**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# AUDIO FINGERPRINTING USING WAVELET TRANSFORM

**Evren KANALICI**

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Advisor
Assoc. Prof. Dr. Gökhan BİLGİN

July, 2019

**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# AUDIO FINGERPRINTING USING WAVELET TRANSFORM

A thesis submitted by Evren KANALICI in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 23.07.2019 in Department of Computer Engineering, Program of Computer Engineering.

Assoc. Prof. Dr. Gökhan BİLGİN
Yildiz Technical University
Advisor

**Approved By the Examining Committee**

Assoc. Prof. Dr. Gökhan BİLGİN, Advisor
Yildiz Technical University _____

Asst. Prof. Dr. Erkan USLU, Member
Yildiz Technical University _____

Prof. Dr. Oğuzhan KÜLEKCİ, Member
Istanbul Technical University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Audio fingerprinting using wavelet transform supervised by my supervisor, Assoc. Prof. Dr. Gökhan BİLGİN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Evren KANALICI

Signature

*Dedicated to my family*

# ACKNOWLEDGEMENTS

Thanks to my family and friends for all their support.

Also, I want to express my greatest gratitude to my supervisor, Assoc. Prof. Dr. Gökhan BİLGİN, for his encouragements, supervision and advices within the process of this study.

Evren KANALICI

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| | |
|---|---|
| $D$ | Distance function |
| $\widehat{x}$ | Fourier Transform of function $x$ |
| $\lvert\cdot\rvert$ | L1 norm |
| $\lVert\cdot\rVert$ | L2 norm |
| $L$ | Loss function |
| $M$ | Mel-frequency transform |
| $\mathbb{R}$ | Real numbers |
| $x_r$ | Real part of $x$ |
| $x_i$ | Imaginary part of $x$ |
| $\Phi$ | Transform mapping |
| $\nabla$ | Vector differential operator |
| $\psi$ | Wavelet function |
| $\phi$ | Wavelet scaling function |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ANNS | Approximated Nearest Neighbour Search |
| AUC | Area Under Curve |
| BFCC | Bark Frequency Cepstrum Coefficents |
| CNN | Convolutional Neural Network |
| CQT | Constant-Q Transform |
| CWT | Continuous Wavelet Transform |
| kNN | k-Nearest Neighbour algorithm |
| LSH | Locality Sensitive Hashing |
| MFCC | Mel Frequency Cepstrum Coefficents |
| MIR | Music Information Retrieval |
| ReLU | Rectifier Linear Unit |
| ROC | Receiver Operating Characteristic |
| SIFT | Scale-invariant Feature Transform |
| STFT | Short-Term Fourier Transform |
| SVM | State Vector Machine |
| SWT | Scattering Wavelet Transform |
| WFT | Windowed Fourier Transform |
| WT | Wavelet Transform |

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

## Audio fingerprinting using wavelet transform

Evren KANALICI

Department of Computer Engineering
Master of Science Thesis

Advisor: Assoc. Prof. Dr. Gökhan BİLGİN

Audio fingerprinting systems have many real-world use-cases such as digital rights management/copyright detection, duplicated audio detection, untagged audio labelling or identify/query-by-example recognition systems. Nowadays, there are popular online platforms that offer identify/query-by-example music recognition services where users can query by snippets of recorded audio to retrieve the matched song metadata.

The compact, robust and fast retrieving fingerprint design is the cornerstone of these systems. Although short-term Fourier transform and Mel-spectral representations are common tools that come to mind, these feature extraction methods suffer from being unstable and having somehow limited resolution. In order to overcome these challenges, scattering wavelet transform (SWT) provides an alternative solution to these limitations by recovering information loss, while ensuring translation invariance and stability.

In this study, a two-stage audio fingerprint characteristic/feature extraction framework is introduced using SWT integrated with Siamese neural network hashing model for musical audio identification. Similarity-preserving hashes provided by the Siamese neural network model correspond to sound fingerprints and can be defined by a similarity distance metric in the embedded hashing space. The Siamese neural network hashing model was trained by two-layer scattering wavelet transform coefficients using relatively aligned segments of the same music files and segments of different music files. The proposed system achieves successful performance scores under environmental noise, modeling the challenges of detecting music and audio

data that may be encountered in everyday life. Using very compact storage, it has been shown to achieve high *ROC-AUC* scores both by one-to-one comparison and by using locality-sensitive hashing (LSH) for content storage.

**Keywords:** Audio fingerprinting, music information retrieval, wavelet transform, scattering wavelet transform, Siamese neural networks, triplet loss function, convolutional neural networks, embedding hash models.

# ÖZET

## Dalgacık dönüşümleri ile ses parmak izi kontrolü

Evren KANALICI

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Gökhan BİLGİN

Ses parmak izi tespit sistemlerinin günlük hayatta dijital hak yönetimi/telif hakkı tespiti, kopya ses verisi tespiti, ses verisi etiketleme, örnek ile tespit/sorgu gibi birçok kullanım alanı mevcuttur. Günümüzde de kullanıcıların canlı ses kayıtları ile sorgulayıp dönüş alabildikleri müzik tespit ve sorgu servisi sunan çevrimiçi popüler platformlar faaliyettedir.

Kompakt, gürbüz ve hızlı erişimi hedefleyen ses parmak izi tasarımı bu sistemlerin temel taşını oluşturur. Kısa-süreli Fourier dönüşümleri ve Mel-spektral gösterimleri ilk akla gelen araçlar olmakla birlikte bu çıkarım yöntemleri kararsızlık gösterir ve bir bağlamda düşük çözünürlüğe sahiptirler. Saçılım dalgacık dönüşümü (SDD) bu kısıtlamaların üstesinden gelebilmek maksadıyla, sinyal dönüşümleri sırasında kaybolan enformasyonu telafi ederek ve öteleme-değişmezliği ve kararlılık sağlayarak alternatif bir çözüm sunar.

Bu çalışmada, müzik ses verisi tanıması için, siyam sinir ağları karım modeli ile tümleşik bir şekilde saçılım dalgacık dönüşümü kullanılarak iki aşamalı bir ses parmak izi karakteristik/özellik çıkarım sistemi sunulmaktadır. Siyam sinir ağı modelinin sağladığı benzerlik muhafaza eden karımlar ses parmak izlerine denk gelmekte ve bu gömülü karım uzayında benzerlik mesafe ölçütü ile tanımlanmaktadır. Siyam sinir ağları karım modeli, aynı müzik dosyalarının belirli bir komşuluk sınırı içerisinde görece hizalı bölütleri ve farklı müzik dosyalarının bölütleri kullanılarak, iki-katmanlı saçılım dalgacık dönüşümü katsayıları ile eğitilmiştir. Önerilen sistem, günlük hayatta karşılaşılabilecek müzik ses verisi tespit & sorgu zorluklarını modelleyen çevresel gürültü altında, başarılı performans skorları elde etmektedir. Oldukça kompakt

depolama alanı kullanarak, hem bire bir karşılaştırma yapılarak hem de depolama için yerellik-duyarlı karım (YDK) kullanılarak yüksek *ROC AUC* skorları elde ettiği gösterilmektedir.

**Anahtar Kelimeler:** Ses parmak izi tespiti, müzik bilgi çıkarımı, dalgacık dönüşümü, saçılım dalgacık dönüşümü, siyam sinirsel ağları, üçüz zarar fonksiyonu, evrişimsel sinirsel ağları, gömülü karım modelleri.

# 1
# Introduction

This thesis presents the results of our study on musical audio identification system via a two-stage fingerprinting framework, and reports detailed evaluations and experiments. Audio identification involves the retrieval of metadata as associated with an unidentified audio extract. Our attempts are based on generating compact fingerprint for relatively long durations of audio segments, and provide robustness for audio signal query execution under additional environmental noise which can model real-world obstacles of music identification problem.

Nowadays fingerprinting systems used in various aspects consumable multi-media of audio streams including monitoring applications, copyright detection, recommenders systems and labelling applications. The increasing large amounts of audio data, attract rousing interest in the ability to identify audio content. This can be done via *audio fingerprinting* which is a representative or discriminative method that aims at operating with precision and certainty.

## 1.1 Literature Review

Audio fingerprinting systems are well researched Music Information Retrieval (MIR) topic and various efficient fingerprinting methods have been introduced. Many audio-fingerprinting methods prefer to use low-level features that aims to characterize the audio signal (i.e. content-based approach) without any other high-level semantic meanings. Also it's very common to approach to the fingerprinting as a computer vision problem after change of domain executed that enables working on spectral data, thanks to the insights of previous studies. The maintain robustness to shifting (i.e. the query is not perfectly aligned with the candidate feature from the content score) overlapping feature extraction is a common practice. Another common preference is to combine all sequentially retrieved candidate matches by some means of constraints to provide a final match (e.g. temporal alignments of the probe candidate matches).

**Table 1.1** Literature by years published

| Literature | Year |
|---|---|
| Allamanche *et al.* [2] | 2001 |
| Haitsma, Kalker, and Oostveen [3] | 2001 |
| Haitsma and Kalker [4] | 2002 |
| Herre, Hellmuth, and Cremer [5] | 2002 |
| Wang [6] | 2003 |
| Ke, Hoiem, and Sukthankar [7] | 2005 |
| Baluja and Covell [8] | 2006 |
| Bellettini and Mazzini [9] | 2008 |
| Wang *et al.* [10] | 2009 |
| Zhu *et al.* [11] | 2010 |
| Ellis, Whitman, and Porter [12] | 2011 |
| Fenet, Richard, and Grenier [13] | 2011 |
| Ramona and Peeters [14] | 2013 |
| Ouali, Dumouchel, and Gupta [15] | 2014 |
| Coover, and Han [16] | 2014 |
| Malekesmaeili and Ward [17] | 2014 |
| Six and Leman [18] | 2014 |
| Sonnleitner, Arzt, and Widmer [19] | 2016 |
| Gfeller *et al.* [20] | 2017 |

Noted, it is argued that many MIR publications are hard to verity due to the fact that often only a textual descriptions made available code remains unpublished leaving many implementation issues uncovered [1]. Also copyright issues of publicly available datasets and the MIR research involving complex systems prevents reproducibility of the works published. Even though most notable works should be overviewed to have a detailed or leastways coarse understanding what foregoing studies propose and give insights, including the widely referenced ones. Table 1.1 shows the notable works in literature by the years they are published.

Allamanche *et al.* [2] evaluate a content based audio identification system using MPEG-7 features, namely "AudioSpectrumFlatness" and similar low-level descriptors, relating signal's power spectrums and tone-like characteristics to form feature vectors for each windowed segments. Each feature vector from subsequent time steps then combined to form a composite vector. These feature vectors are used at trained stage for clustering using Nearest Neighbour and Vector Quantisation to generate minimal codebooks. In this scheme, identification task for a query signal becomes aggregating the distance of query segments to available codebooks and proposing a candidate with minimum distance.

Haitsma, Kalker and Oostveen [3] propose robust hashes extracted from 0.4-second long windowed time frames with 31/32 overlap factor (i.e. slowly varying time

segments). 33 non-overlapping frequency bands selected in each frame in $300Hz - 2000Hz$ range using logarithmic scale (Bark-scale). Bandwidth is spaced in ratio of one musical tone ($2^(1/12)$ per band). Sign of energy difference over frequency-bands and frames is suggested to build robust hashes.

Later, in their work, Haitsma, and Kalker [4] build their fingerprints using Bark Frequency Cepstrum Coefficients (BFCC). While extracting fingerprints from mono-audio signals, overlapping frames with a small step size are used to ensure precision and maintain shift-invariance for queries with arbitrary time offsets. The spectral representation of the audio is constructed using BFCC of 33 log-frequency bands in range ($300Hz - 2000Hz$). The overlapping windows are $372ms$. long and windows are stridden with $11.6ms$. step size giving $\sim$ 100 fingerprints per each second. The short step size provides robustness to arbitrary alignment problem. Window and striding scheme is depicted in Fig. 1.1. Each extracted sub-fingerprints are 32-bits which indicates the difference between successive BFCC bands in consecutive frames. It's claimed these sub-fingerprints are insensitive to small changes or distortions in the audio signal by being sparse, since only quantized binary values of consecutive frames are kept instead of actual differences and they are compact and fast to compute. Fingerprints are then the concatenated sub-fingerprints, and *Hamming distance* is used for comparison.

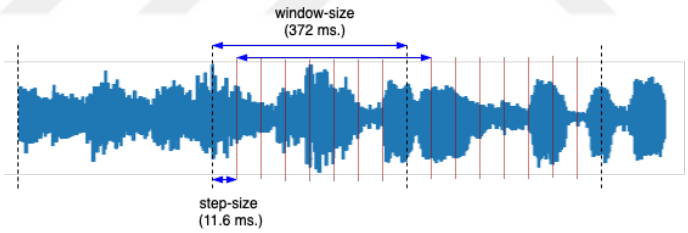



**Figure 1.1** Haitsma, and Kalker audio signal timing scheme

Herre, Hellmuth and Cremer [5] follow up the scheme in [2] which uses characteristics from MPEG-7 content descriptors, and evaluate various trade-offs between fingerprint compactness, temporal coverage and robustness of recognition.

Wang [6] [21] describe a system that creates geometric hashes from local spectrogram maxima. While individual hashes are localized with low specificity, sequences of matches over time show high specificity. The proposed method of searching for sequences of matched hashes constitutes a very efficient algorithm. This has become a classic and influential method, generally known as "Shazam". The hashing model, however, does not exhibit robustness to any type of scale modifications. The system is considered to be highly robust to additive noise. It is prosed a robust scheme using only spectrogram peaks. Wang's method is somehow not dense like other methods, since it

works by extracting spectrograms over long durations of time and examines power peaks of time-frequency representation. It is argued selecting spectrogram peaks provide: (1) robustness to noise, surviving from additional noise, (2) property of linear superposition, that is both components of acoustic audio and noise would provide same peaks. Their sub-fingerprinting scheme illustrated in Fig. 1.2. For possible pairs of peaks $(t_1, f_1), (t_2, f_2)$ in a neighborhood distance, triplets of values $(f_1, \Delta t, \Delta f)$ is calculated, quantized and concatenated vector values are used as fingerprints. Quantized values use $8, 6, 6$ bits for triplets $(f_1, \Delta t, \Delta f)$ respectively resulting 20-bits for each fingerprint.

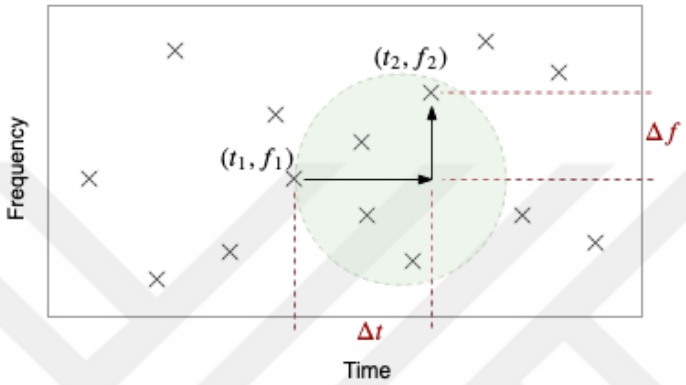



**Figure 1.2** Wang *et al.* frequency-delta triplet selection scheme

Wang also proposes a simple and fast approach for *temporal alignment* and it works like this: (1) for all fingerprint matches a time-vs-time plane is plotted for query fingerprints against matched metadata, (2) diagonals are examined on time-vs-time plot, the strong diagonal indicates a valid match.

Ke, Hoiem, and Sukthankar [7] show that audio fingerprinting can be approached via computer vision techniques. The system can identify query pieces that are severely distorted by noise. It is proposed a method which improves the performance of [4]. They use the similar scheme but design their novel fingerprints using AdaBoost method, parlaying from computer vision field practiced in applications like face recognition tasks [22]. The important insight is audio signals can be processed efficiently when transformed into time-frequency representation. AdaBoost, being a learning method, customized to learn energy concentration of selected frequencies and the discriminative power of its features similar to *boxlets* in [22] enables differentiating two spectrogram sub-rectangles given. The training data also includes degraded versions of original signals. The output of each classifiers yields a binary value based on the differences between two consecutive sub-rectangles. Out of 25000 candidate filters 32 classifiers are selected generating 32-bit features. For querying, same timing scheme is used as [4] and the learned model is used for feature extraction. Two fingerprints are considered to be a match by Hamming distance threshold equal

to 2.

Beluja and Covell [8] also benefit from the insights of computer vision approach combining with wavelet transform. In their work, first, the audio signal is transformed to time-frequency representation. They use multi-resolution haar-wavelets and convert overlapping segments of spectrograms to wavelet coefficients. Wavelets are chosen due to their efficiency in the image retrieval work presented in [23]. Wavelet transform coefficients are equal to number pixels in spectrogram image and authors notice the sparsity of coefficients so they select top-$t$ values. Then similar to the scheme as in [23] coefficients quantized to only sign information and the Min-Hash method used to generate a set of $p$ bytes that as final fingerprints. The fingerprints are compared directly by byte-wise Hamming distance. Since for a large values of $p$-bytes (e.g. 100) the neighborhood search would be infeasible, authors use Locality Sensitive Hashing (LSH) to execute approximate nearest neighborhood search in fingerprints space. Their systems's pipeline is shown in Fig. 1.3.

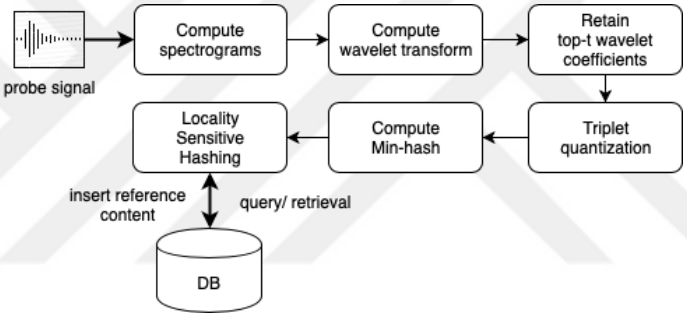
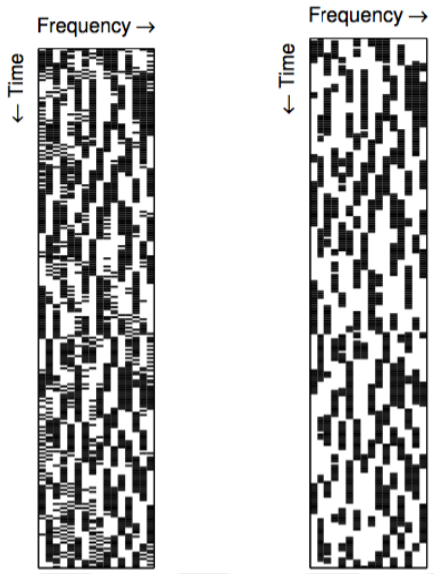



**Figure 1.3** Beluja *et al.* Waveprint pipeline

The window duration of overlapping spectrograms are 372$ms$ and they use step size of 0.09$ms$. which makes about $\sim$ 10 fingerprints per second. Top 200 wavelets are selected for each fingerprint and $p$ is chosen to be 100 (i.e. each fingerprints are 100-bytes). For approximate nearest neighborhood search each fingerprint is divided to twenty five 4-bytes sub-hashes for LSH, and voting mechanism is used for retrieval from LSH for each candidate sub-hashes.

Bellettini and Mazzini [9] propose a framework, tracking down commercial radio or TV broadcast transmission with robustness to pitch-shifted audio. They use energy difference among subsequent spectral sub-bands of audio using STFT with 372ms. long windows and highly overlapped frames. It is argued sequence of bit vectors provides robustness both for alignment problem and pitch modifications. 16 sub-bands are used, resulting 16-bit vectors for each time frame and compared with normalized Hamming distance along with a threshold-ed match algorithm. To ensure robustness to pitch distortion, a limited bit-shift operation is applied with a predefined range after feature extraction step. A further observation is made on altering energy

difference on audio excepts and a $mrl$ pre-processing (i.e. Minimum Run Length for alterations) is opted with a value of 3, in to provide more robustness (see Fig.1.4). The proposed framework is tested both by exhaustively brute-force comparison and with a improved search method.




**(a)** No $mrl$ pre-processing **(b)** With $mrl = 3$ pre-processing

**Figure 1.4** Minimum Run Length for sub-band energy alterations on 256 × 16-bit vector - Bellettini *et al.* [9]

Zhu *et al.* [11] approach to both time-scale modification and pitch-shifting as stretch and translation of the audio signals 2D representation, and build their framework using compute vision techniques to extract robust local descriptors. Audio signals are converted to images by STFT over 2048-sampled windows with %50 overlapping. Spectrograms contain 97 log-scale frequency bands. It's claimed a local descriptor, namely Scale Invariant Feature Transform (SIFT) is utilized in their work, which is to robust to affine transforms also provides robustness in audio domain. 128-dimensional descriptors are used for comparison using Euclidian distance. For a given local descriptor $a$ from audio signal $A$, also $b$ and $b'$ from audio signal $B$ being first and second closest descriptors to $a$ measured by distance function $D$ respectively; $D(a,b) < T * D(a,b')$ should hold to consider $a$ and $b$ to be a match, where $T$ is the threshold value (used as equal to 0.6 in the their work). The study does not suggest a way to construct a complete scheme for the SIFT characteristics and focuses on testing the applicability of the feature. The efficiency of identification is assessed on a database consisting of approximately 20.7 hours of audio material, each produced from 1241 extracts of 1 minute duration. The work is reported to obtain outcomes in the ranges of 64% to 150% and 50% to + 100% respectively for speed and pitch

changes.

Ellis, Whitman, and Porter [12] present a music identification systems providing robustness to spectral modifications and noise encountered in the audio signal. Their method depends on relative timing between successive percussive onsets in the audio. Onset detection is performed independently in 8 frequency bands, corresponding to the lowest 8 bands in the MPEG-Audio 32 band filter-bank (spanning 0 to 5512.5 Hz). The magnitude of the complex band-pass signal in each band is compared to an exponentially-decaying threshold, and an onset recorded when the signal exceeds and adaptive threshold. Pairs of successive inter-inset-intervals (IOIs) in each band, quantized into units of 23.2 ms, are combined to make a hash. To provide robustness against spurious or missed onsets, each onset is considered along with its four successors. Six different hashes (IOI pairs) are created by choosing all possible pairs of succeeding onsets from the four resulting overall hash rate 8 (bands) $\times$ 1 (onset per second) $\times$ 6 (hashes per onset) $\approx$ 48 hashes/sec.

Fenet, Richard, and Grenier [13] first remark on that most fingerprinting techniques provide robustness to transmission distortions and focus pitch alterations by expanding Wang's [6] hashing model. Their technique, based on a hashing coupled with a CQT-based fingerprint, claimed as a solution to capture representation of quantized pitch offsets. CQT transform provides geometrically spaced frequency bins capturing the characteristics of western scale, thus pitch-shifting corresponds to a translation in the CQT domain. In their study, 3 frequency bins per note is used for each 10 ms. apart overlapping frames. Moreover, for more compact representation, spectrogram peak-picking is used inspired by [6]. Evaluated data reflects a broadcast monitoring use case where audio excepts may be shortened for time limitations resulting as pitch modification, including 7 hours long stream of a radio station recording. For experiments, both small and large reference content are used for comparison and scalability purposes respectively. Comparative setup is executed with small reference database which includes approximately 122 hours of audio consisting 1 min. long excerpts from 7309 songs. Compared against the baseline of Wang's [6] framework (which is not robust to pitch alterations), which detects with score of 83%, the proposed technique outperforms with 97.4% detection score (achieving to detect 447 out of 459 occurrences).

Ramona and Peeters *et al.* [14] propose an cosine-filter based extension to IRCAM audio fingerprint system that considers the evolution of the audio characteristics over time instead of instantaneous fingerprints. IRCAM audio fingerprint system labor large temporal scope with few seconds duration to compute each fingerprint using what called "Double-nested Fourier Transform" resulting spectral band energies, where

the second STFT is performed over time for each set of short-term spectral bands. It's argued in the work that, using sums of squared amplitudes for spectral energy provide a weak robustness to scale modifications, since it's equivalent to applying rectangular filters where a pitch alteration ends up messing with both spectral energy peaks. Instead, a cosine-filter based approach (having a smooth shape that enables continuous change of spectral energy as shown in Fig. 1.5) is proposed to compute short-term band energies, making rather robust to moderate frequency distortions.

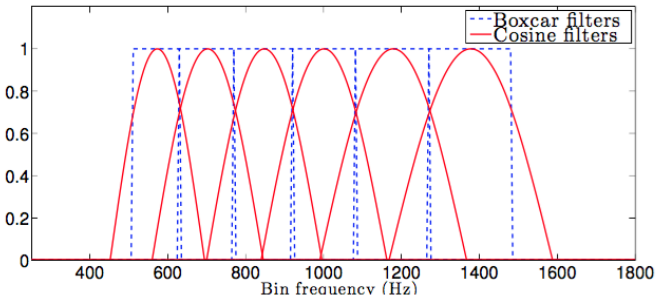



**Figure 1.5** Cosine vs. rectangular filter profiles[14]

The evaluation is made 240 hours long corpus of radio broadcast encoded with low bit-rate with reference content of 1000 items. Stream includes about 2000 occurrences of reference items with 30 seconds long excerpts. Combined with a proposed frame synchronization scheme which detects frame positions that are robust to audio alterations, their method outperforms the re-implementations of *Philips* system by Haitsma and Kalker [4] and Shazam system by Wang [6] by %8.5 and %13.4 points respectively.

Malekesmaeili and Ward [17] propose an audio copy detection system providing strong robustness tempo changes and pitch shifts which is based on local audio fingerprints of the signal that are extracted from two-dimensional "time-chroma" image of spectrogram patches. "Time-chroma" representations are stressed as having advantages as: the pitch shift in the audio signal appears as a circular shift along the chroma axis of the image and any change of tempo in the audio signal appears as a change of scale along this image's time axis. Feature vectors are calculated based on a set candidate feature points which are local maxima (i.e. are robust to noise, quantisation and filtering attacks). These candidate local maxima points are then compared with up to 30 two-dimensional image patches of different width (i.e. along time-axis between 1s to 4s), centered around the particular local maxima. A candidate point is selected as feature point if most of the patches manifest a similarity criterion. The similarity between two patches of different scales are quantified by a dictionary of arbitrary number of ($c$) representative patterns. For a pair of patches to be said are similar, they are both mapped to the same pattern in the dictionary, and

the similarity metric is correlation of low-frequency discrete cosine transform (DCT). Later the fingerprint vectors are scaled and translated into zero-mean unit-variance vectors. Thereby, the tempo and pitch invariance is archived as argued in [17]:

> If the tempo of an audio signal is changed, the time-chroma image is stretched or squeezed along the time axis. This accordingly affects the scales assigned to the feature points, but has no effect on the contents of the patches around them, thus providing tempo invariance. Also, if a song is pitch-shifted, the time-chroma image circularly shifts along the chroma axis. This moves the feature points vertically (along with the whole image), thus has no effect on the content of the patches around them and provides pitch invariance.

The proposed method carries out feature extraction for an average of 20 candidate points per second of audio signal, and about 40% of which were selected as feature points by similarity constraints. For performance evaluation, query probe signals are prepared by mash-up versions of the original signal (i.e. modifying speed, pitch, tempo, etc) up 100 song snippets that were randomly extracted from the songs in the reference content database (about 250 songs). Te distance between two fingerprints is measured by the angle between them, and fingerprint matching is performed by nearest-neighbour search. Results are compared with re-implemented "AudioSIFT" and "Shazam" frameworks. Their proposed algorithm delivers out-performing results for various audio degradations, including tempo and pitch modification attacks, though considering on a rather small content database.

Six and Leman [18] present a scalable audio identification system named "Panako", where their fingerprints use condensed representation of audio signals. Proposed method is inspired by three previous works namely: Wang [6] (for finding local maxima in spectral domain which provide robustness to noise, compression and quantization effects), Artz *et al.* [24] (to benefit a method to align performances and scores of key time points) and Fenet *et al.* [13] (by using Constant-Q Transform for fingerprint extraction where fingerprints remain constant when a pitch-shift occurs). Combining these key concepts they provide granular acoustic finger-printer that is robust to noise, time-scale modification and pitch-shifting. Time-scale modification robustness is provided by storing the quantized values of the spectral peak triplets' time differences. In their evaluation a database of 30000 songs is used for reference data with about 10 million seconds of audio. From this dataset random fragments with modifications (i.e. pitch-shifting, time-stretching, time-scale modification echo, flanger, chorus and filtering) are selected, with a duration between 20, 40 and 60

seconds, and then compared with a baseline system. The findings indicate that the efficiency of the system reduces by more than eight percent with a time-scale alteration. The scheme is shown to handle pitch-shifting, time-stretching, serious compression, and other changes as echo, flanger, and filtering.

Sonnleitner and Widmer [19] propose an efficient audio fingerprinting method that meets the multiple robustness requirements including noise, audio quality degradation and also to large amounts of speed, tempo or frequency and pitch scaling. A simple and fast geometric hashing technique is adapted to achieve representations of fingerprints that are invariant to translation and scaling, and thereby overcome the inherent robustness limitations. The algorithm uses a compact four-dimensional continuous hash representation of quadruples of points which referred to as "quads" that are build by extracting spectral peaks (as in [6]) from the two-dimensional time-frequency representation of reference audio material, then group quadruples of peaks into quads, and create a compact hashes. Quads are constructed for possible spectral points $A, B, C, D$ by the following constraints: (1) $A_t < C_t, D_t \leq B_t$ and (2) $A_f < C_f, D_f \leq B_f$ ; where $t$ and $f$ corresponds to time and frequency axis respectively. This scheme is reported to have high accuracy of more than 95% and an precision of 99% on queries that are modified in pitch and/or time scale by up to 30% with an average query run time of under two seconds for query snippets of 20 seconds.

Gfeller *et al.* [20] present a continuous music recognition system named "Now Playing", combined with background deep modelled music detector focused on energy consumption to awake when a musical audio signal is present. Their neural network finger-printer is generate compact and discriminative fingerprints at a rate of one 96 dimensional embedding per second. Their system is able to detect, recognise and inform users which song is playing without and client-server architecture and any need to network access. A deep modelled neural network fingerprinter is used in their work, which is trained over the audio signal's STFT patches with triplet loss function. The model training involves a dataset of noisy music sections aligned in their reference song with the respective segment. The optimized fingerprint hash model analyzes audio for a few seconds and generates a single fingerprint embedding at a rate of one per second.

The work of Gfeller *et al.* [20] is highly influencial in this study, as we later explain in following Chapter Concepts using similar methods of training with noisy aligned dataset and triplet network training for audio patches under different transformations and generating final embeddings.

Before continuing to following section, where the objectives of this study is given, in

**Table 1.2** Literature by robustness types

| **Degraded audio quality** |
|---|
| Herre, Hellmuth, and Cremer [5] |
| Wang [6] |
| Haitsma and Kalker[4] |
| Ellis, Whitman, and Porter [12] |
| Allamanche *et al.* [2] |
| Coover, and Han [16] |
| Gfeller *et al.* [20] |
| **Pitch-shifts** |
| Fenet, Richard, and Grenier [13] |
| Bellettini and Mazzini [9] |
| Ramona and Peeters [14] |
| Ouali, Dumouchel, and Gupta [15] |
| **Both pitch and time-scale modification** |
| Zhu *et al.* [11] |
| Malekesmaeili and Ward [17] |
| Wang *et al.* [10] |
| Six and Leman [18] |
| Sonnleitner, Arzt, and Widmer [19] |

brief the following Table 1.2 overviews the notable works in literature against various robustness settings.

## 1.2 Objective of the Thesis

The problem at the heart of audio fingerprinting systems can be summarised with two components that are *fingerprint design* and *matching search* methods. In practice the audio signal data may under many kinds of degradations arising as challenges of the study of the research.

In this study, we labor an open research problem based on the problem definition and try to tackle to common challenges and obstacles that encountered while carrying out methods proposed for the actual problem on musical audio domain.

### 1.2.1 Challenges

Audio identification systems usually operate on large data scale and are expected to meet several robustness requirements depending on the use cases, and the approach to these systems basically consist of two major steps: (1) fingerprint design and (2) matching search:

- fingerprint step: aims to generating of robust and compact audio features.
- search step: requires database access and fast search algorithms.

In real-world, the challenges to meet these requirements are varied base on the context of application to be designed and some could be abandoned for other requirements tradeoffs. That been said, some specific challenges that considering the input signal may undergo could be:

- Additional background noise
- Acoustic reverberations
- Quantization errors
- Audio compression artifacts (i.e. GSM or MP3)
- Inference in the transmission
- Pitching (playing faster of slower)
- Equalization
- Quantization errors and D/A and A/D conversion artifacts

And since it is aimed the fingerprints to be compact and robust, while converting an audio signal into a sequence of characteristic features as an input to the fingerprint model, the design choices should include:

- Discrimination power (over large number of other fingerprints)
- Dimensionality reduction (i.e. by/or change of domain)
- Perceptually meaningful parameters (e.g. regarding to the human auditory system)
- Strong robustness (to distortions, additional noise, transmission artefacts etc.) and/or invariance.
- Temporal correlation (being able to capture spectral dynamics).

### 1.2.2 Audio Fingerprinting

An audio fingerprint is a *content-based* compact signature of audio data that characterise audio signal data, which be compared or matched reliably to or against large set of fingerprints from a reference content store. Each fingerprints that extracted from a query audio data and afterwards compared to indexed reference content. As a result a match is found (with an additional match score) or it is reported it not present in reference content (by using thresholding techniques). Audio fingerprint features should have discriminative properties rather than being representative in nature of the requirements of verification/recognition tasks.

The requirements for an audio fingerprinting system described by [25] includes being granular, robust, reliable and economic in terms of storage footprint and computational complexity while responding to a query.

Also as it stated in [1]:

> *Robustness* is determined by various degradations a real-world query can be subjected to while remaining recognisable. Degradations include additional noise, low-quality encoding, compression, equalisation, pitch-shifting and time-stretching. To allow scaling to large reference items, an economy in terms of storage is needed.

Considering retrieval process, economy means computational load. The tradeoff between each requirements can adjust depending on the context of the application. A typical finger-printer scheme is depicted in Fig. 1.6.

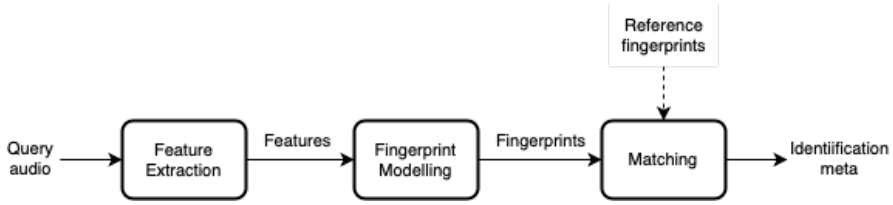



**Figure 1.6** A general finger-printer scheme

### 1.2.3 Audio Retrieval and Identification

Such audio fingerprinting system have many use-cases including digital right management/copyright detection, identifying duplicates [26], labelling untagged audio with metadata or query-by-example recognition. For instance Shazam [21] and SoundHound [27] are popular query-by-example music recognition services where users make a query by snippets of recorded audio to back-end services to retrieve

matched song metadata. A common pipeline steps of general architecture can be summarised as below [28]:

1. Fingerprints of the reference content audio collection and its corresponding metadata (e.g., audio-file ID, name, time frame data, etc.) are systematically stored in a database.
2. The fingerprint extraction generates a set of relevant features combined with an optional post-processing and feature modelling.
3. For a given a short recording of audio as query, its feature vectors (i.e fingerprints) are computed in the same way as reference content database generated.
4. Afterwards, a *searching algorithm* will find the candidate matches for the given query from the fingerprints those were stored in the reference database.
5. Additionally, *adaptive scoring* techniques will be applied to derive final match or response to query as it is not available in content store based on the candidate matches derived in step (4).

At the query stage, the fingerprints could be extracted at *uniform rate* with an arbitrary time window, or they will be extracted with *random offsets* (within an increasing time frame) to avoid unlucky alignments while querying the content database. Another approach would be to extract fingerprints from a range with points of interest based on the application requirements (e.g. ranges where RMS power or some other attribute is higher than a given threshold).

Having the features are extracted for the query, then it is compared with a database of reference content to evaluate candidate matches. Since a naive pairwise comparison on a large dataset is not feasible, the database can be partition with hashing enabling the retrieval process to correspond a reverse lookup. Techniques for approximate nearest neighbour search like *locality sensitive hashing* (LSH) or *vector quantisation* may be used on very large-scale datasets since direct hashing may not scaleable for millions or billions of entries. The retrieved metadata by reverse lookup includes IDs song and the time offsets within the song. The set of fingerprints for the query probe from the reverse lookup are the potential shortlist of candidates.

Storing the time offsets as metadata in the content store is crucial since candidate matches of queried fingerprints may fed to another pipeline to promise a final candidate for a whole sequential query. Based on the candidate shortlist, various

constraints may be applied to propose final candidate match namely *temporal alignment* for linear correspondence. Temporal alignment method is used to avoid false positives and increasing precision of retrieval. Techniques like Expectation Maximization [29][7] or Dynamic Time Warning [8] may be used for temporal alignment in the nature of tempo constraint, that is the tempo between query and match should be same or close enough.

## 1.3 Hypothesis

The scope of this study aims implementing full-fledged musical audio identification system. We explored a two-stage feature extraction method combining Scattering Wavelet Transform (SWT) and deep embedding hash model to generate final fingerprints. Combined with concept of neural network finger-printer, embedding hash model is highly influenced by the work of Gfeller *et al.* [20], using convolutions and divide-any-encode layer of segmented transformed signal data. Our contribution also compasses investigations and evaluations of concepts SWT, divide-encode block, piece-wide threshold quantisation, adaptive scoring schemes and LSH content storage for audio identification problem.

In this research, we have attempted to address the most prevalent obstacle to audio identification from the end-user view, which is musical audio tampered with environmental noise, not concentrating on pitch and tempo change kinds of robustness. The identification system is first experimented with naive comparative technique not to be affected by artefacts of database accuracy and later also with LSH methods to be possible to use large-dimensional audio fingerprint information to collect and identify a real-world application situation using our compact and discriminative audio fingerprints.

## 1.4 Organisation

In this chapter we started by giving the problem definition for audio fingerprinting recognition systems. We explained the details of fingerprint design and methodology for retrieval and identification tasks. Also we enlisted and various challenges that can be encountered for a real-world applications and coarsely explained design decisions.

In following Chapter Concepts we delve into technical concepts and the terminology used in entire study in a detailed way. Namely it is explained how our two-stage fingerprinting framework works based on its technical foundations. Next, Chapter Evaluation and Musical Audio Identification Application presents

our findings of evaluations and various experiments of the framework proposed in Chapter Concepts. Finally, the study is summed up in Chapter Conclusion and Recommendations.

# 2
# Concepts

Given the problem definition in Chapter Introduction and various approaches to the specific task in the literature, what follows the theoretical background should be given which our technique and implementation are build upon. This chapter introduces coarse overview of concepts as building blocks of our method presented in this study, and provides materials for the subsequent chapters for completeness. Readers can investigate the given references for a comprehensive follow up.

The following pages in this chapter explain the concepts practiced while developing our framework in detail, and the content is organized as follows: Section 2.1 gives a brief overview of the key points of our proposed method. In Section 2.3 we explain scattering wavelet transform (SWT), the preliminary stage of our feature extraction pipeline, executing *change of domain* on audio signals to 2D representation, and before in Section 2.2 we delve into wavelet theory to have an integral understanding. Section 2.4 provides necessary information for convolutional neural networks, a deep neural network block used in our embedding hash model. In Section 2.5 we present a deep neural network model coupled with SWT coefficients as input to generate final audio fingerprints as embeddings, and in Section 2.6 the mentality how the model is trained examined.

## 2.1 Introduction

Followings sections explain various concepts used in this study to have grasp understanding the contextual background information used. Namely, first we start with wavelet theory in Section 2.2 and continue to explain Scattering Wavelet Transform in Section 2.3 to construct our first stage of fingerprinting scheme. In Section 2.4 and Section 2.5 we build our deep fingerprint hash model as the second stage based on scattering-wavelet-transform output coefficients. Section 2.6 explains the final embedding hash model and strategies how it is trained. Methodology sections (if available) describe how the concrete implementation is applied based on the

technical background explained in detail. The overall scheme of audio fingerprinting is depicted in following Fig. 2.1 stroked with the dashed rectangular area:

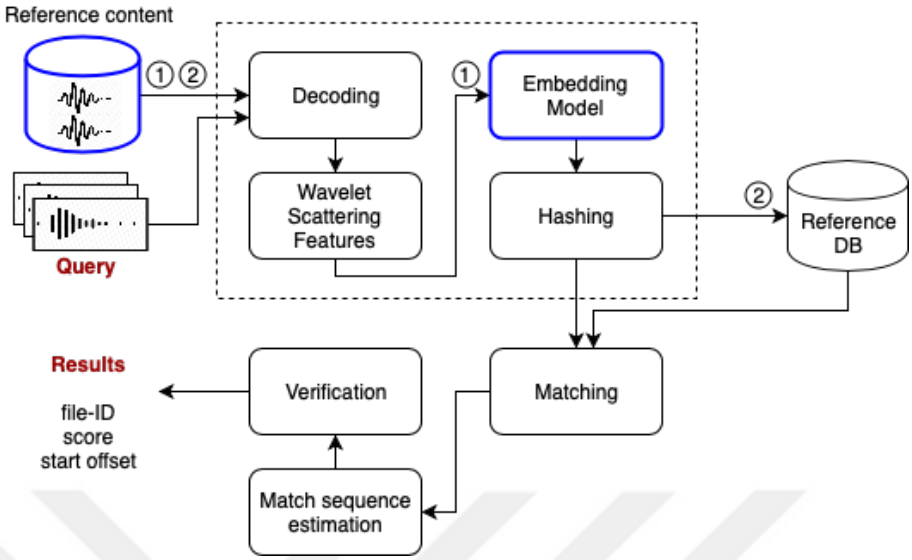



**Figure 2.1** Concepts - system scheme overview

Scattering Wavelet Transform have a significant importance in this study, thus to evaluate definitive characteristics of the transform on musical audio data, it is first studied on *musical genre classification* task and the results are reported with following publication:

- E. Kanalici and G. Bilgin, "Music Genre Classification via Sequential Wavelet Scattering Feature Learning." In International Conference on Knowledge Science, Engineering and Management, pp. 365-372. Springer, Cham, 2019.

where wavelet scattering coefficients are used as features providing both translation-invariant representation and transient characterisations of audio signal to predict musical genre. Extracted features are fed to sequential architectures to model temporal dependencies of musical piece more efficiently.

Later, the main subject of this study evaluated with limited configuration, that is only using one-to-one comparison against content storage for audio labelling task to avoid hash storage artefacts. Similarly as in Fig. 2.1, a two-stage feature extraction framework using SWT coupled with deep Siamese hashing model for musical audio labelling is proposed. Similarity-preserving hashes are the final fingerprints and in the projected embedding space, similarity is defined by a distance metric. Hashing model is trained by roughly aligned and non-matching audio snippets to model musical audio data via two-layer scattering spectrum. The results reported in the following publication:

- E. Kanalici and G. Bilgin, "Scattering Wavelet Hash Fingerprints for Musical Audio Recognition", In International Journal of Innovative Technology and Exploring Engineering, pp. 1011-1015. BEIESP, 2019.

## 2.2 Wavelet Theory

Wavelet theory as an independently developed framework is a signal analysis/synthesis tool which can represent non-stationary signals with dynamically scaling window size and translations.

> [30] Wavelet theory provides a unified framework, some foundations developed independently, for various signal processing applications. Including multi-resolution signal processing for computer vision, sub-band coding in speech and image compression, and wavelet series expansions developed in applied mathematics . The *Wavelet Transform* (WT) interests with the analysis of *non-stationary* signals and it provides an alternative approach to Short-Time Fourier Transform (STFT) or Gabor transform. In contrast to the STFM which uses a single analysis window, the WT uses short windows at high frequencies and long windows at low frequencies[31].

WT can be considered as a signal decomposition into a set of basis functions, which are called as *wavelets*. Wavelets are generated from a single prototype wavelet (*mother wavelet*) by scaling (i.e. dilations and contractions) and shifts. The prototype wavelet is analogous to a bandpass filter, and since the obtained wavelets are dilated and contracted versions of the prototype, they satisfy "constant-Q" property, having multiple of frequency bandwidths consecutively. WT presents the notion of *time-scale* plane instead of *time-frequency* as in the case of STFT, which the signal interested in mapped into.

### 2.2.1 Fourier Transform

The signal analysis interests extracting relevant information from a signal by a means of some transformation. Some analysis techniques are based on a priori assumptions on the signal like being *stationary* (i.e. not evolving over coarse of time). Fourier transform, given in Eq. (3.1), is such a stationary transform, the analysis coefficients $X(f)$ define the notion of trend of frequency $f$ in whole time-dependent signal. It transforms the signal to frequency domain and the energy concentrations in the

frequency spectrum imply the most dominant frequencies in the analyzed signal without any time information.

As a result Fourier analysis works well if x(t) is composed of a time-dependent stationary components without any abrupt change over the coarse of time. Most signals posses non-stationary characteristics in their nature especially musical audio, thus some notion of time-frequency localization should be defined to well describe these signals of interest that have a dynamic nature.

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-2j\pi f t} dt \tag{2.1}$$

Eq. (3.1) can be interpreted as the similarity detection between frequency components and a given signal since the dot product of the signal $x(t)$ with the complex sinusoids amounts for. Certain frequencies will overlap more with the signal $x(t)$ that results large amplitude for the Fourier transform for specific frequency values. Thus although Fourier transform has well defined frequency resolution, it lacks of time resolution as demonstrated in following figure. In Fig. 2.2 (a) a stationary signal with immanent four different frequencies and its frequency spectrum is shown. Fig. 2.2 (b) shows another signal with four different frequencies but changing over course of the time. The frequency spectrum of the second signal is present. As it can be seen both spectrum plots have similar characteristics with amplitude peaks at the same frequency instances. Also frequency spectrums have no information about time.

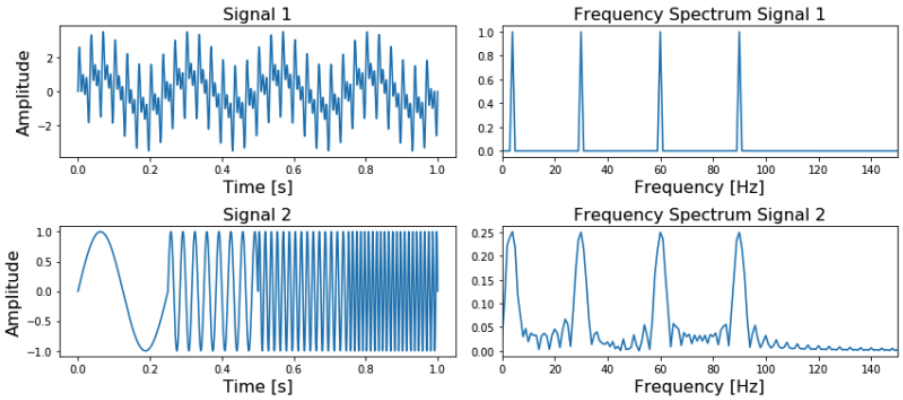



**Figure 2.2** Frequency spectrums of stationary and non-stationary signals

Time-frequency localization can be introduced by looking the whole signal by only through segmented windows. The Windowed Fourier transform (WFT) is an analysis tool specialized for extracting *local-frequency* information from a signal. Since for non-stationary signals Fourier transform fails to represent localized information, WFT provides localization using segmented time frames. WFT is applied with sliding frames

of length $T$ from a time series of time step $\delta t$ and with total length equals $N\delta t$, thus extracting frequency in range $[T^{-1}, (2\delta t)^{-1}]$ at each time step. The frames can be windowed with an arbitrary window functions such as a a Gaussian window [32].

As discussed in [32], the WFT time-frequency localization is inaccurate and inefficient. The inaccuracy comes from having limited frequency range, gives result to the aliasing frequency components that do not fall within the range. The inefficiency comes from the $T/(2\delta t)$ frequencies which must be analyzed at each time step. Thus, wavelet transform, being a scale independent analysis method may be employed for signals that are non-stationary and have wide range of frequency characteristics.

### 2.2.2 Short-time Fourier Transform

Describing a signal on time-frequency domain requires a 2D representation retrieved from a defined transformation. Since this instantaneous frequency concept can not be defined over infinitely small time ranges, averaged frequency may be the solution to describe spectral characteristics over time domain. Thus the transformation maps a signal of interest to a time-frequency plane using time windows for spectral averaging.

To introduce frequency dependence on time, the Fourier transform was adapted by Gabor [31] defining instantaneous frequency coefficients $S(t, f)$ by using a finite support window function $g(t)$ centered at $\tau$:

$$\mathrm{STFT}(\tau, f) = \int x(t) g^*(t-\tau) e^{-2j\pi f t} dt \tag{2.2}$$

STFT maps windowed signals $x(t)g^*(t-\tau)$ into two-dimensional function in a *time-frequency* plane $(\tau, f)$. In Eq.(3.2) the Fourier analysis is applied to windowed signals and the analysis depends critically on the choice of window function $g(\cdot)$. Fig. 2.3 shows time series analysis with vertical lines on time-frequency plane which amounts to time localization (a), and the Fourier transform corresponds to horizontal regions on the plane for stationary signals that amounts to frequency localization (b). Using a sliding windowed analysis enables localized view for both time and frequency for STFT as depicted in (c).

STFT can be thought of an either windowed time analysis or modulated filter-bank since it divides time-frequency plane in both axis. But an arbitrary resolution for both domain has limitations. For given window function $g(t)$ and its Fourier transform $\widehat{g}(f)$, the square of frequency resolution (frequency bandwidth) $\Delta f$ and the square of time resolution (spread in time) $\Delta t$ are given:

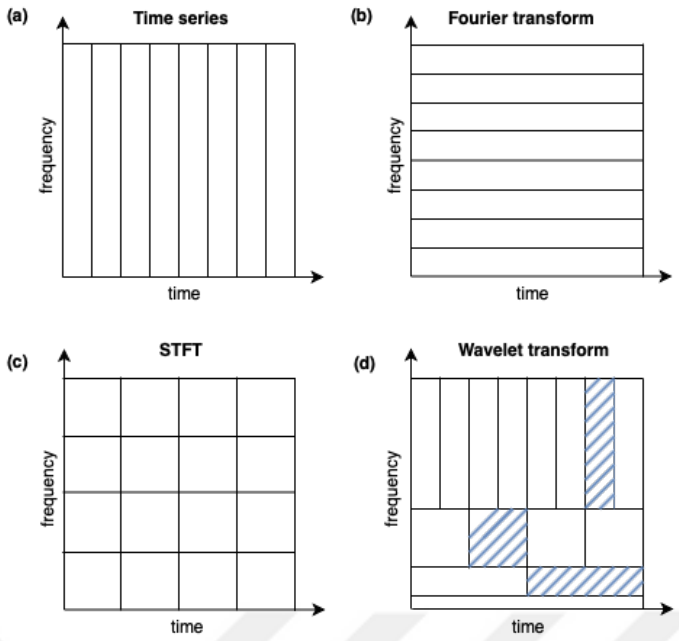



**Figure 2.3** Wavelet transform time-scale resolution vs. STFT time-frequency resolution

$$\Delta f^2 = \frac{\int f^2 |\widehat{g}(f)|^2 df}{|\widehat{g}(f)|^2 df} \tag{2.3}$$

$$\Delta t^2 = \frac{\int t^2 |g(t)|^2 df}{|g(t)|^2 dt} \tag{2.4}$$

where $|\widehat{g}(f)|^2 df$ and $|g(t)|^2 dt$ are the energy of the signal.

Two spectral signals (e.g. sinusoids) can be discriminated if they are $\Delta f$ apart and two pulses can be discriminated if they are $\Delta t$ apart on frequency and time domains respectively. The resolution in time and frequency cannot be arbitrarily small since their product is lower bounded by Eq.(3.5), resulting a trade-off between time and frequency resolution by namely *uncertainty principle* (or *Heisenberg inequality*).

$$\Delta t \Delta f \geq \frac{1}{4\pi} \tag{2.5}$$

Wavelet transform provides a different approach which enables more fine grained time-frequency resolution by introducing the concept of scale. By analyzing the signal with different varying scales a better tradeoff for frequency component extraction (i.e. high resolution for small frequencies and low resolution for large frequencies) is achieved which is coarsely depicted in Fig. 2.3 (d).

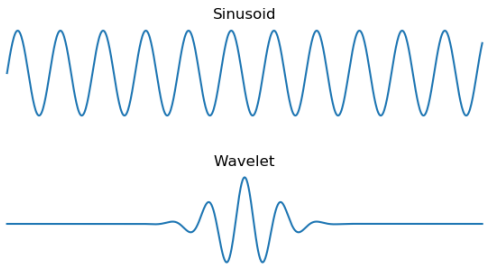


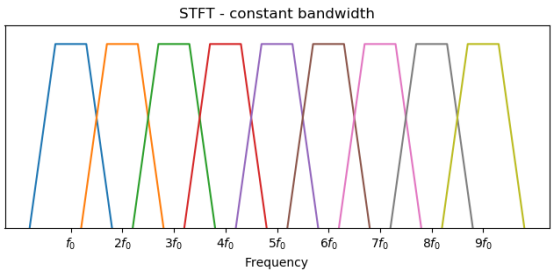

**Figure 2.4** Sinusoid vs. wavelet

### 2.2.3 Wavelet Transform

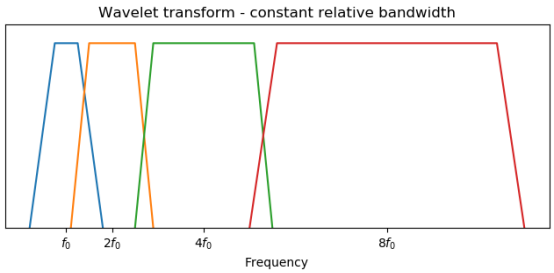
Fourier transform were linear combinations of the signal with complex sinusoids $e^{-2j\pi f}$, whereas in the wavelet transform case it is a combinations of functions called as *wavelets*. Generated from a prototype wavelet (namely *mother wavelet*), wavelet functions are localized in time with a limited support while a sinusoid is spread in time infinitely (see Fig. 2.4). Also prototyping a wavelet provides scale changes (i.e. dilations and contractions) and since its localized in time it can be time shifted, thus localization in both frequency and time domain can be achieved by applied convolutions on the signal with scaled and translated wavelets. Resulting time-scale representation of the signal is a 2D representation and called *Scaleogram*.

To tackle resolution limitation of STFT, multi-resolution analysis can be applied by varying $\Delta f$ and $\Delta t$ whose bounded by Eq.(3.5). In multi-resolution approach time resolution increases subject to central frequency of the filter-bank analysis filters. Thus, proportionally the frequency resolution ($\Delta f$) is subject to central frequency and the relation can be imposed as: $\Delta f = cf_0$ for a constant value $c$. Having analysis filter-bank satisfying this relation is called *constant-Q* analysis. Thus each analysis filters have constant relative bandwidth which are regularly spread in logarithmic scale rather than a uniform spread as in STFT case (Fig. 2.5). While $\Delta f$ changes with spread, satisfying the bounded relationship of uncertainty principle, $\Delta t$ also changes and time-resolution improves high-frequencies while the frequency-resolution improves at low frequencies.

Wavelet transform maps a given signal to time-scale plane. Relating concept of scale to a pseudo-frequency can be given by $f = a^{-1}f_0$ where $f_0$ is the central frequency of the prototype wavelet and $a$ is the scaling factor. Following the relation, the scaling factor $a$ has an inverse relation with pseudo-frequency values. Noted that, the term pseudo-frequency ($f$) has a narrow relation to frequency modulation as in STFT case, which depends only prototype wavelet time-scalings. Thus, the term "scale" is preferred to "frequency" for wavelet transform.




**(a)** STFT frequency bandwidth




**(b)** Wavelet transform frequency bandwidth

**Figure 2.5** STFT and Wavelet transform bandwidths on frequency spectrum

There are different families of wavelets developed in literature, each have their own characteristics which fits best with the designed features for different purposes. These characteristics include the shape, smoothness and compactness of wavelet functions.

To be *admissible* as a wavelet function, it must have have zero-mean and be localized in both frequency and time. Localization in both time and frequency provided by having a finite energy, that it a wavelet function should be integrable and the inner product with any signal should exists. Some describing characteristics of wavelets are:

- Orthogonality: orthogonal, non-orthogonal or bi-orthogonal.
- Symmetry: symmetric or non-symmetric.
- Wavelets can be real or complex values. In complex case, the real part amounts to the amplitude and the imaginary part amounts to the phase.
- Wavelets are generally used with normalization factor, providing a unit energy.

Also noted, *wavelet function* and *wavelet basis* have conceptually different meanings. The term *wavelet function* is used generally to refer to either orthogonal or non-orthogonal wavelets. Whereas the term *wavelet basis* refers only to an orthogonal set of functions. The use of an orthogonal basis implies the use of the *discrete wavelet transform*, while a nonorthogonal wavelet function can be used with either the discrete or the *continuous wavelet transform* [33].

Based on characteristics, various families of wavelets are developed in the literature. In Fig. 2.6 some of the several different families of wavelets are plotted.

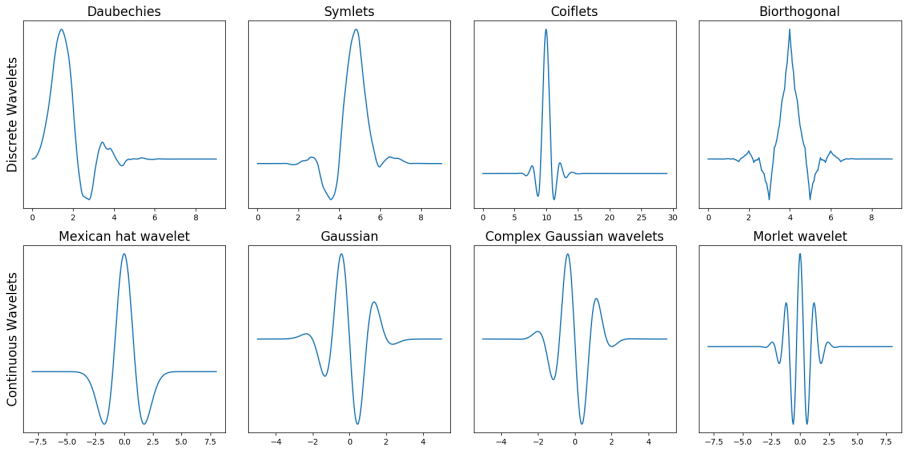



**Figure 2.6** Several families of wavelets

Also for the wavelet families there can be different subcategories distinguished by order (*vanishing moments*) and decomposition level.

The scaling function of a wavelet with $p$ vanishing moments can approximate polynomials up to a degree of $p-1$. The "vanishing" term refers to that wavelet coefficients goes zero for approximated polynomials of degree at most $p-1$, thus the scaling function alone is enough to represent such functions. More vanishing moments amounts to that the wavelets can approximate more complex polynomials.

Some instances of the "Daubechies" family wavelets are plotted in Fig. 2.7. In the figure columns enumerates the order of Daubechies wavelets and rows corresponds to the level of decompositions. Vanishing moments determines the smoothness and approximation order of wavelets. As it can be seen as the number of vanishing moments goes up along a row, the wavelet becomes smoother since its polynomial degree increases. And while the decomposition level increases along a column, the number of wavelet samples increases.

An example of complex wavelet function is the complex Morlet wavelet, which is a Gaussian modulated wave:

$$\psi_0(n) = \phi^{-1/4} e^{iw_0 n} e^{-n^2/2} \tag{2.6}$$

where $w_0$ is the non-dimensional frequency.

Complex Morlet wavelet filter function and its frequency response are plotted in Fig 2.8.

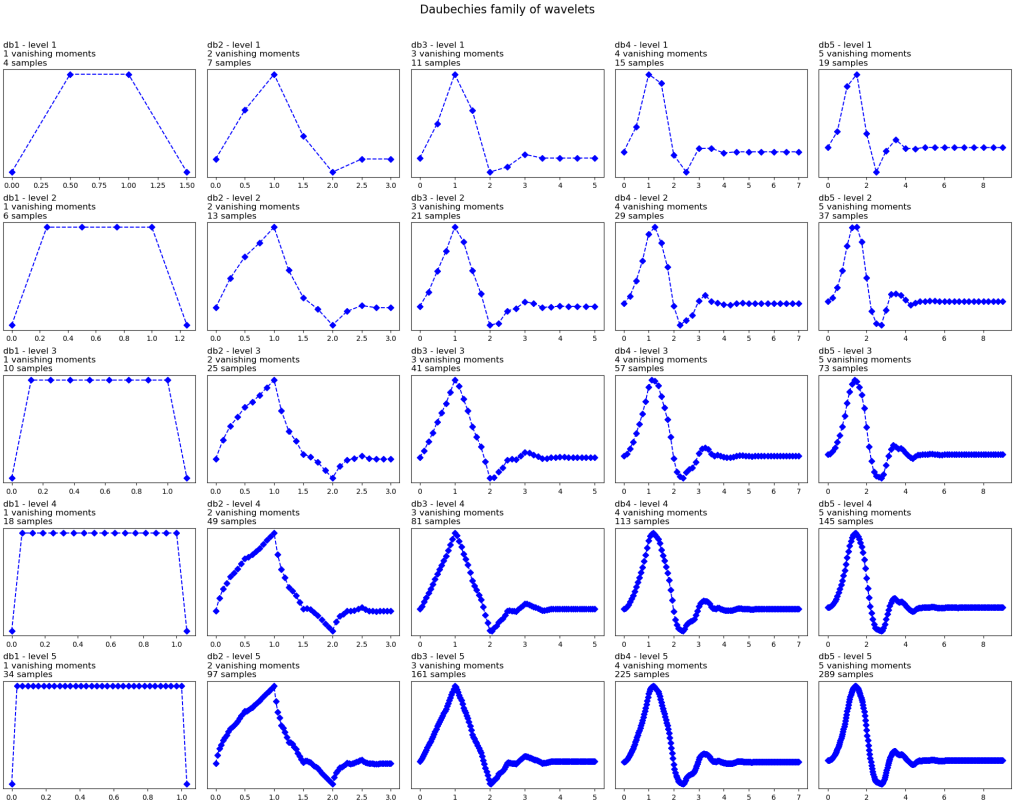



**Figure 2.7** Daubechies family wavelets

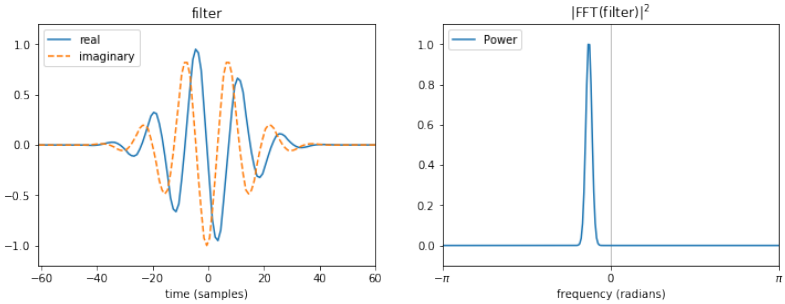



**Figure 2.8** Complex Morlet wavelet filter and its frequency response

Wavelet transform has two different and theoretical approaches, namely the continuous wavelet transform and the discrete wavelet transform.

The continuous wavelet transform (CWT) follows the resolution limitation with an additional scheme: all impulse responses of the filter-bank are defined as the scaled (dilated or contracted) versions of the prototype function $h(t)$:

$$h_a(t) = \frac{1}{\sqrt{|a|}} h(t/a) \tag{2.7}$$

where $a$ is a scale-factor and the constant $1/\sqrt{|a|}$ is normalization factor.

The definition of CWT is given as:

$$CWT_x(\tau, a) = \frac{1}{\sqrt{|a|}} \int x(t) h^*(\frac{t-\tau}{a}) dt \tag{2.8}$$

By varying the wavelet scale-factor $a$ and translating along the time variable $\tau$, time-scale localization is achieved. In Eq. (3.8) the scale-factor and translation time variable are continuous, thus the number of wavelets is not finite.

Noted that, the modulated window used in STFT can be defined in form of the basic wavelet $h(t)$:

$$h(t) = g(t) e^{-2j\pi f_0 t} \tag{2.9}$$

Then the frequency responses of the analysis filters satisfy "constant-Q" relation ($f = f_0/a$). But more generally, $h(t)$ can be chosen to be any band-pass function.

Discrete wavelet transform (DWT), as distinct from CWT, the scale-factor and translation time values are not continuous but discretely sampled. The scale-factor is two-powered integer values ($a \in 2^j$), and the translation is discrete integer values ($\tau \in \mathbb{Z}$). The DWT is only discrete for the scale and translation values, but not in the time-domain. To be able to make analysis of signals which are discrete in the time-domain the wavelet transforms should also be discretized in the time-domain. These wavelet transform formations are referred to namely the discrete-time wavelet transform and the discrete-time continuous wavelet transform.

### 2.2.4 Discrete Wavelet Transform

To make discrete-time analysis of wavelet transforms possible, sub-band coding [34] and multi-resolutional analysis [35] were developed independently as backing foundation of *discrete wavelet transform*. These coding methods introduce critical sampling (acquiring minimum samples for information). In coding scheme of discrete wavelet transform, scale corresponds to up- down-sampling for small and large values respectively.

In *multi-resolutional analysis*, given a sequence $x(n), n \in \mathbb{Z}$, lower resolutional signal is derived by lowpass filtering by a half-band low-pass filter having impulse response $g(n)$. following the Nyquist's rule subsampling by two corresponds to doubling the scale in analysis:

$$y(n) = \sum_{k=-\infty}^{k=+\infty} g(k)x(2n-k) \tag{2.10}$$

In Eq.(3.10) lowpass filtering corresponds to resolution change and sub-sampling by two corresponds to scale change. Retrieving lowpass and downsampled version of the signal, the approximation is obtained by upsampling $y(n)$ by two and interpolation:

$$\begin{aligned} y'(2n) &= y(n) \\ y'(2n+1) &= 0 \end{aligned} \tag{2.11}$$

$$a(n) = \sum_{k=-\infty}^{k=+\infty} g'(k)y'(n-k) \tag{2.12}$$

If $g(n)$ and $g'(n)$ are perfect half-band filters (having a frequency passband equal to one in frequency range $(-\pi/2, +\pi/2)$ and zero elsewhere), the the Fourier transform of approximation $a(n)$ would be equal to Fourier transform of original signal $x(n)$ over the frequency range $(-\pi/2, +\pi/2)$ and zero elsewhere, that is $a(n)$ would be a perfect half-band lowpass approximation of $x(n)$. But in general case for non-perfect approximations, the error term given:

$$d(n) = x(n) - a(n) \tag{2.13}$$

$x(n)$ can be reconstructed by sum of approximation and error term $(a(n) + d(n))$ as shown in Fig. 2.9, but there exists some redundancy since original signal $x(n)$ with

sampling rate $f_s$ is disjoint into two composed signals $d(n)$ and $y(n)$ with sampling rates $f_s$ and $f_s/2$ accordingly. In case of perfect half-band low-pass filter, error term $d(n)$ contains frequencies above $\pi/2$ of $x(n)$, thus shows $d(n)$ can be subsampled by two without information loss thus namely via *critically sampling*.

In the nature, the coding scheme of separating the signal $x(n)$ into coarse approximation $a(n)$ and additional detail $d(n)$ involves halving the resolution and doubling the scale (lowpass filtering followed subsampling by two), and this scheme can be iterated on $y(n)$ creating tree of lower resolution signals at lower scales.

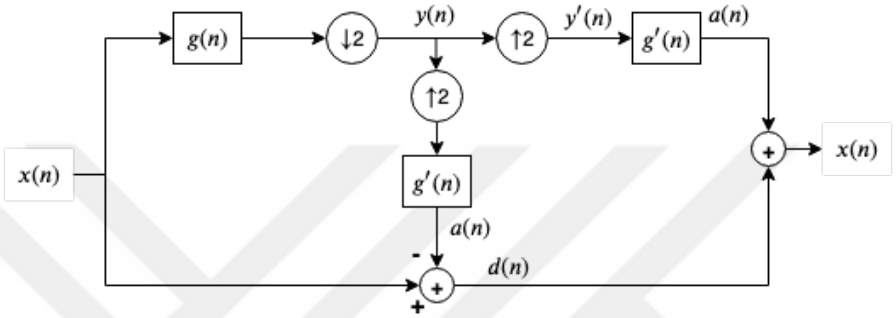



**Figure 2.9** Multiresolution scheme

Multi-resolutional analysis creates redundant set of components, that is one stage of decomposition leads half rate approximation and full rate difference, resulting 50% increase in number of samples. It's shown this *oversampling* may be avoided if lowpass filters meet certain conditions [36].

*Sub-band coding* (or filter-bank) scheme provides no redundancy where the lowpass subsampled approximation is obtained same, but instead of calculating difference error term, the additional detail term is calculated by high-pass filter $h(n)$ and subsampling on by two on original signal $x(n)$ as shown in Fig. 2.10. Assuming that lowpass filter $g(n)$ is perfect half-band, a perfect half-band high-pass filter $h(n)$ would lead to perfect reconstruction.

This scheme corresponds to one step of wavelet decomposition using *sinc* (i.e. $sin(x)/x$) filters decomposing the signal to lowpass and high-pass components at double scale. In particular these ideal filters are used, the discrete version is identical to continuous wavelet transform [30]. It's shown that without using ideal filters, yet it is possible to recover original signal $x(n)$ from its two filtered and subsampled components $y_0(n)$ and $y_1(n)$. To reconstruct both upsampled and filter by $g'(n)$ and $h'(n)$ respectively and superimposed (see. Fig 2.10). The reconstructed signal $\widehat{x}(n)$ is not identical to original signal $x(n)$ unless the filters meet specific constraints. Filter that meet these specific constraints are called having *perfect reconstruction* property,

and practically have techniques for filter design [37].

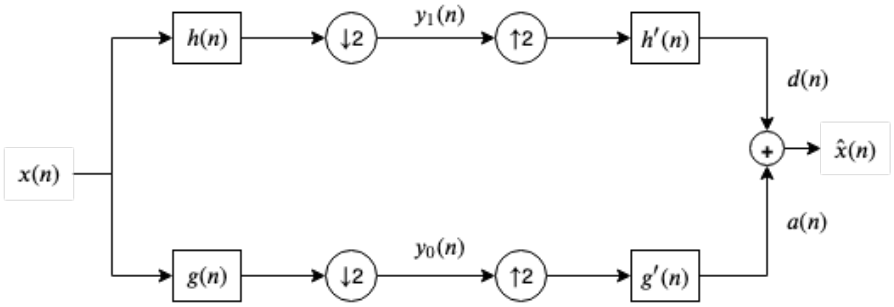



**Figure 2.10** Subband coding scheme

Assuming that the analysis and synthesis filters are identical (provided they are time-reversed versions of one another) and perfect reconstruction is satisfied (i.e. $x(n) = \hat{x}(n)$), it is shown that this sub-band analysis/synthesis filter pair corresponds to an *orthonormal basis* and the reconstruction achieved by summing up orthogonal projections. Using FIR filters, the relation between analysis and filters are given:

$$h(L-1-n) = (-1)^n g(n) \tag{2.14}$$

where $L$ is the length of the signal and the modulation $(-1)^n$ transforms lowpass filter $f$ to high-pass filter $g$.

In Fig.2.10 the first stage of convolutions followed by subsampling of two evaluates the inner product of signal $x(n)$ and the sequences $g(-n+2k)$, $h(-n+2l)$ where time-reverse come from convolution:

$$\begin{aligned} y_0(k) &= \sum_n x(n)g(-n+2k) \\ y_1(k) &= \sum_n x(n)h(-n+2k) \end{aligned} \tag{2.15}$$

Since the set of filter impulse responses is orthonormal, it is then reconstruction of $x(n)$ is given by sum of weighted impulse responses:

$$\hat{x}(n) = \sum_{k=-\infty}^{k=+\infty} [y_0(k)g(-n+2k) + y_1(k)h(-n+2k)] \tag{2.16}$$

The weights are inner products of signal with the impulse responses. Also from Eq.(3.14) and Eq.(3.16) it is clear that the synthesis filters are also the analysis filters

with time-reversal [30].

Fig. 2.10 show only one step of sub-band coding scheme, which may be iterated for multiple levels to achieve decomposition of a signal. Each iteration extracts the approximation coefficients and detail coefficients by low-pass filter $g(n)$ and high-pass filter $h(n)$ respectively. To retrieve wavelet transform coefficients iteratively, sub-band coding scheme is applied on the approximation coefficients from the previous levels. At each subsequent level, signal approximation is first down sampled by factor of two as in Fig. 2.10, then the sub-band filters (i.e. $g(n)$ and $h(n)$) separates new coarse low-pass and detailed high-pass components.

Fig. 2.11 depicts the sub-band coding decomposition of chirp signal. A chirp signal has time-dependent frequency spectrum that is increasing or decreasing linearly. The plotted signal is a up-chirp signal with an increasing frequency along the time axis. In Fig. 2.11 6-level of decomposition is applied. As the level of decomposition increases, detailed coefficients have decreasing frequency bands at each level as expected since DWT is applied from previous subsequent approximations.

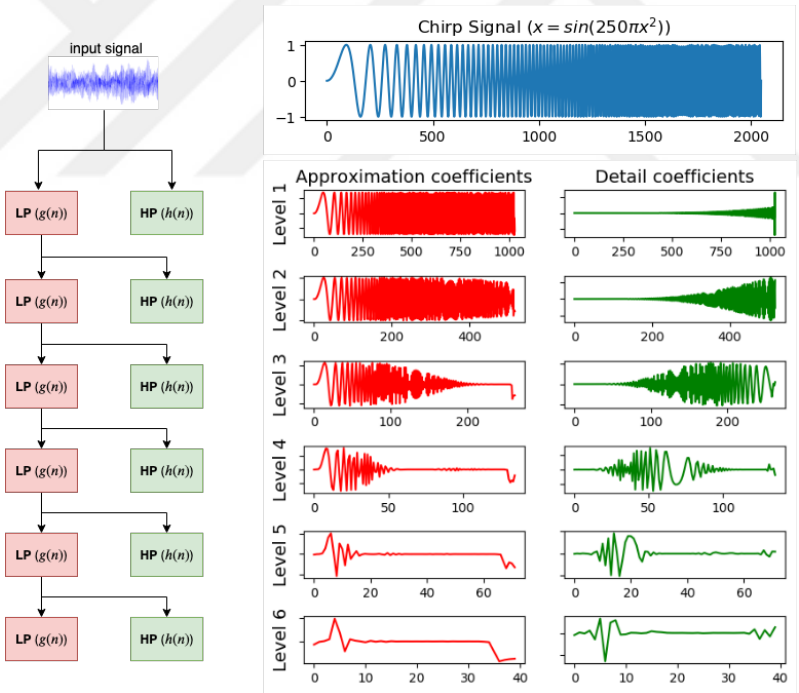



**Figure 2.11** Discrete wavelet transform - subband coding of chirp signal

## 2.3 Scattering Wavelet Transform

Wavelet Scattering Transform (SWT), presented by Mallat [38], is a signal decomposition based on arguments of translation-invariance and stability under dilation and transposition. SWT is implemented as similar to convolution networks

whose filters are not learned, but fixed being wavelet filters, and computed through a cascade of wavelet decompositions iteratively.

**Definition 2.1.** Translation invariance Discrete equivalent of time-invariance, that is for a given translation-dependent input function $x(t)$ and translation-dependent output function $y(t)$; the system will be considered translation-invariant if $y(n-k)$ is the system response to input $x(n-k)$ [39].

**Definition 2.2.** Lipschitz continuity A real-valued function $f : \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous if a positive constant $K \in \mathbb{R}$ exists such that:

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{2.17}$$

Let $\widehat{x}(w)$ be Fourier transform of $x(t)$. The Fourier modulus is translation invariant for given $x_1(t) = x(t - t_1)$:

$$|\widehat{x}_1| = |\widehat{x}| \tag{2.18}$$

For a given deformation $x_\tau(t) = x(t - \tau(t))$ which satisfies Eq. (3.19), a mapping function $\Phi(x)$ is *stable under deformation* $\tau$ if a small deformation implies as small change in the mapping function that is Lipschitz continuity in Eq. (3.20) holds for all $x(t)$ and $\tau(t)$:

$$|d\tau(t)/dt| < 1 \tag{2.19}$$

$$\|\Phi x - \Phi(x_\tau)\| < C \sup_t |\nabla \tau(t)| . \|x\| \tag{2.20}$$

**Corollary 2.0.1.** *Modulus of short-time Fourier transform (spectrograms) are unstable.*

Dilation defined as $x_\tau(t) = x(t - \epsilon t)$ is a deformation function which satisfy Eq. (3.19): $\sup_t |d\tau(t)/dt| = \epsilon$. Armed with the definition, log power spectrums of the original and dilated version of a harmonic signal with two time instances is depicted in Fig. 2.12.

As it can be seen in Fig. 2.12, frequency shifts in high frequencies is greater than the bandwidth, so short-time Fourier transform (STFT) modulus is not deformation invariant under dilation:

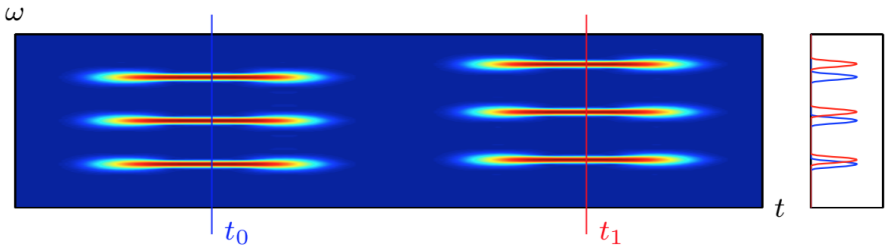



**Figure 2.12** STFT of dilated harmonic signal [40]

$$\|\widehat{x} - \widehat{x}_\tau\| \not\leq C \sup_t |d\tau(t)/dt| . \|x\| \tag{2.21}$$

High-frequencies are more prune deformation instabilities, the major difficulty is to main the Lipschitz continuity over high-frequency bands [38].

**Corollary 2.0.2.** *Mel-frequency representations are stable.*

With insights that short-time Fourier transform (STFT) is unstable, mel-frequency representation provides stability by band-pass averaging:

$$Mx(t,\lambda) = \frac{1}{2\pi} \int |\widehat{x}(w,t)| \left|\widehat{\psi}_\lambda(w)\right|^2 dw \tag{2.22}$$

where $\widehat{x}$ is STFT and $\widehat{\psi}_\lambda$ is a band-pass filter at mel-frequency $\lambda$. Frequency bins of mel-scale defined as:

$$\lambda(f) = 1127 ln(1 + \frac{f}{700}) \tag{2.23}$$

The mel-scale logarithm prevent the progressive instability of harmonics (i.e. multiples of *base frequency*) $f = nf_0$ since:

$$\begin{aligned} \lambda(f) \approx & 1127 \ln \frac{f}{700} \\ = & 1127(\ln n + \ln f_0 - \ln 700) \end{aligned} \tag{2.24}$$

And thus mel-frequency representation, thanks to band-pass averaging has stability under deformations, including $\epsilon$ dilations:

$$\begin{aligned} \|Mx(\lambda,t) - Mx_{\tau}(\lambda,t)\| &< C \sup_{t} |d\tau(t)/dt| . \|x\| \\ &= C\epsilon \|x\| \end{aligned} \tag{2.25}$$

It can be seen in Fig (3.2), frequency peaks now overlap in high-frequencies.

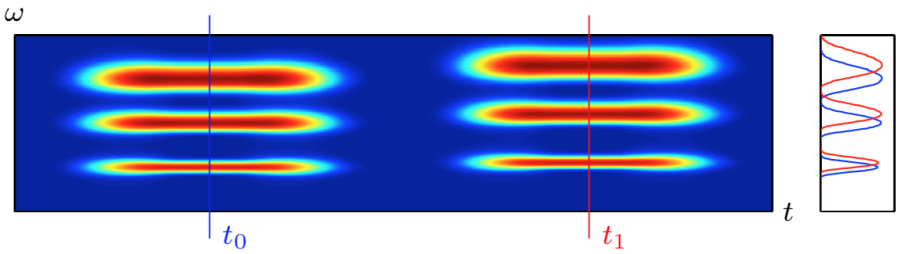



**Figure 2.13** Mel-frequency scale averaging with overlapping high-frequencies [40]

**Corollary 2.0.3.** *Time averaging results information loss.*

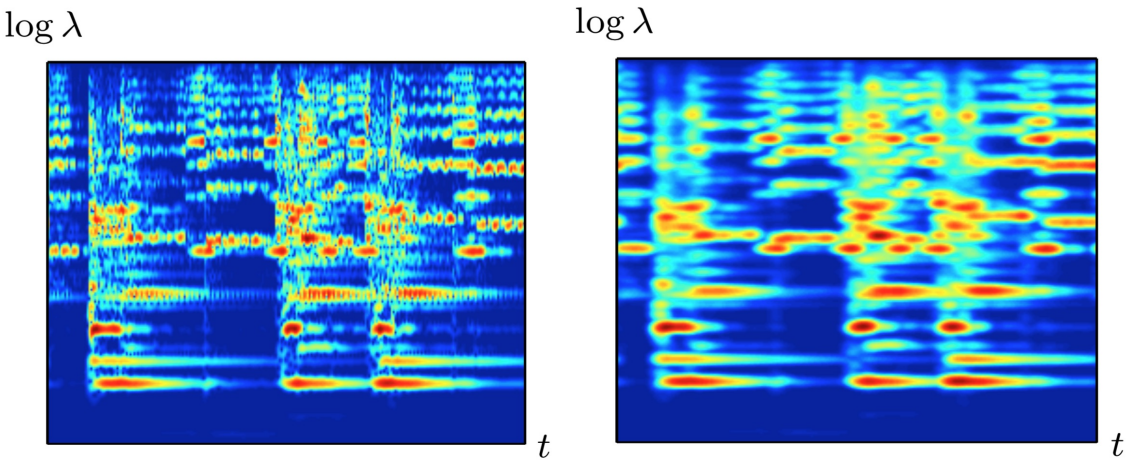
The deformation stability comes at the cost of information loss (1) in high frequencies that are averaged and (2) in short-time windowed temporal structure. Mel-frequency can be approximated by convolutions and time-averaging [40]:

$$Mx(\lambda,t) \approx |x \star \psi_{\lambda}|^2 \star |\phi|^2(t) \tag{2.26}$$

where the windowing function $\phi$ acts as a local time-averaging of the mel-frequency response $|x \star \psi_{\lambda}|$.

Actually in Eq. (3.26) it is shown that frequency averaging is equivalent to a time averaging of the filter bank output [40]. Keeping windows (i.e. support of lowpass filter $\phi$) short prevents the mel-spectrogram from averaging away too much information. Considering $|\phi|^2(t)$ as a low-pass filter, the information loss is contained in the high-frequency components of system response (i.e. amplitude modulations of $|x \star \psi_{\lambda}|$). In Fig. 2.13 its apparent the loss of information when a lowpass time-averaging filter applied with an arbitrary temporal window size.

Fig. 2.14 (a) and Fig. 2.14 (b) shows original ($|x \star \psi_{\lambda}|^2$) and time-averaged ($|x \star \psi_{\lambda}|^2 \star |\phi|^2$) scaleograms of a musical recording respectively. The window duration for averaging filter $\psi$ is $T = 190ms$. This time averaging discards detailed transient information of the signal such as attacks and tremolo. Mel-frequency spectrogram is often computed over arbitrary small time windows of and thus it is unable to capture large-scale structures.




**(a)** Scaleogram $\log |x \star \psi_\lambda|^2$

**(b)** Averaged scaleogram with lowpass filter $\phi$ of duration $T = 190ms$.

**Figure 2.14** Scaleograms of musical signal - time averaging information loss [40]

Most of the transient characteristics (e.g. attacks, modulations and tremolo) of the signal is lost by this time-averaging (Fig. 2.14). As a result, one should be able to increase $T$ without losing too much information and capture the transient characteristics of $|x \star \psi_\lambda|$ at scales smaller than $T$. Multi-layer scattering transform, as a solution, provides both mel-frequency and modulation features by recovering the mentioned averaged lost information.

For a given signal $x$, the wavelet transform $Wx$ was defined as convolutions with a averaging operator (low-pass filter) $\phi$ and higher frequency wavelets $\psi_\lambda$:

$$\begin{aligned} Wx(t,\lambda) &= \Big(x \star \phi(t), x \star \psi_\lambda(t)\Big)_{t\in\mathbb{R},\lambda\in\Lambda} \\ \psi_\lambda(t) &= 2^{-jQ}\psi(2^{-jQ}t); \qquad\qquad \lambda = 2^{-jQ}t \end{aligned} \tag{2.27}$$

In Eq.(3.27):

- Wavelets $\psi_\lambda$ are high-order ($\lambda > 0$), and low-pass filter $\phi$ corresponds to *zeroth-order* ($\lambda = 0$).
- $\psi_\lambda$ are dilated band-pass filters with constant mel-scale bandwidth.
- $Q$ is the *quality-factor* of Q-constant band-pass filters $\widehat{\psi}_\lambda$, that is number of wavelets per octave.
- The representation oversampled and redundant.
- Wavelet transform preserves energy.

To have an informative invariant which is not zero (as opposed to being zero-sum as $\int x \star \psi(t)dt = 0$), it has to have non-linearity (sigmoid, arc-tangent, rectification etc.). The non-linear requirements are:

- *Diffeomorphism* (i.e. deformation) stability: $\Phi$ commutes with diffeomorphism.
- $\mathbf{L}^2$ stability (*contractive*): should be stable to additive perturbations (the norm should not explode).

Coarsely, the basic intuition of SWT is based on applying a non-linear map $\Phi$ to time-averaged signal (Eq. (3.28)), which is translation-invariant and commutes with diffeomorphism and $\mathbf{L}^2$ stable to additive perturbation since its contractive and preserves the norm. Modulus operator $|\bullet|$, is the only choice having these particular properties i.e. being a non-linear map, translation-invariant and stable to deformations as argued in [40] (Eq. (3.29)).

$$\int \Phi(x \star \psi_\lambda)(t)dt \tag{2.28}$$

$$\begin{aligned} \|\Phi(h)\| &= \|h\| \\ \|\Phi(g) - \Phi(h)\| &\leq \|g - h\| \\ \Rightarrow \Phi(h)(t) = |h(t)| &= \sqrt{|h(t_r)|^2 + |h(t_i)|^2} \end{aligned} \tag{2.29}$$

A modulus computes a smooth lower frequency envelop for complex waveforms. The integral of modulus is $\mathbf{L}^1$ norm which is non-zero and stable invariant (Eq. (3.30)):

$$\int |x \star \psi_\lambda(t)|dt = \|x \star \psi_\lambda\|_1 \tag{2.30}$$

The wavelet power spectrum extracts time-windowed envelopes at different resolutions (Eq. (3.31)). Modulus discards phase information but retains sufficient information in the nature of wavelet transform being redundant.

$$|W|\, x = \left( x \star \phi(t), \left| x \star \psi_{\lambda_1}(t) \right| \right)_{t \in \mathbb{R}, \lambda_1 \in \Lambda_1} \tag{2.31}$$

Whereas zeroth-order coefficients $S_0 = x \star \phi(t)$ is locally translation-invariant thanks to modulus operator, the high-frequency information is lost by time-averaging $\phi$.

Although lost information is obtained with modulus of wavelet coefficients $\left|x \star \psi_{\lambda_1}\right|$, yet these coefficients are not time-shift invariant. Provided that local time-shift invariance is obtained by time averaging as in $S_0$, applying averaging again gives first-order of scattering coefficients:

$$S_1 x(t, \lambda_1) = \left|x \star \psi_{\lambda_1}\right| * \phi(t) \tag{2.32}$$

If the frequency resolution of wavelets $\psi_{\lambda_1}$ are the same as the standard mel-scale i.e. $\lambda(f) \approx 1127\ ln(f/700)$, **the first-order coefficients $S_1 x$ approximate the mel-spectrogram** as it is shown in [40]. What's more, scattering transform enables to recover lost information in higher-order components by feeding the modulus of coefficients $\left|x \star \psi_{\lambda_1}\right|$ to a bank of next-order wavelets $\psi_{\lambda_2}$ (Eq. (3.33)), and then applying time-averaging operator gives second-order coefficients in a time-shift invariant manner (Eq. (3.34)):

$$|W_2|\left|x \star \psi_{\lambda_1}\right| = \left(\left|x \star \psi_{\lambda_1}\right| \star \phi, \left|\left|x \star \psi_{\lambda_1}\right| \star \psi_{\lambda_2}\right|\right)_{\lambda_2 \in \Lambda_2} \tag{2.33}$$

$$S_2 x(t, \lambda_1, \lambda_2) = \left|\left|x \star \psi_{\lambda_1}\right| \star \psi_{\lambda_2}\right| \star \phi(t) \tag{2.34}$$

**Repeatedly applying modulus for stability to deformations (*diffeomorphism*) and time-averaging for translation invariance** in layer-wise fashion develops the scattering spectrum as depicted in Fig. 2.15.

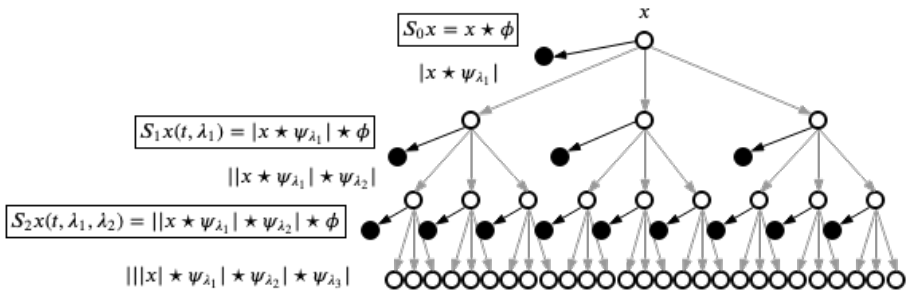



**Figure 2.15** Scattering wavelet spectrum

By using Eq. (3.34) in a cascaded fashion, windowed scattering for any path of scales $p = (\lambda_1, ..., \lambda_m)$ of order $m$ is given in Eq. (3.35). All the output coefficients at each layer will be averaged by the scaling function $\phi$ (similar to pooling operation as in convolution neural networks). Noted, the interval nodes of the scattering tree are

used as coefficients instead of leaves, since averaging is not applied to leaves, contrary to convolution neural networks. Also, the energy of the decomposition goes to zero as the depth $m$ increases.

$$S[p]x(t) = ||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}| ... | \star \psi_{\lambda_m}| \star \phi(t)$$

$$\{S[p]x\}_{p \in P} \tag{2.35}$$

An example of 5-seconds long audio clip shown in Fig. 2.16. Also zeroth-order SWT decomposition coefficients are plotted. Note that, the scaling function $\phi$ (behaving as the averaging operator), the large values of coefficients reflects the energy concentration of audio signal.

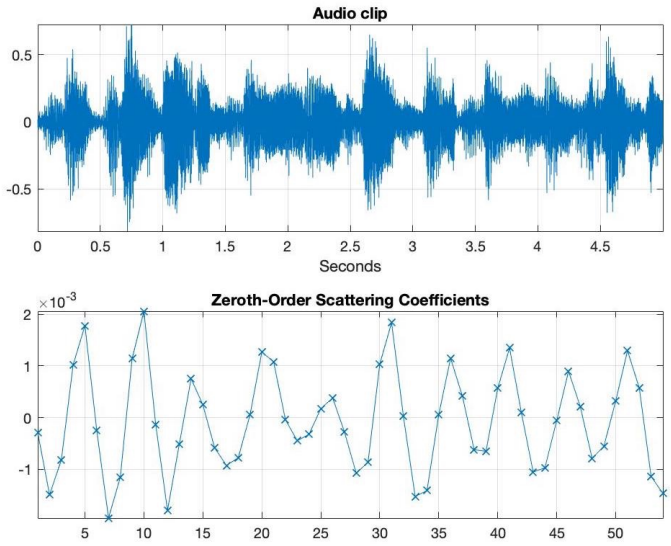
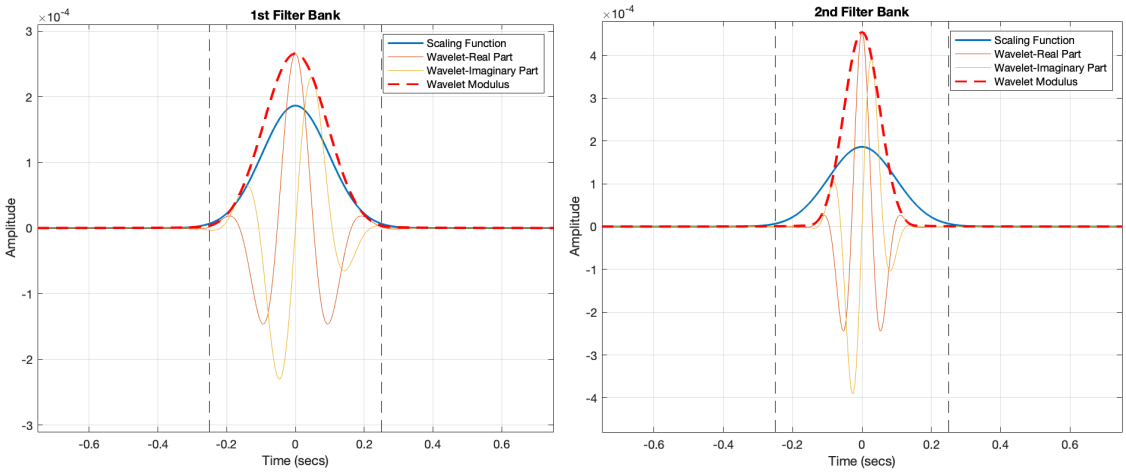



**Figure 2.16** Zeroth-order scattering coefficients

Fig. 2.17 shows wavelet scattering framework with 2 filter banks. Wavelets are in form of modulated Gaussian by sine and cosine wave as a complex waveform. It can be seen that the scaling filter is localized in invariance-scale (0.5 seconds) by design. Also the time support of the coarsest-scale wavelet is bounded by the invariance-scale of the wavelet scattering. Whereas second order filter bank wavelets are more localized in time with shorter support.

Fig. 2.18 shows frequency response of two wavelet filter-banks.

Comparing to mel-frequency cepstral coefficients (MFCC), which is well-known for music and speech related information retrieval tasks, being lossless is beneficial for various aspects (inverse property). Substantially, 1st layer outputs of SWT decomposition $S[\lambda_1]x$ approximates MFCC values and Scale-invariant feature




**(a)** 1st-order filter bank **(b)** 2nd-order filter bank

**Figure 2.17** Scattering Wavelets with 2 filter banks

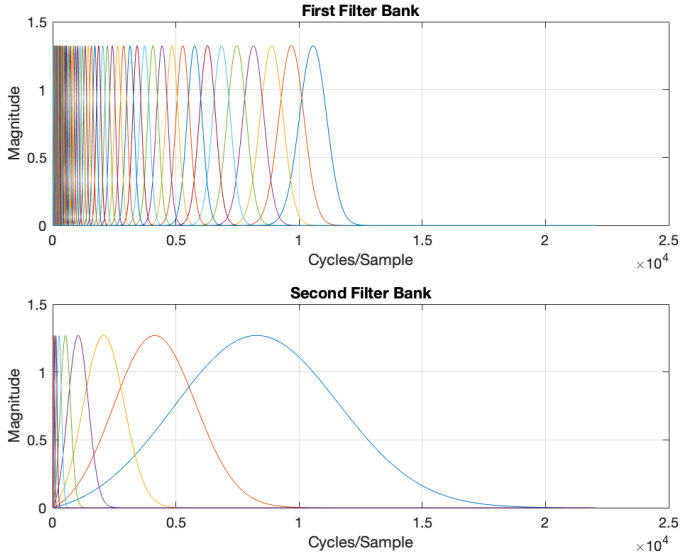



**Figure 2.18** Wavelets frequency response

transform (SIFT, an algorithm in computer vision to detect and describe local features in images) [41] values for 2D transform in particular. In [40] it's also explored the similarities between SWT and convolutional neural networks.

Although 2-layered scattering is successful enough for most of audio signal applications capturing mel-spectrogram and modulation features, noted also coefficients goes to zero as higher layer transforms are applied. In practical level i.e. for feature extraction, parameters for support of averaging low-pass filter $\phi(t)$ (*invariance-scale*) and filter-bank quality per each order ($\Lambda_i$) may be considered.

### 2.3.1 Scattering Transfer Normalization

To increase the invariance, the scattering coefficients should be normalized. First-order scattering normalization is given in Eq. (3.36) where $\epsilon$ is silence detection threshold of case $x = 0$:

$$\widetilde{S}_1 x(t, \lambda_1) = \frac{S_1 x(t, \lambda_1)}{|x| \star \phi(t) + \epsilon} \tag{2.36}$$

Also first-order coefficients may be normalized by the average of $|x|$, creating invariance only to the amplitude change over an infinite window of $\phi$.

For coefficient values of any order $m \geq 2$ , normalization applied considering the delegation (*scattering transfer*) from the previous layer, that is dividing with the coefficients of previous order:

$$\widetilde{S}_m x(t, \lambda_1, \ldots, \lambda_m) = \frac{S_m x(t, \lambda_1, \ldots, \lambda_m)}{S_{m-1} x(t, \lambda_1, \ldots, \lambda_{m-1}) + \epsilon} \tag{2.37}$$

### 2.3.2 Amplitude Modulation

First-layer of scattering coefficients basically contain wavelet coefficients which extract information in different frequency (i.e. scale) bands. All the inner structure of modulations appear in the second-layer of the scattering network.

Audio signal (i.e. voiced signal; musical or speech signal) is simply modeled as vocal chord with vibrations in a sense of serious of impulses $e$, which is filtered by the throat of musical instrument $h$ and with additional possible successions or amplitude modulation $a$. If the audio signal is coarsely modeled as:

$$x(t) = (h \star e)(t) \cdot a(t) \tag{2.38}$$

where:

- $e(t) = \sum_n \delta(t - n/\xi_1)$ is pulse-train excitation with pitch $\xi_1$.
- $h(t)$ is resonance cavity impulse response; where $\widehat{h}(w)$ amounts to the formant.
- $a(t)$ is amplitude modulation.

Audio signals usually involve amplitude modulations by their nature, whose variations may correspond to transient characteristics in time domain of the signal. It is shown that these modulations can be characterized by normalized second-order scattering coefficients for voiced and unvoiced sounds [40]. Then the first two order of normalized scattering coefficients approximates ($\widetilde{S}_1$ and $\widetilde{S}_2$ indicate normalized first-order coefficients and normalized second-order coefficients respectively):

$$\widetilde{S}_1 x(t, \lambda_1) \approx |\widehat{\psi_{\lambda_1}}(n\xi_1)| \frac{|\widehat{h}(\lambda_1)|}{\|h\|_1} \tag{2.39}$$

where $\|h\|_1(t) = \int |h(t)| dt$ and $n$ is such integer satisfies $|n\xi_1 - \lambda_1| < \xi_1/2$.

$$\widetilde{S}_2 x(t, \lambda_1, \lambda_2) \approx \frac{|a \star \psi_{\lambda_2}| \star \phi(t)}{a \star \phi(t)} \tag{2.40}$$

Thus, given $\lambda_1 \approx n\xi$ is close to harmonics, the first-order coefficients are proportional to the spectral envelope $|\widehat{h}(\lambda_1)|$ and extract the formant in a sense of global structure of the signal. While second-order coefficients $\widetilde{S}_2$ are not depended on $h$ and $\xi_1$ but extracts the modulation $a(t)$ (see Fig. 2.19 (c)).

Fig. 2.19 (a) shows scaleograms of $\log |x \star \psi_{\lambda_1}|$ for three voiced and three unvoiced signals. First three voiced signals generated by same pitch of $\xi = 600Hz$ and by same impulse response $h(t)$ with following amplitude modulations $a(t)$:

- a smooth attack
- a sharp attack
- a tremolo of frequency $\eta$: $a(t) = 1 + \epsilon \cos(\eta t)$

Following three unvoiced signals are generated with the same impulse $h(t)$ and same amplitude modulations $a(t)$ as the first three voiced sounds.

Fig. 2.19 (b) shows first-order scattering log $\widetilde{S}_1 x(t, \lambda_1)$ with $T = 128$ms. of the signals in (a). Since they are the wavelet coefficients averaged ($|x \star \psi_{\lambda_1}| \star \phi(t)$), they have similar harmonic structure depending on formant $|\widehat{h}(\lambda_1)|$, but attack and tremolo information is lost by averaging $\phi$ and normalization and they are not differentiable.

Fig. 2.19 (c) shows second-order scattering log $\widetilde{S}_2 x(t, \lambda_1, \lambda_2)$ displayed only for partial $\lambda_1 = 4\xi$ (the vertical band indicated in (a)), as a function of $t$ and $\lambda_2$. The frequency $\eta$ of tremolo and both soft and sharp attacks are obvious as concentrations of the energy in the scaleogram, thus the transient inner structure is apparent. Also it can be seen that the oscillation frequency of the tremolo which is inherent in the third sound creates large amplitude coefficients for $\lambda_2 = \eta$.

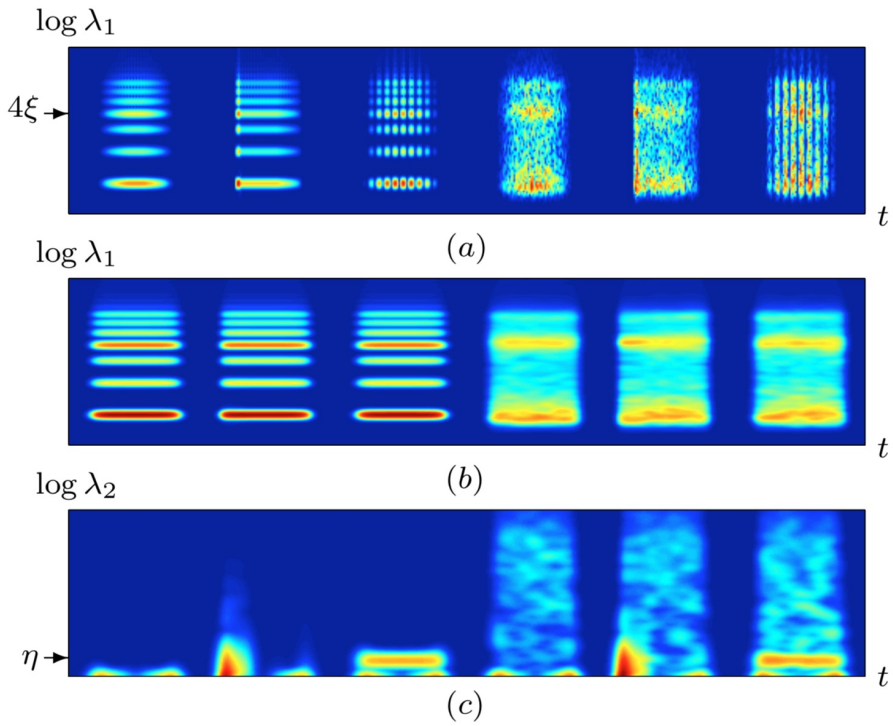



**Figure 2.19** Scattering wavelet coefficients and modulation [40]

### 2.3.3 Computation Complexity

For given signal of size $N$ the scale varies for the set of $2^j$, thus resulting an order of $O(\log N)$ complexity for wavelet transform. For an arbitrary order of scattering, window of size $N$ yields $O(Q^m \log^m N)$ coefficients of order $m$ for a given quality-factor value $Q$.

Wavelet scattering have more indices than spectrograms, but being sparse not all

combinations are of relevant. Scatterings where $\lambda_2 > K\lambda_1$ yield:

$$\left|\left|x \star \psi_{\lambda_1}\right| \star \psi_{\lambda_2}\right| \approx 0 \tag{2.41}$$

Since support of filter frequency responses do not overlap. This limits the number of coefficients to compute. Total computational complexity is given by $O(N\log N)$.

### 2.3.4 Relation with Human Auditory System

The scattering transform has similarities with auditory processing models that integrate cascade of constant-Q filter banks combined by non-linearities [40]. Thus, it is shown the wavelet scattering transform is relatively analogous to human auditory physiological system based on processing model of [42]. Especially the first two orders of wavelet scattering is modeled similarly in auditory perception process as depicted in Fig. 2.20.

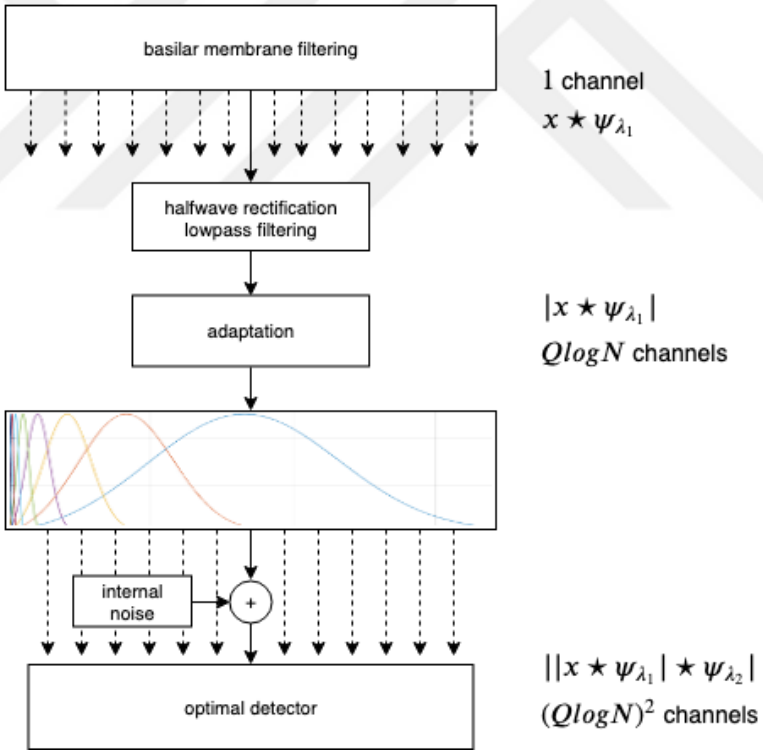



**Figure 2.20** Scattering wavelet spectrum - human auditory system relation

The first filter bank with $Q_1 = 8$ models the cochlear filtration, while the second filter bank corresponds to later processing in auditory pathway incorporating filters with $Q_2 = 1$.

### 2.3.5 Methodology

Given an audio signal either in training stage and DB indexing or for query stage it follows the same pipeline. First the signal is decoded and resampled to target sampling rate $f_s = 16kHz$. Then SWT coefficients are extracted from signal data. For scattering transform we chose the support of averaging filter $\phi$ to be $2^9$ samples, giving an invariance-scale of $\sim 32ms$ $(2^9/f_s)$, and the number of first-order wavelets per octave to be 8 as suggested in [40]. Feature vectors consist of one-second long scattering coefficients resulting $299 \times 31$ dimensional sub-rectangles (i.e. $1/(2^9/f_s) = 31$). Sub-rectangular feature coefficients are depicted in Fig. 2.21 containing both first-order and second-order scattering for a 1-second long audio segment.

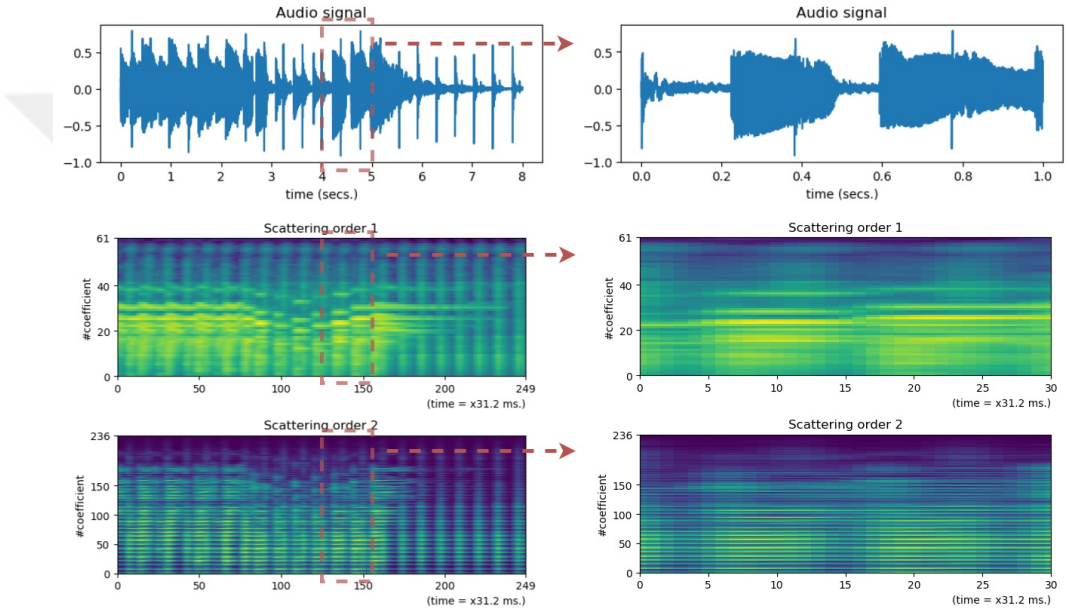



**Figure 2.21** Scattering coefficients sub rectangles including both first and second orders

Regarding to quality-factor values (i.e. octave frequency resolutions) for scattering, for a given audio signal $x$, it has been shown that $Q_1 = 8$ wavelets per octave at the first-order transform provide sparse depiction of a mixture of sounded (i.e. speech, music and ambient) signals [43]. This scheme almost corresponds to of mel-scale frequency resolution. At the second order, selecting $Q_2 = 1$ enables wavelets with more narrow time support, that are better suited to characterize transients and attacks inherent in the signal [40].

The first-order scattering features are mean and variance normalized. As of the second-order scattering features, they are mean and *scatter transfer* normalized using following defined equation:

$$\widetilde{S}_2x(t,\lambda_1,\lambda_2) = \frac{S_2x(t,\lambda_1,\lambda_2)}{S_1x(t,\lambda_1)+\epsilon} \tag{2.42}$$

The overview of the scheme for audio signal scattering wavelet features used in this study is depicted in Fig. 2.22

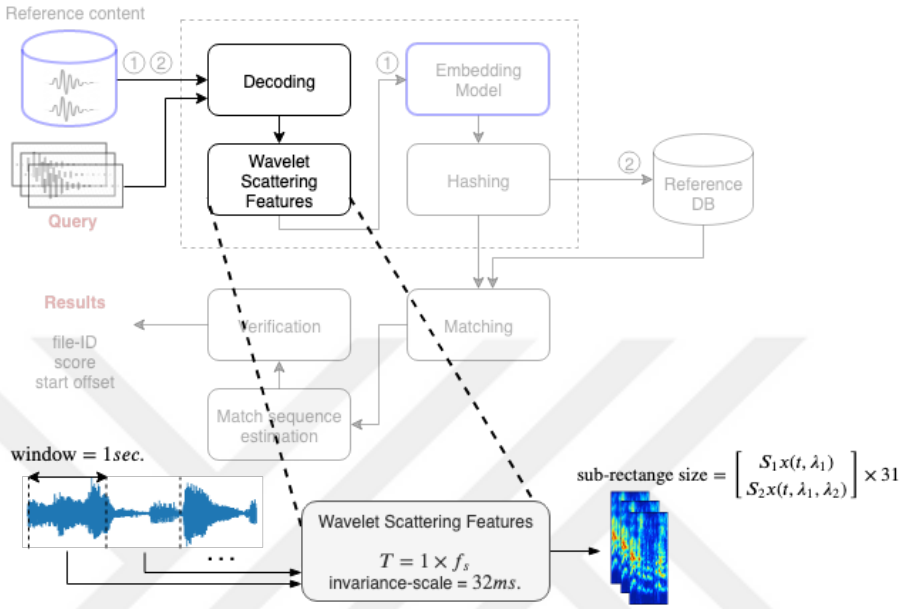



**Figure 2.22** Scattering wavelet transform - methodology overview

## 2.4 Joint Learning and Convolutional Neural Networks

**Definition 2.3.** Metric is a function that calculates *distance* which is defined for every element in a given arbitrary set ($\mathbb{X}$). Metric semantically can be thought as a measure of similarity. Given $x_0, x_1, x_2 \in \mathbb{X}$, a metric function $D(x_0, x_1)$ must satisfy following constraints:

- Non-negativity: $D(x_0, x_1) \geq 0$
- Identity of discernible: $D(x_0, x_1) = 0 \Leftrightarrow x_0 = x_1$
- Symmetry: $D(x_0, x_1) = D(x_0, x_1)$
- Triangle inequality: $D(x, x_2) \leq D(x_0, x_1) + D(x_1, x_2)$

Metrics can be an instance from two broad categories: (1) pre-defined metrics (e.g. Euclidian Distance) and (2) learned/data-driven metrics (e.g. Mahalanobis Distance).

**Definition 2.4.** Euclidian Distance

$$D(x_0, x_1) = (x_0 - x_1)^T (x_0 - x_1) \tag{2.43}$$

**Definition 2.5.** Mahalanobis Distance weighted variant of the Euclidian distance. The distance between two points is modified by a weight matrix $M$. Weights are estimated from a given arbitrary data.

$$D(x_0, x_1) = (x_0 - x_1)^T M (x_0 - x_1) \tag{2.44}$$

The standard deviation of the data may be used as weight matrix [44].

$$D(x_0, x_1) = (x_0 - x_1)^T \Sigma^{-1} (x_0 - x_1) \tag{2.45}$$

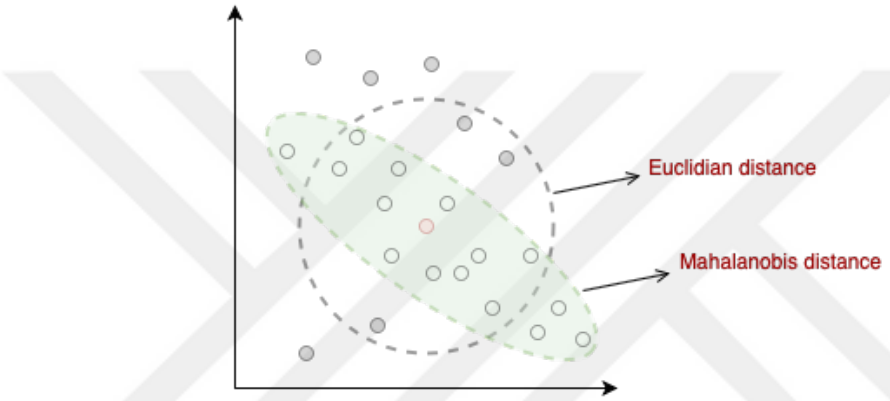



**Figure 2.23** Euclidian and Mahalanobis distances

Data-driven metrics can be learned in supervised or unsupervised fashion. For a supervised scenario a typical approach can be summarized in a 2-step procedure [45] (see Fig. 2.24):

1. Apply a **supervised** domain transform (e.g. Linear Discriminant Analysis).
2. Perform mapping by an unsupervised metric projection.

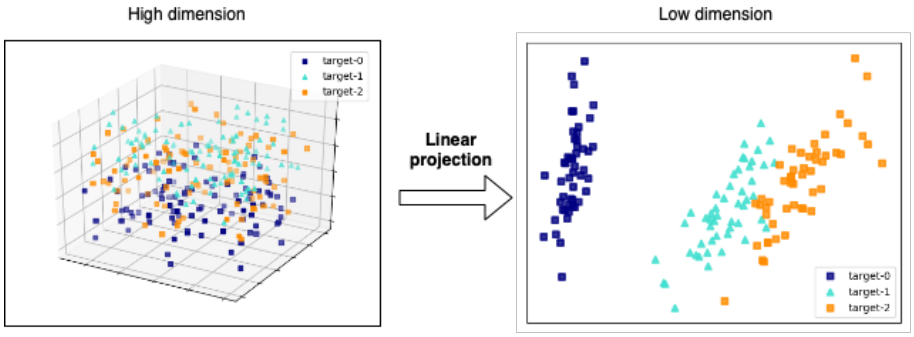



**Figure 2.24** A Metric learning scheme

For instance the Linear Discriminant Analysis (LDA) [46], tries to find a projection where class separation is maximized while keeping the in-class variance small (see Fig. 2.25). It is formulated by the ratio of between-class covariance and within-class covariance to achieve maximizing component axes for class separation:

$$J_{\text{LDA}}(\mathbf{w}) = \max_{\mathbf{w}}(\mathbf{w}^T \Sigma_b \mathbf{w})/(\mathbf{w}^T \Sigma_w \mathbf{w}) \tag{2.46}$$

where $\Sigma_b$ and $\Sigma_w$ are between-class and within-class covariance matrices respectively.

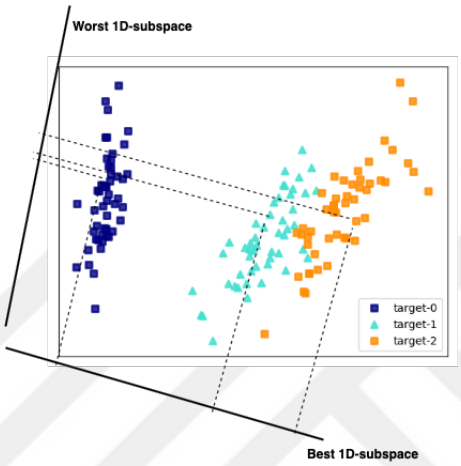



**Figure 2.25** Linear Discriminant Analysis (LDA) learning scheme

Or the pipeline of a somehow opposite approach for similarity matching tasks can be summarized as follows:

1. Extract features in **static** (not learned) way: STFT coefficients, color histograms etc.
2. Learn the similarity: by using a metric over the features

Whereas these briefly mentioned traditional approaches have the same shortcoming, that is **the feature representation of the data and the similarity metric are estimated/learned separately**. On the other hand Convolutional Neural Networks (CNNs) can jointly optimize the representation of the input data conditioned on the selected arbitrary distance metric namely *similarity measure* being used, by what's called *end-to-end learning*. That is CNNs by nature are able to reduce spectral variations while modeling the spectral correlations [47].

A "Deep Neural Network" fundamental model consists of neurons arranged in various layers. Each neural network has an input and output layer, and depending on the

complexity of the problem, many hidden layer augmented to the network. The neurons learn and recognize patterns while the information is passed through layers. Once the model is optimized through training (i.e. trained), precise projections can then be made by the network. CNNs, proposed by Yann LeCun *et al.* [48], is a particular sort of neural network that operates on 2D representations exceptionally well, equipped with the usage of the concept called *local connectivity*. By means that each neuron is only linked to a local area of the quantity of the input. This minimizes the amount of parameters by enabling various sections of the network to be specialized in high-level characteristics such as patterns and textures.

Initially, such a neural network should understand all the raw components in a 2D representation, such as edges and other somehow low-level characteristics. These are then identified and paved the way for complex functions later on, by obtaining the low-level characteristics first, followed by higher-level ones. Filters (or kernels) provide a means of extracting the information required in parallelized fashion, rather than just transmitting the data, which would prevent the network from understanding the intrinsic structures. At early layers of the network, specific filters would extract the low-level characteristics based on their weights. The filters can be understood as the layers of logical units in the compound network which are connected to by their weights. Extractions are executed by convolutions. Filters (or kernels) are convolved with the input data to reach at the *intermediate images*, corresponding to **the partial knowledge of the image by the network**. For a deep network, these retrieved by-products, in turn convolved with more filters of followings layers, to model the inherent structure of the prior intermediate representations. The overview CNN architecture with its operators is depicted in Fig. 2.26.

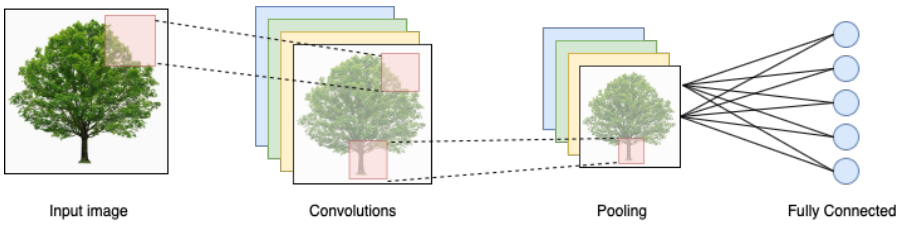



**Figure 2.26** CNN operators overview

In Fig. 2.26 the convolution layer uses the number of filters performing convolution operations as it scans the input $I$, and scanning is based on hyperparameters, namely filter (i.e. kernel) size $F$ and stride $S$. The resulting output is called by term *feature map* or *activation map* (see Fig. 2.27). After a convolution layer, the pooling layer execute a downsampling operation by means of *spatial invariance* (regarding to local connectivity). Maximum or average pooling are mostly used pooling operations (see Fig. 2.28). To optimize arbitrary objectives, CNN architectures mostly use

fully-connected layers as hindmost layers. The fully-connected layer works on a flattened intermediate layer that connects each input to all neurons (see Fig. 2.29).

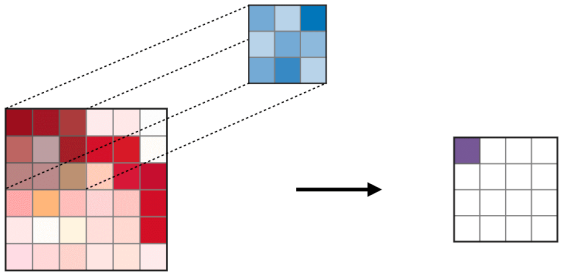
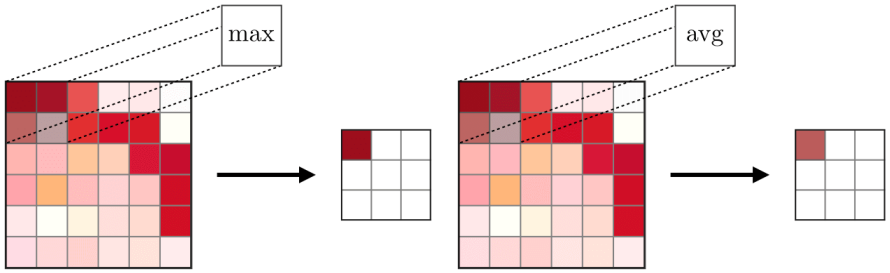


**Figure 2.27** CNN convolution operator [49]




**(a)** Max pooling **(b)** Avg. pooling

**Figure 2.28** CNN pooling operator [49]

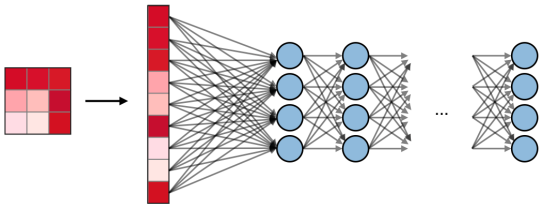


**Figure 2.29** CNN fully-connected layers [49]

Kernel operations involve hyper-parameters of filter dimensions and stride value. A filter of size $(F_0 \times F_1)$ applied to a $C$-channel input is a $(F_0 \times F_1 \times C)$ volume that performs convolutions on an input of size $(I_0 \times I_1 \times C)$ and the product is an output feature map of size $(O_0 \times O_1 \times 1)$. For a convolution or a pooling operation, the stride $S$ denotes the number of pixels by which the filter window slides following each operation [49].

### 2.4.1 Methodology

Methods exists in the literature which encourage integration of CNNs for hash learning models. These models utilize joint learning property CNNs. Training input scheme can be varied based on optimization requirements and more importantly attributes of the input data available. Given properties may propose discrete class labels, similarity

decisions etc. of the input data. Acquainted with input attributes, these models could instrument pair-wise input [50] or point-wise input [51] schemes using deep CNNs.

In this study, we leverage CNNs for hash learning in an endeavor to build our *similarity-preversing* model. Although SWT coefficients may distinguish the characteristics of audio signals, such transform can not be only used to understand the dynamical behavior of musical data perfectly. We will introduce a simple deep CNN module in our embedding hash model to capture variations of SWT coefficients. But since the second-order SWT coefficients are sparse, we adapt the CNN module only for first-order coefficients with small kernel windows. The overview of the CNN model used in this study is depicted in Fig. 2.30.

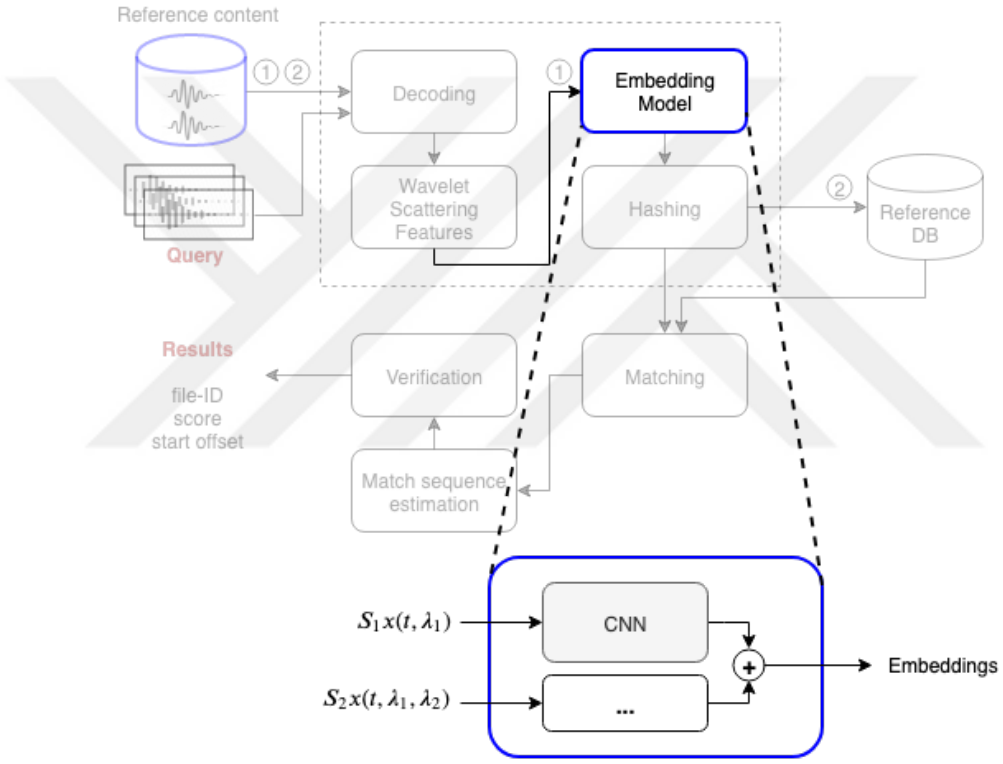



**Figure 2.30** CNN - methodology overview

CNN layers also use Parametric ReLU activation function and batch normalization.

## 2.5 Embedding Network Model

*Similarity-preserving* embedding hash modeling is a widely-used method for nearest neighbor search in a large-scale context. In most common existing embedding hash methods, the input is first encoded as a representative vector by means of a possible domain change, followed by another separate projection or quantization step that generates final hashes. However intermediate feature vectors may not be suitable with the coding process resulting in non-optimal hash codes. A carefully designed

end-to-end deep neural network models can optimize both feature generation hashing process [52].

Learning-based hashing methods aiming to encode input data while preserving similarity is an emerging open search topic since requirements of various real-world application include nearest neighbor search on large-scale. Compact bit-wise representations are beneficial regarding efficiency of both storage and search/retrieval speed and complexity especially for large-scale applications.

Data-driven proposed embedding models may work in unsupervised fashion on data like iterative quantization [53] or kernelised locality-sensitive hashing [54]. Or, if available, supervised methods are optimized with labeled information (e.g. categories, similarity/dissimilarity of samples), combined with deep architectures tailored for learning-based hashing.

Most existing methods of hashing model based on two-stage process where input data first represented by traditional hand-engineer features of vector descriptors followed by a separated projection and quantization steps to encode these descriptors as final hashes. However the mentality of separate process of data representation can fail to generate optimal hashes since such descriptors may not be optimally suitable with the hashing process. Such as, for *similarity-preserving* needs, intermediate representation may small or large values for a defined metric for similar and dissimilar pairs accordingly and thus, such intermediate bottleneck layer as representation may not be able to generalize other data. Ideally, the two stage can be combined and learnt joint by overall hash learning process.

An supervised embeddings hash model pipeline can designed with following components: (1) a sub-network to produce robust intermediate representation, (2) hash-generating block to convert intermediate representation to final hashes and (3) suitable loss function selection as depicted in Fig. 2.31. Supervised learning-based hashing models are used for compact representation of domain specific data for various tasks including classification, recognition and verification.

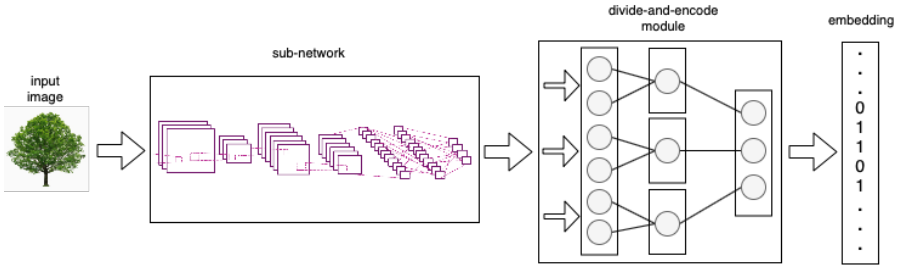



**Figure 2.31** Embedding hash model

The trained model provides a mapping from input space to an embedding space. Embeddings learned in such way can be used as features vectors. After a non-linear projection of embedding space is modeled, a distance metric (also the training loss of the model) may be defined. For instance in if Euclidian embeddings are learnt then the similarity is defined by $\mathbf{L}^2$ distance. Provided with a distance metric various tasks can be defined as:

- Verification: thresholding of distance between two embeddings.
- Recognition: can be approached as a simple k-nearest-neighbor (k-NN) clustering task
- Clustering: can be achieved using k-means or other clustering techniques on embedding space.

### 2.5.1 Divide and Encode layer

Having the intermediate features obtained from the sub-network a hash-generating block to convert intermediate representation is used weather is a simple fully-connected layer or a more complex block. Traditional approach for hash generation can be applied by a simple fully-connected layer. Whereas *Divide-and-encode* module enables splitting intermediate representation into multiple branches then concatenate each into the designed order of hash components as shown in Fig. 2.32.

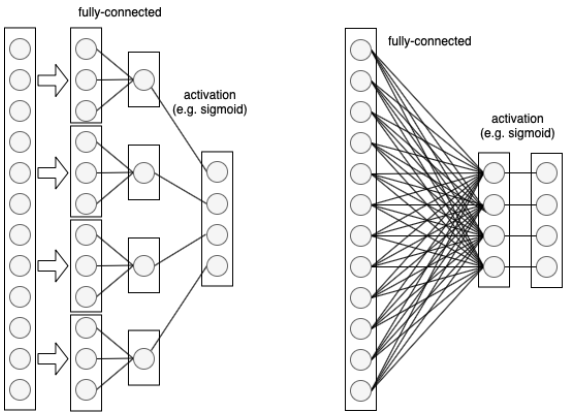



**Figure 2.32** Divide-and-encode module

The main idea of using divide-and-encode block instead of a fully-connected layer is to reduce the redundancy among the intermediate representation branches which go through to generate final hash bits. For $q$-bit hash codes, first divide-and-encode module splits intermediate representation to $q$-slices, then each slice is fed to internal

fully-connected layers to generate final values using a arbitrary activation function (e.g. sigmoid for a unit interval output $\in [0,1]$). The activation function may be preferred according to designed model, generating real values or unit interval range output (which later may quantized to binary).

Since each hash component is generated from a separated slice of features (i.e. containing information from split path of intermediate features), the generated hash bits would be less redundant to each other which can be advantageous considering the entropy of the input data.

### 2.5.2 Quantization

For binary hash embedding model, the activation outputs should be quantized to encourage binary values. For any input $I$, a binary hash embedding model can defined as $q$-bit hash generator $F$ as: $F : I \rightarrow \{0,1\}^q$. If sigmoid activation is used in final layer, using piece-wise threshold function would deduce quantized binary outputs by forcing the model to binary values to at training stage [52]. Given the fully-connected output $\text{fc}_i(x^{(i)}) = \mathbf{W}_i x^{(i)}$ for each slice-$i$ and sigmoid activation in Eq.(3.47), Eq.(3.48) shows piece-wise threshold function for sigmoid activation as its input $z$, where $\epsilon$ is a model hyper-parameter. The piece-wise threshold function is plotted in Fig. 2.33.

$$\text{sigmoid}(y) = \frac{1}{1+e^{-\beta y}} \tag{2.47}$$

$$f(z) = \begin{cases} 0; & z < 0.5-\epsilon \\ z; & 0.5-\epsilon \leq z \leq 0.5+\epsilon \\ 1; & s > 0.5+\epsilon \end{cases} \tag{2.48}$$

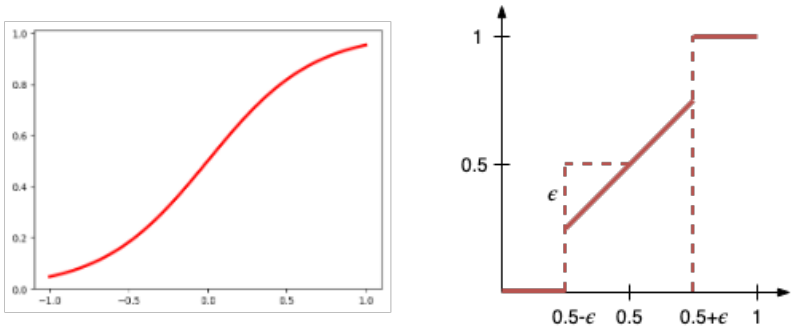



**Figure 2.33** Piece-wise threshold function plot

Using a piece-wide threshold layer will force the model to output binary values, where

the values that are not in range of $[0.5-\epsilon, 0.5+\epsilon]$ will be truncated to be 0 or 1. Noted that, in the training stage, the proposed model with piece-wise threshold function will only generate real-valued hash codes that most of them forced to binary for input data. To generate binary hash bits, later the output of the optimized model is quantized by Eq. (3.49):

$$b = \text{sign}(F(I) - 0.5)) \tag{2.49}$$

where $\text{sign}(v_i) = 1$ if $v_i > 0$, or otherwise $\text{sign}(v_i) = 0$.

On the other hand, quantization for real-valued hashes only can be implemented for bit reduction that is for reducing code size.

### 2.5.3 Methodology

Provided with SWT the extracted coefficients of are fed to embedding network for the next stage of the pipeline as depicted in Fig. 2.34. The model is trained with some portion of the reference content, but the training data is not indexed in reference database for querying (action ① in Fig. 2.34). The optimized model after training will be used for both database indexing (action ② in Fig. 2.34) and querying. However the embedding model is not final and can be improved with arbitrary data and/or can be adapted using harmonic embeddings [55] in a transfer learning fashion. Harmonic embeddings is a technique of adapting newly designed models to be able to improve verification accuracy while maintaining compatibility to less accurate embeddings of initial model.

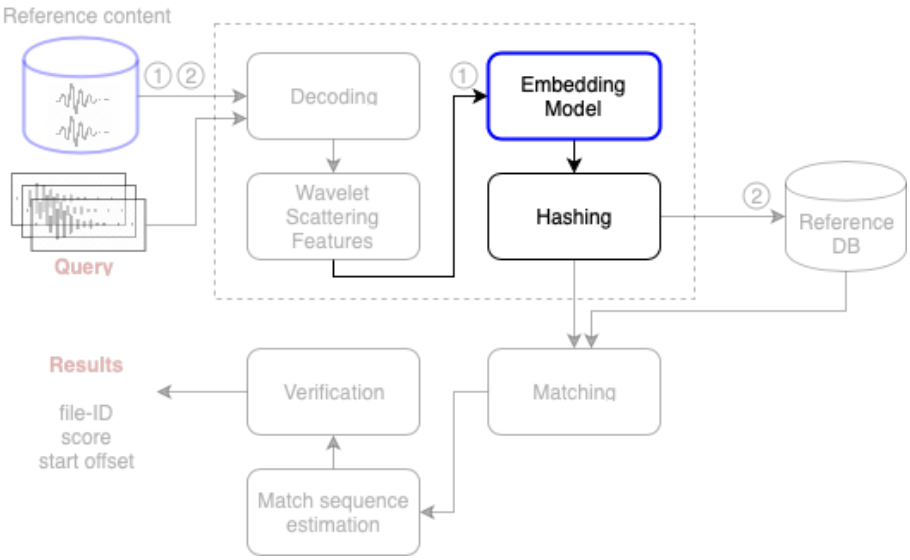



**Figure 2.34** Embedding hash model - methodology overview

We build our fingerprint hashing model by the combination of stack of convolutional layers (CNNs), fully-connected layers and divide-and-encode block [52] on top. CNNs can model spectral correlations of our first-order scattering coefficients $S_1(t, \lambda_1)$. While second-order scattering $S_2x(t, \lambda_1, \lambda_2)$ which is the decomposition of modulation features in each sub-band of the first-order filter-bank ($|x \star \psi_{\lambda_1}|$) preserves the local information for given sub-band $\lambda_1$, being sparse with few non-zero coefficients, are fed to fully-connected layer. Designed model is shown in Fig. 2.35.

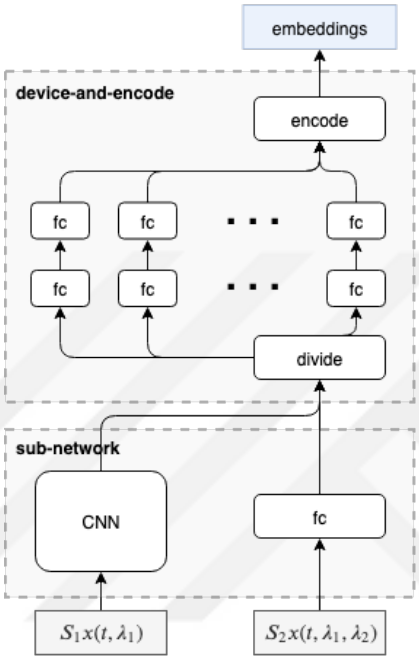



**Figure 2.35** Overview of embedding hash model with scattering wavelet coefficients

Two-level of divide-and-encode block splits the intermediate representation before combining into the final embeddings. The idea of choosing divide-and-encode block is to reduce the redundancy among intermediate hashes as suggested in [52]. Final embeddings may be retrieved as real values or quantization can be opt in using a piece-wise threshold function at the learning stage. Except final divide-and-encode layer all layers use Parametric ReLU activation function and batch normalization.

## 2.6 Siamese and Triplet Networks

For a *similarity-preserving* network model, input can be any representation and the model is responsible to provide an output either binary value (i.e. identical/imposter or similar/dissimilar) or a real value indicating how similar a pair of inputs are. This similarity-preserving network model is called *Siamese network*. Siamese network is a type neural network that use shared weight while working in tandem on pair of inputs to compute comparable output vectors [56]. Output vectors then can be

used similarity score armed with a distance metric. Area of utilization of siamese networks include generating invariant and robust descriptors [57], face-recognition [58], objects re-identification tasks [59] and imposter detections.

A variant of siamese network is *triplet network* combining both similarity-preserving and marginalization of dissimilarity that is discriminating dissimilar pairs. Triplet networks are trained with triplet of exemplars including an *anchor sample* (baseline sample), a positive sample and a negative sample, making two pairs for each triplet: anchor against positive and anchor against negative.

Not all the models of Siamese networks share the same architecture, the design of architecture should be evaluated empirically by what performs well on the specific task. A typical architecture of Siamese network with **shared model weights** is depicted in Fig. 2.36(a). Another architecture with unshared block of layers is shown in Fig. 2.36(b) [60].

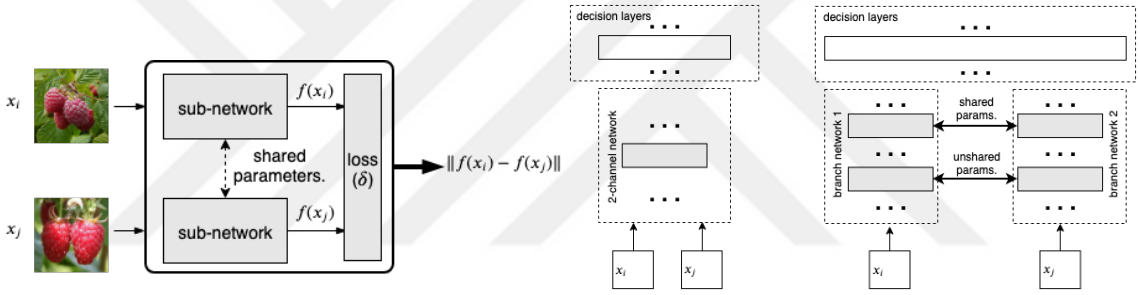



**Figure 2.36** Siamese network models with sub-network blocks ((a) A typical Siamese network model with shared weights, (b) A Siamese network model with unshared block of layers)

Siamese networks with shared weights can be generalized by following equation:

$$J_{\text{Siamese}}(\mathbf{W}) = \begin{cases} \min\limits_{\mathbf{W}}\{\delta[f(x_i), f(x_j)]\}; & (i,j) \rightarrow \text{similar} \\ \\ \max\limits_{\mathbf{W}}\{\delta[f(x_i), f(x_j)]\}; & (i,j) \rightarrow \text{dissimilar} \end{cases} \tag{2.50}$$

where $i, j$ are indices of of input samples, $f(\cdot)$ is embedding model implemented by siamese network and $\delta$ is network layer joining outputs from the siamese network.

### 2.6.1 Loss Functions

Siamese networks should be trained with both positive pairs and negative pairs (i.e. similar and non-similar pairs respectively) as depicted in Fig. 2.37, since learning with using only similar pairs would force the model to embed every input to a same point in embedding space. That will result the model to output a constant and categorize every input pair as similar [61]. As a result the loss function should include both positive loss ($L^+$) and negative loss ($L^+$) optimizations.

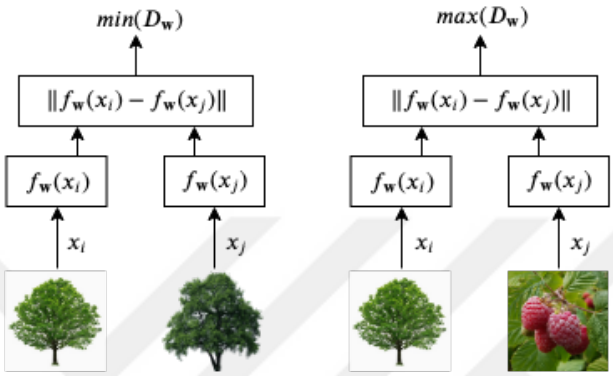



**Figure 2.37** Dissimilar pair learning

Possible selection of functions could be *Euclidian loss* and *Hinge loss* functions for positive loss and negative loss respectively.

**Definition 2.6.** Hinge loss is a loss function used for *maximum-margin* classification most notably in support vector machines (SVMs). For an intended target value $t \in -1, +1$ and prediction $y$, Hinge loss is defined as:

$$L(y) = \max(0, 1 - t.y) \tag{2.51}$$

Euclidian loss and an extended version of Hinge loss functions for input pair learning is given in Eq. (3.52) (see Fig. 2.38):

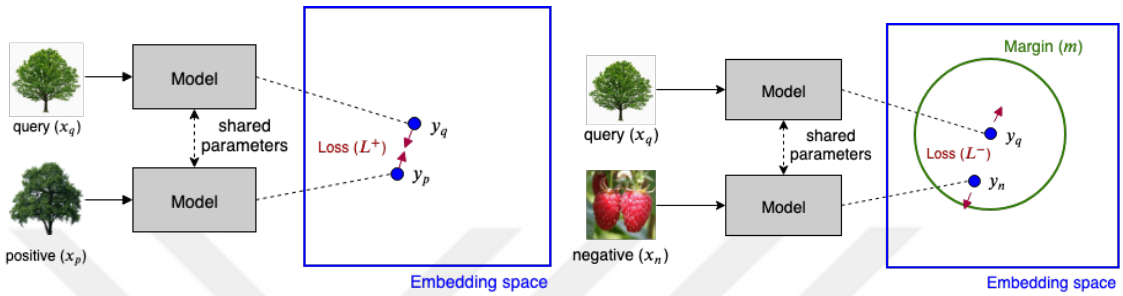
$$\begin{aligned} L^+(x_q, x_p) &= \left\| x_q - x_p \right\|^2 \\ L^-(x_q, x_n) &= \max(0, m^2 - \left\| x_q - x_n \right\|^2) \end{aligned} \tag{2.52}$$

where $m$ is margin, $(x_q, x_p, x_n)$ are query, positive and negative samples accordingly.

**Definition 2.7.** Contractive loss function combines both Euclidian loss and Hinge loss and is able to learn a margin of separation for negative pairs. Contractive loss is suitable for both Siamese and triplet networks [62]. For an embedding mapping $f$, contractive loss function is given in Eq. (3.53):

$$
\begin{aligned}
L^+ &= \|f(x_0) - f(x_0)\|^2 \\
L^- &= \max(0, m - \|f(x_0) - f(x_1)\|)^2 \\
L_{contractive}(x_0, x_1, y) &= \frac{1}{2} t L^+ + \frac{1}{2}(1-t)L^-
\end{aligned}
\tag{2.53}
$$

where $t$ is binary target value $t \in 0, 1$, $m$ is distance margin and $(x_0, x_1)$ are sample pairs that can be positive or negative.




**(a)** positive pair learning **(b)** negative pair learning

**Figure 2.38** Positive and negative pair losses

**Definition 2.8.** Triplet ranking hinge loss tries to enforce a margin between each dissimilarities, while closing the distance between similarities. This allows the similar embeddings to place on a manifold, while still ensuring the distance and thus discriminability of others. For an embedding mapping $f$, *triplet loss* function is given in Eq. (3.54):

$$
\begin{aligned}
L^+ &= \left\|f(x_a) - f(x_p)\right\|^2 \\
L^- &= \|f(x_a) - f(x_n)\|^2 \\
L_{triplet}(x_a, x_p, x_n) &= \max(0, m + L^+ - L^-)
\end{aligned}
\tag{2.54}
$$

where $m$ is distance margin and $(x_a, x_p, x_n)$ are anchor, positive and negative samples accordingly.

In triplet loss function in Eq. (3.54), the negative value $L^-$ will force learning in the network, while the positive value $L^+$ will act as a regularizer. Similarly for the learning optimization by the contrastive loss in Eq. (3.53), there must a regularization applied to the learned weights like a weight decay, or some similar operation like a normalization.

The embedding space represented by $f(x) \in \mathbb{R}^d$ is a mapping from $x$ to $d$-dimensional Euclidean space. Triplet ranking loss function will ensure that an anchor sample is closer to its similar items than the dissimilar ones as visualized in Fig. 2.39.

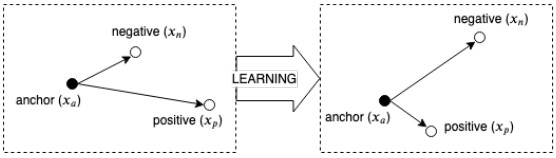



**Figure 2.39** Triplet loss learning

### 2.6.2 Data Mining

While training the network not all triplets will have significant impact on optimization, only the possible triplets whose violate triplet loss constraint (i.e. having a non-zero loss) will optimize the the model. In Eq.(3.55) constraint for triplet loss is given. Generating all the possible triplets on a large training data would result many triplets satisfying the triplet constraint. These triplets would not contribute to the training and result in slower convergence, while they are being fed through the network. Thus it reveals the necessity of selecting *hard triplets*, whose have loss values bigger than zero and can therefore contribute to improve the model [55].

$$\left\|f(x_a) - f(x_p)\right\|^2 + m < \|f(x_a) - f(x_n)\|^2 \tag{2.55}$$

Therefore, selecting the contributing triplets in the learning stage is crucial. Selection decisions should consider triplets that violates the triplet constraint in order to ensure optimal model and fast convergence. Sampling strategies for indices of samples $(i)$ that provides possible violation of the triplet loss constraint is given below:

- **hard positive**: $\underset{x_p}{\operatorname{argmax}} \left\|f(x_a^{(i)}) - f(x_p^{(i)})\right\|_2^2$
- **hard negative**: $\underset{x_n}{\operatorname{argmin}} \left\|f(x_a^{(i)}) - f(x_n^{(i)})\right\|_2^2$
- **semi-hard negative**: $\|f(x_a^{(i)}) - f(x_p^{(i)})\|_2^2 < \|f(x_a^{(i)}) - f(x_n^{(i)})\|_2^2$

For a given anchor sample, semi-hard negative strategy selects negatives that are further away from the anchor than the positive sample but still hard because the squared distance is within the margin value. For semi-hard negative strategy, number of exemplar can be sampled in sorted way by decreasing distance or picked up by randomly.

Also as it is argued that it is infeasible to compute argmax and argmin on the whole training set. Additionally, it might lead to poor training for outlier or mislabelled data [55]. To avoid these to happen, two possible options are:

- Offline data mining: selecting argmax and argmin on the subset of data by using the most recent network checkpoint.
- Online data mining: selecting hard positive/negative exemplar from with a mini-batch.

Online data mining method instruments mini-batch usage, thus the mini-batch size should be adjusted accordingly. Since to have a meaningful representation of any anchor sample, the mini-batch should also be ensured to include enough number of exemplars for anchor positives.

### 2.6.3 Methodology

The embedding hash model in our systems is the module that is responsible for generating audio fingerprints, and the module is trained in triplet network fashion. Training split of reference content is used to train the embedding model (action ① in Fig. 2.40). The overview of the scheme of triplet learning used in this study is depicted in Fig. 2.40.

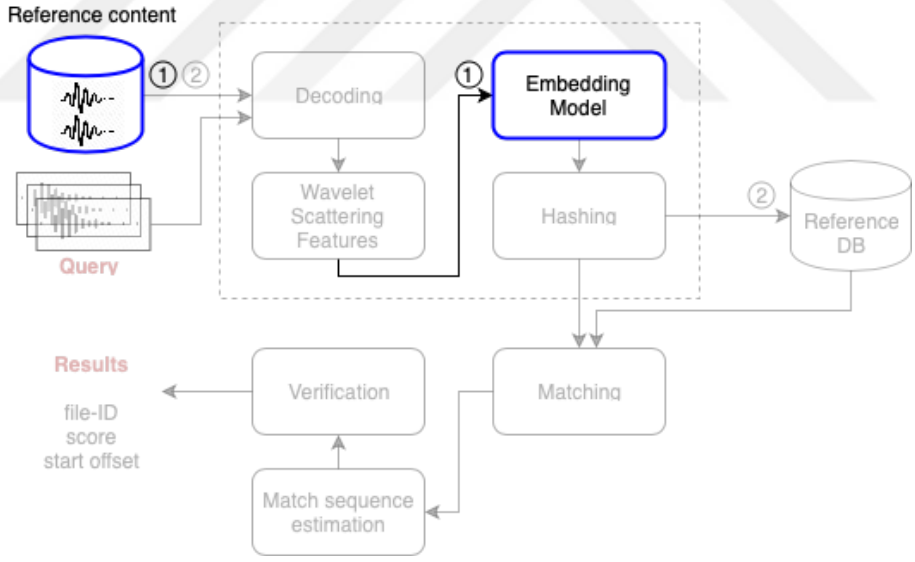



**Figure 2.40** Siamese networks - methodology overview

Triplet loss is more suitable for the systems of fingerprint identification, motivated by the triplet loss enforces the discrimination of arbitrary musical signal data more than representativity. That is, we require discriminative power over being able to be descriptive. Triplet learning corresponds forcing embeddings $f(x)$ from input audio signal $x$ into a feature space $\mathbb{R}^d$ such that square distance of semantically similar signals are small, whereas the squared distance between a pair different signals of are large. Thus a semantic similarity of signals should be described, and we define it by

a *neighborhood distance*. And we select the anchor-positives within the neighborhood distance of an anchor audio segment shifted by a step size equal to invariance-scale of SWT as depicted in Fig. 2.41.

Considering the online mining of samples, all possible anchor-positive pairs within a neighborhood distance of anchor are included in each mini-batch instead of just selecting hardest positives. Additionally mini-batches contain randomly sampled negative samples. A sampling strategy is applied to the random negative sample set as a selection filtering. As stated in [55], selecting the hardest negatives may have consequences such as leading to stuck optimizations of the model at local minima in the training or may result as a collapsed model (i.e. for embedding function $f$; $f(x) = 0$), thus the semi-hard strategy is preferred in order to avoid these observed issues.

Also, the overall mini-batch size should be ensured allowing enough all-possible positives and subset of negative exemplars. And on the other hand we would like to use small mini-batches as these tend to improve convergence speed.

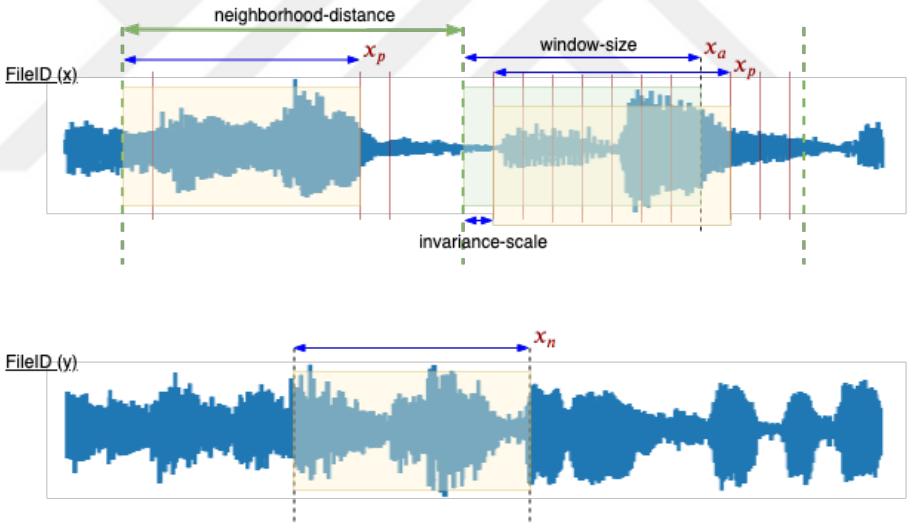



**Figure 2.41** Similarity and online mining scheme

# 3
# Evaluation and Musical Audio Identification Application

In previous Chapter Concepts detailed technical concepts are given with their methodological approaches for a concrete implementation of fingerprinting scheme and the musical audio identification system investigated in this study. In this Chapter, provided with the design decisions, we will overlay the evaluation of the overall system and presented experimental results. Later analyzing the result of the experiments conducted we propose the audio musical audio identification application.

As indicated Chapter Concepts, we trained our model as siamese network of audio snippet similarities using triplet loss function. The constraints for audio snippets to be considered as same were that they should be from same audio file and their starting position should differ only couple of hundred milliseconds of *neighborhood distance*. Siamese networks with $L^2$ loss functions are useful to learn mappings from input data to a compact Euclidean space where distances correspond to a measure of similarity [55].

We used GTZAN dataset [63] [64] for our experiments. In training stage of our embedding hash model, various additional noise applied to audio signals to preserve similarity for degradations. In evaluation step, selected environmental and artificial noises are applied with adjusted SNR values as explain in next section. For implementation, to build our embedding hash model PyTorch framework [65], and for scattering transform of audio signals Kymatio framework [66] are used.

GTZAN dataset includes 1000 audio files from 10 musical genres each 30-seconds long and with sampling rate $f_s = 22050Hz$. We prepared 10-seconds longs audio snippets downsampled to $f_s = 16kHZ$, having total number of 3K snippets, then randomly split at ratio of $(0.8, 0.2)$ for training and test accordingly. For scattering transform, after our experiments we chose the support of averaging filter $\phi$ to be $2^9$ samples, giving an invariance-scale of $\sim 32ms$ $(2^9/f_s)$, and the number of first-order wavelets per octave

to be 8. Feature vectors consist of one-second long scattering coefficients resulting $299 \times 31$ dimensional sub-rectangles (i.e. $1/(2^9/f_s) = 31$).

We define a *neighbor distance* that equals to $370ms$, to label audio segments considered as same or not by their starting position while training. While we are training our model, we use strides equal to invariance-scale duration for each audio snippets, whereas database indexing stage use features per every one-second without overlapping. General overview of audio signal fingerprint scheme is depicted in Fig. 3.1.

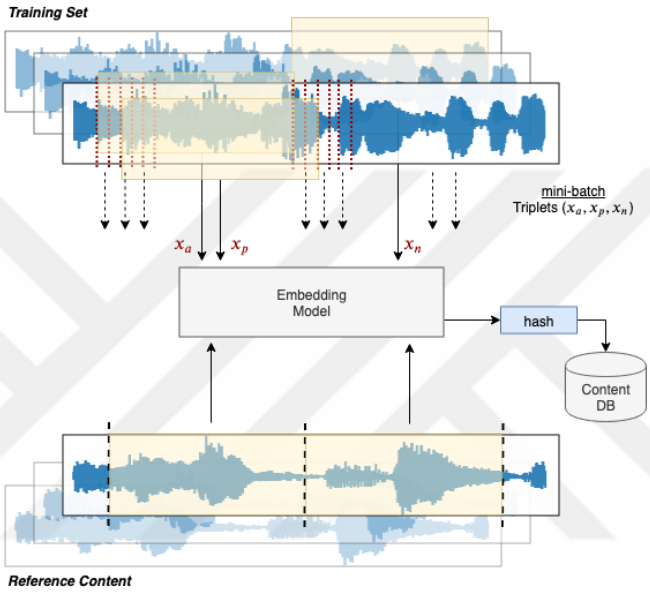



**Figure 3.1** Audio signal fingerprinting scheme. Embedding model training (top), Database indexing (bottom)

To prevent biased learning, first-order $S_1x(t, \lambda_1)$ scattering coefficients were mean and variance normalized, whereas for second-order $S_2x(t, \lambda_1, \lambda_2)$ coefficient values, first *scattering transfer* normalization [40], later mean-normalization were applied. While training, for each mini-batch online triplet selection strategies are executed to prevent poor training of the network [55] [67]. Each mini-batch contains all possible positive triplets in the neighborhood distance while negatives are selected using semi-hard exemplars, that are further away from the anchor than positive exemplar, but within the radius of margin. We prepared a noisy variant (superimposed with various environmental noise) of the dataset with **same alignment**, and in our scenario, anchor and positive samples regard to clean and noisy audio segments within the defined neighborhood accordingly.

**Table 3.1** Embedding net. architecture for SWT coefficients

| **CNN module** | | | |
|---|---|---|---|
| Input shape: | *first-order coefficients* $(62 \times 31)$ | | |
| **Layer (type)** | Output shape (#Channel x Height x Width) | Kernel | Stride |
| Conv2d | [32, 60, 29] | 3x3 | 1 |
| PReLU | [32, 60, 29] | - | - |
| MaxPool2d | [32, 30, 14] | 2x2 | 2 |
| Conv2d | [32, 28, 12] | 3x3 | 1 |
| BatchNorm2d | [32, 28, 12] | - | - |
| PReLU | [32, 28, 12] | - | - |
| MaxPool2d | [32, 14, 6] | 2x2 | 2 |

| **Fully-connected module** | |
|---|---|
| Input shape: | *second-order coefficients* $(237 \times 31) = 7347$ |
| **Layer (type)** | Output shape |
| Linear | [1024] |
| BatchNorm1d | [1024] |
| PReLU | [1024] |

## 3.1 Coefficients and Hash Dimensions

Extracted $(299 \times 31)$ dimensional sub-rectangles by scattering transform over 1-second long audio segments, the coefficients then fed to embedding model to generate audio fingerprints. The embedding model contains two sub-modules for first and second order of scattering transform which are trained on their respective separate path of input output pipeline as well as in divide-and-encode layer to remove the redundancy among the intermediate representation branches of scattering coefficients and encoded them separately. For fully-connected module constant 1024 units is used. The CNN and fully-connected module parameters of embedding model are given in Table 3.1.

Fig. 3.2 and Fig. 3.3 shows CNN kernels of trained model and top layer heatmaps for random #3 audio segments against various models dimensions respectively. From the heatmap figure kernel specialization can be seen coarsely.

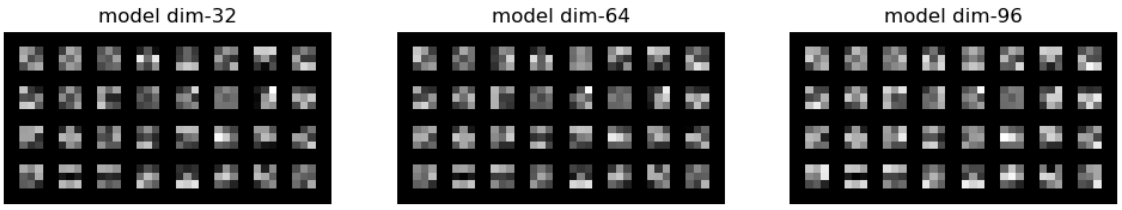
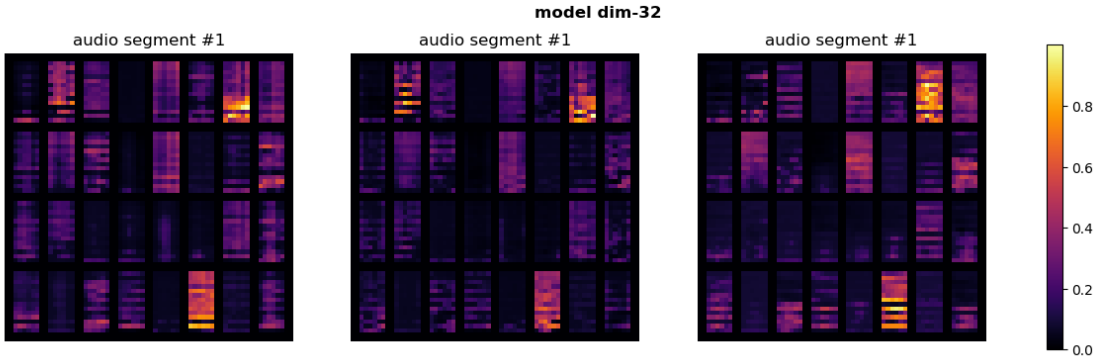
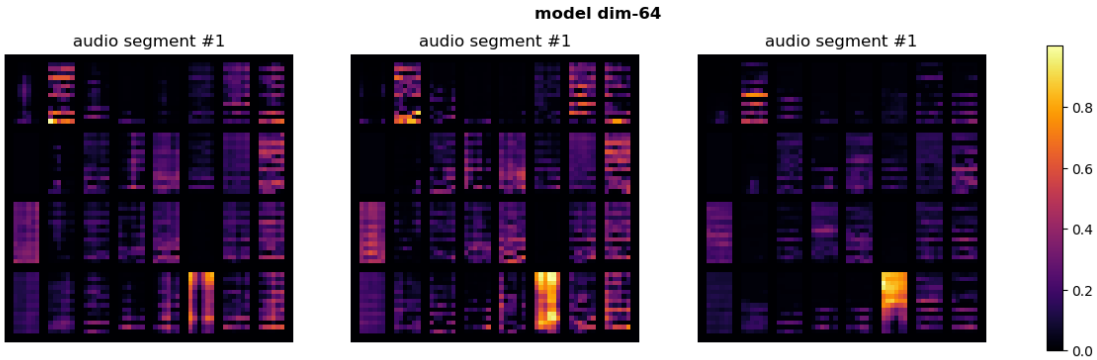



**Figure 3.2** CNN kernels for various model dimensions

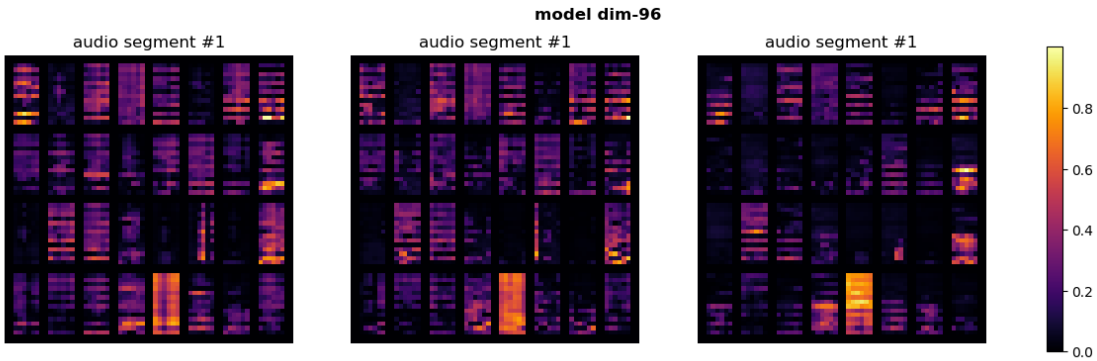
Table 3.2 shows hindmost layer of embedding model device-and-encode block




**(a)** 32-dim. model




**(b)** 64-dim. model




**(c)** 96-dim. model

**Figure 3.3** CNN heatmaps for audio segments' first-order SWT coefficients

**Table 3.2** Hash dimensions agains two-layer SWT coefficients

| | | **Divide-and-encode block** | |
|---|---|---|---|
| | | *first-order input size* $(32 \times 14 \times 6) = 2688$ | |
| | | *second-order input size* 1024 | |
| **Output dim.** | Output dim./orders | 1st-order encode size | 2nd-order encode size |
| 8 | [7, 1] | 384 | 1024 |
| 16 | [14, 2] | 192 | 512 |
| 32 | [57, 7] | 96 | 256 |
| 96 | [85, 11] | 32 | 93 |
| 128 | [114, 14] | 24 | 7 |
| 256 | [228, 28] | 12 | 36 |

configurations for various output dimensions. The output dimensions are adjusted with about $(9,1)$ ratio between first and second order of scattering coefficients. In the table, encoded sizes are given for inputs (i.e. 2688 and 1024 for first order and second-order scattering intermediate paths respectively) according to desired output dimension.

## 3.2 Similarity and Online Mining

Similarity for audio segments were defined by a neighborhood distance of their starting offset. Audio segments that are within the neighborhood distance were considered similar and the pair in the mini-batch were evaluated as anchor-positive pairs. For negative exemplars *semi-hard* segments were preferred and they were contained in the mini-batch for each triplets.

## 3.3 Database Indexing and Query Matching

Equipped with the final finger-printer which includes the trained and optimized embedding hash model, takes and input of audio signal with a determined length (e.g. 1sec. $\times f_s$) and generates hashes. The reference content is indexed on the database after passed through the finger-printer pipeline. Each song file in the content has its own file-ID and audio file data is segmented with a window length of 1sec. without overlapping for indexing on the database. Inserted key-value pairs are fingerprint-metadata information and each metadata will contain file-ID and starting offset of the audio file segment (see Fig. 3.4).

At query stage, up to a number of candidate matches (top-$t$) is proposed by the system for each segment of the query probe signal. In the common scenario where the query is longer than the window length (1sec.) and top-$t$ is bigger than one, the

match sequence is two-dimensional. Fig. 3.5 shows a query execution and the system response of candidate matches for 8 segment probe signal with top-$t$ equals to 5. In the lower table candidate matches are represented in a form of "*file-ID* : *start-offset*" symbolically. Only the **longest sequence** is shown with an almost perfect offset match for the query. Such perfect alignments are unrealistic and hard to achieve in real-world examples, thus in general adaptive scoring techniques is used for matches sequence estimations.

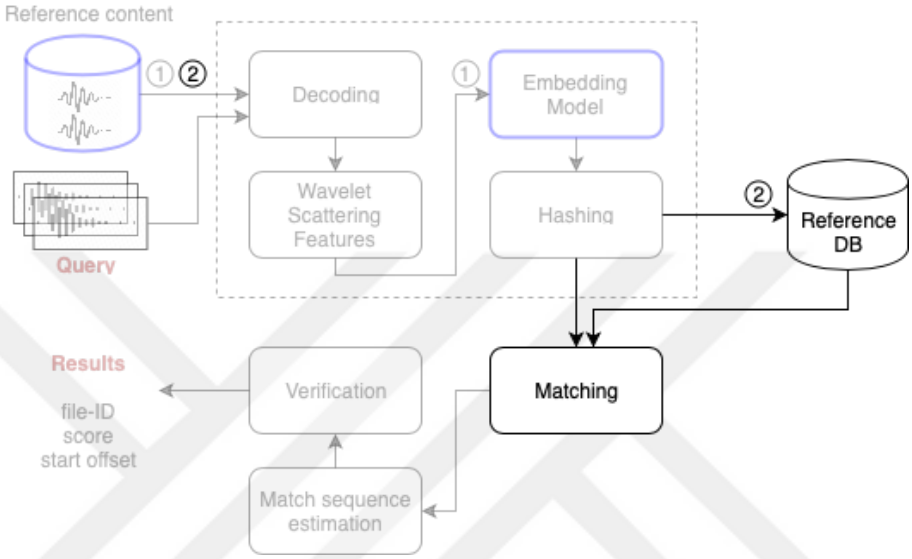



**Figure 3.4** Database indexing and query probe matching

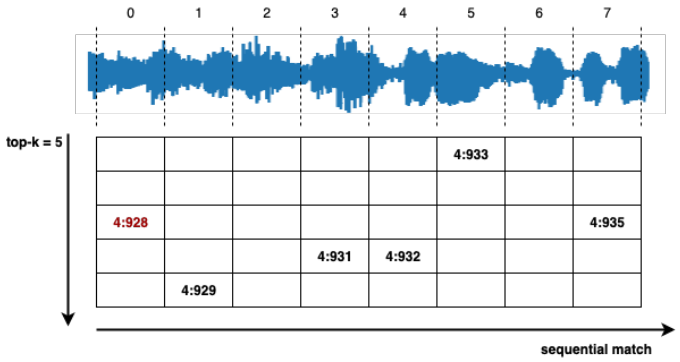



**Figure 3.5** Match sequence for query probe signal

## 3.4 Adaptive Scoring

The matches from database includes time-offset metadata, so we define an *adaptive scoring* method for sequential match estimation of an audio signal using following temporal constraints as a basis of dynamic-programming (similar to constraints in [68]) to decide whether proposed candidates are a match or not:

1. Define an anchor(initial) match for all of top-$t$ candidates (i.e. column values in Fig. 3.5).
2. For the tail of the anchor match, we define following temporal constraints:
   - Do not allow temporal backtracking of matched features.
   - Allow only matches the within the defined temporal cone for anchor match.
3. Promote the longest match sequence as the candidate.

Before retrieving the final results for the query probe, optional verification against final candidate match can be opt in as a separate module as depicted in system pipeline Fig. 3.6. Verification may include sub-thresholding for sequential match estimation or limiting sub-matches per query.

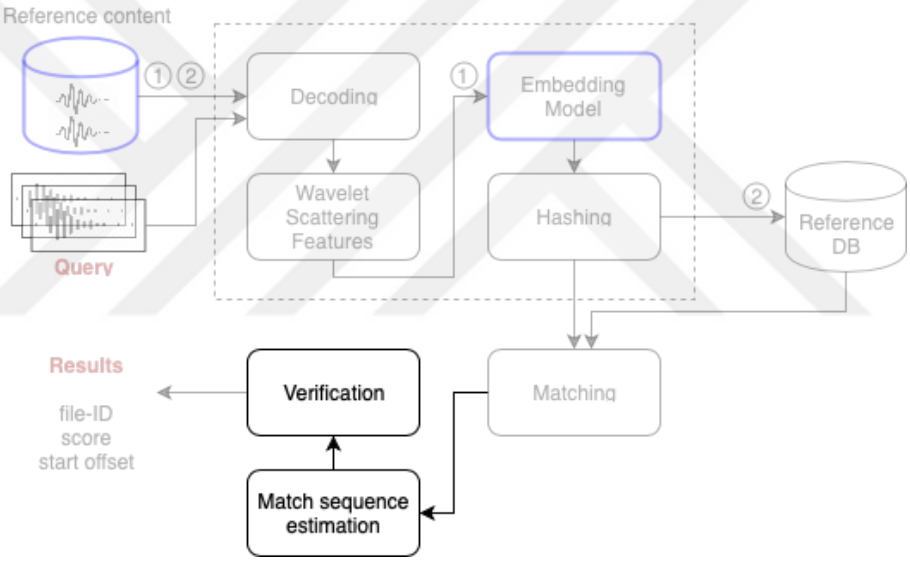



**Figure 3.6** Adaptive scoring for match sequence estimation

## 3.5 Experiments

After training the embedding hash model, the content database is built from clean test samples using the embedding model as the feature extractor. Content database consist of test split of GTZAN having #600 10-seconds long snippets. Each snippet is represented by features per every one-second without overlapping. Retrieval is first experimented by **one-by-one naive comparison** not to be influenced by database precision artifacts. One-second long segments are randomly sampled from query snippets with *arbitrary alignments* and features are extracted using the same embedding model. For each compared features agains database, $L^2$ distances are

**Table 3.3** Mean top-$t$=1 positive retrieval probability

| **Hash Dim.** | margin=0.2 | margin=1.0 | margin=2.0 |
|---|---|---|---|
| 8 | 0.8622 | 0.8731 | 0.8589 |
| 16 | 0.9266 | 0.9185 | 0.9011 |
| 32 | 0.9310 | 0.9343 | 0.9297 |
| 64 | 0.9404 | 0.9358 | 0.9263 |
| 96 | 0.9354 | **0.9411** | 0.9297 |

calculated and the features which regards top-$t$ smallest distance values are selected as candidates.

First we query the database with existing clean samples and calculate the top-$t$=1 mean score for multiple models. For each 10-seconds long snippets #10 randomly sampled segments are selected and their features are compared against the database. If the first result is a match (i.e. same song within the neighborhood offset) we increment the score. The reason of this experiments is to evaluate how margin value and hash dimension affect the retrieval probability of alignment and shift-invariance without any additional noise. Results are given in Table 3.3. As it can be seen best similarity preserving embeddings are obtained for 96-dimensional vectors and for $margin = 1.0$, although after 32-dimensional hash model, having more dimension doesn't seem to have great impact.

By selecting best $margin = 1.0$ value from Table 3.3, we calculate the same scores for noisy samples with various signal-to-noise-ratio (SNR) values. Experimented additional noise include (1) chatting people, in down-town streets, (2) noisy wind sound effect, (3) sound recorded inside a window of rainy day and (4) artificial synthetic glitchy noise, all retrieved from freesound.org [69] website. Results are given in Table 3.4.

We can see the dimensionality factor of the embeddings on discrimination in Table 3.4 clearly, for hash dimension equal to 96 we can retrieve positive samples with about %93 recall from a sparse database having features per only every one-second of audio signals with a very compact representation ($96 \times 4 = 384$ bytes/sec.). Also noted, the positive retrieval score for noise type (3) is poor having less than %50 for SNR below 6.

Lastly, a concrete recognition search is done against our indexed DB with top-$t$=20, i.e. each fingerprint can propose up to twenty potential matches. Query set includes both positive and negative samples and query snippets are superimposed with randomly selected noise type with SNR values varying from 0 to 3 (i.e. between half and same energy of noise applied). We execute the retrieval process with a sequence of snippets

**Table 3.4** Mean top-$t$=1 positive retrieval probability for noisy samples

| **Hash Dim.** | SNR=9 | SNR=6 | SNR=3 | SNR=0 |
|---|---|---|---|---|
| **(1) group-people-chatting-city** | | | | |
| 16 | 0.8468 | 0.7710 | 0.6327 | 0.4100 |
| 32 | 0.8685 | 0.8264 | 0.7252 | 0.5281 |
| 64 | 0.8958 | 0.8608 | 0.7737 | 0.6206 |
| 96 | 0.9064 | 0.8770 | 0.8020 | 0.6625 |
| **(2) wind-on-microphone** | | | | |
| 16 | 0.8581 | 0.7947 | 0.6618 | 0.4522 |
| 32 | 0.9025 | 0.8854 | 0.8341 | 0.7166 |
| 64 | 0.9245 | 0.9010 | 0.8833 | 0.8168 |
| 96 | **0.9312** | 0.9197 | 0.8884 | 0.8333 |
| **(3) raindrops-on-the-windows** | | | | |
| 16 | 0.6497 | 0.4625 | 0.2975 | 0.1412 |
| 32 | 0.7479 | 0.5954 | 0.3816 | 0.2006 |
| 64 | 0.8052 | 0.6697 | 0.4779 | 0.2737 |
| 96 | 0.8397 | 0.7279 | 0.5722 | 0.3677 |
| **(4) artificial-synthetic-noise** | | | | |
| 16 | 0.8120 | 0.6897 | 0.5008 | 0.2768 |
| 32 | 0.8658 | 0.8102 | 0.6737 | 0.4710 |
| 64 | 0.8966 | 0.8502 | 0.7845 | 0.6416 |
| 96 | 0.9056 | 0.8816 | 0.8166 | 0.6912 |

of audio with **random alignments** but with **increasing starting offsets**. Randomly selecting the stride amount is important to avoid problems of unlucky alignments; if the sampling of the probe is kept constant, it may be possible to repeatedly find samples that have uniformly large offsets from the sampling used to create the content storage [8]. Finally match sequence estimation is applied with adaptive scoring technique.

Fig. 3.7 shows Receiver operating characteristic (ROC) curve of the retrieval performance for various dimensional embeddings.

## 3.6 Application

In practice of applicability, having fingerprint with $p$-dimensions,

effectively searching near-neighbor in a $p$ dimension space is not a trivial job (particularly if $p$ is large); thus naive comparisons are inconvenient. For looking up for items similar to a query as a retrieval, it is not feasible to make comparisons over the entire data set. Rather, mostly methods of Approximated Nearest Neighbor (ANN) are preferred. For application considerations, in this study an ANN technique is used, termed locality-sensitive hashing (LSH). LSH techniques both efficient by the amount

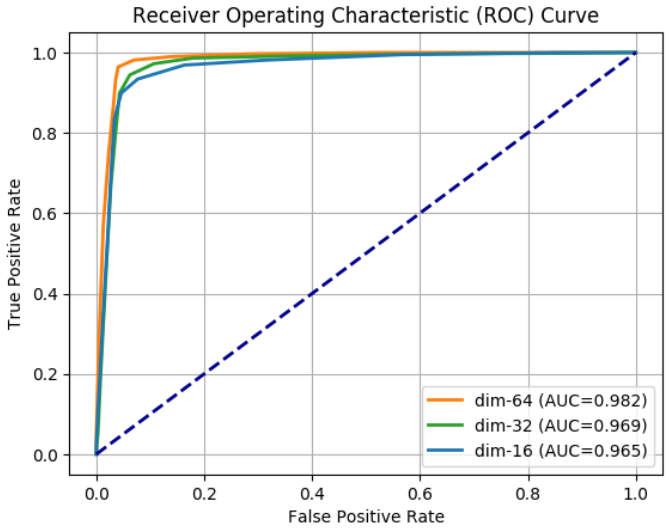



**Figure 3.7** Retrieval performance with adaptive query and naive one-by-one comparison against real-valued hash dimensions

comparisons needed (a tiny percentage of the dataset will be examined) and also provides noise-robustness properties through localization [70].

### 3.6.1 Locality-sensitive Hashing

Unlike more conventional hashing method, LSH conducts a sequence of hashes, each of which examines input vector portions or projections.

The overview of Locality-Sensitive Hashing technique is depicted in Fig. 3.8.

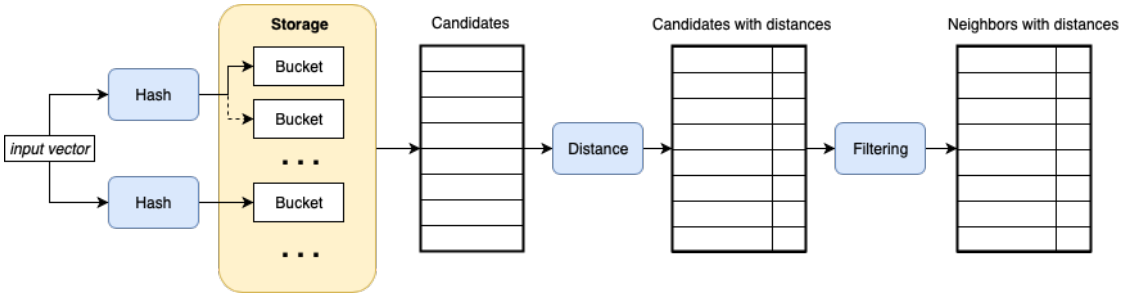



**Figure 3.8** Locality-Sensitive Hashing pipeline overview

LSH also promotes flexible limitations to further examine applicants from the individual hash tables as part of the final list of match candidates (by means of the sub-thresholding) shown as "Filtering" in the figure. This additional constraint corresponds "Verification" step depicted in Fig. 3.6. After optionally applying the filtering, the vector data with minimum distance is the best match for the probe data. For more than one candidate, LSH techniques support retrieving top-$t$ candidates sorted by the distance values.

In Fig. 3.8 given a query probe vector as input, each hash functions generate one (or more bucket) key(s) for content indexing. The vector is then stored for each key in a particular bucket during indexing. Later in search step, the near-neighbour candidates are gathered from the buckets in storage. The LSH application requires that the hashes generated should be locality-sensitive in general, thus to some extent **preserving the spatial structure**. Close vectors required to be placed into same buckets. After collecting from all the corresponding matching buckets in storage, the distance to the query vector is calculated using a distance measure (e.g. cosine distance) for all candidates. Lastly in the pipeline, an optional filtering is applied as a sub-thresholding step.

To preserve the spatial structure of vectors by means of locality, hash generating is executed as follows: the input vector is projected onto $N$-random vectors and for each projection a binary/discrete value is assigned based on the vector's location to the random vector. The projection value for the specific vector is string of binary/discrete values in the feature space as the hyperspace of $N$-random *normalized* vectors span. The projection is defined as the product of random normals $\mathbf{N}$ and input vector $\mathbf{v}$ ($\mathbf{p} = \mathbf{N} \times \mathbf{v}$) is given in Eq. (4.1);

$$\begin{bmatrix} n_{1,1} & \cdots & n_{1,D} \\ \vdots & \ddots & \\ n_{P,1} & & a_{P,D} \end{bmatrix}_{\#proj.\times dim.} \times \begin{bmatrix} v_1 \\ \vdots \\ v_D \end{bmatrix}_{dim.\times 1} = \begin{bmatrix} p_1 \\ \vdots \\ p_P \end{bmatrix}_{\#proj.\times 1} \tag{3.1}$$

where $\#proj.$ is number of random projection and $dim.$ is input vector dimension.

The random projection application of LSH is intended to approximate the *cosine distance* between vectors. Each hyperplane defined by the normal unit vector $\mathbf{r}$ is used to hash input vectors [71].

For given vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, the distance is settled as angle between two vectors, $\theta(\mathbf{u}, \mathbf{v}) = \arccos(\frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||\cdot||\mathbf{v}||})$. Based on angle as the distance between vectors, for a hyperplane defined by $\mathbf{r} \in \mathbb{R}^d$, the hash is defined as $h(\mathbf{v}) = \text{sing}(\mathbf{v}\cdot\mathbf{r})$, that is $h(\mathbf{v}) = \pm 1$ depending on which side of the $\mathbf{v}$ lies on partitioned hyperplane. Each possible choice of $\mathbf{r}$ defines a single hash function.

The probability of the hashes being equal (i.e. collusion) is given [72];

$$Pr[h(u) = h(v)] = 1 - \frac{\theta(\mathbf{u}, \mathbf{v})}{\pi} \tag{3.2}$$

**Table 3.5** LSH bucket-key examples

| Binary bucket-key | Discrete bucket-key |
|---|---|
| "1111010010" | "-3,0,0,0,0,-2,-1,-2,-2,-1" |
| "1011101010" | "-4,2,-2,0,0,0,-4,-5,-3,-1" |
| "0110000100" | "0,-1,-1,-5,0,-3,-1,-1,3,2" |
| "1100111100" | "-2,2,-2,-1,-2,-1,-1,3,0,1" |
| "0111010001" | "-1,-1,0,-1,1,0,-2,-1,1,0" |
| "0010100100" | "5,-2,1,-3,-1,0,3,1,-2,0" |
| "0000110110" | "2,-2,2,-1,-3,-3,-2,1,-3,0" |
| "0101110011" | "-3,-1,0,-3,0,-1,0,0,0,0" |
| #projection=10, LSH bin-width=4 | |

$\theta(\mathbf{u},\mathbf{v})/\pi$ is closely related to $\cos(\theta(\mathbf{u},\mathbf{v}))$ for small angles as can be seen in Fig. 3.9.

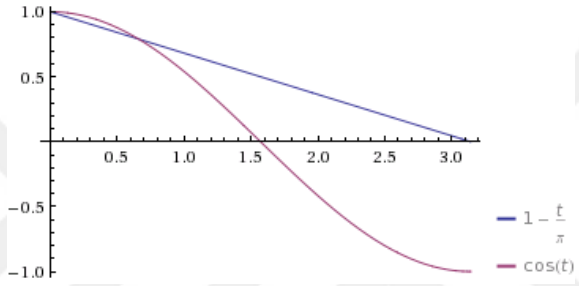



**Figure 3.9** $1-\theta/\pi$ approximation to $\cos(\theta)$

Hashing generates just a single bit (i.e. binary) in this setting. The probability of hash bits of two vectors match is proportional to the cosine of the angle between them.

In binary setting, $n$-th hash bit is calculated based on input vector $\mathbf{v}$ whether lies on the positive or negative side of the hyperplane defined by $n$-th normal vector (Eq. (4.3)). The permutation of the hash bits define the bucket-key, and for the possible buckets where the input vector belongs. If a *bin-width* value is used, the projection value is not quantized to a binary, but instead divided by the *bin-width* value (Eq. (4.4)), and using the bin index in each random projection as part of the bucket-key. Examples of generated bucket-key scheme for both binary and discrete values are given in Table. 3.5.

$$\text{hash}(\mathbf{P})_i = \begin{cases} 1; & p_i > 0 \\ 0; & \text{otherwise} \end{cases} \tag{3.3}$$

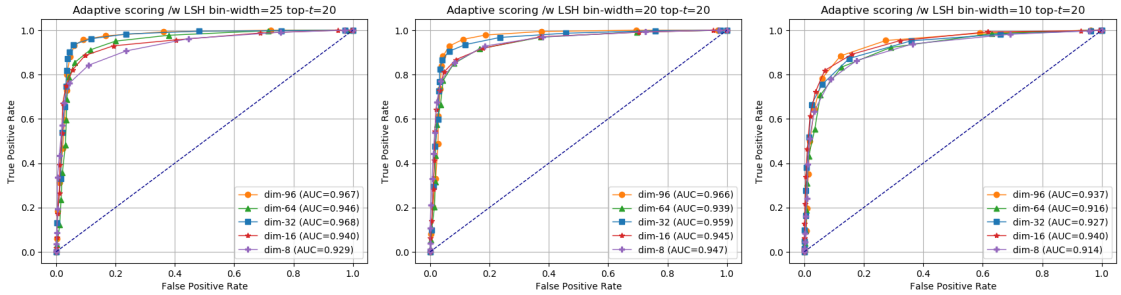
$$\text{hash}(\mathbf{P})_i = \lfloor p_i/\text{bin-width} \rfloor \tag{3.4}$$

**Table 3.6** ROC AUC scores using LSH

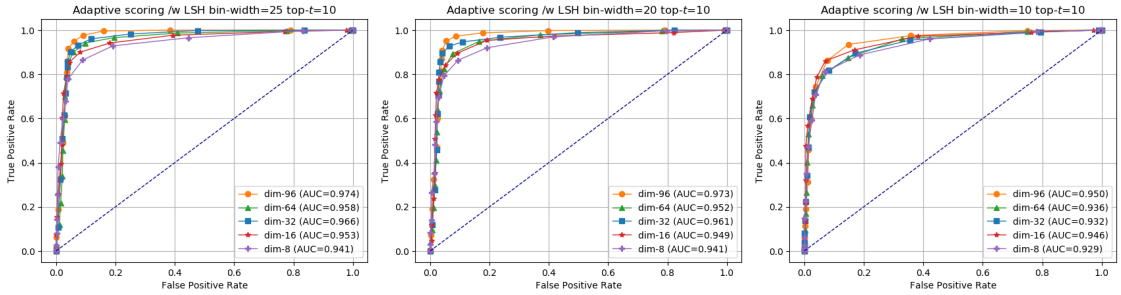
| **Real-valued hashes** | | | | | |
|---|---|---|---|---|---|
| **LSH bin-width** | dim=96 | dim=64 | dim=32 | dim=16 | dim=8 |
| LSH top-$t$=20 | | | | | |
| 25 | 0.966 | 0.945 | **0.967** | 0.940 | 0.929 |
| 20 | 0.965 | 0.939 | 0.959 | 0.944 | 0.946 |
| 10 | 0.936 | 0.916 | 0.927 | 0.940 | 0.913 |
| LSH top-$t$=10 | | | | | |
| 25 | **0.974** | 0.958 | 0.966 | 0.952 | 0.941 |
| 20 | 0.973 | 0.958 | 0.966 | 0.952 | 0.941 |
| 10 | 0.949 | 0.935 | 0.932 | 0.946 | 0.928 |

| **Binary-valued hashes (piece-wise thresholding)** | | | | | |
|---|---|---|---|---|---|
| **LSH bin-width** | dim=256 | dim=128 | dim=64 | dim=32 | dim=16 |
| LSH top-$t$=20 | | | | | |
| 25 | 0.886 | 0.916 | 0.907 | 0.914 | 0.906 |
| 20 | 0.851 | 0.892 | **0.917** | 0.914 | 0.905 |
| 10 | 0.606 | 0.699 | 0.792 | 0.874 | 0.913 |
| LSH top-$t$=10 | | | | | |
| 25 | 0.919 | 0.923 | **0.932** | 0.931 | 0.917 |
| 20 | 0.867 | 0.914 | 0.913 | 0.935 | 0.921 |
| 10 | 0.687 | 0.722 | 0.845 | 0.903 | 0.922 |

Retrieval results using LSH technique and querying 10-second long probe signals with additional noise superimposed with a range of SNR values between $[0, 3]$ are shown in followings table and figures. Results are conducted for various fingerprint dimensions for both real and binary (using piece-wise threshold embedding model) hashes. Table 3.6 shows ROC AUC scores for real-valued and binary-valued hashes. Fig. 3.10 and Fig. 3.11 show ROC curves for real-valued and binary-valued hashes with various output dimensions respectively.

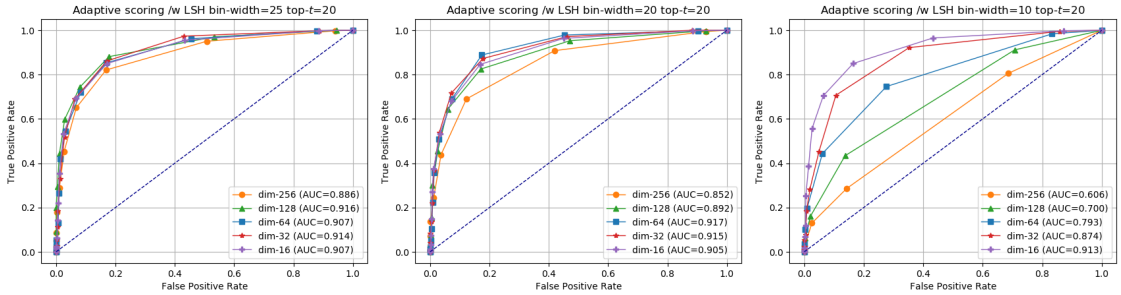



**(a)** bin-width=25 top-$t$=20 **(b)** bin-width=20 top-$t$=20 **(c)** bin-width=10 top-$t$=20

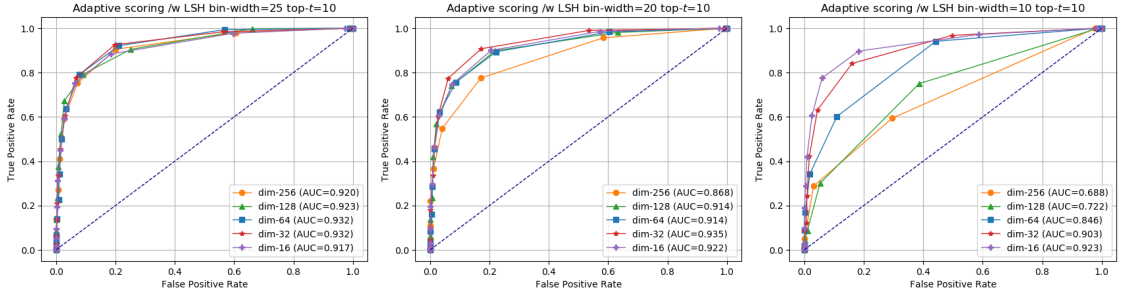



**(d)** bin-width=25 top-$t$=10 **(e)** bin-width=20 top-$t$=10 **(f)** bin-width=10 top-$t$=10

**Figure 3.10** ROC curves using LSH with real-valued hashes




**(a)** bin-width=25 top-$t$=20 **(b)** bin-width=20 top-$t$=20 **(c)** bin-width=10 top-$t$=20




**(d)** bin-width=25 top-$t$=10 **(e)** bin-width=20 top-$t$=10 **(f)** bin-width=10 top-$t$=10

**Figure 3.11** ROC curves using LSH with binary-valued hashes

# 4
# Conclusion and Recommendations

An audio fingerprint is a content-based compact signature that represents audio signal. Fingerprinting/hashing is an engineering task which includes fingerprint design, similarity metric and matching search for verification and recognition/identification of data in the interested domain. Real-world applications for audio fingerprinting varies from query-by-example music recognition, audio labeling, content-based integrity verification to copyright detection systems. For instance popular online music labeling systems are available including Shazam [6] [21], SoundHound [27] and Google Sound Search [73] with client-server architecture. A general architecture for a fingerprinting & recognition systems is depicted in Fig. 4.1.

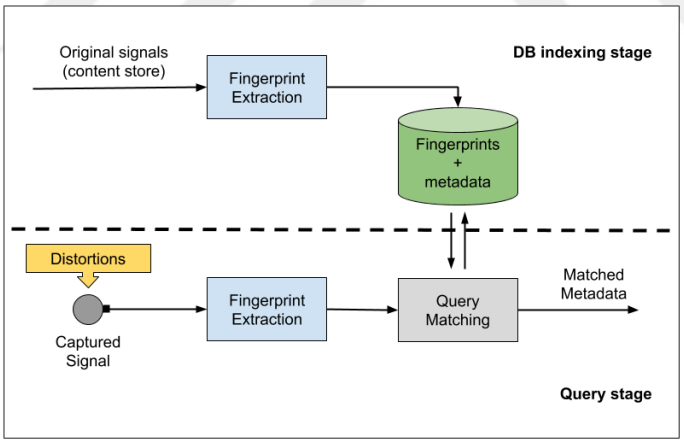



**Figure 4.1** General architecture of an audio fingerprinting system

Considering the tradeoff of system resources of real-world scenarios, precision/recall and response time requirements, what the academic research is mainly focused on is design of fingerprints that are essential i.e. being robust to various degradations, alignment problems in matching step, feature extraction/calculation complexity and compactness for fast retrieval. An audio signal may undergo many kind of distortions like additional background noise, reverberations, pitch shifting, interference in transmission, quantization and/or compression artifacts (i.e. GSM or MP3). The designed system should tackle with these obstacles.

In this study we explored a two-stage feature extraction method combining scattering wavelet transform (SWT) and deep embedding hash model to generate final fingerprints. SWT generates contractive representation of signals and provides Lipschitz continuity to deformations [74]. That is the distance between the transforms of the degraded and the original signals are bounded for a group of deformations satisfying necessary conditions. Then we build our deep hashing network based on combination of first-order and second-order scattering coefficients to model musical data efficiently. We first experimented with naive comparison method not to be influenced by database precision artifacts and later also with LSH techniques to be feasible working with large dimensional audio fingerprint data for retrieval and identification.

With an adaptive scoring scheme we can retrieve 0.982 ROC score for 96-dimensional hash fingerprints using naive comparisons. Our database is sparse, indexing only 96 floating-values (i.e $96 \times 4 = 384$bytes) per second for indexed audio signals without overlapping. Also limiting the hash dimension doesn't seem to have great impact as we suspect that because the database size is limited. For 16-dimensional hashes (64 bytes/sec.) the ROC score is above %95. Also in a feasible application settings, by using LSH method %97 ROC score is retrieved for 96-dimensional model.

In this study we tried to tackle most common obstacle for audio identification from the end-user perspective that is musical audio **tampered with environmental noise**. Noted that, this study does not focus on robustness of types pitch and tempo changes, which makes the application not suitable for scenarios like copyright detection where same content can be played back with different pitch and time-scale modifications. To be more comprehensive other group of degradation should be considered, such as how sampling rate changes effect the output our framework or could using features of long durations (i.e. one second) compensate linear-speed modification of audio signal should be answered. If the latter is not, keeping minimal duration of features vs. storage footprint trade-off should be adjusted carefully. Also other and/or larger datasets should be experimented with, and GTZAN dataset's faults should be considered (e.g. mostly repetitions in the song structures) [75] while extending this study.

# References


[1] J. Six, F. Bressan, and M. Leman, "A case for reproduciblity in mir-replication of'a highly robust audio fingerprinting system'," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 1, no. 1, 2018.

[2] E. Allamanche, J. Herre, O. Hellmuth, B. Fröba, T. Kastner, and M. Cremer, "Content-based identification of audio material using mpeg-7 low level description.," in *ISMIR*, 2001.

[3] J. Haitsma, T. Kalker, and J. Oostveen, "Robust audio hashing for content identification," in *International Workshop on Content-Based Multimedia Indexing*, Citeseer, vol. 4, 2001, pp. 117–124.

[4] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system.," in *Ismir*, vol. 2002, 2002, pp. 107–115.

[5] J. Herre, O. Hellmuth, and M. Cremer, "Scalable robust audio fingerprinting using mpeg-7 content description," in *2002 IEEE Workshop on Multimedia Signal Processing.*, IEEE, 2002, pp. 165–168.

[6] A. Wang *et al.*, "An industrial strength audio search algorithm.," in *Ismir*, Washington, DC, vol. 2003, 2003, pp. 7–13.

[7] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 597–604.

[8] S. Baluja and M. Covell, "Content fingerprinting using wavelets," 2006.

[9] C. Bellettini and G. Mazzini, "Reliable automatic recognition for pitch-shifted audio," in *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, IEEE, 2008, pp. 1–6.

[10] A. L.-c. Wang and D. Culbert, *Robust and invariant audio pattern matching*, US Patent 7,627,477, Dec. 2009.

[11] B. Zhu, W. Li, Z. Wang, and X. Xue, "A novel audio fingerprinting method robust to time scale modification and pitch shifting," in *Proceedings of the 18th ACM international conference on Multimedia*, ACM, 2010, pp. 987–990.

[12] D. P. Ellis, B. Whitman, and A. Porter, "Echoprint: An open music identification service," 2011.

[13] S. Fenet, G. Richard, Y. Grenier, *et al.*, "A scalable audio fingerprint method with robustness to pitch-shifting.," in *ISMIR*, 2011, pp. 121–126.

[14] M. Ramona and G. Peeters, "Audioprint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 818–822.

[15] C. Ouali, P. Dumouchel, and V. Gupta, "A robust audio fingerprinting method for content-based copy detection," in *2014 12th International Workshop on Content-Based Multimedia Indexing (CBMI)*, IEEE, 2014, pp. 1–6.

[16] B. Coover and J. Han, "A power mask based audio fingerprint," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 1394–1398.

[17] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Processing*, vol. 98, pp. 308–321, 2014.

[18] J. Six and M. Leman, "Panako: A scalable acoustic fingerprinting system handling time-scale and pitch modification," in *15th International Society for Music Information Retrieval Conference (ISMIR-2014)*, 2014.

[19] R. Sonnleitner, A. Arzt, and G. Widmer, "Landmark-based audio fingerprinting for dj mix monitoring.," in *ISMIR*, 2016, pp. 185–191.

[20] B. Gfeller, R. Guo, K. Kilgour, S. Kumar, J. Lyon, J. Odell, M. Ritter, D. Roblek, M. Sharifi, M. Velimirović, *et al.*, "Now playing: Continuous low-power music recognition," *arXiv preprint arXiv:1711.10958*, 2017.

[21] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[22] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[23] A. Finkelstein and D. Salesin, "Fast multiresolution image querying," in *Proceedings of the ACM SIGGRAPH Conference on Visualization: Art and Interdisciplinary Programs*, Citeseer, 1995, pp. 6–11.

[24] A. Arzt, S. Böck, and G. Widmer, "Fast identification of piece and score position via symbolic fingerprinting.," in *ISMIR*, 2012, pp. 433–438.

[25] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 3, pp. 271–284, 2005.

[26] C. V. Cotton and D. P. Ellis, "Audio fingerprinting to identify multiple videos of an event," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2010, pp. 2386–2389.

[27] A. S. Master, T. P. Stonehocker, B. J. Levitt, J. Huang, and K. Mohajer, *Systems and methods for sound recognition*, US Patent 9,280,598, Mar. 2016.

[28] N. Q. Duong and H.-T. Duong, "A review of audio features and statistical models exploited for voice pattern design," *arXiv preprint arXiv:1502.06811*, 2015.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[30] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE signal processing magazine*, vol. 8, no. ARTICLE, pp. 14–38, 1991.

[31] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[32] G. Kaiser, *A friendly guide to wavelets*. Springer Science & Business Media, 2010.

[33] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological society*, vol. 79, no. 1, pp. 61–78, 1998.

[34] D. Esteban and C. Galand, "Application of quadrature mirror filters to split band voice coding schemes," in *ICASSP'77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, vol. 2, 1977, pp. 191–195.

[35] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983.

[36] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE transactions on signal processing*, vol. 40, no. 9, pp. 2207–2232, 1992.

[37] P. P. Vaidyanathan and P.-Q. Hoang, "Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction qmf banks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 1, pp. 81–94, 1988.

[38] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[39] A. V. Oppenheim and R. W. Schafer, "Digital signal processing(book)," *Research supported by the Massachusetts Institute of Technology, Bell Telephone Laboratories, and Guggenheim Foundation. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975. 598 p*, 1975.

[40] J. Andén and S. Mallat, "Deep scattering spectrum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.

[41] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features.," in *iccv*, vol. 99, 1999, pp. 1150–1157.

[42] T. Dau, B. Kollmeier, and A. Kohlrausch, "Modeling auditory processing of amplitude modulation. i. detection and masking with narrow-band carriers," *The Journal of the Acoustical Society of America*, vol. 102, no. 5, pp. 2892–2905, 1997.

[43] E. C. Smith and M. S. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, no. 7079, p. 978, 2006.

[44] P. C. Mahalanobis, "On the generalized distance in statistics," National Institute of Science of India, 1936.

[45] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[46] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[47] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, IEEE, 2010, pp. 253–256.

[48] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[49] A. Amidi and S. Amidi, *Convolutional neural networks cheatsheet*. [Online]. Available: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks` (visited on 07/23/2019).

[50] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[51] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 27–35.

[52] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3270–3278.

[53] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2916–2929, 2012.

[54] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search.," in *ICCV*, vol. 9, 2009, pp. 2130–2137.

[55] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[56] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[57] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[58] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[59] L. Wu, Y. Wang, J. Gao, and X. Li, "Where-and-when to look: Deep siamese attention networks for video-based person re-identification," *IEEE Transactions on Multimedia*, 2018.

[60] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4353–4361.

[61] S. Chopra, R. Hadsell, Y. LeCun, *et al.*, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR (1)*, 2005, pp. 539–546.

[62] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1735–1742.

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

[64] ——, *Gtzan genre collection*, 2002. [Online]. Available: `http://marsyas.info/downloads/datasets.html` (visited on 07/23/2019).

[65] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[66] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, E. Belilovsky, J. Bruna, *et al.*, "Kymatio: Scattering transforms in python," *arXiv preprint arXiv:1812.11214*, 2018.

[67] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[68] S. Baluja and M. Covell, "Audio fingerprinting: Combining computer vision & data stream processing," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, IEEE, vol. 2, 2007, pp. II–213.

[69] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*, ACM, 2013, pp. 411–412.

[70] A. Gionis, P. Indyk, R. Motwani, *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, 1999, pp. 518–529.

[71] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, ACM, 2002, pp. 380–388.

[72] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, p. 117, 2008.

[73] *Google sound search*. [Online]. Available: `https://support.google.com/googleplaymusic/answer/2913276` (visited on 07/23/2019).

[74] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[75] B. L. Sturm, "The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use," *arXiv preprint arXiv:1306.1461*, 2013.

# Publications From the Thesis

**Contact Information:** kanalici.evren@gmail.com

## Conference Papers

1. E. Kanalici and G. Bilgin, "Scattering Wavelet Hash Fingerprints for Musical Audio Recognition", In International Journal of Innovative Technology and Exploring Engineering, pp. 1011-1015. BEIESP, 2019.

2. E. Kanalici and G. Bilgin, "Music Genre Classification via Sequential Wavelet Scattering Feature Learning." In International Conference on Knowledge Science, Engineering and Management, pp. 365-372. Springer, Cham, 2019.