aim is to identify the aesthetic properties of music from the audio content, a dataset that is free from *audio-semantic* noise is needed. Furthermore, to test *transfer learning*, a large dataset is needed for the *source task* (ref. 3.2.1). Popularly used *Million Song Dataset* (MSD) [11] contains a cluster of complimentary datasets, most of them annotated with *social tags*. For instance, *Last.fm* which forms a part of MSD contains annotations from users of an online radio application. But such social tags contribute to the *audio-semantic* noise which we want to eliminate. The dataset that is mostly used for evaluating content-based algorithms is *Magna Tag A Tune* dataset [7], where annotations are gathered through a game application that attracts users who are familiar with technical terms related to music. Hence the tags in this dataset are usually clean. Hence this would be a decent choice for our *source task*.

### 4.1.1 Dataset for source task

The MagnaTagATune dataset consists of 25,856 clips of 29.1-s, 16 kHz-sampled mp3 files with 188 tags. These annotations are gathered from an online game called *Tag a Tune*. A player is partnered up with another random player who cannot be communicated. Both listen to some track, and have to select appropriate tags. Then the players are asked one simple question : "Are we listening to same song?". Answering this correctly will earn them points. This dataset is the largest available

that comes close to minimizing the *audio-semantic* noise. However, social factor still plays a role here. This is partly indicated by tag frequency where the most frequent tag is used 4,851 times while the $50^{th}$ most frequent one used 490 times in the training set. We use only the top-50 tags for training from this dataset.

### 4.1.2 Dataset for target task

To gather annotations with clean mapping to aesthetic properties, we adopt the most straightforward and costly method - ask someone to listen to songs and tag them. Around 900 songs approximately 5 - 8 min long, were tagged by my supervisor Prof. Paolo Bientinesi in association with Prof. Marco Aluno (Professor of Composition and Theory at University EAFIT, Columbia). Out of 900 songs, 100 are used for validation.

### 4.1.3 Evaluation metrics

For multi-label classification with L labels, the performance of L binary classifiers is computed and averaged. Each label can belong to one of the class - *positive* (1) or *negative* (0). To discuss about a fair performance measure, lets first recall some important terminologies,

**True Positives (TP) :** If the classifier admits a label as positive when the ground truth is also positive.
**True Negative (TN) :** If the classifier admits a label as negative when the ground truth is also negative.
**False Positives (FP) :** If the classifier admits a label as positive when the ground truth is negative.
**False Negative (FN) :** If the classifier admits a label as negative when the ground truth is positive.

For a general tagging problem, most of the tags are *negative* for most of the clips (That is, out of 900 songs, if 20 songs have the tag '*electro*', then for this tag there are 20 *ground truth positives* and 880 *ground truth negatives*.

**Accuracy :** To see why *accuracy* will be an unfair measure, lets look at the definition of *accuracy* for the label 'electro',

$$Accuracy = \frac{\sum TP + \sum TN}{900} = \frac{0 + 880}{900}$$

Even if the classifier did not classify one 'electro' as positive, accuracy will be 0.98 with the contribution from 880 true negative samples.

**Precision :** This is also not a comprehensive measure to indicate the strength of the classifier because it does not tell anything about the percentage of negatively classified samples (false negatives). That is, even if the classifier *correctly* admits one 'electro' as positive and 899 as negative, *precision* would be 1.0

$$Precission = \frac{\sum TP}{\sum TP + \sum FP} = \frac{1}{1 + 0}$$

**Recall :** This is also not comprehensive because it does not tell anything about false positives or in other words, it does not say if your classifier is a *liar*. That is, even if the classifier admits 900 samples as positive (20 true positives and 880 false positives), *recall* will be 1.0

$$Recall = \frac{\sum TP}{\sum TP + \sum FN} = \frac{20}{20 + 0}$$

To strike a balance between *recall* and *precision*, the harmonic mean of both is often used, which is called *F1* score. But to calculate all the metrics mentioned so far, some classifier threshold is required (That is, a binary classifier spits a number between 0 and 1 and if the number is above the threshold, the sample is classified as positive, otherwise negative). If someone changes the threshold then the performance can change. To find a comprehensive measure for classifier performance, the metric should consider all threshold values.

**Area under precision-recall curve :** When the *precision* and *recall* are plotted for various threshold and the area under the precision-recall curve is found, the metric is termed as *average precision*. Averaging the *average precision* of L labels gives *Mean average precision*, which is a useful metric and can be found in some research works.

**Area under receiver operating characteristic curve (AUC) :** The *fall-out* and *recall* are plotted for various threshold and the area under the this curve is found. *Fall-out* is defined as

$$Fall\_out = \frac{\sum FP}{\sum FP + \sum TN}$$

*Recall* answers the question, 'when the ground truth is positive, how often does the classifier admit it as a positive'. *Fall-out* answers the question, 'when the ground truth is negative, how often does the classifier admit it as positive'. A random classifier would have an AUC of 0.5, meaning, for a binary random classification of an unknown sample there is 50% chance for it to be true. Therefore, AUC can be thought of as a probability that a classifier would rank a randomly chosen positive ground truth higher than a randomly chosen negative observation. AUC measures how well the positive and negative classes are separated. AUC is computed for L labels and averaged. Since the publications reviewed in previous chapter report this metric, we will also use the same. But in addition, we use *weighted average* because our validation set is small and number of occurrences of each label is not balanced.

Therefore, in all our experiments, *Weighted averaged AUC* (WAUC) will be reported.

## 4.2 Experiments

As with any MIR task, the raw audio signal containing amplitude values in time domain is first down sampled and representation parameters are fixed (ref. 2.1.4). This is because computational cost is heavily affected by the size of the input layers.

**Sampling Parameters :** Although most of the available audio in digital format are sampled at 44KHz (ref. 2.1.2), it is important to note that most of the information lie in the lower range of the spectrum. In [18], a pilot experiment was conducted to demonstrate similar performance with 12KHz and 16KHz for top 50 tags (Recall that MTT clips are already downsampled to 16KHz). Hence we sample all our tracks to 12KHz.

Next step is to extract relevant features. General pipeline is *sampling* (ref. 2.1.2), *representation* (ref. 2.1.4), stacks of *dimensionality reduction* (ref. 2.2) followed by *temporal summarization* (ref. 2.3). Feature learning can be introduced at different stages. Introducing in earlier stages would require training with huge amount of data. Feature learning on raw sampled signal proved sub-optimal in [16]. Same was the case when convolved over STFT frame [18]. Convolutions over mel-spectrogram performed better than MFCC in many previous work [18][16]. Hence we stick to engineered features until the extraction of mel-spectrogram.

**Mel-Spectrogram Parameters :** The signal in the time domain is converted to frequency domain by Short-Time-Fourier-Transform ($STFT$) using Fast-Fourier-Transform(FFT) algorithm. The arguments for doing this were presented in Chapter 2 (ref. 2.1.4). The parameters for FFT are the size (often referred as FFT Size) and stride (often referred as hop-length) of the window function. FFT size was fixed to 512 (42 ms) and hop-length was fixed to 256. Motivated by the human auditory system, the frequency axis is binned to mel-scale and log of squared STFT coefficients (proportional to loudness) are calculated. In [18], it was stated that 96 mel-bins were optimum.

Feature learning over mel-spectrogram still requires large dataset, but our target dataset is small. Hence by questing the effectiveness of feature learning over spectrogram with MFCCs, we decided to compare both.

### 4.2.1 Experiments with pre-trained CNNs as feature extractor

In Chapter 2 (ref. 2.2.2), it was shown how Convolution Neural Networks (CNN) are motivating to be used as feature extractor for music signal. In Chapter 3 (ref. 3.1.4) some of the successful mel-spectrogram convolution architectures trained on MTT dataset showing state-of-art performance were discussed. Now we would like to see if such features extracted through these models can be used for auxiliary tasks. That is to say, the learned CNN parameters (weights) from *source* dataset are used as initialization setting for *target* tasks.

The CNN architecture from [18] achieves the best AUC score on MTT dataset and hence this architecture is used for feature extraction. The algorithm for their model ($CHOI\_CNN$) is explained in section 3.1.4 (Algorithm 9). The input to their CNN was 29.1s mel-spectrogram with representation parameters mentioned above (Thus resulting in 1366 time samples). So for our task, features are extracted every 29.1s and sequentially sent to RNN. The temporal summarization is done by 2 layer *Long Short-Term Memory Recurrent Neural Network* (ref. 2.3.2). The RNN module does sequence to one mapping ($Seq2One$) of the input features. This entire model is then trained for 150K iteration on MTT dataset with top 50 tags. Their model was already trained on Million Song Dataset [11] with top 50 *Last.fm* tags. We just fine-tune their model on MTT dataset after

merging clips from same song. Features of clips from same song are sequentially given as input to RNN with a dropout (ref. 2.4) of 0.3 after each layer, which then projects to a fixed sized feature vector. The output of RNN is then passed to a fully connected layer with 50 output units and *sigmoid* activation. ADAM optimizer with *binary-cross-entropy* loss function is used for training. The starting learning rate is 0.001, decaying at $1^{-8}$ and beta 0.99 (ref. 2.4). Algorithm for $L$ labels is described below and the notations used are consistent with formalisms in Chapter 2. The algorithm is implemented in *Torch* [torch] and mel-spectrogram was extracted using *Librosa* [librosa]

**Algorithm 10** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^L$

$\mathbf{C} = STFT(\mathbf{a})$ ▷ $\mathbf{C} \in \mathbb{C}^{M \times P}$
$\mathbf{Y}_r = Log(\mathbf{C} \odot \mathbf{C})$ ▷ $\mathbf{Y}_r \in \mathbb{R}^{M \times P}$
$\mathbf{R} = MEL(\mathbf{Y}_r)$ ▷ $\mathbf{R} \in \mathbb{R}^{96 \times P}$
$W = floor(\frac{P}{1366})$
**for** $i \in \{0, .., W\}$ **do**
    $\mathbf{X} \leftarrow \mathbf{R}[:][i : (i+1).1366]$ ▷ $\mathbf{X} \in \mathbb{R}^{96 \times 1366}$
    $\mathbf{Y}[i] \leftarrow CHOI\_CNN(\mathbf{X})$ ▷ $\mathbf{Y} \in \mathbb{R}^{1024 \times W}$
**end for**
$\mathbf{Y}_1 = Drop_{(0.3)}(Seq2Seq\_LSTM(\mathbf{Y}))$ ▷ $\mathbf{Y}_1 \in \mathbb{R}^{1024 \times W}$
$\mathbf{y}_2 = Drop_{(0.3)}(Seq2One\_LSTM(\mathbf{Y}_1))$ ▷ $\mathbf{y}_2 \in \mathbb{R}^{1024}$
$Pred = \sigma(L(\mathbf{W})\mathbf{y}_2))$ ▷ $\mathbf{W} \in \mathbb{R}^{L \times 1024}$

This CNN model can either be used as a *black-box* feature extractor (That is, weights of the model are not modified while training the *target-task*) or certain layers can be *fine-tuned* (That is, we continue the training on *target-task*). Both the cases are looked separately,

**Blackbox CNN + RNN :**
The weights of CNN trained on the source task are not modified (no fine-tuning). The weights of RNN are also initiazied with those trained on source task. The fully-connected layer in the source task is changed to 65 output units. (i.e 65 labels). The network is trained by back-propagating through the fully connected layer and RNN with *binary-cross entropy* loss function (ref. 2.4). The optimization parameters are same as that of *source task*. Training is stopped after 25K iterations, after which the model begins to over-fit. Weighted averaged AUC (WAUC) was **0.65**

**Fine-tune CNN + RNN :**
With the same parameter settings, the last layer of CNN was finetuned after 5K iterations. Training was then continued until 25K iterations and WAUC went up to **0.69**. When last two CNN layers were finetuned WAUC further improved to **0.71**. Fine-tuning earlier layers proved sub-optimal.

This tells us that in the final layers CNN tend to find features specific to task. (Recalling that labels for source and target tasks are different). However, we still do not know if convolutions over

mel-spectrogram will be better than MFCCs which is proven to model audio discriminants. Before going there, the effectiveness of RNN should also be questioned. It was hypothesised in Chapter 3 (3.2) that Bag-of-Frames (2.3.1) features using K-Means (which was actually used in [15] to attain state of art performance) may not be suitable for summarizing features for longer audio. So we test this by replacing RNN with Bag Of Frames (BoF) features.

**CNN + BoF :** The CNN is first finetuned for 10K iterations with algorithm 10. Then 1024 centroids of CNN features are found by unsupervised training on both MTT and our target dataset. This is followed by multi layer perceptron with a hidden layer of 512 units and ReLU activation. Having hidden size of 1024 did not improve the result. WAUC was **0.67**. The algorithm is described below,

**Algorithm 11** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^{65}$

$\mathbf{C} = STFT(\mathbf{a})$ ▷ $\mathbf{C} \in \mathbb{C}^{M \times P}$
$\mathbf{Y}_r = Log(\mathbf{C} \odot \mathbf{C})$ ▷ $\mathbf{Y}_r \in \mathbb{R}^{M \times P}$
$\mathbf{R} = MEL(\mathbf{Y}_r)$ ▷ $\mathbf{R} \in \mathbb{R}^{96 \times P}$
$W = floor(\frac{P}{1366})$
**for** $i \in \{0, .., W\}$ **do**
    $\mathbf{X} \leftarrow \mathbf{R}[:][i : (i+1).1366]$ ▷ $\mathbf{X} \in \mathbb{R}^{96 \times 1366}$
    $\mathbf{Y}[i] \leftarrow CHOI\_CNN(\mathbf{X})$ ▷ $\mathbf{Y} \in \mathbb{R}^{1024 \times W}$
**end for**
$\mathbf{y}_1 = BagOfFrames(\mathbf{Y}, 1024)$ ▷ $\mathbf{y}_1 \in \mathbb{R}^{1024}$
$Pred = \sigma(L(\mathbf{W}_2)ReLU(L(\mathbf{W}_1)\mathbf{y}_1))$ ▷ $\mathbf{W}_2 \in \mathbb{R}^{65 \times 512}, \mathbf{W}_1 \in \mathbb{R}^{512 \times 1024}$

### 4.2.2 Experiments with MFCCs as feature extractor

MFCCs are still de-facto standard for classifications on small datasets. If CNNs had to outperform MFCCs, the learned parameters should have to encode discriminants similar to MFCCs. MFCCs are computed by taking discrete-cosine transform on log mel-spectrogram (ref. 2.2.1). Following the comparison strategies from [18], we retain 30 coefficients, their first and second derivative, resulting in a vector of size 90 for each STFT frame.

**MFCC + RNN :**
Now, MFCCs from every STFT window is passed in a *30s Batched sequence* to a Sequence to one LSTM, which results in a projection for every 30s window. These sequence of 30s frames are then passed to another Sequence to One LSTM to get a final projection. This is done because a MFCCs from STFT frame will result in a long sequence and RNNs tend to forget the information in the earlier sequence samples. This network was first trained with MTT dataset before our target dataset. The parameter setting are same as those used while training **CNN+RNN**. The resulting WAUC was **0.74**

**Algorithm 12** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^{65}$
$\mathbf{R} = MFCC(\mathbf{a})$ ▷ $\mathbf{R} \in \mathbb{R}^{90 \times P}$
$W = floor(\frac{P}{1366})$
**for** $i \in \{0, .., W\}$ **do**
    $\mathbf{X} \leftarrow \mathbf{R}[:][i : (i+1).1366]$ ▷ $\mathbf{X} \in \mathbb{R}^{90 \times 1366}$
    $\mathbf{Y}[i] \leftarrow Drop_{(0.3)}(Seq2One\_LSTM(\mathbf{X}))$ ▷ $\mathbf{Y} \in \mathbb{R}^{1024 \times W}$
**end for**
$\mathbf{y} = Drop_{(0.3)}(Seq2One\_LSTM(\mathbf{Y}))$ ▷ $\mathbf{y} \in \mathbb{R}^{1024}$
$Pred = \sigma(L(\mathbf{W})\mathbf{y}))$ ▷ $\mathbf{W} \in \mathbb{R}^{65 \times 1024}$

**MFCC + BoF :**
RNN is repalaced with Bag of Frames features. Now WAUC drops to **0.62**

**Algorithm 13** $Pred = \text{MODEL}(\mathbf{a})$

**Input :** $\mathbf{a} \in \mathbb{R}^N$
**Output :** $Pred \in \mathbb{R}^{65}$
$\mathbf{R} = MFCC(\mathbf{a})$ ▷ $\mathbf{R} \in \mathbb{R}^{90 \times P}$
$\mathbf{y} = BagOfFrames(\mathbf{R}, 1024)$ ▷ $\mathbf{y} \in \mathbb{R}^{1024}$
$Pred = \sigma(L(\mathbf{W}_2)ReLU(L(\mathbf{W}_1)\mathbf{y}))$ ▷ $\mathbf{W}_2 \in \mathbb{R}^{65 \times 512}, \mathbf{W}_1 \in \mathbb{R}^{512 \times 1024}$

## 4.3 Summary of Results

Summary of results is shown in the table below. It is seen that *transfer learning* of convolutions over mel-spectrogram with architecture in [18] cannot match with MFCCs for small datasets. This indicates that convolutional features from source dataset are more task-specific. It is also seen that *Recurrent Neural Networks* perform better in summarizing features for longer audio. This indicates the existence of rhythmic patterns that discriminate music.

| **Model** | **AUC** |
|---|---|
| Finetune CNN + RNN | 0.71 |
| CNN + BoF | 0.67 |
| MFCC + RNN | **0.74** |
| MFCC + BoF | 0.62 |

# Chapter 5

# Conclusion

The model settings for the task of *aesthetic tagging* have been analysed in this thesis. From the machine learning end, the performance can be further pushed by

- Searching or testing other CNN models[22]
- Working with the label space. For instance, there can be broad subsets and each tag can be belong to one or many of the subset. Then a separate model is trained for each subset

However, it is seen that the performance gap is still huge to serve any real-time application arising from aesthetic auto-tagging. Hence, to develop an algorithm that could come close to human artist, just a dataset with clean tags is not sufficient but also proper understanding of mathematical modelling of musical discriminants (ref. 2.1.1) is important.

**TODO:** more ideas, discussion

# Appendix A

# Appendix

## A.1 Basis Transformation

Here we discuss only transformation from standard basis or Cartesian basis.

The standard basis for $\mathbb{R}^N$ is the ordered sequence $\mathbf{I}_n = [\mathbf{e}_1, \mathbf{e}_2, .., \mathbf{e}_n]$, where $\mathbf{e}_i$ is a vector with 1 in $i^{th}$ place and 0 elsewhere. Any vector $\mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^N$ can be represented as a linear combination of $\mathbf{I}_n$ as,

$$\mathbf{x} = \sum_{i=1}^{N} x_i \mathbf{e}_i = \mathbf{I}_n \mathbf{x}$$

**Basis transformation** from standard basis is defined as representing the same vector $\mathbf{x}$ with the new co-ordinates $[y_1, y_2, ..., y_m]$ in basis $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_m] \in \mathbf{R}^{N \times M}$.

$$\mathbf{x} = \sum_{i=1}^{M} y_i \mathbf{v}_i = \mathbf{V}\mathbf{y} \qquad \mathbf{y} \in \mathbb{R}^M$$

$\mathbf{V}$ is also known as **change of coordinates matrix** (also stated as any matrix whose columns form a basis). If $\mathbf{V}$ is orthogonal, then $\mathbf{V}^{-1} = \mathbf{V}^T$ and hence $\mathbf{y} = \mathbf{V}^T\mathbf{x}$

## A.2 Convolution

Only discrete convolutions with finite support are discussed below,

### A.2.1 1D Convolution

Convolution of a vector $\mathbf{f}$ with filter $\mathbf{w}_k$ of stride $s$ is defined as,

$$\mathbf{C}(k, i) = \sum_{n=i.s}^{i.s+F} \mathbf{f}(n)\mathbf{w}_k(n - i.s) \qquad \mathbf{f} \in \mathbb{R}^N, \mathbf{w}_k \in \mathbb{R}^F, \mathbf{C} \in \mathbb{R}^{K \times I} \tag{A.1}$$

$$\mathbf{C}(k,i) = \mathbf{f}(n) \star \mathbf{w}_k(n - i.s)$$

Where:

$K$ is the number of filters. $k \in 0, 1...K-1$

$I$ is the number of contractions. $i \in 0, 1...I-1$

**Short Hand Notation :** 1D Convolution of $\mathbf{f}$ with filter $\mathbf{w}_k$ with stride $s$

$$\boxed{\mathbf{C}(k,:) = \mathbf{f} \star \mathbf{w}_k^{(s)}}$$

### A.2.2 2D Convolution

Convolution of a matrix $\mathbf{F}$ with filter $\mathbf{W}_k$ of row-stride $s$ and column-stride $t$ is defined as,

$$\mathbf{C}(k,j,i) = \sum_{n=i.s}^{i.s+F} \sum_{m=j.t}^{j.t+G} \mathbf{F}(m,n) : \mathbf{W}_k(m - j.t, n - i.s) \qquad \mathbf{F} \in \mathbb{R}^{M \times N}, \mathbf{W}_k \in \mathbb{R}^{G \times F}, \mathbf{C} \in \mathbb{R}^{K \times J \times I} \tag{A.2}$$

$$\mathbf{C}(k,j,i) = \mathbf{F}(m,n) \star \mathbf{W}_k(m - j.t, n - i.s)$$

**Short Hand Notation :** 2D Convolution of $\mathbf{F}$ with filter $\mathbf{W}_k$ with row-stride $s$ and column-stride $t$

$$\boxed{\mathbf{C}(k,:,:) = \mathbf{F} \star \mathbf{W}_k^{(s,t)}}$$

# Bibliography

## Proceedings


[9] Taemin Cho, Ron J. Weiss, and Juan P. Bello. "Exploring common variations in state of the art chord recognition systems". In: *In Proc. of the Sound and Music Computing Conf. (SMC.* 2010.

[11] Thierry Bertin-mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. "The million song dataset". In: *In Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR.* 2011.

[12] Simon Lemieux, Yoshua Bengio, Douglas Eck, and Universit De Montral. "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio". In: *In: Proc. 12th International Society for Music Information Retrieval Conference.* 2011.

[15] Sander Dieleman and Benjamin Schrauwen. "Multiscale Approaches To Music Audio Feature Learning". In: *ISMIR.* 2013.

[16] S. Dieleman and B. Schrauwen. "End-to-end learning for music audio". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2014, pp. 6964–6968. DOI: 10.1109/ICASSP.2014.6854950.

[17] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Transfer learning by supervised pre-training for audio-based music classification". eng. In: *Conference of the International Society for Music Information Retrieval, Proceedings.* Taipei, 2014, p. 6.

[18] Keunwoo Choi, George Fazekas, and Mark Sandler. "Automatic tagging using deep convolutional neural networks". In: *International Society of Music Information Retrieval Conference. ISMIR.* 2016.

[21] J. Pons, T. Lidy, and X. Serra. "Experimenting with musically motivated convolutional neural networks". In: *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI).* 2016, pp. 1–6. DOI: 10.1109/CBMI.2016.7500246.


## Articles


[5] Carlos N. Silla, Alessandro L. Koerich, and Celso A. A. Kaestner. "A Machine Learning Approach to Automatic Music Genre Classification". In: *Journal of the Brazilian Computer*

*Society* 14.3 (2008), pp. 7–18. ISSN: 1678-4804. DOI: 10.1007/BF03192561. URL: http://dx.doi.org/10.1007/BF03192561.

[6] Giorgos Tsiris. "Aesthetic Experience and Transformation in Music Therapy: A Critical Essay". In: *Voices: A World Forum for Music Therapy* 8.3 (2008). URL: https://voices.no/index.php/voices/article/view/416.

[7] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. "Evaluation of algorithms using games: The case of music tagging". In: (2009), pp. 387–392.

[8] Luis M. de Campos, Juan M. Fernndez-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks". In: *International Journal of Approximate Reasoning* 51.7 (2010), pp. 785 –799. ISSN: 0888-613X. DOI: http://dx.doi.org/10.1016/j.ijar.2010.04.001. URL: http://www.sciencedirect.com/science/article/pii/S0888613X10000460.

[10] S. K. Kopparapu and M. Laxminarayana. "Choice of Mel filter bank in computing MFCC of a resampled speech". In: (2010), pp. 121–124. DOI: 10.1109/ISSPA.2010.5605491.

[13] M. Muller, D. P. W. Ellis, A. Klapuri, and G. Richard. "Signal Processing for Music Analysis". In: *IEEE Journal of Selected Topics in Signal Processing* 5.6 (2011), pp. 1088–1110. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2011.2112333.

[14] M. Slaney. "Web-Scale Multimedia Analysis: Does Content Matter?" In: *IEEE MultiMedia* 18.2 (2011), pp. 12–15. ISSN: 1070-986X. DOI: 10.1109/MMUL.2011.34.

[20] Patrik N. Juslin, Laura S. Sakka, Gonalo T. Barradas, and Simon Liljestrm. "No Accounting for Taste? Idiographic Models of Aesthetic Judgmentin Music". In: *Psychology of Aesthetics, Creativity, and the Arts* 10.2 (2016), pp. 157–170. DOI: 10.1037/aca0000034.

## Pre-Prints

[19] Keunwoo Choi, Gyorgy Fazekas, Mark Sandler, and Kyunghyun Cho. *Convolutional Recurrent Neural Networks for Music Classification*. Version 3. Dec. 21, 2016. arXiv: 1609.04243.

## Books

[1] D. O'Shaughnessy. *Speech communication: human and machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Pub. Co., 1987. ISBN: 9780201165203.

[3] R.L. Allen and D. Mills. *Signal Analysis: Time, Frequency, Scale, and Structure*. Wiley, 2004. ISBN: 9780471660361.

## Misc

[2] Jean julien Aucouturier and Francois Pachet. *Music Similarity Measures: What's The Use ?* 2002.

[4] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. *Content-Based Music Information Retrieval: Current Directions and Future Challenges*. 2008.

[23] Lecture Notes. *Spectral Leakage and Windowing*. https://mil.ufl.edu/nechyba/www/__eel3135.s2003/lectures/lecture19/spectral_leakage.pdf. Online; accessed 20 March 2017.